

**SUPPORTING FUNCTIONALITY-BASED DESIGN IN
COMPUTER-AIDED DESIGN SYSTEMS**

by

Obinna Stan Muogboh

Bachelor of ENGR, in Electronics Engineering, University of Nigeria, 1996

Master of Science, in Industrial Engineering, University of Pittsburgh, 2000

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

2003

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This dissertation was presented

by

Obinna Stan Muogboh

It was defended on

February 4th, 2003

and approved by

Dr. Bopaya Bidanda, Professor, Department of Industrial Engineering

Dr. Mary Besterfield-Sacre, Assistant Professor, Department of Industrial Engineering

Dr. Michael Lovell, Associate Professor, Department of Mechanical Engineering

Dr. Dipo Onipede, Assistant Professor, Department of Mechanical Engineering

Dissertation Director: Dr. Bartholomew O. Nnaji, Professor, Department of Industrial Engineering

ABSTRACT

SUPPORTING FUNCTIONALITY-BASED DESIGN IN COMPUTER-AIDED DESIGN SYSTEMS

Obinna Stan Muogboh, PhD

University of Pittsburgh, 2003

Designs are conceptualized in terms of the functions they need to accomplish. The need for a new product design arises as a result of the identification of a new functionality to be accomplished by the product. That is, design is functionality driven. However, existing CAD tools are not equipped to capture functionality or reason in such a fashion to support design for product functionality. This research proposes a new design formalism to enable functionality-driven design of mechanically engineered products. This procedure provides a methodology that allows a designer to model product functionality and to carry out conceptual design with the aid of a computer. It also serves as a bridging tool between the conceptual design phase and detailed design phase of a product. Thus, the primary objective of this research is to develop a methodology that will support the following activities in CAD systems: functionality modeling, functionality data structuring, and form conceptualization.

The functionality modeling methodology developed in this work includes the use of operands, operators, and coupling bonds to describe product functionality in CAD systems. The Universal Modeling Language (an object-oriented programming technique) is used to model product functionality in computer systems.

The tools developed in this research provide a means of modeling and propagating product functionality information to downstream design activities. The propagation of functionality as a constraint is achieved using Extensible Markup Language (XML) data files. These tools also provide a mechanism for verifying and enforcing constraints on solid CAD models. The functionality definition interface is implemented with a customized Microsoft Visio graphics engine.

The tools developed in this research provide a means of modeling and propagating product functionality information to downstream design activities. It also provides a mechanism for verifying and enforcing constraints on solid CAD models. The functionality definition interface is implemented with a customized Microsoft Visio graphics engine.

DESCRIPTORS

CAD	Computer-Aided Conceptual Design
Computer-Aided Design	Conceptualization
Coupling Bond	Design
Functional Marker	Functional Model
Functionality-based Design	Functionality Constraint
Functionality Modeling	Functionality Primitive
Operand	Operator
Product Function	XML

ACKNOWLEDGEMENTS

I received a lot of support from many people during the course of this research. I owe special gratitude to my academic advisor, Prof. Bart. O. Nnaji, for his advice, inspiration, and support in this work. My deep gratitude goes to the entire faculty and staff of the Department of Industrial Engineering at the University of Pittsburgh, for the support and kindness they have shown me over the years. I thank Drs. Mary Besterfield-Sacre, Bopaya Bidanda, Dipo Onipede, and Mike Lovell, for accepting to serve in my dissertation committee, and for providing invaluable advice and constructive criticism to this work.

I appreciate the efforts of all the team members of Pegasus *e*-Designer project, whose constructive criticism and discussions lend a great impetus to the success of this project. My heartfelt thanks go to all the Automation and Robotics laboratory members for their support and suggestions in this work. I acknowledge the help of Kyoung Kim in providing useful suggestions and assistance with the formatting of this report. The numerous discussions with him contributed to the success of this work. I also acknowledge the help of Pamela Ajoku in proof reading parts of this dissertation.

I am very grateful to my wife, Roseline Muogboh, for all the love, support, kindness, and for proof reading some of my work. And for their continued understanding, prayers, and support through this study, I thank my entire family, and friends. Finally, for His divine guidance and blessings, I am grateful to the Almighty God.

TABLE OF CONTENTS

ABSTRACT.....	iii
ACKNOWLEDGEMENTS	v
LIST OF TABLES	xi
LIST OF FIGURES	xiv
1.0 INTRODUCTION.....	1
1.1 Problem Statement.....	4
1.2 Research Objective	6
1.3 Contributions.....	7
1.4 Methodology.....	8
1.4.1 Information Flow in Functionality-enabled CAD System.....	8
1.4.2 Functionality Modeling Process	11
1.5 Research Organization.....	14
2.0 TECHNICAL BACKGROUND	18
2.1 Design of Mechanically Engineered Products.....	18
2.1.1 Design Process.....	20
2.1.2 Feature-Based Design	27
2.2 Functionality-based Design	28
2.2.1 Functionality Modeling and Verification.....	28

2.2.2 Input-Output Model (Black-box Model)	36
2.2.3 Functionality Decomposition.....	37
2.2.4 Functionality-based Product Design Conceptualization	39
2.3 Spatial Relationship in Product Design	44
2.4 Design Constraints	48
3.0 PROPOSED FUNCTIONALITY-BASED DESIGN MODEL.....	52
3.1 Modeling Scope	55
3.2 Major Challenges	56
3.3 Functionality Operations.....	58
3.3.1 Basic Functionality Operations (<i>BOPN</i>).....	62
3.3.2 Compound Functionality Operations (<i>COPN</i>).....	63
3.3.3 Generic Functionality Model	64
4.0 FUNCTIONALITY OPERANDS.....	68
4.1 Material Operand: Solid Operands	72
4.1.1 Functional Position Marker.....	75
4.1.2 Degree of Freedom: Kinematic Constraints	80
4.1.3 Physical Constraint Set	82
4.1.4 Material Type.....	84
4.1.5 Mass Properties	87
4.1.6 Topology	87
4.2 Energy Operand – Mechanical Energy Operand	90
4.2.1 FORCE Operand	92
4.2.2 TORQUE Operand.....	98

5.0 FUNCTIONALITY RELATIONS AND STATES	103
5.1 Functionality Relations	103
5.1.1 Coupling Bond (CB)	108
5.1.2 Representation of Functionality Relations	112
5.1.3 Derivation of Functionality Relations	114
5.2 Functionality States	121
5.3 Functionality Modeling Repository	127
6.0 IMPLEMENTATION AND TESTING	132
6.1 Functionality Object Model	132
6.1.1 Functionality Operation	137
6.1.2 Operand Set	138
6.1.3 Sub-Functionality Set	141
6.1.4 Form	141
6.1.5 Coupling Bond (CB) Set	142
6.1.6 States	142
6.2 Computer Implementation	142
6.2.1 Functionality-Based Design Procedure	144
6.2.2 Functionality-Based Design (FbD) Architecture	148
6.2.3 XML Functionality Data Format	151
6.2.4 Graphic User Interface and General Capability	159
6.3 Testing and Validation	178
6.3.1 Functionality-based Design (FbD) Procedure	179
6.3.2 Functionality Modeling of an Automotive Space-Frame Sub-assembly	184
6.3.3 Evaluation of Application of Methodology	221

7.0 CONCLUSION AND FUTURE WORK	229
7.1 Conclusion	229
7.2 Future Work	231
7.2.1 Integration with CAD System.....	232
7.2.2 Extension to Commercial Product Level	234
APPENDIX A	237
A.1 Definition of Aluminum Alloy	237
A.2 Test Condition 1	238
A.3 Test Condition 2	239
A.4 Test Condition 3	240
A.5 Result of Design 1, Test condition 1	241
A.6 Result of Design 1, Test condition 2	242
A.7 Result of Design 1, Test condition 3	243
A.8 Result of Design 2, Test condition 1	244
A.9 Result of Design 2, Test condition 2	245
A.10 Result of Design 2, Test condition 3	246
APPENDIX B	247
B.1 Transformation Operation	247
B.2 Transmission Operation	249
B.3 Joint Operation	252
B.4 Load bearing Operation.....	255
B.5 Energy Converter Operation	257
B.6 Frictional Operation	258
B.7 Offset Operation.....	260

B.8 Channel Operation.....	262
B.9 Block Operation	264
BIBLIOGRAPHY.....	268

LIST OF TABLES

Table 1 Definition of a functional prototype (proposed by Tomiyama et al ^[24])	33
Table 2 Mathematical description of the transmission operation shown in Figure 19	70
Table 3 Mathematical description of the transformation operation shown in Figure 20	70
Table 4 Inter-operand relationship matrix.	114
Table 5 Listing of functionality operation states	126
Table 6 Functionality models of commonly used mechanical product functionality	130
Table 7 Summary of the master shapes in the basic functionality stencil	163
Table 8 Summary of the master shapes in the solid operand stencil	166
Table 9 Summary of the master shapes in the energy operand stencil	170
Table 10 Inter-functional feature constraints representing geometric constraint set	183
Table 11 Task - functional features QFD for the car frame	187
Table 12 Functionality attributes of cross-beam solid operand	196
Table 13 Functionality attributes of tee-beam solid operand	197
Table 14 Functionality attributes of F ₁ force-energy operand	198
Table 15 Functionality attributes of F ₄ force-energy operand	198
Table 16 Functionality attributes of F ₂ force-energy operand	199
Table 17 Functionality attributes of F ₃ force-energy operand	199
Table 18 Functionality attributes of F ₅ force-energy operand	200

Table 19 Functionality attributes of F_6 force-energy operand	200
Table 20 Coupling bond elements	206
Table 21 Joint operation: crossbeam-teebeam coupling bond (x_{12}).....	208
Table 22 Load bearing: crossbeam- F_1 coupling bond (x_{13})	212
Table 23 Load bearing: crossbeam- F_2 coupling bond (x_{14})	213
Table 24 Load bearing: crossbeam- F_3 coupling bond (x_{15}).....	213
Table 25 Load bearing: crossbeam- F_4 coupling bond (x_{16}).....	214
Table 26 Load bearing: teebeam- F_5 coupling bond (x_{27}).....	214
Table 27 Load bearing: teebeam- F_6 coupling bond (x_{28}).....	215
Table 28 Equilibrium constraint: global coupling ($F_1, F_2, F_3, F_4, F_5, F_6$)	217
Table 29 ANSYS test result showing maximum normal stress in space-frame	227
Table 30 FbD capability versus Existing Commercial CAD Systems	228
Table 31 "Aluminum alloy" properties.....	237
Table 32 "Aluminum alloy" stress limits.....	237
Table 33 Model : parts	238
Table 34 Contact conditions	238
Table 35 Structural loading.....	238
Table 36 Structural supports	238
Table 37 Model : parts	239
Table 38 Contact conditions	239
Table 39 Structural loading.....	239
Table 40 Structural supports	239
Table 41 Model : parts	240

Table 42 Contact conditions	240
Table 43 Structural loading.....	240
Table 44 Structural supports	240
Table 45 Structural results	241
Table 46 Structural results	242
Table 47 Structural results	243
Table 48 Structural results	244
Table 49 Structural results	245
Table 50 Structural results	246
Table 51 Force to torque transformation functionality operation.....	249
Table 52 Solid transmission functionality operation.....	251

LIST OF FIGURES

Figure 1 Relationship between product development cycle and design commitment in term of cost and changeability.....	5
Figure 2 Functionality information flow diagram for the evolution of design in FbD.....	9
Figure 3 Sample transmission design	12
Figure 4 Product design phases.....	15
Figure 5 Conceptual design unit interactions with the rest of the designer system.....	16
Figure 6 Classification of design	22
Figure 7 Design process.....	23
Figure 8 Feedback control loop depicting the design process	24
Figure 9 Design as the mapping from functional space to physical space	25
Figure 10 Input-output model of system.....	36
Figure 11 Product function decomposition.....	38
Figure 12 Physical effect: earth's gravitational pull on object, O	43
Figure 13 Six types of spatial relationships: (a) against, (b) parallel-offset, (c) include -angle, (d) parax-offset, (e) aligned, (f) incline-offset.....	46
Figure 14 An example of assembly by spatial relationships.....	47
Figure 15 Mechanical product sample – dumb-bell	53
Figure 16 Energy conversion functionality.....	56
Figure 17 Load bearing as an example of mechanical functionality	60

Figure 18	Functionality operands	68
Figure 19	Transmission functionality operation illustrating the interaction between FORCE and MEDIUM operands: transmission of force from point "A" to plate "B"	69
Figure 20	FORCE to TORQUE transformation operation functionality	71
Figure 21	Material operand structure illustrating a detailed attribute modeling of a solid material	74
Figure 22	Operand coupling at the functional points, A1 and A2	76
Figure 23	Wire-frame representation of a simple solid material operand	77
Figure 24	Wire-frame representation of multipoint solid material operand with four functional points A, B, C, and M	77
Figure 25	The markers on functional coupling features	78
Figure 26	Inter-functional feature constraints representing geometric constraint set	80
Figure 27	Aligned spatial relation specification example	82
Figure 28	Material properties	86
Figure 29	Boundary modeling in product conceptualization with <i>functional points</i> only	88
Figure 30	Boundary modeling in product conceptualization illustrating use of <i>functional points</i> (M_1 , M_2 , and M_4), <i>functional region</i> (M_3), and <i>reference functional point</i> (M_{ref})	89
Figure 31	Mechanical energy converter	90
Figure 32	Mechanical energy operands	92
Figure 33	FORCE operand model	95
Figure 34	Rotational energy in a rotating shaft	98
Figure 35	Direction of torque is given by the right hand rule	100
Figure 36	TORQUE operand model	102
Figure 37	Functionality operator linking four operands	103
Figure 38	Solid operands in sliding contact	104

Figure 39 Operator-operand-attribute relation for sliding contact example	105
Figure 40 A compound functionality showing inter-primitive interaction.....	107
Figure 41 A compound functionality showing inter-primitive interaction between the coupling operands	107
Figure 42 Functionality operand coupling bond.....	108
Figure 43 Two solid blocks in sliding contact.....	111
Figure 44 Coupling bond for the two solid blocks in sliding contact.....	112
Figure 45 Functionality primitive A	119
Figure 46 Brake rotor states.....	122
Figure 47 A pulley system	123
Figure 48 A three-operand functionality operation	125
Figure 49 Example of compound functionality in a cart	131
Figure 50 Product functionality decomposition.....	133
Figure 51 Functionality class diagram.....	136
Figure 52 Functionality-based design flow chart.....	145
Figure 53 Mapping from design task to functional requirement	146
Figure 54 Mapping from functional requirement to functionality model.....	147
Figure 55 Functionality-based design flow diagram for the evolution of design.....	149
Figure 56 Sample transmission design	150
Figure 57 Functionality modeling interface using Microsoft Visio.....	161
Figure 58 Functionality operation start-up definition dialog box.....	162
Figure 59 Modeling of functional line in Visio	165
Figure 60 Modeling of functional arc in Visio	166
Figure 61 Functional point user form	168

Figure 62 Functional line user form.....	168
Figure 63 Functional arc user form.....	169
Figure 64 Solid operand definition user form.....	169
Figure 65 Force operand definition user form	171
Figure 66 Torque operand definition user form.....	172
Figure 67 Coupling bond definition user form	173
Figure 68 Quantitative functional relations definition user form	175
Figure 69 Spatial relations definition interface.....	176
Figure 70 Quantitative constraints definition user form.....	177
Figure 71 Functionality modeling flow diagram	181
Figure 72 Space-frame of a car	184
Figure 73 Solid operand features of the frame.....	188
Figure 74 Solid operands: SOLID{A, B, X} and SOLID{C}	189
Figure 75 Resultant loading forces on the frame structure.....	192
Figure 76 Car frame operand interaction graph.....	205
Figure 77 Application of constraints on functional markers A, B, and C	209
Figure 78 Application of constraint on joint location.....	210
Figure 79 Conceptual form set for joint operation (1 = 1, 2, 3)	210
Figure 80 Dialog box showing the start-up screen of the Visio interface	218
Figure 81 Screen capture of a wireframe model of the crossbeam solid operand	218
Figure 82 Screen capture of the space-frame operand interaction graph.....	219
Figure 83 Screen capture of a wireframe model of the coupled solid operands.....	220
Figure 84 CAD model of test design	226

Figure 85 Loading condition for test condition 1	238
Figure 86 Loading condition for test condition 2	239
Figure 87 Loading condition for test condition 3	240
Figure 88 Normal stress distributions for design 1, test condition 1	241
Figure 89 Normal stress distributions for design 1, test condition 2	242
Figure 90 Normal stress distributions for design 1, test condition 3	243
Figure 91 Normal stress distributions for design 2, test condition 1	244
Figure 92 Normal stress distributions for design 2, test condition 2	245
Figure 93 Normal stress distributions for design 2, test condition 3	246
Figure 94 Transform operation	247
Figure 95 FORCE to TORQUE transformation functionality operation.....	248
Figure 96 Transmission operation functionality	250
Figure 97 Solid transmission functionality example	251
Figure 98 Joint functionality operation.....	252
Figure 99 Load bearing functionality operation	255
Figure 100 Energy converter functionality operation.....	257
Figure 101 Frictional functionality operation.....	258
Figure 102 Offset functionality operation.....	261
Figure 103 Channel functionality operation	263
Figure 104 Block functionality operation.....	265

1.0 INTRODUCTION

In the past two decades, the application of computers in the design of mechanical products has seen tremendous improvements ranging from the use of advanced graphic engines for solid modeling to automatic imposition of geometric constraints. ^[1, 2, 3] This evolution, however, has focused on improving the tools for the processes involved in the detailed design phase of product design. Little advancement has been made to allow computer-aided design (CAD) systems to capture and model product functionality during conceptualization. Such acquisition would have made it possible to impose functionality as a design constraint that may be used during the detailed design and analysis phase of the product.

In this research, a new functionality modeling formalism is proposed to enable functionality-based design of mechanically engineered products. This procedure provides a framework that allows a designer to model product functionality and carry out conceptual design with the aid of a computer. It also serves as a bridge between the conceptual design phase and detailed design phase of a product. It is envisaged that this new design procedure will improve the creativity and productivity of the designer.

Existing design tools are not equipped to capture design intent (functionality) at a high level. Hence, they cannot impose functionality as a product design constraint during the rest of the design phase. This inadequacy implies that the designers still have to apply both experience and extensive design tests to ensure that the original design intents (functionality) are maintained throughout the design process. A truly integrated CAD system would require that such tests be performed by the CAD system in a transparent manner. Transparency implies that test for conformity to functionality is performed concurrently during the design process in the same

CAD environment. The development of such integrated system will drastically reduce the product development time and the associated costs. It will also improve the efficiency and productivity of the engineering design process.

Current CAD packages have no means of checking designs in a transparent manner to ensure that the product functionality is maintained throughout the design phase. The implication of this is that most designers will at the end of their “design” phase; export their design to independent analysis systems, where such systems exist, to test for each individual product functionality. For example, the design of a load-bearing beam subject to high temperatures, a designer would at the end of the design process, employ Finite Element Analysis (FEA) packages to perform analysis on the stress response of the beam. In addition, a separate analysis will be performed for the thermal response. A failure in thermal analysis and pass in the stress analysis will result in a design change to meet the thermal need. This change, however, may result in a violation of the stress requirement (after another analysis), and hence, another change in design. This iteration continues until a suitable compromise is reached. A suitable translation of the product functionality as a design constraint and subsequent imposition of such constraint to down stream design activities will eliminate this time consuming sequential design-analysis process.

Engineering design is a mapping of a specified product onto a (description of a) realizable physical structure – the designed artifact.^[8] The desired function of the artifact is what it is supposed to do. The physical structure is the actual physical parts out of which it is made, and the part-whole relationship among them. Typically, the need for a new design arises as a result of identification of a new functionality to be accomplished by the product. Designs are conceptualized in terms of the functions they need to accomplish. That is, design is

functionality driven. Although, designs are functionality driven, modern CAD tools are not equipped to capture and reason in such a fashion to design for product functionality. Because of this, designers usually carry out the most important phase of design -- conceptual design, without the aid of any CAD tool. This is usually done by reasoning about the concepts involved in the particular application to arrive at a design concept.

The benefits of concurrent engineering ^[4, 5, 6] has directed recent research effort in the design of CAD/CAM systems towards a seamless computer integration of various processes involved in product realization. Such integration is hampered by lack of suitable representation and modeling techniques for the various design issues. This inadequacy (which was emphasized in a recent NSF sponsored workshop on *e-Product Design and Realization*) ^[7] is pronounced in the representation of product functionality. Consequently, no CAD tool exists to effectively aid the conceptualization of products and subsequent propagation to the conventional CAD systems for detailed design activities. With the available CAD systems, it is difficult to link geometric solid models (from detailed design phase) with the conceptual models developed during the early design phases as they are based on different representation schemes. This lack of continuity implies that the conceptual and detailed design phases are treated as isolated units that inherit all the disadvantages of the traditional over-the-wall product development process.

Another inadequacy of available CAD packages is the lack of a flexible design environment that supports the re-use of past design experiences and knowledge. Past design experiences are stored as product solid models and features with no direct provision to capture the function of the product. With the high shortage and rising cost of domain experts, such a system is of great importance. A knowledge-based CAD system (using the tool proposed in this

work) will overcome this problem by providing the expert knowledge in a virtual form through the use of artificial intelligence (AI) techniques.

This work is focused on developing a suitable way of representing and propagating product function as a design constraint. A suitable representation of Design for Functionality (DFF) will ensure that a design constraint defined early in the design process (usually during conceptual design phase) is visible and enforced throughout the entire design process. This task will require the development of a generic procedure for propagating functionality description to downstream design activities. This propagation is accomplished in this work by representing functionality in an Extensible Markup Language (XML) file format that is tagged to the CAD data and file.

1.1 Problem Statement

Design is the process of constructing a description of an artifact that satisfies a functional specification, meets certain performance criteria and resource limitations, is realizable in a given target technology, and satisfies criteria such as testability, manufacturability, reusability, etc.^[8] Decisions made early in the design process have a significant impact on other aspects of a product's life cycle. A study conducted by Lotter^[9] indicates that about 75% of the entire cost of a product is committed during the design phase. This commitment also extends to design changeability (ability to change or influence the final product design) as illustrated in Figure 1.^[10] Conceptual design goes a long way in defining the nature and amount of work required during the detailed design phase and other subsequent activities such as manufacturing. A poorly conceived design cannot be compensated for by good detailed design, since the design

direction and possibly scope has already been laid down during the conceptual stage. This is to say that detailed design phase merely works within the scope defined during the conceptual stage.

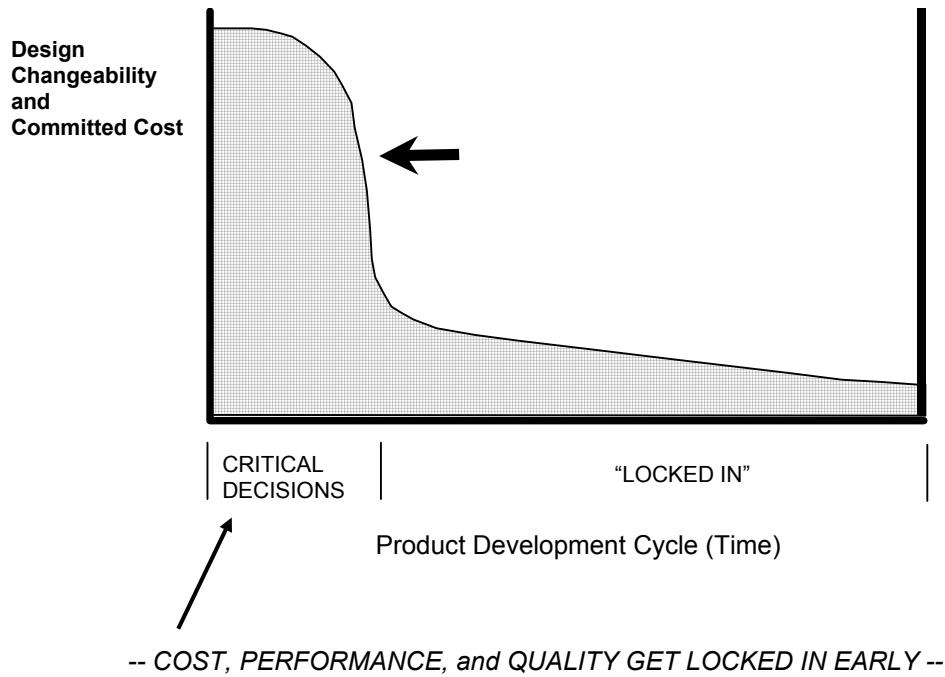


Figure 1 Relationship between product development cycle and design commitment in term of cost and changeability

Unfortunately, product requirement and constraints such as safety, reliability, manufacturability, and assemblability are imprecise and vague during the conceptual design. Usually, the designer only knows the need / function of the product. Hence, the need to utilize the functionality requirement effectively and efficiently during product conception is of paramount importance. However, existing commercial CAD systems are not equipped to support and use this essential information – functionality – during the design of a product.

Reasoning about product functionality are largely done in the designer's head with very limited or no aid of computing tools.

Product functionality modeling (representation, propagation, and satisfaction) is a very challenging problem that involves the amalgamation of various engineering fields such as kinematics, artificial intelligence, feature and geometric reasoning, constraint reasoning, and domain specific knowledge such as material properties. The design of a physical system involves additionally reasoning about the artifact's behavior, both internal and external. The external behavior of a system is what it does from the viewpoint of an outside observer. Internal behavior is based on observing what the parts of the system do. With increasing complexity in mechanical designs and the need for design reuse and CAD/CAM integration, there is a need for a system that will adequately model and propagate product functionality during the conceptual design phase of a product.

This research will develop product functionality model that are suitable for computer implementation. The model will enable CAD systems to support functionality-based design at both the early and detailed stages of design. Hence, definition, representation, propagation, and satisfaction of functionality as a design constraint is made possible.

1.2 Research Objective

Representation of product functionality as a constraint will assure that functionality defined early in the design process is visible and enforced throughout the entire design process. The primary objective of this research is to develop, for a mechanically engineered product, a

functionality modeling methodology that will enable functionality-based design (FbD) activities in CAD systems. The specific tasks to be accomplished in this research include:

- *Functionality modeling:* The work to be done under this research task consists of the development of a scheme for representing, propagating, and enforcing the product functionality as a design constraint. This will enable CAD systems to capture the functionality-related information during the conceptual design phase.
- *Computer data structure development:* This includes the development of a computable modeling data structure for representation and propagation of product functionality in a transparent manner.

1.3 Contributions

A functionality modeling methodology together with associated computer tools is developed to allow for the development of computer tools to link design functions with the structural (physical) embodiment used to realize the functions. Hence, a seamless integration between specification, conceptualization, and detailed design process can be facilitated.

The following are the expected or anticipated contributions of this work in the field of Computer-Aided Design (CAD) of mechanical products.

- Provision of a functionality modeling methodology that supports the definition and representation of product functionality as design constraints.

- Provision of a means for propagating product functionality developed during the early design stage to the detailed design stage. This provision is made possible by the description of the functionality information as an extensible markup language (XML) data file with functionality constraints and CAD form linkage embedded as tags.
- Introduction of the concepts of operands, operators, and coupling bonds in the modeling of product functionality.
- Development of a framework to support the functionality-based design (FbD) of mechanical products.
- Provision of a bridge between the conceptual and detailed design stages of design.

1.4 Methodology

1.4.1 Information Flow in Functionality-enabled CAD System

The information flow of a functionality-based designer system is shown in Figure 2. This work will focus on the functionality modeling interface section of the designer system.

Functionality Modeling Interface (FMI) Module

The functionality-based design (FbD) aids in the conceptualization of product design to solve the design problem using functionality modeling approach. In this research, product conceptualization will be accomplished through a functionality-driven approach.

The functionality-modeling interface (*FMI*) in an FbD system takes design needs and converts them into functionality-structured representation and constraint. The accomplishment of this task provides a structured functional data that makes it possible to use computer tools to

reason and perform the following two key design operations central to functionality-driven design. These tasks are (1) functionality model generation operation, and (2) functionality-based constraint verification. To enable the *FMI* accomplish these tasks, three functionality engines are to be developed in this research: *functionality definition engine*, *functionality representation engine*, and *functionality constraint engine*.

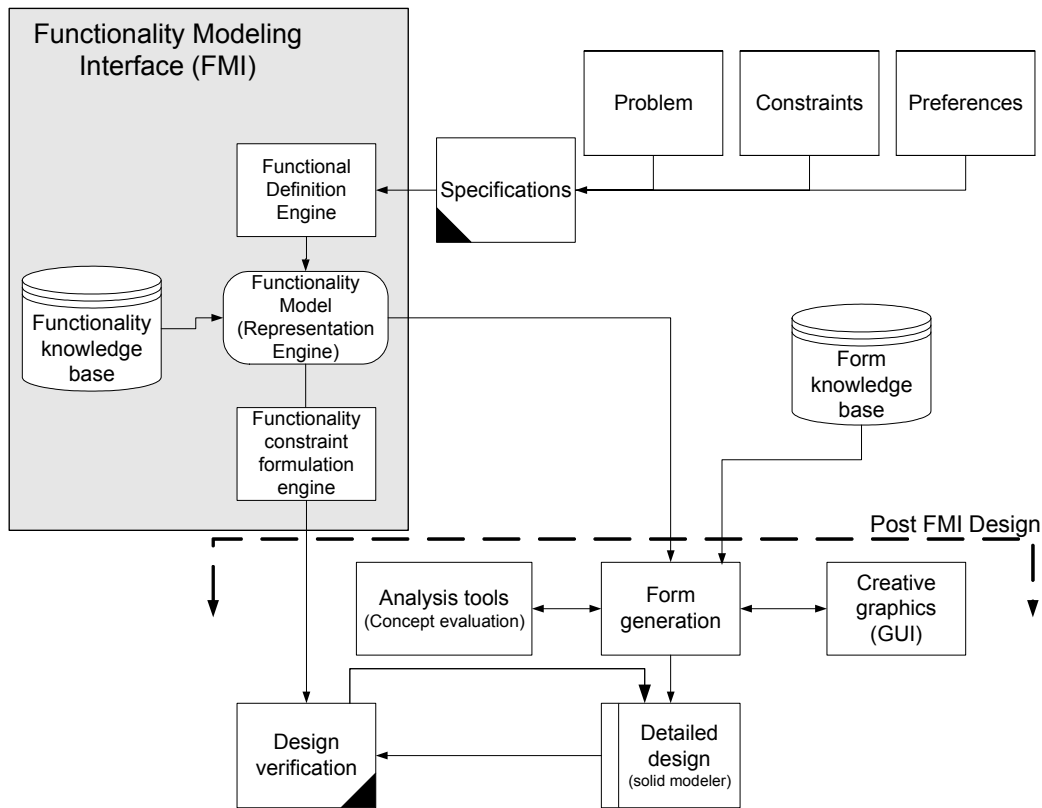


Figure 2 Functionality information flow diagram for the evolution of design in FbD

The *functionality definition engine* serves as an interface to the designer. It accepts the definition of the user needs in a human friendly manner and transforms them into a function-oriented problem definition.

The *functionality representation engine* converts the functionality-oriented problem definition into a functionality-based data structure as described in Section 3. This representation serves as our design objective and is a standard against which all design proposals will be evaluated. In a supply chain management structure, such functionality model may now serve as a design order for suppliers to compete on. For an in-house design operation, such model may now be used in the conceptualization of design parameters (DP) to satisfy the functionality constraints.

The *functionality constraint engine* extracts functionality constraints from the functionality representation. This constraint is used to evaluate design proposals to ensure that they satisfy the original functional requirement. It is also possible to extend this scheme to be used for decision-making in selecting designs of competing proposals from bidders.

Post FMI Design Stage

The form generator is used as an aid to the design by providing tools that will enhance the creativity of the designer. With the use of knowledge base tools, the system could use the functionality model together with the vast expert knowledge in its knowledge base system [*] to infer the possible forms the artifact can assume. The concepts generated by this module are sent to the detailed design module. The form generator is part of the future extension of this work.

In the detailed design module, the concepts from the form generator are developed into a detailed design, with all the design parameters fully specified in this module. Hence, this

* The implementation of a supporting knowledge base for functionality-based design is not the focus of this research. Hence, it is not implemented in this work. However, the implementation of the design repository discussed in Section 5.3 could form the basis for the functionality-based knowledge system. Each industry could also customize its design repository to include domain-specific product functionality information.

module completes the design process by providing a detailed description of an artifact that can satisfy the functionality proposed in functionality builder module. A verification engine within the detailed design module is used to verify that the detailed design satisfy the original function. This module provides an interface to analysis tools, which are used to evaluate the detailed design to verify if they satisfy the functionality constraints. This technique can aid in the evaluation of goodness of a design. The functions of the detailed design and verification have traditionally been provided by commercial CAD systems. An integration of the functionality model developed in this research to existing CAD systems will significantly enhance design process. This research will provide an XML data structure to support such future integration.

1.4.2 Functionality Modeling Process

This is a very crucial part of this research. Two issues are critical to the success of functionality-based design: *functionality definition*, *functionality constraint extraction* and *functionality propagation*. The operation of the functionality modeling process will be illustrated using the design of a transmission mechanism (shown in Figure 3(a)) as an example.

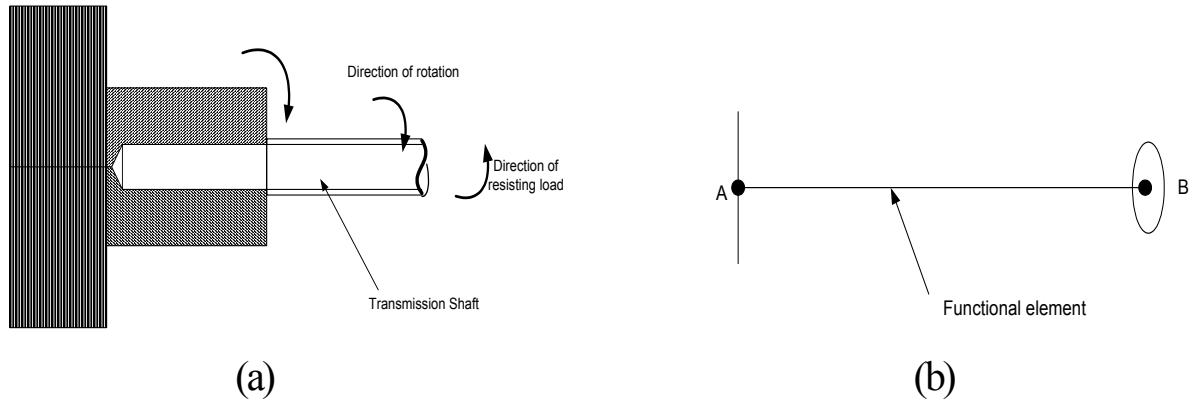


Figure 3 Sample transmission design

Definition: The functionality definition module is responsible for accepting designer's specifications (problem definition) and transforming them into a functionality model compatible format. For example, in the case of shaft design, it will typically receive information stating that a means of transmitting rotational motion is needed. Once the primary function of transmission is selected, the knowledge base ^[x] that supports transmission will be invoked and instantiated. At this point, functional relationships are also defined. These relationships will prescribe the kind of motion, degrees of freedom, expected forces on the product, and other interacting entities (components). In addition, any unique feature or designer's preferences required by the design is also defined at this stage.

Constraint Extraction: Having defined the functionality structure all the implied functionality constraints from the functionality model are extracted. For example, in the shaft design, some of the implied constraints are: material constraints (solid, elastic limit, and tensile

^x The implementation of a supporting knowledge base for functionality-based design is not the focus of this research. Hence, it is not implemented in this work. However, the implementation of the design repository discussed in Section 5.3 could form the basis for the functionality-based knowledge system. Each industry could also customize its design repository to include domain-specific product functionality information.

strength), degree of freedom (DoF), clearance, tolerance, motion trajectory, and maximum length. These are the constraints that any design proposal must satisfy in order to be functionally acceptable. The advantage of this technique is that these constraints are made available from the start of the design process.

Propagation The functionality information from the functionality model is made available in such a way that: (1) They may be used to retrieve acceptable design structures from the knowledge base during the form generation stage by presenting a template for matching functions to stored knowledge; (2) They may also be used in the transformation of functionality representation to a *creative graphic* (wireframe) representation to aid the creativity of the designer (this concept is illustrated in Figure 3(b) for the shaft example); and (3) The functionality information in the form of constraints are integrated with evolving CAD model and propagated to and from the detailed design phase.

The above research will culminate in the development of an object-oriented data structure for the representation and propagation of functionality as design constraints. This data structure (and corresponding model) will serve as an interface during the functional requirement definition phase. It will be used to capture the designer's intent as functionality constraint during the conceptual design phase. The definition of functionality as constraint will make it possible to evaluate and enforce functionality as *design constraint* just as it is currently being done for other product issues such as manufacture, assembly, and geometry reported in most design literatures.^[11]

1.5 Research Organization

Three distinct phases are identifiable in the sequence of operations involved in product design (see Figure 4). During the requirement specification phase, the general information on the desired product is specified, albeit in imprecise manner (because of the limited knowledge of the product). In phase 2, the conceptual design phase, the product that will meet the needs specified in phase 1 is conceived. Phase 3, the detailed design phase, defines the detailed geometric features, tolerances, and material properties of the conceived product.

This research focuses on developing a practical modeling technique for product functionality and its use as a tool in the design of mechanically engineered products. The modeling of product function will enable us to define it as a design constraint in *computer-aided conceptual design* (CACD). Consequently, we will come up with generic representation scheme for functionality that will enable the propagation and enforcement of functionality constraint throughout the product design and possibly manufacturing (through CAM) phases of a product lifecycle.

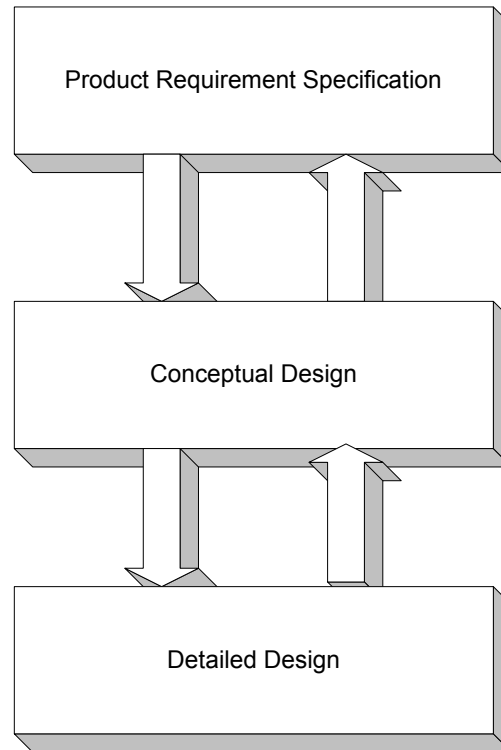


Figure 4 Product design phases

A computer implementation of the functionality model is developed to test and validate the principles developed in this research. This tool is responsible for accepting functional specifications from the user and then translating those functional specifications to functionality design constraint that are applied throughout the remainder of the design process. It is also responsible for the transformation from *functional specification* to *functionality model*, as depicted in Figure 5.

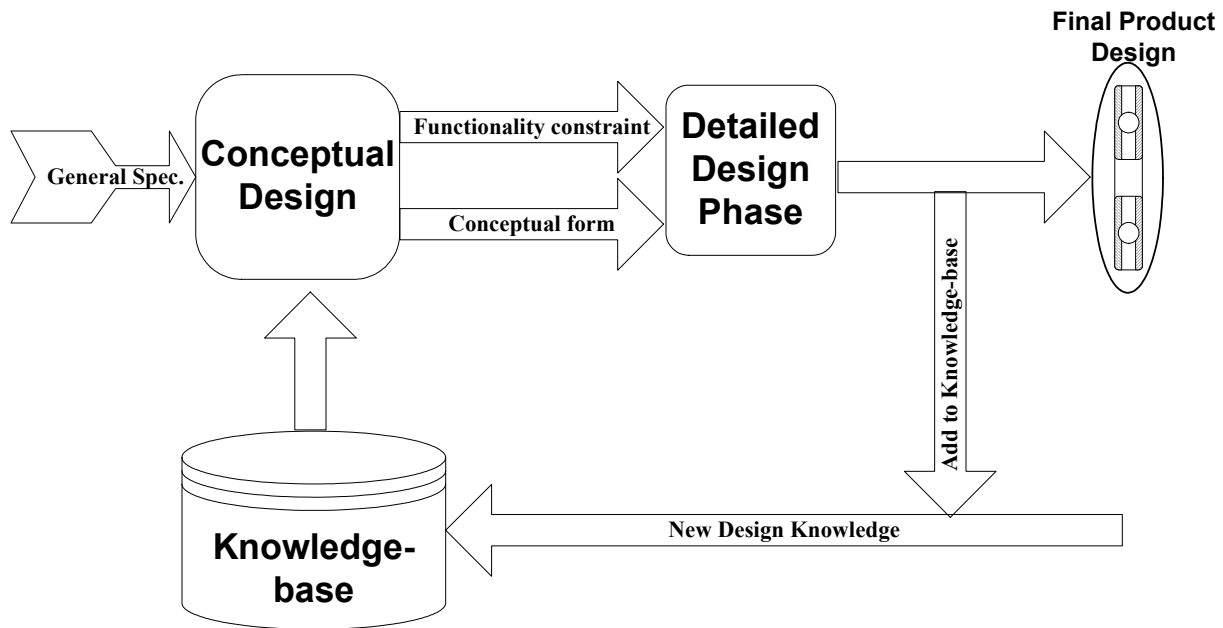


Figure 5 Conceptual design unit interactions with the rest of the designer system

The functionality constraints together with the initial product form (conceptual form) are generated as the output of the conceptual design phase (see Figure 5). In the detailed design phase, the system merely varies the design parameters generated during the conceptual design phase to come up with an optimal or near optimal design. This subsequent variation ensures that other design issues such as manufacturing and assembly requirements are taken into consideration. The conceptual design phase has an input from the system knowledge base,^[+] which represents previous design experiences and some expert knowledge built into the system. This knowledge base could be domain specific. The presence of a knowledge base simplifies

⁺ The implementation of a supporting knowledge base for functionality-based design is not the focus of this research. Hence, it is not implemented in this work. However, the implementation of the design repository discussed in Section 5.3 could form the basis for the functionality-based knowledge system. Each industry could also customize its design repository to include domain-specific product functionality information.

the conceptual design process by tapping from previous design experience and expert knowledge.

This thesis is composed of seven chapters. Chapter 2 looks at the previous work in functional modeling and constraint-based modeling and provides some technical background on relevant concepts related to this research. Chapter 3 is on functionality modeling methodology implemented in this work. It provides an overview of the modeling approach developed in this work. It introduces the concepts of functionality operation, operands, operators, and relationships. Representation of the various components of functionality is also discussed. Chapter 4 focuses on the application of functionality operands in the functionality models. Chapter 5 is on functionality relations. It covers functionality operators, functionality states, the use of coupling bonds, and the functionality design repository. Chapter 6 is on the computer implementation, testing, and validation of the functionality model. Data structures of the functionality model are also discussed. Chapter 7 is on the conclusion and future extensions of this research.

2.0 TECHNICAL BACKGROUND

2.1 Design of Mechanically Engineered Products

Engineering design is very important because it determines the ultimate outcome of engineering activities, including manufacturing of goods, improvement in quality of life, and provision of defense needs. The design of mechanically engineered products is a very challenging task because of the nature of the products and the design process. A given design artifact (product of design) may perform different functions depending on the context of application. Thus, product function is very subjective, depending largely on the context of application (i.e. design intent). This unique characteristic has made it very challenging to develop a procedure that directly relates design to functionality. The lack of a generally accepted systematic procedure for the design of mechanically engineered products has further complicated the existing problem. This often leads to cases where a single design specification may result in several designs for the same product specification. This in itself is not a problem, as it introduces variety in the product selection. However, the problem is that there is no scientific or systematic procedure for the evaluation of such design proposals. Thus, the evaluation process again is very subjective, and the outcome depends on personal experience and preferences of the design evaluators. For example, a company that solicits for design proposals from its supplier's has no sound engineering or scientific procedure for selecting the best design that meets its engineering goals.

The design process can be thought as the detailing of shape as the designer's idea evolves.^[12] Thus, CAD software as a design aid is just a tool to facilitate this detailing process.

Although some research has been done to support product conceptualization, ^[13, 14] existing methods for supporting the geometric aspects of design have little impact at the conceptual design stage for the following reasons: ^[15]

- CAD systems have concentrated on the capture and representation of geometric shape, as opposed to providing support for conception;
- Systems which attempt to provide conceptual design support are based on little or no relations to function; and
- CAD systems require a detail of representation, which is too restrictive for conceptual design.

Two major challenges of Computer-aided Conceptual Design (CACD) are: ^[15] (i) how to allow a concept representations evolve as detail accumulates in the design process; (ii) even if it were possible to commit early on to a certain abstract shape, complete information needed to display the object may be absent.

The use of CAD/CAM tools for design work has significantly improved designer's productivity and the quality of designs. ^[16] The full potential of computers in design has not been fully utilized in conceptual design process, because most of the available CAD tools focus on the downstream activity of detailed design or assembly design. Detailed design is carried out after the design concept and geometry are well established. CAD/CAM tools help the designer in drafting work, FEM analysis, NC cutter path analysis, etc. However, the key design activity is the conceptual design stage, where the designer works with the functional requirements of a part. The functional requirements need to be decomposed, and at a certain stage, the functionality has to be mapped to the geometry or form. The designer may update the functional decomposition, update the mapped geometry, and proceed through iterations until the concept is well defined.

The above problem is associated with the creative nature^[17] of design (involving diverse problem-solving techniques and many kinds of knowledge) and hence subjective to the designer's previous experience and preferences. Hence, the more experienced a designer is, the more likely he/she is to find an acceptable design solution within a short period. The questions that interest us are: Can the creativity of the designer be improved with modern computer tools? Can the vast experience and information accumulated in over a century of design be captured, stored, and made available to the designer using computer tools? Design decisions made at the initial or upstream stage of engineering affect all subsequent outcomes. In the following sections, basic design procedures relevant to this research are presented. Some of the techniques to be used in this research work are also described.

2.1.1 Design Process

Design has been defined as the process of constructing a description of an artifact that satisfies a functional specification, meets certain performance criteria and resource limitations, is realizable in a given target technology, and satisfies criteria such as testability, manufacturability, reusability, etc.^[8] This definition clearly highlights the key features of design as:

- It must satisfy the functional specification. That is it must be able to accomplish the need for which it was designed. This definition, however, presupposes the existence of a well-defined functional specification. Existing CAD technologies are ill equipped to accept product functionality as part of the design specification. In fact, the inputs of

most CAD systems are well-conceived artifacts, albeit in designers' brain, with the CAD only acting as an information recorder.

- The design output is a description of the mechanical artifact that may now be converted to a physical artifact through a manufacturing operation.
- Existence of design constraints such as resource limitations, and other technological and social constraints during a design process. For example, any conceived artifact must be subject to acceptable physical laws such as Newton's law, $F = ma$.

The design process may be classified into three categories: ^[18] generative, adaptive, or variant design. This is illustrated in Figure 6. In *generative design*, new design tasks and problems are realized by original design incorporating new solution principles. These can be realized either by selecting and combining known principles and technology, or by inventing completely new technology. Generative designs usually proceed through all design phases; depend on physical and process fundamentals; and require a careful technical and economic analysis of the task. In *adaptive design* one keeps to known and established solution principles and adapts the embodiment to changing requirements. It may be necessary to undertake original designs of individual assemblies or components. In this type of design the emphasis is on geometrical, analytical (strength, stiffness etc), production and material issues. In *variant design*, the sizes and arrangements of parts and assemblies are varied within the limits set by previously designed product structures (e.g. size ranges and modular products). Variant design requires original design effort only once. It includes designs in which only the dimensions of individual parts are changed to meet a specific order.

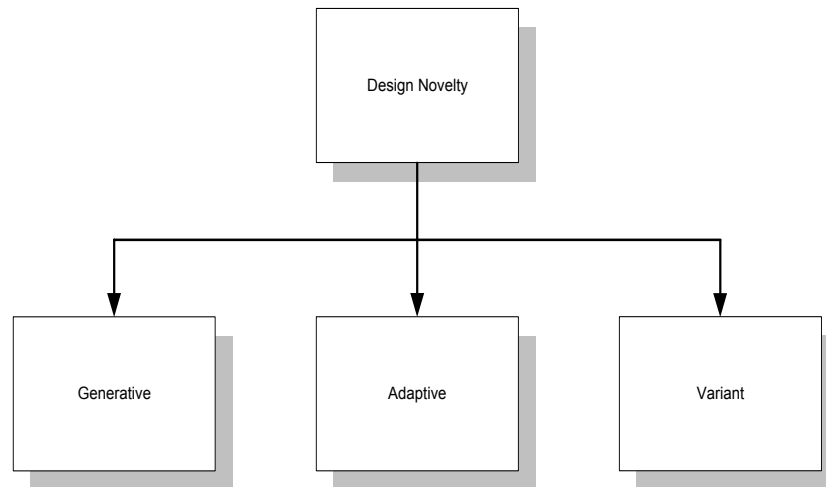


Figure 6 Classification of design

Design involves four distinct aspects of engineering and scientific endeavor: the *problem definition* from a “fuzzy” array of facts and myths into a coherent statement of the question; the *creative process* (concept generation) of devising a proposed physical embodiment of solutions; the *analytical process* (concept evaluation) of determining whether the proposed solution is correct or rational; and the *ultimate check* (design verification) of the fidelity of the design product to the original perceived needs. The sequence of events involved in the design of a product is illustrated in Figure 7.

It is sometimes difficult to judge whether or not the problem definition does correctly represent the perceived needs until the final output of the design process in the form of products, processes, or systems is compared with the perceived needs. This process is usually carried out through design verification procedures such as physical prototyping or computer techniques such as simulation or virtual prototyping. The major challenge of computer verification techniques is that the functional requirements are usually not clearly defined in such a way to

allow for automated computer verification operation. This research provides a tool to allow product functionality to be modeled in such a way that automated function verification might be enhanced.

The second step of the design process is the creative process of synthesizing a design solution in the form of physical embodiment. The creative ideas and the synthesis process depend on the specific knowledge possessed by the designer, and on his or her ability to integrate knowledge.

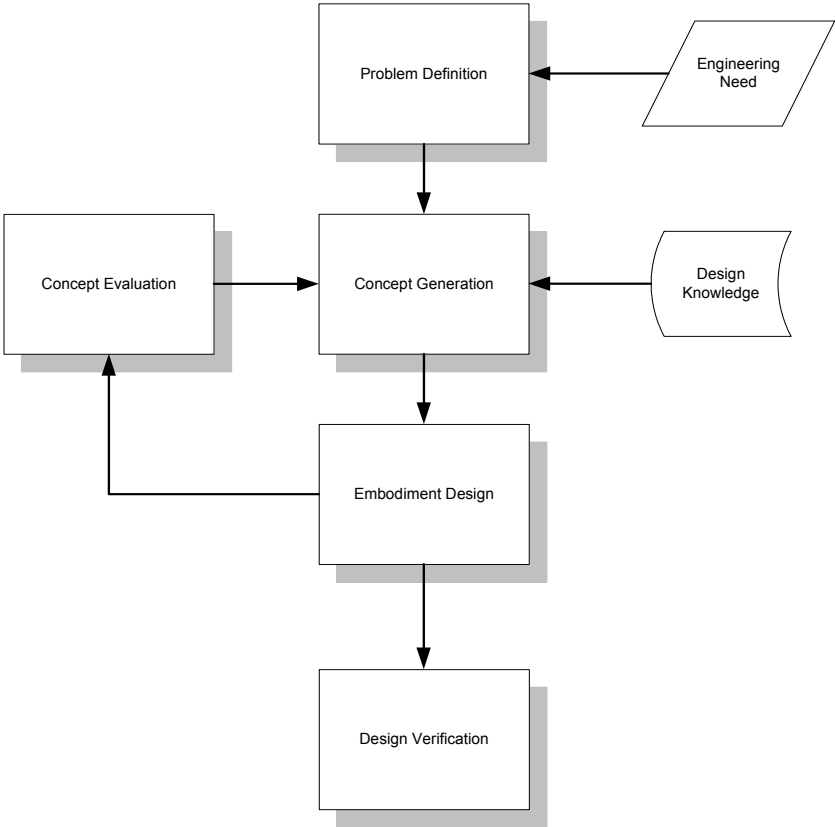


Figure 7 Design process

Design creativity is complemented by analytical process. This aspect is illustrated by Suh ^[62] as shown in Figure 8, which depicts the design process as a feedback control loop. It shows how the creative process must be checked through analysis and corrected for differences between the perceived problem definition and the proposed solution. In the figure, Y is the desired outcome and X the input. The gain of the feedback loop should be as large as possible to converge to a correct solution quickly; that is, the ability to judge the quality of the outcome of the creative process improves the creative process itself. In the figure, the former is depicted by the function H and the latter by G . The relationship between X , Y , H , and G is given by:

$$\frac{Y}{X} = \frac{G}{1+GH} \approx \frac{G}{GH} = H^{-1} \quad \text{for } GH \gg 1$$

Equation 1

When $G \times H$ is much larger than unity, the gain is equal to $1/H$. If we cannot analyze a design solution, then we cannot rapidly generate the “best” design since we cannot distinguish a good design from a bad design. In the absence of a criterion for selecting a good design, we cannot make good design decisions. The “analysis” of design implies making correct decisions as well as evaluating the details of specific features.

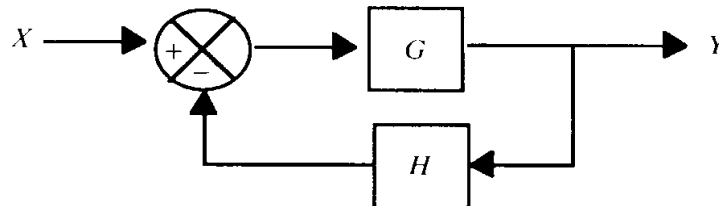


Figure 8 Feedback control loop depicting the design process

Design involves a continuous interplay between *what we want to achieve* and *how we want to achieve it*. The objective of design is stated in the functional domain, whereas the physical solution is generated in the physical domain. The design process involves relating the *functional* requirements (FR) of the functional domain to the *design* parameters (DP) of the physical domain. This is illustrated in Figure 9 (adapted from Suh ^[62]), where DPs in the physical domain are chosen to satisfy FRs in the functional domain.

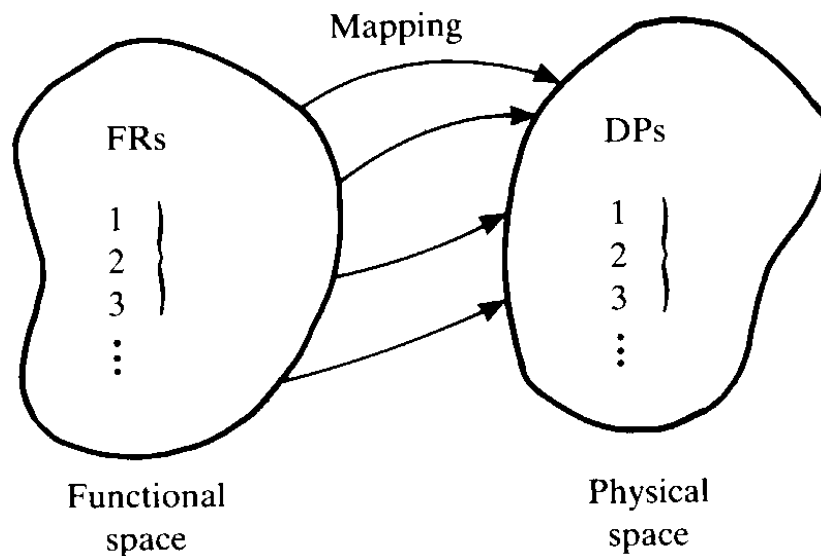


Figure 9 Design as the mapping from functional space to physical space

Hence, design may be described as the creation of synthesized solutions in the form of products, processes or systems that satisfy perceived needs through the mapping between the FRs in the functional domain and the DPs of the physical domain, through the proper selection of DPs that satisfy FRs. This mapping process is nonunique; therefore, more than one design

may ensue from the generation of the DPs that satisfy the FRs; in other words, the actual outcome depends on a designer's individual creative process.

The design process begins with the establishment of FRs in the functional domain to satisfy a given set of needs, and ends with the creation of an entity that satisfies these FRs. The determination of a good set of FRs from diffused and often poorly defined perceived needs is an important step in the design process. The final design cannot be better than the set of FRs that it was created to satisfy. A problem may be ill conceived, resulting in the formulation of a wrong set of functional requirements and consequently, a wrong set of design parameters. Hence, it is very important that FRs are clearly and concisely defined to represent the correct design problem or need. One of the objectives of this research is to come up with a methodology for the description of FRs (through functionality modeling) that will facilitate the use of computers in the mapping process of FRs to DPs.

In generative product design (evolution), the need for a product design arises because of identification of a need to be accomplished by the product. In variant (or adaptive) design, however, the need for new design or re-design process arises because of the identification of a modification or improvement that need to be made to the product. In this work, discussion on design evolution is largely focused on generative designs. To perform variant design however, one only need to identify the functional elements associated with the existing design and then performs the functionality modeling for the product. The integration of the functionality-modeling tool to existing CAD systems will greatly enhance the ease with which one can perform variant or adaptive designs.

2.1.2 Feature-Based Design

Feature-based design is a technique that permits a designer to express geometric design intent while creating the geometry of the product. It requires and permits the designer to think beyond mere shape and to state explicitly what portions of part are important and why. There is no universally accepted definition for features. Various researchers have defined feature differently. Some have given general definitions like “*a subset of geometry on an engineering part which has a special design or manufacturing characteristic*”,^[19] “*generic shapes with which engineers associate certain properties or attributes and knowledge useful in reasoning about the product*”,^[20] or “*a set of geometric entities (surfaces, edges and vertices) together with specifications of the bounding relationship between them and which imply an engineering function on an object*”.^[21] Using feature concepts in the product model offers several convenient properties as follows:^[22]

- Recurring characteristics of products can be modeled as feature types, and used as a repository of reusable product knowledge that may be related to a particular shape or geometric pattern.
- Specific products can be modeled through their constituent features, providing a more natural basis of interaction with the designer than mere geometric models.
- Manufacturing knowledge can be associated with features, and accessed to determine the producibility of a designed object or for planning its actual manufacture.

In design by features, the part is constructed directly from pre-defined features.^[22] The part is represented by the features and the relationships among them. Parts are designed by

adding, manipulating and subtracting generic features stored in a database and the user (i.e. the part designer) is provided with a library of features to work with. The user may be allowed to modify existing features in the library or create new features and add it to the library. A major advantage of this approach is its ability to capture and propagate the designer's intentions for use by downstream applications. The functionality model developed in this work offers the advantage of extending current feature modeling techniques by associating each feature with specific engineering functionalities (which may be context dependent).

In feature extraction on the other hand, the part is constructed using geometric entities (lines, curves etc.) on conventional CAD geometric modelers. The features are then identified, i.e. recognized from the geometric model using some geometric reasoning techniques. The main attraction of this approach is design freedom and the possibility of using current CAD modelers.

2.2 Functionality-based Design

2.2.1 Functionality Modeling and Verification

There is no clear and uniform definition of function. Functions are intuitive concepts depending on intentions of designers or users. Rodenacker^[23] defined function as a relationship between input and output of energy, material, and information and this representation is widely accepted in design research. This definition is limited in application e.g. when there is no energy flow involved. An alternative definition is provided by Tomiyada et al.;^[24] function is “a description of behavior abstracted by human through recognition of the behavior in order to utilize the behavior.”

A list of classifications of technical artifacts based on criteria such as function, working means, complexity, production, and product was drawn up by Hubka.^[25] It is, however, difficult

to agree on a generally acceptable system of classification – the tasks, applications and forms are much too varied and complex. Hubka suggested that technical systems should be treated as systems connected to the environment by means of inputs and outputs. A system can be divided into sub-systems and what belongs to a particular system is determined by the system boundary. The inputs and outputs cross the system boundary. With this approach, it is possible to define appropriate systems at every stage of abstraction, analysis or classification. While this description may be appropriate for general description and to some extent functional description of a system, a designer obviously will need a working knowledge of the systems internals in order to come up with a good design concept.

Many researchers have approached the functionality representation problem by providing taxonomy of possible product functionality.^[26] Grabowski et al.^[27] divided product function model into three layered function models with different abstractions: the logical model, the status model and the relation model. Their functional classification is based of the function taxonomy provided by Pahl.^[18] The *logical model* is used to represent a high level topology and connectivity of sub functions, with the aid of Boolean operators: AND, OR, and NOT. The *status model* assumes every sub-function / module has some working states, which are numeric coded in the status model. Unlike the logic model, this model describes the relation of more than one sub functions. *Relation model* defines the mathematical or physical relations between several physical variables including relations between input and output within a component. The Grabowski's group claim that each function model can completely describe a mechatronic product with its related abstraction level, and can fluently be converted into another model. This claim is very restrictive as each model only represents a particular view of the product. Instead of this conversion attempt, the models could play a more complementary role to one another.

Another limitation of this function classification is that they are defined at an abstract level, in such a way that it is not easy to apply in basic design activities that require more concrete special functions, which vary with application tasks and are usually realized in form of components.

Some researchers have tried to map functionality to form through the use of *function-form matrix* scheme ^[16, 27]. A data structure of function-form matrices was employed by Mukherjee et al. ^[16] to provide a relationship between a functional decomposition and the sketching abstraction of the parts. They carried out this process using a hierarchical decomposition of functions into sub-functions through the process of functional representation to the point of initiating the physical design of the part. Their representation takes into account the fact that form-function relationships are usually many-many in nature. There are however, some serious limitations in this approach because of lack of suitable functionality representation and reasoning schemes.

The use of graphs in the description of relationships among interacting entities has been used in the field of CAD and artificial intelligence. Al-Hakim et al. ^[28] used graph theory to represent a product and define the relationships between its components. They employed the graph-theory concepts of the “tree” and the “forest” to represent a functional design artifact and idle condition, respectively. In their approach, components of a product being designed are represented as vertices of a graph while the edges of a graph represent the relationship between the components (vertices). The number of vertices of a graph representing a product in an idle status may be different from the graph representing the same product during operation. They used this approach to consider expected mechanical failures and other constraints at the conceptual design stage resulting from the flow of energy between the components of a product.

It is however difficult to extend this approach to model generic product functionalities at both the conceptual and detailed design phases of design.

Object-oriented principles of encapsulation and inheritance have been used by Gorti and Sriram ^[15] to localize knowledge representation of product design and process. They described a symbolic evolution approach, symbol-form mapping. In geometry representation, they used an object-oriented paradigm to provide a solution. They also proposed a structure information model and primitives for an integrated representation of the design product and process. ^[29] Their symbolic evolution approach has two main components: a structured model that defines some primitive constructs and their interaction; and a reasoning approach that operates on the representation. The primary objects are context and artifact. A *context* consists of the design tasks (goal), the user specifications, the decisions that have been made, and the artifacts that are created as part of the design process. The *artifact* is comprised of *function*, *form*, and *behavior*. Design relationships provide the overall functional and spatial coherence for the design. They were concerned primarily with four broad classes of relationships: functional relationships, composition, aggregation, and spatial relationships. The basic object model of the symbolic evolution is required to provide for explicit representation of these relationships. Gorti's symbol-form mapping framework relies heavily on explicit representation of relationships. This implies that the utility of their approach is limited to that class of design, which deals with assembled systems, where disaggregated knowledge is coordinated together with a common process. In general, the current implementation of the symbol-form mapping framework is geared towards conceptual design. The current approach to dealing with iteration in design is somewhat limited. The extent of innovativeness the framework permits is largely limited by the representation of behavior in the system.

Hierarchical decomposition approach has been employed in the modeling of product design functionality. [30, 16] Cole [30] employed a hierarchical decomposition of the primary system functions into sub-functions, at ever-increasing levels of detail. They 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 functions. Mukherjee et al. [16] in their work noted that the identification of a set of functions for the part is the first step in part design. They also carried out this process using a hierarchical decomposition of functions into sub-functions through the process of functional representation to the point of initiating the physical design of the part. While the functional decomposition process follows a sequence of higher level functions being decomposed to intermediate and lower level sub-functions, their generic representation of a function is given as follows:

$$[<\mathbf{v}><\mathbf{n}><\mathbf{m}><\mathbf{d}><\mathbf{o}>]_{locn} <\mathbf{keywords}>$$

where,

v, n, m, d, and o are sets and

v represents verbs,

n represents nouns,

m represents a magnitude attribute such as 10 N of force,

d represents a direction attribute, and

o is another set of nouns representing objects on which the function applies.

locn distinguishes between more than one similar functions,

keywords is an additional set of specialized words used to enhance the functional representation.

These models, however, are very restrictive and do not provide a flexible and practical way to model functionality relationships and constraints that support implementations and propagation to downstream design activities.

Tomiyama et al. ^[24] proposed the use of function-behavior-state (FBS) modeling to deal with function in conceptual design. They represented a function by two concepts; i.e. its symbol represented in the form of to do something and its semantics represented by the relationship between function and behavior (which they called F-B relationship). In their work, Tomiyama's group assumed that the representation of function includes human intention, whereas, the representation of behavior of an entity can be determined objectively by its relations to other entities based on physical principles. They called attributes and relations of an entity a state of an entity. For representing functions, they construct a knowledge base of functions by collecting prototypes of functions from existing design results. Table 1 shows the scheme of functional prototypes. *Name* of a function is described in the form of "to *verb objectives*." *Decomposition* describes feasible candidates for detailing this function in the form of networks of sub-functions.

Table 1 Definition of a functional prototype (proposed by Tomiyama et al ^[24])

Item	Contents
Name	<i>verb + objectives</i>
Decomposition	<i>networks of sub-functions</i>
F-B Relationship	<i>networks of views</i>

Although there has been a lot of work on design evaluation, for example, the weighted objective method ^[31] and value engineering, ^[32] relatively little work has been carried out on design verification. ^[33] Design evaluation focuses on evaluating different alternatives against

specified criteria, whereas verification involves checking that the design proposal satisfies functional and other specifications. There are two families of methods for design verification. First, attributes of interest can be directly calculated or estimated by means of domain-specific algorithms or formulae. These methods have limited application, as they require a well-defined problem space. Second, verification can be achieved by simulation. Developing a structural model of the device and allowing this to change state from specified initial conditions simulates the behavior of a system or device.

In the industry, several design verification approaches are used. The most common approach is to use CAE software to simulate the behavior of a design and then to manually check this against the desired performance targets. One extension to this technique involves automatic optimization of design variables (within a specified range) for a given working environment, e.g. MSC CAE and iSIGHT software. Some advanced user automated design verification systems that use Product Data Management (PDM) systems include iMAN, and expert systems such as ICAD. The PDM and expert system approaches are domain and problem specific.

Deng et al. ^[33] proposed a functional design verification model based on their previous work ^[34, 35, 36, 37] on *Function-Environment-Behavior-Structure* (FEBS) design model. Design verification is achieved by identifying input and output design variables, developing a variable dependency graph, propagating constraints over the variable dependency graph and checking the values of the design variables against the functional requirements. The propagated constraints are determined by first deriving algebraic expressions for each output variable. The algebraic expressions are then substituted for the output variables in the expression of the constraint-to-be-propagated. This technique results in some constraints not being able to propagate to input

variables, designers are therefore required to either verify them in a manner similar for constraints on input variables or use heuristic methods for constraint solution. Their approach is limited because the behavior must be known or specified by the designer. That is it assumes the existence of a working design concept. In addition, this model is based on the fact that product models are treated as in-output systems. This assumption, however, places some restrictions on systems that cannot be described with this input/output model.

Another major challenge faced by researchers in this field is taxonomy. The National Institute of Standards and Technology (NIST) design repository ^[38, 39, 40, 41] and other research groups ^[18, 26, 27, 42, 43, 44] have tried to approach this problem by developing an extensive library of possible product functions. There are still unresolved issues on how to achieve a computational model of this taxonomy that can serve as an aid in product design. The NIST design repository is based on the description of function using functional basis. The models are based on the use of flows to describe functions as inputs and outputs. This assumption places a restriction on the type of functions that can be modeled with this approach to those systems with identifiable inputs and outputs (thus, assuming a casual relationship between entities). Unfortunately, input-output models cannot be used for functions where no clear casual relationship is identifiable. Moreover, the model developed has no transparent and flexible means of modeling and propagating functionality as constraint on basic design elements to detailed design. This research will address this problem by providing a scheme for mapping the functionality description into mathematical relationships and constraints.

2.2.2 Input-Output Model (Black-box Model)

The *input-output* (or *black-box*) model views mechanical systems as composed of different levels of abstractions with inputs and outputs described in terms of *energy* and *material*. Energy and matter have been described as basic concepts of the input-output model of an engineering artifact. ^[18] This is illustrated in Figure 13. Force is the means by which the motion of an artifact (matter) is changed. Ultimately, this process is explained in terms of energy. Thus, in engineered systems, energy is a fundamental element, the flow or transformation of which, can result in the accomplishment of a functional requirement. Whenever a change or flow is involved, time is introduced as a fundamental quantity. By reference to time, the interplay of energy and matter can adequately describe the state of a system.

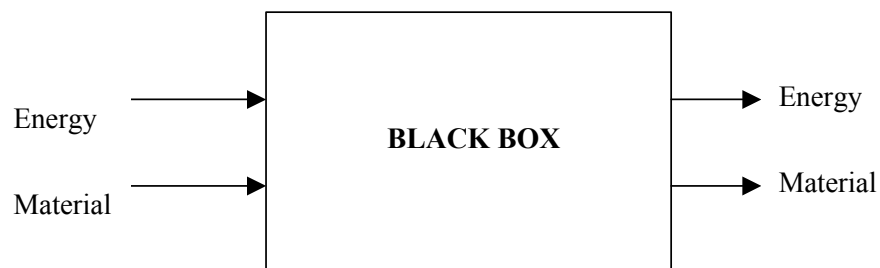


Figure 10 Input-output model of system

In technical sphere, the previous description is usually linked to concrete physical or technical representations. For example, the manifest forms such as mechanical, electrical, optical energy etc is used to describe energy. For matter, it is usual to substitute material with properties as weight, color, condition etc.

Analysis of technical systems makes it clear that all of them involve technical processes in which energy and material are channeled and/or converted. This model considers these conversions as flows, and the prevailing one the main flow. A second type of flow usually accompanies the main flow, and quite frequently all two come into play. Thus, there can be no flow of material without an accompanying flow of energy, however small.

While the input-output model is very useful in understanding and modeling design functionality, it has limitation by considering functional entities as flows and by the assumption of causal relationship. Hence, the input-output model, excludes systems that have no obvious or identifiable input and output flows, with no identifiable motion when in operation. Examples of such systems with no identifiable input / output flows include a computer chassis and an electric power outlet cover. The functionality model proposed in this research extends this basic model of functionality entities to support all classes of mechanically engineered products. The basic entities are not considered only as time dependant elements – flows. Rather, they are considered simply as basic entities involved in the realization of mechanical functions.

2.2.3 Functionality Decomposition

The physical realization of product functionality is hierarchical in nature. As illustrated in Figure 14, product functionality can normally be described with an overall functionality by bringing together separate entities of sub-functionalities and basic components consisting of energy and material. Each sub-functionality accomplishes a specific function that contributes towards the accomplishment of the overall function. If an *overall task* has been adequately defined – that is, if the attributes and required behavior of all the quantities involved and their

actual or required properties are known – then it is possible to specify the *overall function*. An overall function can often be divided into identifiable sub-functions corresponding to sub-tasks. The relationship between sub-functions and overall function is very often governed by certain constraints such as functional and physical relationships.

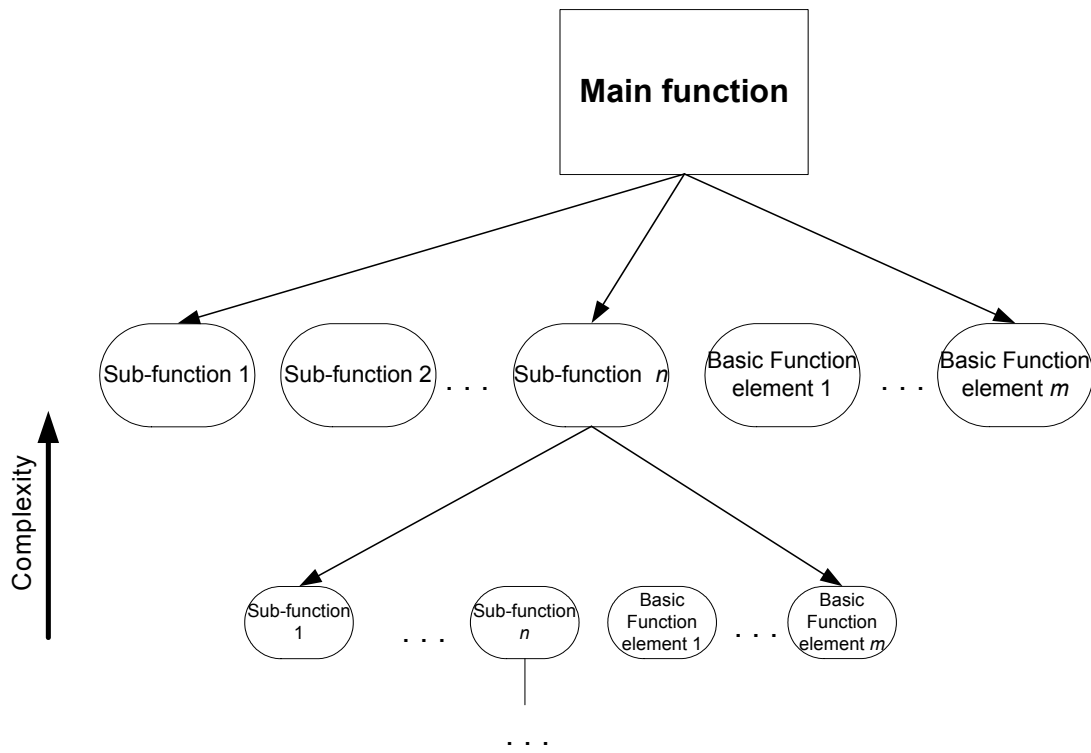


Figure 11 Product function decomposition

The meaningful and compatible combination of sub-functions into an overall function produces a *function structure*, which may be varied to satisfy the overall function. To that end, it is useful to make a block diagram in which the processes and sub-systems inside a given block (black box) are at first ignored.

Functions are usually defined by statements consisting of a verb and a noun, for example “increase pressure”, “transfer torque” and “reduce speed”. They are derived from the interaction of energy and material. It is useful to distinguish between main and auxiliary functions. While *main functions* are those sub-functions that serve the overall objective directly, *auxiliary functions* are those that contribute to it indirectly. They have a supportive or complementary character and are often determined by the nature of the solution.

A logical analysis of functional relationship starts with the search for the essential ones that must appear in a system if the overall problem is to be solved. There may be some conditional elements in sub-functional satisfaction (if-then condition). Logical relationships must also be established between the entities of a particular function / sub-function. In most cases, there are several entities whose relationships can be treated like propositions in binary logic. For example, a simple AND-function may be illustrated by a clutch system; a trigger force must be sent and clutch engaged before the torque can be transmitted.

2.2.4 Functionality-based Product Design Conceptualization

Conceptual design involves the process of obtaining a pool of feasible concepts from which the most promising one is selected. This is the process of creation, the most difficult and least understood step in the design process. ^[45] So far, no existing method is available to guide designers directly and precisely to invent devices, products, systems, or processes.

An approach for mapping an evolving symbolic description of design into a geometric description was developed by Gorti and Sriram. ^[15] They identified symbol-form mapping as the crucial issue to be addressed in developing *Computer-Aided Conceptual Design* (CACD)

framework. That is, the process by which a logical description of an engineered system is mapped into physical description during the conceptual design phase. They argued that the design process has two aspects – a symbol aspect, which leads to a logical product description, and a geometric aspect, which leads to a physical description. The distinct elements of the symbol-form mapping are: (a) deriving spatial relationships between objects as a consequence of the functional relationships; (b) instantiating alternative feasible solutions subject to these relationships; and (c) presenting the evolving descriptions of geometry. They called this part of the design a function-symbol mapping. At this stage, the specific functional requirements coupled with the design context leads to: (a) a selection of components; (b) establishment of functional relationships between components; and (c) establishing important parametric constraints. In order to physically realize the design, they noted that the logical description must somehow be mapped into a geometric description. This mapping is often iterative – the geometry may lead to further refining the logical description. Another stress is the importance of localized knowledge in relationship mapping. They focused on developing a domain-independent methodology to support form conception, by localizing domain-dependent aspects of the symbol-form mapping.

A computer tool called *function-behavior-state* (FBS) modeler was proposed by Umeda's group^[46] to support conceptual design. To accomplish this, they used the functional decomposition and physical features in knowledge base of the modeler. The key objectives of the FBS are:

- To distinguish subjective parts of a design object (*function symbols and F-B relationships*) and the objective parts (*behaviors and states*);

- To represent a function as an association of subjective concepts (*function symbols*) and objective concepts (*behaviors*) rather than just either of them; in other words, function relates subjective concepts and objective concepts; and
- To represent a design object hierarchically to support a modeling process that details functional and behavioral descriptions concurrently.

Wang and Nnaji ^[47] proposed the use of “modus” to model product functionality and form conceptualization. They defined modus as the basic design operation unit that embodies functionality instead of feature or basic geometric elements such as lines, circles, etc. This approach has a great limitation in modeling product functionality as a constraint and propagation of functionality to downstream design activities.

Mukherjee et al. ^[16] proposed the use of representation methodology, *sketching abstraction*, as the schema for a CAD tool to support conceptual design of parts. The principle of the sketching abstraction is to use *wireframe-based data structure* to represent the functionally essential parts of the geometry, using features and workpiece face information. The remaining geometry is abstracted using a linkage system. The sketching abstraction is annotated with a set of primitives. They developed a grammar to extract canonical pairwise relationships between the functionally essential parts of the geometry. These relationships form the functional signature of the part. Sketching abstractions have a close relationship with the part functionality. They developed a group technology (GT) based coding schema for representing functions and to enhance the retrieval of parts for the part library. Through the process of decomposition, the designer can select the most appropriate set of features from a library of features. The geometry is related to the functionality using data structures called function-form

matrices. The remaining geometry of the part is abstracted using a set of linkages to create the sketching abstraction.

Establishing a function structure facilitates the discovery of solutions because it simplifies the general search for them and because solutions to sub-functions can be elaborated separately. A graphical display of this decomposition may also stimulate the creativity of the designer. Most mechanical product functions are realized by selection of appropriate physical processes. Hence, in conceptual design, physical processes and working principles are selected to accomplish mechanical functions. A physical process (realized by the selected *physical effects* and the determined *geometric and material characteristics*) results in a *working relationship* that fulfils the function in accordance with the task.

Physical Effects

Physical effects are described quantitatively by means of the physical laws governing the physical quantities involved. For example, in Figure 12, the earth's gravitational force (F_g) on an object of mass (m) is described by: $F_g = m \cdot g$, where g is the earth's gravitational force constant. Several physical effects may have to be combined in order to fulfill a sub-function. For example, the operation of a bi-metallic strip is the result of a combination of two effects, namely thermal expansion and elasticity. A physical effect chosen for a particular sub-function, however, must be compatible with the physical effects of other, related sub-functions.

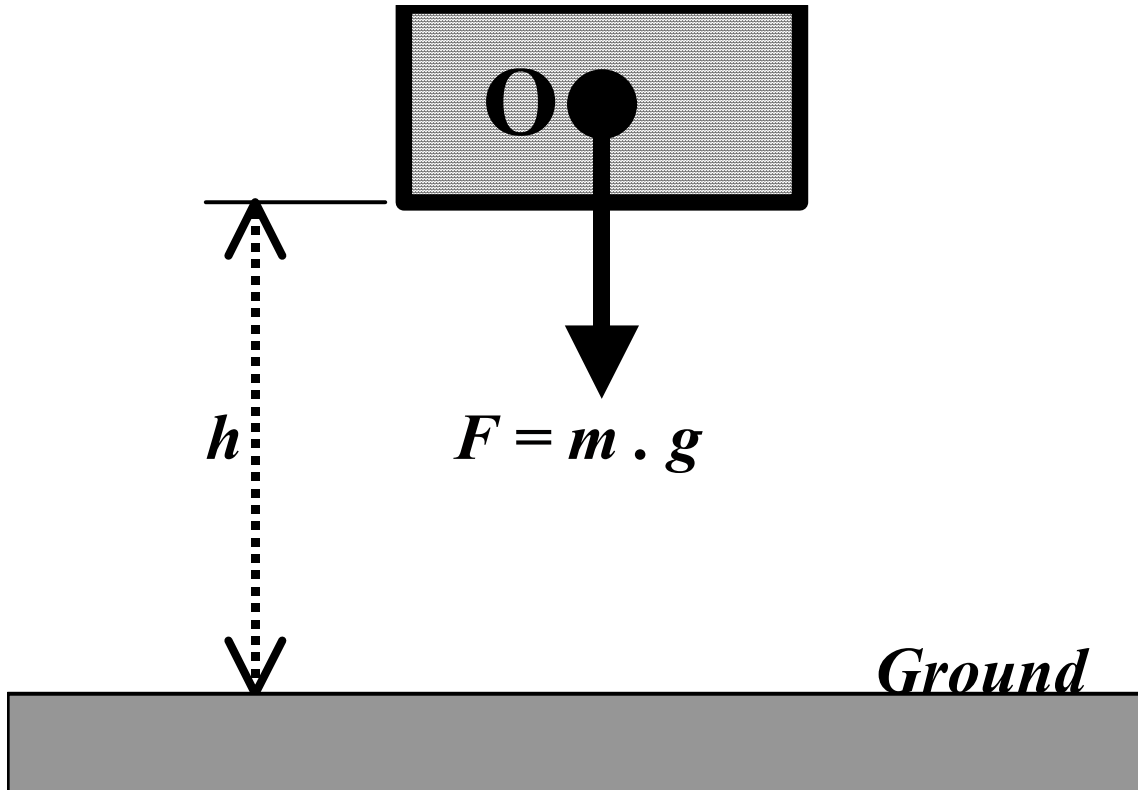


Figure 12 Physical effect: earth's gravitational pull on object, O

Geometric and Material Characteristics

The place where the physical process actually takes effect is the *working location* (active location). A function is fulfilled by the physical effect, which is realized by the *working geometry*, i.e. the arrangement of *working surfaces* (or working spaces) and by the choice of *working motions*. In addition, we need a general idea of the type of material with which the working surfaces are to be produced, for example, whether it is solid, liquid or gaseous; rigid or flexible; elastic or plastic. A general idea of the final embodiment is often insufficient; the main material properties must be specified before the functional interrelationship can be formulated adequately.

Only the combination of the physical effect with functional entities (geometry and material characteristics) allows the principle of the solution to emerge. This interrelationship is called the working principle, and it is the first concrete step in the implementation of the solution - conceptualization. The combination of several working principles results in the working structure of a solution. It is through this combination of working principles that the solution principle for fulfilling the overall task can be recognized. The working structure derived from the function structure thus represents how the solution will work at the fundamental principle level.

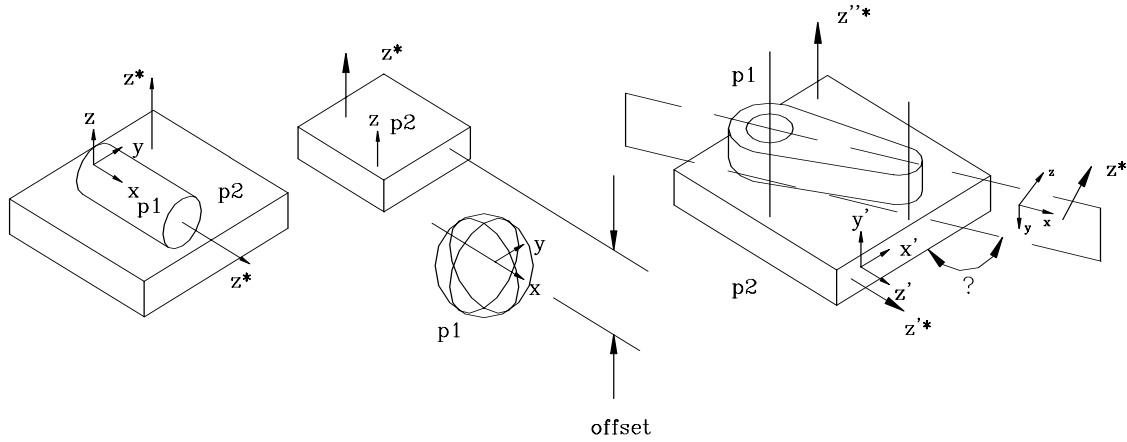
2.3 Spatial Relationship in Product Design

Spatial relationships were proposed by Popplestone et al. ^[48,49,50,51] to express the relative positions of parts in a product's final state by specifying spatial relationships between features. A mathematical model is applied to transform a system of spatial relationships into equations, which are solved to determine the location of components in a product. Liu and Nnaji expanded the concept, and not only inferred the assembly position of parts but also captured designers' intent. ^[52, 53] There are six major types of spatial relationships (illustrated in Figure 10) defined as follows:

- *Against*: the mating surfaces touch at some point. The *against* relationship is the most basic spatial relationships and applies to any parts assembly. Any combination of two features can possess this property.

- *Parallel-offset*: the parallel relation holds between planar faces, cylindrical and spherical features. In the case of two parallel planar faces, the outward normals are pointing in the same direction. This relationship exists without physical contact of two features with an offset distance.
- *Parax-offset*: this relationship is similar to *parallel-offset* but the outward normals of the parallel planar faces are in opposite directions.
- *Aligned*: two features are aligned if their centerlines 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 two faces which makes the include angle.
- *Include-angle*: this is similar to *incline-offset*, but having an include angle between two planar faces in their positive normal direction. The rotation is clockwise with respect to a normal of a picking face. The rotational axis has to be parallel to the normals of above two planar faces.

Each spatial relationship can be interpreted as a constraint imposed on the degrees of freedom between relative mating or interacting features. Given a set of combination of spatial relationships, the resultant degrees of freedom can be inferred by a rule-based reduction system. An example is adopted from ^[52] to illustrate the concept as shown in Figure 11.



Degrees of freedom based on geometry:

p1: {plan_z::rot_x,rot_z}
 p2: {cyl_x::lin_y,rot_z}

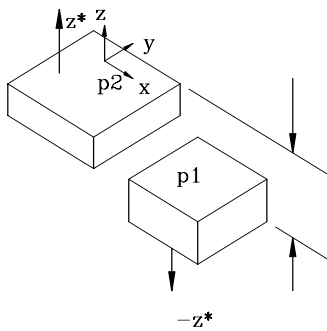
(a)

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'}

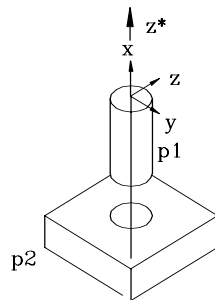
(c)



Degrees of freedom based on geometry:

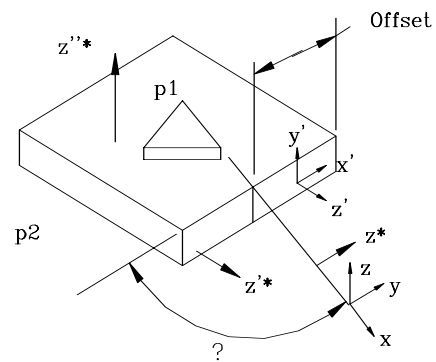
p1: {plane_z::rot_z}
 p2: {plane_z::rot_z}

(d)



p1: {lin_x::rot_x}
 p2: {lin_x::rot_x}

(e)



p1: {plane_z::rot_z}
 p2: {plane_z'::rot_z'}

(f)

Figure 13 Six types of spatial relationships: (a) against, (b) parallel-offset, (c) include - angle, (d) parax-offset, (e) aligned, (f) incline-offset

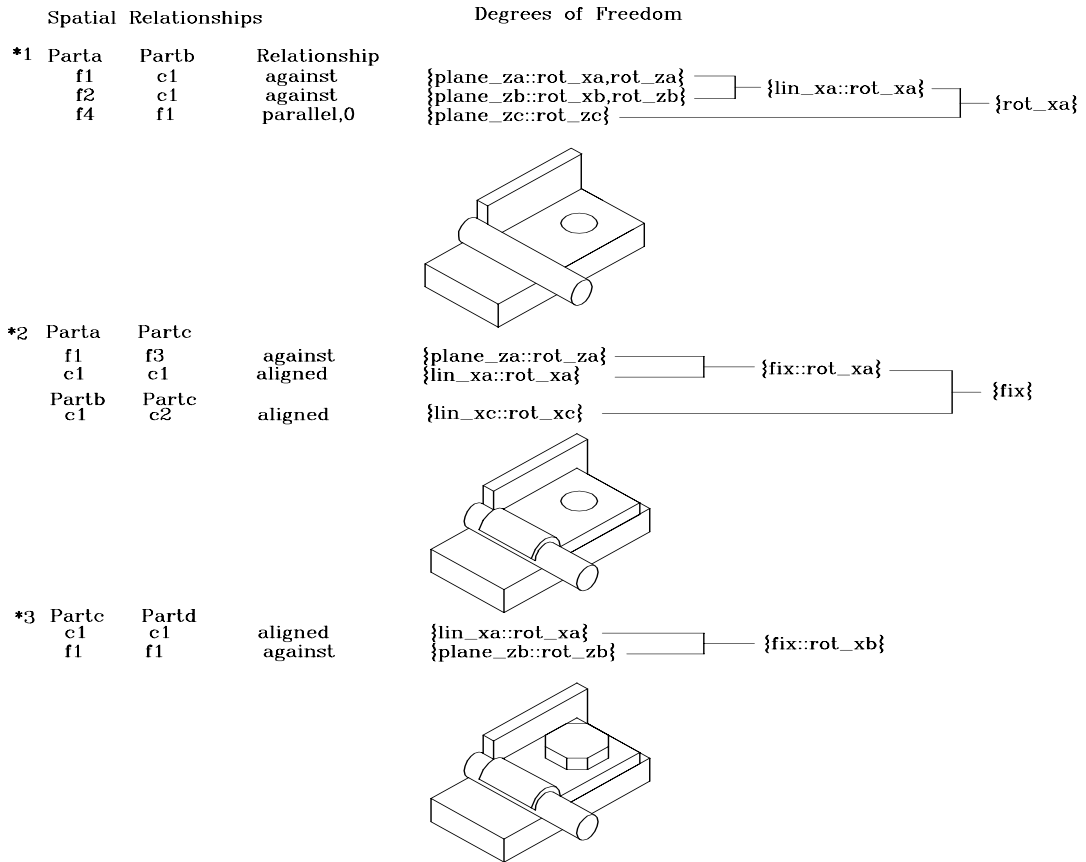
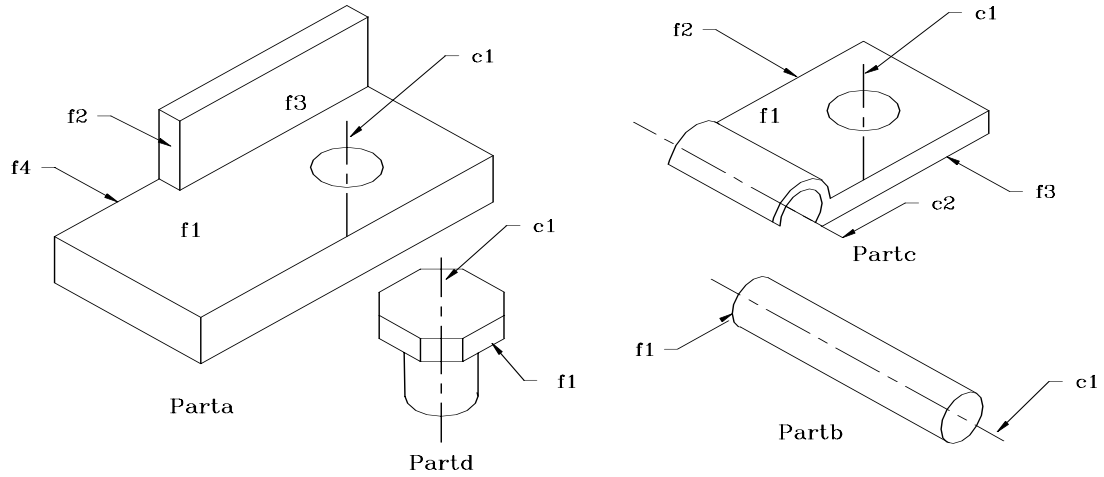


Figure 14 An example of assembly by spatial relationships

2.4 Design Constraints

Representation of constraints is a major challenge facing the design engineers. Due to the vital nature of constraints in design, their representation has received considerable attention in the design research community. Many researchers^[54,55,56,57] believe that engineering design is a constraint-oriented process. Some researchers argue that design process involves the recognition, formulation and satisfaction of constraints.^[58] Stauffer and Slaughterbeck-Hyde^[59] defined constraints as a piece of information that limits design variables to a specific set of permissible values. Van Hentenryck and Saraswat^[60] considered a constraint to be a restriction on a space of possibilities. In summary, a constraint may be considered as either the bound on a single design parameter or the relation among a set of design parameters. Although the characteristics of constraints within the domain mapping process have been studied, no explicit representation scheme is available for modeling design constraints.

Criteria and design knowledge include the preferences and constraints for evaluating design concepts. Consequently, constraints must be expressed explicitly with a systematic and consistent representation scheme. The constraints can then be examined and manipulated by engineers or computers with several advantages;^[61] constraints can be checked for accuracy, sufficiency and completeness of decisions.

Suh^[62] discussed input constraints and system constraints in axiomatic design. Input constraints are those given in design specifications, usually expressed as bounds on size, weight, materials, and cost. System constraints are imposed by the system in which the design solution must function.

Feature based design methods use geometric constraints to model the geometric relationship within a feature or among different features. ^[63] Although constraint modeling of features and geometry is well supported by the feature-based design methodology, it only models the results of a decision, not the constraints that govern the decision. Representation of constraints that affect design decisions is of great importance because a good decision should be based upon explicit and complete information.

A single constraint can be expressed in different formats: domains, equalities, inequalities, or rules. A domain represents the bound of a design parameter, while equalities, inequalities and rules describe the relations among design parameters. ^[61]

A set of constraints may be represented as: (1) the constraint network. (2) the constraint-variable incidence matrix, and (3) the adjacency matrix. A constraint network is a collection of constraints that are interconnected by virtue of sharing variables. ^[64] Constraint networks are useful for constraint propagation and monitoring violations. The purpose of a constraint variable incidence matrix is to study the relationships between constraints and variables. ^[65,66] In the matrix, rows represent constraints and columns represent variables.

Three types of constraint-based design methods identified in the literature are: constraint monitoring, constraint satisfaction, and constrained optimization. Constraint monitoring uses constraints passively to check whether a decision satisfies all constraints. Constraint satisfaction uses constraints actively to derive some values of variables based on given input values of other variables. Constrained optimization, on the other hand, aims at finding the best solution from the alternatives to optimize the objective functions subject to constraints.

Many researchers insist that constraint monitoring is more appropriate for concurrent engineering. ^[67] Bowen et al. ^[68] pointed out that a constraint satisfaction system tends to

automate the design process. Constraint monitoring takes advantage of the engineer's knowledge and intelligence. In constraint monitoring, the engineer assigns values to variables, one after another. The algorithm propagates the values through the constraint network. If an assignment violates any constraint, the engineer has to provide a different value. To use constraints actively, constraint satisfaction methods derive the values of some variables based on the given values of other variables. To solve the constraint satisfaction problem in embodiment design, Thornton and Johnson ^[69] constructed a constraint violation function based on a set of non-linear and mixed equality and inequality constraints. If all constraints are satisfied, the value of this function is zero. If any constraint is violated, the function has a positive value.

Some constraints, such as rules, cannot be expressed mathematically. One solution for considering both is to use the Constraint Logic Programming (CLP) paradigm. ^[70] Constraint Logic Programming is a merger of two declarative paradigms: constraint solving and logic programming. ^[71] Lakmazaheri and Rasdorf ^[70] used Constraint Logic Programming (CLP) successfully to perform the analysis and synthesis tasks in truss structure design.

Although the methods discussed in this chapter have tried to model product functionality, there are limitations to the various approaches. These limitations include:

- A restriction to functions that are can be modeled as input and output flows.
- The function models have generally been based on high-level approach providing little link to actual design elements or product parameters.
- The models are too restrictive and not generic to support the design of variety of products.

- There is a lack of suitable propagation mechanism from conceptual design to downstream product development steps.

The functionality-based design (FbD) methodology discussed in this thesis will address these issues.

3.0 PROPOSED FUNCTIONALITY-BASED DESIGN MODEL

As earlier stated in Chapter 1, the objective of this research is to develop a scheme for mapping product functionality into mathematical relationships and constraints. This mapping process is known as functionality modeling. The concept of functionality operation, operand, operator, relationship, coupling bonds, and constraints modeling is developed in this work to aid functionality-based product conceptualization and propagation to detailed design and analysis phases of product realization. This approach allows for the description of a mechanical device such that functionality-based design can be supported. Specifically, in order to realize the above objective, the following tasks were accomplished:

- Functionality modeling to enable CAD systems to capture the designer's intent during the conceptual design phase. This involves the development of a scheme for representing, propagating, and enforcing the product functionality as design constraints.
- Functionality data structure development to allow for the description of product functionality in a transparent manner.
- Functionality relationship and mapping to develop the set of functionality relationships and mapping operations that enables the description and propagation of product functionalities to downstream design activities.
- Description of functionality model for commonly used mechanical functionalities as a means of support for design repository of reusable product knowledge.
- Development of computational tools to implement the above concepts in a CAD system.

The computational tools developed in this research include:

- Functionality definition interface and tool kits;
- Functionality object model; and
- Functionality data representation in XML.

The concerns of functionality modeling are best illustrated with an example. Given the product (dumb-bell) in Figure 15, how can we describe the function of components P_1 , P_2 , and P_3 in such a way that it captures all the functionality related issues in the design and operation of the product? Also of interest is the question of how we describe the functionality relationship of components P_1 , P_2 , and P_3 in the product a way that any proposed design can automatically be evaluated against such descriptions. In addition, how such descriptions should be propagated to downstream design activities is of interest. What formalism allows for such description to be entered in a computer to become part of design data? Once a possible functionality model is defined, the physical structure of P_1 , P_2 and P_3 could be varied subject to the proposed functionality constraints.

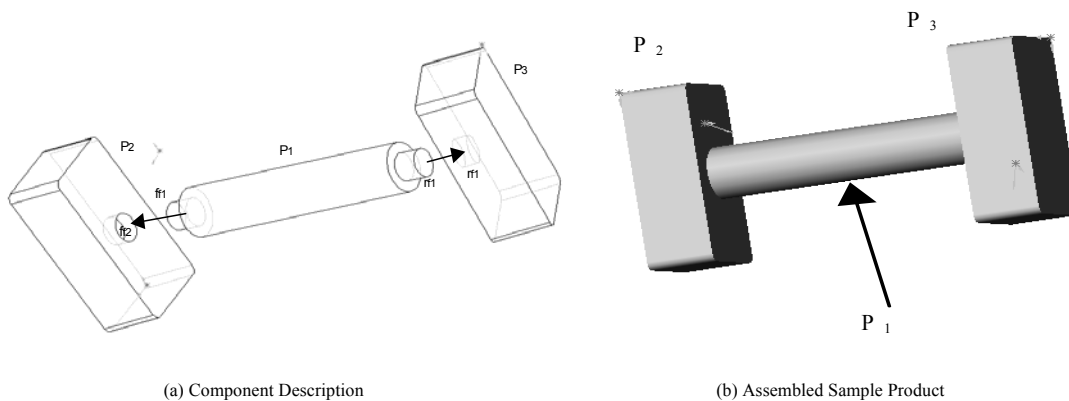


Figure 15 Mechanical product sample – dumb-bell

(Note: Dumb-bells are used as weights in fitness exercises. Hence, the functionality of a dumb-bell is to provide the necessary load used to train the body muscles.)

Functional modeling has to ensure that two important model views are taken into considerations. Hsu *et al.* identified the two model views as computer-oriented and human-oriented models.⁽⁷²⁾ The computer-oriented models ensure that computational reasoning and manipulations can be done in a very efficient manner. Examples of computer-oriented models are structured languages, graphs, and mathematical models. Human-oriented model on the other hand, provides conducive modeling techniques that aid the creativity of human designers. Examples of human-oriented models are images and natural languages.

In general, constraints are restrictions on the design space (form or physical configuration). The inclusion of design constraints reduces the available choice of design to satisfy the design objective. The more constraints a problem has, the less solution space is available. Therefore, constraints must be expressed explicitly with a systematic and consistent representation scheme. The geometry and functionality of a product are represented in terms of engineering constraints in a constraint based modeling system. Constraint definition and representation mechanism are hence required to model these engineering constraints that are imposed on the product design. The representation of product functionality as a design constraint ensures that the possible design space (or choices) is restricted to only design configurations that will satisfy the original product functionality.

The strategy proposed for the functionality-based product model is described here. The reason for the approach is also presented along with the justifications for the proposed methodology.

3.1 Modeling Scope

The work developed in this research is limited to the class of designs that deal exclusively with mechanical devices governed by well-understood mathematical laws of physics such as Newton's laws of motion. A mechanical device is a piece of equipment, mechanical in nature, designed to serve a special purpose or perform a special function.^[73] It generally consists of mechanical members connected by joints. These members are so formed and connected that they transmit constrained motions by moving upon each other as mechanisms, or they transmit forces without any relative motion as structures. By limiting to this class of products, it is believed that a solid foundation that result from this research might then be extended to other classes of engineering systems. Previous researchers^[25] have attempted to generalize this approach to all classes of products. The assumption by some of these groups is the existence of a universal design theory that will apply to all classes of products – mechanical, electrical, chemical, and even biological. This assumption is however still being debated. However, by focusing on a sub-set of products that are well understood, the chance of success is greatly increased by limiting the work to a level that is manageable to produce results that have practical application in the design of mechanical systems.

Another major limitation or assumption in this work is that materials are restricted to idealized solids – rigid bodies – while energy consideration and modeling is restricted to mechanical energies comprising of force and torque. The material restriction implies that fluids (gases and liquids) and highly deformable bodies are not implemented in the functionality model. The system however provides a classification and makes room for possible extension that might include these material categories in future. To accommodate other energy sources in this work, a functionality primitive – energy converter – is used to transform other energy

sources to mechanical form before being applied to the system. This assumption is not too restrictive as the major source of energy in mechanical systems is mechanical energy (force/torque). Another implication of the energy restriction is that functionality objects can be built to act as energy converters and used in a plug-an-play fashion in a fully deployed system. This is illustrated in Figure 16.

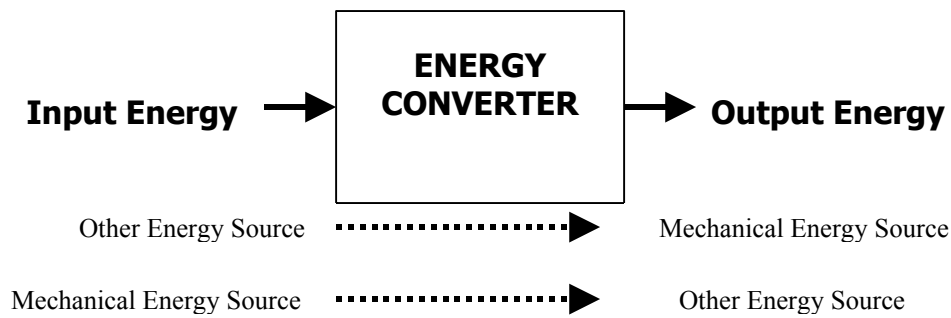


Figure 16 Energy conversion functionality

3.2 Major Challenges

Product functionality modeling is a very challenging problem that involves the amalgamation of various engineering fields such as kinematics, artificial intelligence, feature and geometric reasoning, constraint reasoning, and domain specific knowledge such as material properties. The major challenges involved in this research are highlighted below:

- Mapping between function and structure: the connection between function and structure could be indirect in an artifact.

- Context dependency of product function: a product that performs function “a” under condition “x”, may also perform function “b” under condition “y”.
- Non-functional constraints: additional non-functional constraints (such as the physical processes e.g. energy storage in a compressed spring; and interactions e.g. friction on rubbing surfaces) are difficult to capture.
- Adequate modeling structures: the existing product modeling techniques are inadequate for dealing with capturing functionality requirements during the conceptual design.
- Design phase interoperability: lack of integration between conceptual design and detailed design phases in existing CAD packages.

As pointed out earlier in the Chapter 2, many researchers have tried to model functionality. The models proposed by these researchers have been very restrictive, and lack the robustness necessary to support functionality-driven design. The key factors that have contributed to these inadequacies are:

- Most of the models have only considered a particular aspect of functionality. Functionality is a very broad and encompassing subject that is context dependent. For example, a model that tries to view functionality as inputs and outputs will only result in a description of a particular view of functionality. In this research, a model that tries to accommodate different views/descriptions of functionality is proposed.
- Another major problem encountered by most researchers that have tried to generate *form* from *function* is the lack of a good functionality model that will aid the creativity of the designer. In this work, a functionality representation model that supports the creativity of

the designer is described. This is accomplished by making sure that two important model aspects are represented: computer-oriented model and human-oriented model. For creativity to be simulated, the human model includes both text description and graphical representations.

- One thing that is lacking in most of the previous work is the representation of the functionality model as a design constraint that allows a design to be tested transparently against those constraints.
- A major problem faced by previous research models is the inability to support incremental design procedures. That is, the ability to start conceptual design with scanty information as is usually the case in practice. At the beginning of a design, complete product information is usually not available and is only provided or developed during the course of design process.

3.3 Functionality Operations

Engineering design process involves the mapping of specified product tasks unto a description of a realizable physical structure. This specified task of the design is what its physical realization is supposed to do. The physical structure is the actual physical parts out of which it is made, and the part-whole relationship among them. Consequently, *the functionality of an entity (or a product) is the general task it performs*. The entity may be either a physical artifact (e.g. the brake drum of a car wheel assembly) or an invisible object (e.g. the use of electro-magnetic forces to relocate a ferro-magnetic object). An important attribute of functionality is the existence of an identifiable task that needs to be performed.

Typically, the need for a new design arises as a result of identification of a new functionality to be accomplished by the product. Designs are conceptualized in terms of the functions they need to accomplish. That is, design is functionality driven. The physical realization of product functionality is hierarchical in nature. That is product functionality can normally be described with an overall functionality by bringing together separate entities of sub-functions and functional entities. Each sub-function, accomplishes a specific function that contributes towards the accomplishment of the overall function. The sub-functions are in turn sub-divided into sub-functions and functional entities.

A functionality operation is defined as the realization of a given task. Where a task is the desired functionality of the entity. Thus, an operation defines relationships and imposes constraints on functionality elements (operands and attributes). The key element in the definition of a functionality operation is the existence of an object or entity configuration that performs the given task. Hence, the functionality operation is the solution outcome of the functionality-modeling objective. For example, consider the task of providing support to a beam structure by columns as shown in Figure 17. The functionality and corresponding operations are as follows:

- *Functionality*: support of a beam.
- *Functionality operation*: representation of the act of providing support to the beam.

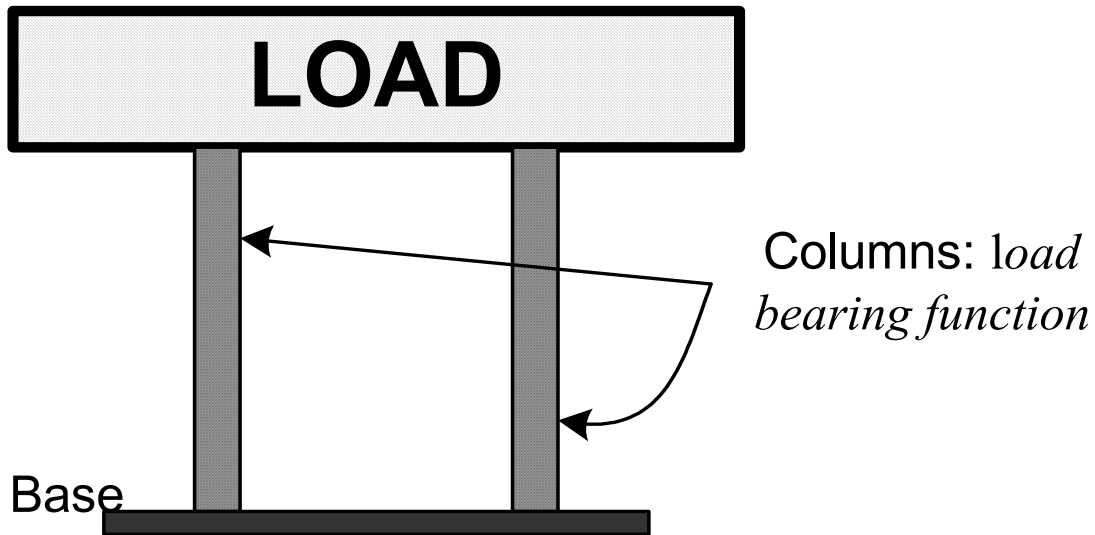


Figure 17 Load bearing as an example of mechanical functionality

In this work, mechanical functionality is considered as being composed of two components: the *operator* and the *operand*. These two entities are defined as follows:

- An *operand* is a distinct element involved in the realization of a given functionality. Operands constitute the building block of mechanical functionalities. They are entities that come together to accomplish any mechanical task. For example (1) Transmission of force (operands: force and transmission medium) (2) Transform of force to torque (operands: force, torque, and transmission medium)
- An *operator* is the established relationship between operands in the performance of a functionality operation.

The use of the term “*flow*” to model product functionality was proposed by previous researchers. [18, 23, 74] Many researchers [26, 27, 40, 43] on product functionality modeling have adopted this definition. The definition of product functionality in terms of *flow* however, places

some restriction on the functions that can be modeled. The two major limitations are highlighted below:

- It assumes causality, which is cause and effect relationship, requiring the definition of mechanical elements as flows with inputs and outputs. This assumption limits the possibility of functionality modeling to the class of functions with identifiable inputs and outputs.
- The definition of functional flow includes a class for signals. However, signal is a perception and hence not considered a basic mechanical operand. A signal may be derived by a combination of other basic functional elements (see example in Section 4.0).

The basic mechanical elements are modeled in this work by excluding *signals* and the *flow restriction* on the nature of mechanical elements. The result of this is renamed operand. The term “operand” is borrowed from computer engineering where it is used to represent data component of a computer instruction code. It represents the portion of the computer instruction that constitutes that actual values and parameters used in execution of an instruction. Similarly, in the modeling of functionality, operand is used to represent the mechanical entity that is used to accomplish a general mechanical task. The term operand is used in this work because it captures the nature of the mechanical elements as the resources that are employed to achieve a particular function. The operand may also be the result or output of a mechanical function. The set of operands that are sufficient to describe the selected class^[*] of mechanical functionality is the focus of this research.

* Scope of this research as described in Section 4.1 is limited to mechanical devices using non-deformable materials (excluding gases and liquids).

An important recognition from the above definitions is that mechanical functionality operation is realized by prescribing relationships at the linkage of operands. This prescription is achieved by the operator of the mechanical functionality operation that needs to be performed. The operands are the entities the operator relates together for the actualization of a given task. A more detailed description of the functionality operand is presented in Section 4.4 while that of the operator is presented in Section 4.5.

A functionality model is composed of an aggregate of functionality operations. Functionality operation is classified according to the level of interaction complexity of its components. Each functionality operation could be either *basic* or *compound* in nature. A *basic functionality operation* is defined as a mechanical operation in which a functionality operator relates only a set of operands together with their corresponding attributes. On the other hand, a *compound functionality operation* corresponds to functionalities that are more complex by relating set of operands with other sub-functionality operations.

3.3.1 Basic Functionality Operations (*BOPN*)

Acts on only operands (with their corresponding attributes). They represent basic mechanical functions. Basic functionality operation yields a special class of mechanical functionality operations that is referred to as primitive. A subsequent combination of primitives yields functionalities that are more complex. Symbolically, we define a basic functionality operation as:

xBOPN: {<*opand 1 : attrib 1*>, <*opand 2 : attrib 2*>, ... <*opand k : attrib k*>}

where,

xBOPN = basic functionality operator
 <*opand i*> = functionality operand *i*, *i* = 1, 2, ..., *k*
 {*attrib i*} = attributes of operand *i*

A more detailed description of this representation and the corresponding example is provided in Chapter 4 of this thesis.

3.3.2 Compound Functionality Operations (*COPN*)

A subsequent combination of the functionality primitives and definition of their unique relationships and interactions yields a more complex functionality object that may be used to construct any mechanical function. Acts on both operands and operations. Compound functionality operation yields a special class of mechanical functionality operations that is referred to as functionality compound operations. Symbolically, we define a compound functionality operation as:

xCOPN: {OPN1:[*copand 1*], OPN2:[*copand 2*], ... OPNk:[*copand k*], <*opand 1 : attrib 1*>, <*opand 2 : attrib 2*>, ... <*opand t : attrib t*>}

Where,

xCOPN = compound functionality operator
OPNi = functionality operator *i*, where *i* = 1, 2, ..., *k*
 [*copand i*] = coupling operand set *i* corresponding to *OPNi*
 <*opand j*> = functionality operand *j*, *j* = 1, 2, ..., *k*
 {*attrib j*} = attributes of operand *j*

Using a more concise representation, a compound functionality operation can also be represented as:

$$x\mathit{COPN}: \{ \langle \mathit{OPN} \rangle \mid \langle \mathit{OPAND} \rangle \}$$

Where,

- $x\mathit{COPN}$ = compound functionality operator
- $\langle \mathit{OPN} \rangle$ = set of lower functionality operations
- $\langle \mathit{OPAND} \rangle$ = set of functionality operand

3.3.3 Generic Functionality Model

A generic functionality model will make it possible to impose functionality as a set of design constraints and preferences that may be used during the detailed design and analysis phase of the product. In this research, new design representation formalism is developed to enable functionality-based design of mechanical products. This methodology provides a framework that will allow a designer to carry out conceptual design with the aid of a computer. It also serves as an interface tool between the conceptual design phase and the detailed design phase of a product.

In this research, the generic functionality model is presented as a scheme for mapping the functionality description into mathematical relationships and constraints. This provides a means for *representing*, *propagating*, and *enforcing* product functionality as design constraints. This functionality representation links product functions with the structural (physical) embodiment used to realize the functions. This representation also offers support for recurring product

functionalities to be modeled as a functionality primitive, and used as a repository of reusable product knowledge that may be related to a particular form or geometric pattern. A subsequent combination of the functionality primitives and definition of their unique relationships and interactions yields a more complex functionality operation that may be used to construct any mechanical function.

Given a set of functionality objects (\mathbf{o}_{ij}), with their corresponding attributes (\mathbf{a}_{ijk}), we define a generic functionality model of a functionality operation (a modeling representation for both *xBOPN* and *xCOPN*) is defined by a ternary relation, Γ_i , given by Equation 2

$$\Gamma_i = \{(r, s, f) \mid r \in \mathbf{R}_i, s \in \mathbf{S}_i, \text{ and } f \in \mathbf{F}_i\}$$

Equation 2

Where,

- i = functionality operation index
- \mathbf{R}_i = a functionality operator, defines a set of relationships between functionality objects
- \mathbf{S}_i = a set of functionality states, defines state each object can assume.
- \mathbf{F}_i = function-form mapping set, defines mappings between relationships and physical forms.

Given a set of functionality objects, \mathbf{O}_i in a functionality operation i , the functionality operator defines the relationship between these functionality objects, it can be represented mathematically as shown in Equation 3. A functionality object (\mathbf{o}_{iq}) is either an operand or an interacting functionality operation (*operand*: material | energy and *operation*: lower level functionality operation). A relation can assume any of the following forms: spatial; physical

aspect / processes; technological constraints; user preferences. A more detailed discussion on functionality relationships is presented in Section 4.5

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

Equation 3

Where,

$\mathbf{o}_{iq} = \{\mathbf{a}_{iqs} \mid \mathbf{a}_{iqs} \in \mathbf{A}_{iq}\} =$ object q in functionality operation i

$\mathbf{A}_{iq} =$ attribute set for functionality object \mathbf{o}_{iq}

$r_{ijk} =$ relation, mapped relationships between objects j and k .

$j, k, q =$ functionality object indices

$i =$ functionality operation index

Similarly, the state of a functionality object is defined by considering the various possible values its time-varying attributes might assume. For instance, the possible states of a load-bearing structure could range from *NO_LOAD* to *MAX_LOAD* condition. Hence, the state of a functionality object, q can be represented mathematically by a state set, \mathbf{S}_i as shown in Equation 4.

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

Equation 4

Where i is the functionality operation index and \mathbf{s}_{iq} is the state of objects q in functionality operation i . Hence, objects may have more than one functionality state. The functionality object state is given by Equation 5.

$$s_{iq} = \{(\mathbf{a}_{iqs}, \mathbf{v}_{iqs}) \mid \mathbf{a}_{iqs} \in \mathbf{A}_{iq}, \mathbf{v}_{iqs} \in \mathbf{V}_{[aiqs]}\}$$

Equation 5

Where

\mathbf{A}_{iq} = *attribute set* for functionality object \mathbf{o}_{iq}

$\mathbf{V}_{[aiqs]}$ = *set of possible values (range) of attribute* \mathbf{a}_{iqs}

Finally, the functionality-form mapping, \mathbf{F}_i defines the mapping between the functionality relationship between the functionality model and the physical structure used to accomplish the given task. This linkage is included in the functionality model to provide support from form conceptualization, propagations to detailed design tools, and design verification. With this inclusion, each functionality description is attached to the physical form / structure as the design evolves. A mathematical representation of this mapping is defined by Equation 6.

$$\mathbf{F}_i = \{(r_{ijk}, f_{ijkl}) \mid r_{ijk} \in \mathbf{R}_i, f_{ijkl} \in \mathbf{FF}\}$$

Equation 6

Where,

r_{ijk} (*relation*): mapped relationships between objects j and k .

f_{ijkl} (*form*): conceptual forms mapped to relation r_{ijk} .

\mathbf{FF} = set of functionality forms

i = functionality operation index

j, k = functionality object indices

l = form index

A more in-depth discussion of the various components of the functionality model is presented in chapters four and five. A case study using FbD is discussed in chapter six.

4.0 FUNCTIONALITY OPERANDS

An *operand* is a distinct element involved in the realization of a given functionality. Operands constitute the building block of mechanical functionalities. They are entities that come together to accomplish any mechanical task. An operand is the mechanical product functionality equivalent of “data” in computer systems. Operands are classified into two broad categories: *material* and *energy* operands. A *material* operand is any physical entity that has some mass and volume. An *energy* operand on the other hand is the functionality component that effects some work outcome. This classification is illustrated in Figure 18.

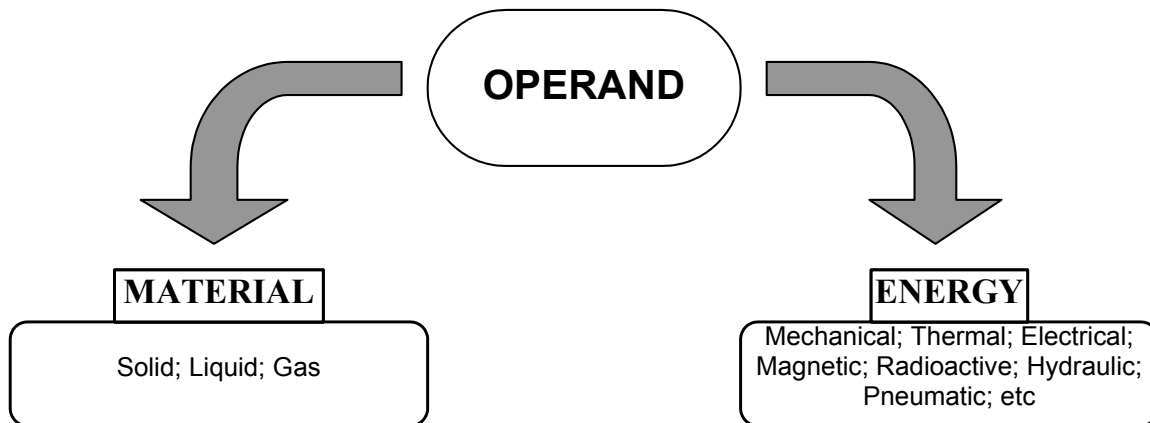


Figure 18 Functionality operands

The existence of an operand alone however, will not yield any meaningful functionality. The operator defines its context of operation by defining the governing relationships. An

important recognition from the above definitions is that mechanical functionality operation is realized by prescribing relationships at the linkage of operands. This prescription is achieved by the operator of the mechanical function operation that needs to be performed. The operands are entities the operator relates together for the actualization of a given task. For example, consider the functionality operation – transmission of force from one point (A) to another point (B) through a medium (see Figure 19). The operands in this case are: the force and the transmission medium (note: that the medium is a solid bar). The existence of force and medium does not provide any meaning or intelligence as to the purpose of these operands. The presence of a transmission operator, however, defines the context and the relationship between the operands – force and medium. A mathematical description of the operation as a primitive (basic) functionality can be represented as shown in Table 2.

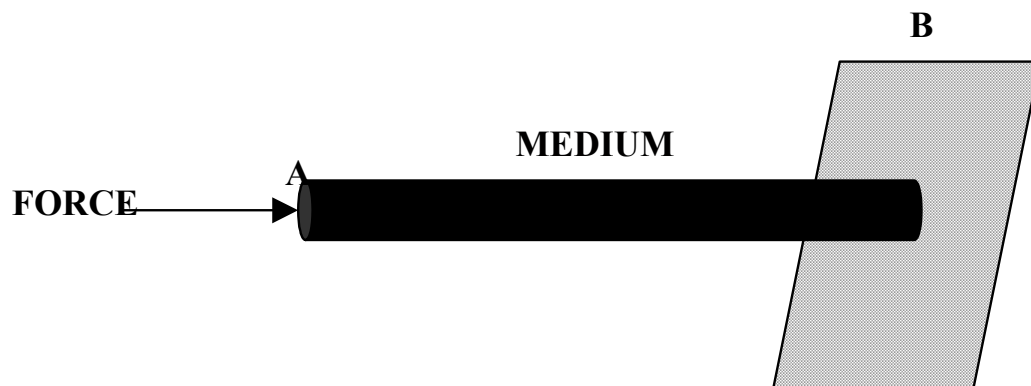


Figure 19 Transmission functionality operation illustrating the interaction between FORCE and MEDIUM operands: transmission of force from point "A" to plate "B"

Table 2 Mathematical description of the transmission operation shown in Figure 19

<i>transmissionBOPN</i> : {<FORCE : attrib FORCE>, <MEDIUM : attrib MEDIUM>}	
where,	
Functionality item	Description
<i>transmissionBOPN</i>	transmission basic functionality operator
<FORCE>	<i>FORCE energy operand</i>
<MEDIUM>	<i>MEDIUM material operand</i>
{attrib FORCE }	<i>attribute set of FORCE operand</i>
{attrib MEDIUM }	<i>attribute set of MEDIUM operand</i>

A detailed functionality model for the transmission example is given in Section 6.32. In the detailed model description, the operator defines the relationship between the attributes of the two operands. For example, how is the magnitude of force related to the deformation or strain in the medium (assuming a solid medium)?

Another example of a functionality operation is the transformation of FORCE to TORQUE (operands: force, torque, & transmission medium). This example is illustrated in Figure 20 and modeled in Table 3.

Table 3 Mathematical description of the transformation operation shown in Figure 20

<i>transformBOPN</i> : {<FORCE : attrib FORCE> <TORQUE : attrib TORQUE> <MEDIUM : attrib MEDIUM>}	
where,	
Functionality item	Description
<i>transformBOPN</i>	transformation basic functionality operator
<FORCE>	<i>FORCE energy operand</i>
<TORQUE>	<i>TORQUE energy operand</i>
<MEDIUM>	<i>MEDIUM material operand</i>
{attrib FORCE }	<i>attribute set of FORCE operand</i>
{attrib TORQUE }	<i>attribute set of TORQUE operand</i>
{attrib MEDIUM }	<i>attribute set of MEDIUM operand</i>

The term flow has been used by some researchers to describe some aspects of the operand. The term operand is used in this work because it captures the nature of the mechanical elements as the resources that are employed to achieve a particular function. The operand may also be the result or output of a mechanical function. The set of operands that are sufficient to describe the selected class^[*] of mechanical functionality is the focus of this research.

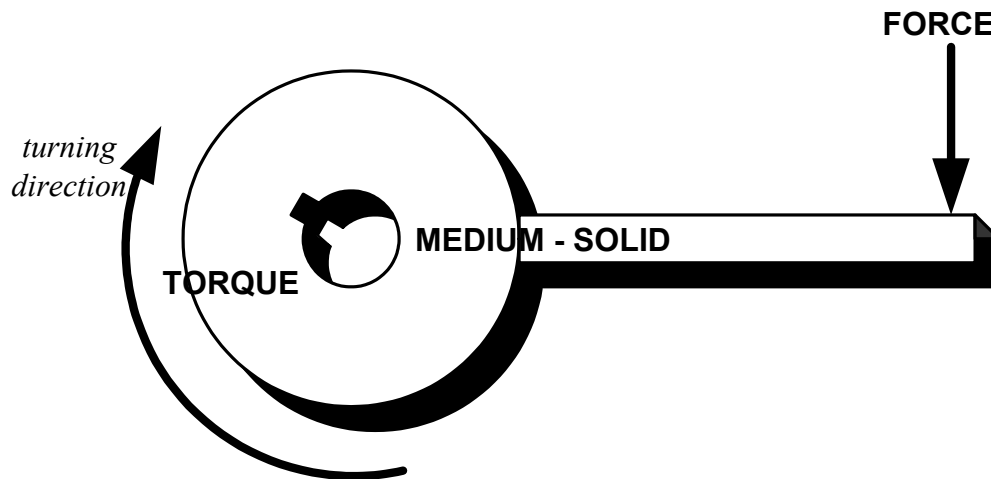


Figure 20 FORCE to TORQUE transformation operation functionality

Functionality operand is restricted to a selected set of materials and energy entities (see Figure 18). This is a modification to the sub-set of *flows* defined by some previous researchers in the field of functional modeling.^[18, 26, 27, 38, 39, 40, 41, 43] The selected subset is sufficient to describe the class of mechanical systems considered in this research. Each operand has its own set of attributes that are used in defining functionality relationships and constraints. An important departure from the work of previous researchers^[18, 38, 39, 40, 41, 43] is the lack of a

* Scope of this research as described in Section 4.1 is limited to mechanical devices using non-deformable materials (excluding gases and liquids).

primitive operand for the signals. This is because signal is a perception and hence not considered a basic mechanical operand. A signal may be derived by a combination of other primitive operations. For example, in a voltmeter, a regulated displacement of a pointer attached to the magnetic coil over a graduated scale may be observed or interpreted as a measurement signal of the potential difference between two points in an electrical system. Szykman's team considered signal as mostly composed of auditory, olfactory, tactile, taste, and visual signals, which is outside the realm of mechanical systems consideration of this work. These signals are related to human sensory system and do not easily translate into physical mechanical design elements.

The following subsections presents a more detailed description of each of the functionality operands employed in this work.

4.1 Material Operand: Solid Operands

A *material* operand is any physical entity that has some mass and volume. Although there are three classes of materials: solids, gases, and liquids; this research is restricted to solids with the assumption that they exhibit the properties of an idealized mechanical body known as rigid body. This restriction is to scope the model development to demonstrate the principles developed here which may then be extended to cover other types of materials.

In this context, a solid material operand is defined as any object with mass having a definite shape and volume. The definition of solid material operands in this work is restricted to "Rigid Bodies", which includes any class of engineering material (metals, alloys, ceramics, polymers, composites, etc) in its solid phase. The rigid body assumption allows us to neglect

highly deformable bodies in the development of solid material operand model. Figure 21 illustrates the structure of a solid material operand. The principal components of the solid operand are: functional features, physical property, degree of freedom, mass property, and material type.

From Figure 21, it should be noted that a solid operand is modeled by: **SOLID {FM, DoF, PyC, MP, MT}**, where **FM** is the set of functional positional markers; **DoF** is the required degree of freedom of operand; **PyC** is the set of physical constraints on the operands; **MP** is the set of mass properties on the operands; and **MT** is the material type constraint. Hence, the attributes of a solid operand is completely defined by **FM, DoF, PyC, MP, and MT**. However, at the beginning of the design process (conceptualization) and functional definition, some of these information may not be known and hence incomplete. The model is however defined by noting the unspecified (unknown) attributes and neglecting them (or using default values) for any intermediate reasoning or computation. As design progresses, these attributes are defined.

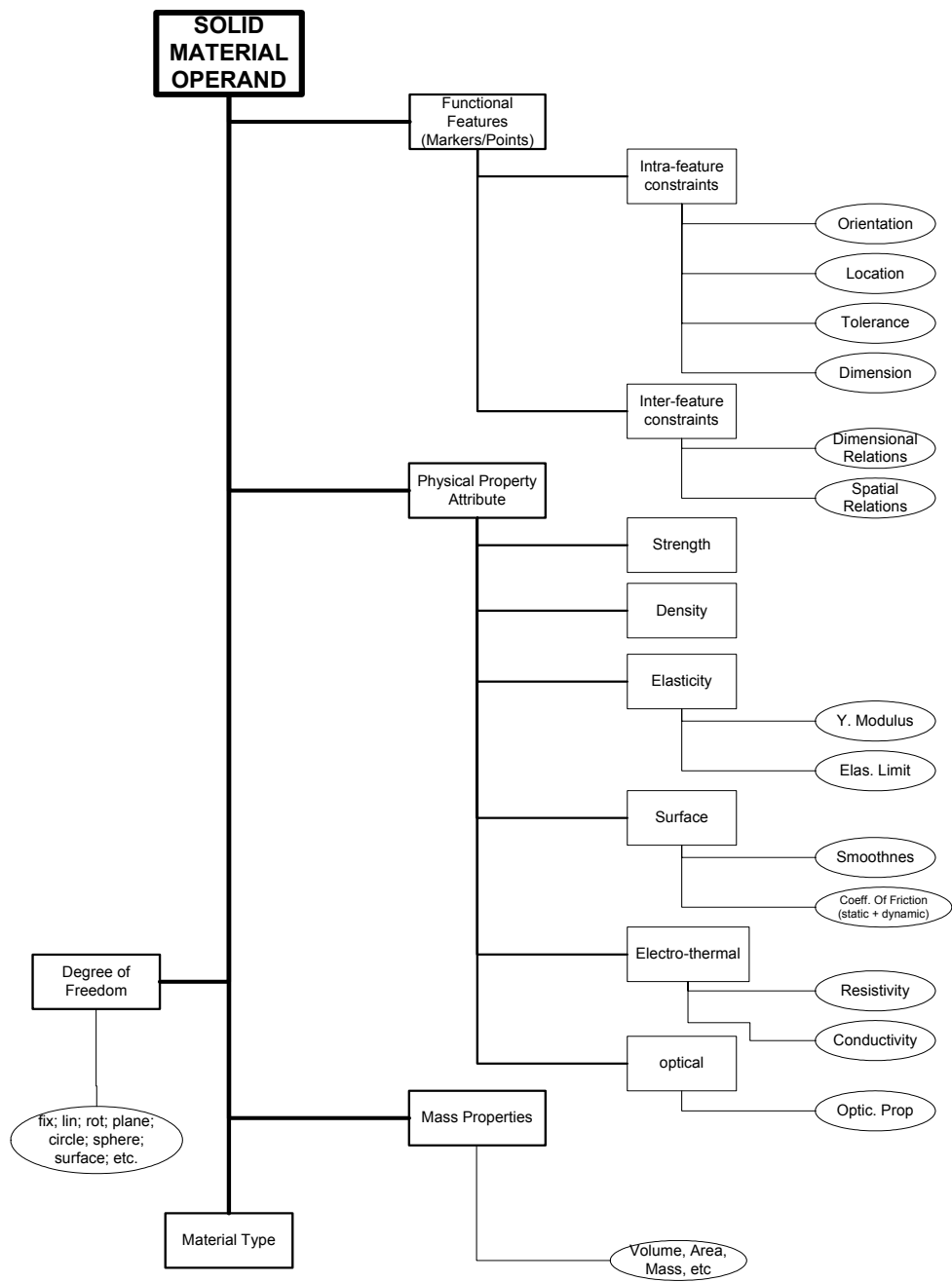


Figure 21 Material operand structure illustrating a detailed attribute modeling of a solid material

4.1.1 Functional Position Marker

Two related operands in a functionality operation do not make contact over their whole surface but only functional regions/point of each operand are in contact. Hence, functional points/regions (markers) are geometric points, locations, or regions on solid operands, which are needed to relatively position the operands according to their coupling relations in the whole functionality operation or structure. They are primarily related to operand's topology and geometry. Each functional position marker must have:

- *Intra-functional marker constraints*: which describes the geometric shape and size, location, orientation, tolerance, and dimension of the functional marker.
- *Inter-functional marker constraints*: which describes the dependencies between functional marker within a solid operand, for example, position of functional point, orientation, feature adjacency, etc.

In the assigning process of operand relationships, the functional points/regions are defined and extracted from the operands. For example, assigning a planar surface of one part against a planar surface of the other part, these two planar surfaces are the functional points/regions of interest in the operands coupled pair. In this fashion, the interpretation of functional markers/points will be mostly lower-level kinds of geometric entities. One reason for that is to easily extract functional points universally from operands, parts or pre-defined features, which are usually diverse because they are intent-oriented. From this point of view, the functional region/point of a material operand could be a point, a planar face, a centerline of a

cylinder or an edge of a face boundary, etc. Therefore, the functional markers/points are determined by the types of operand relationships being specified and the operand types (entities) being selected (see Figure 22). In Figure 22, the functional point involved in coupling the two material operands (P1 and P2) are points A₁ and A₂.

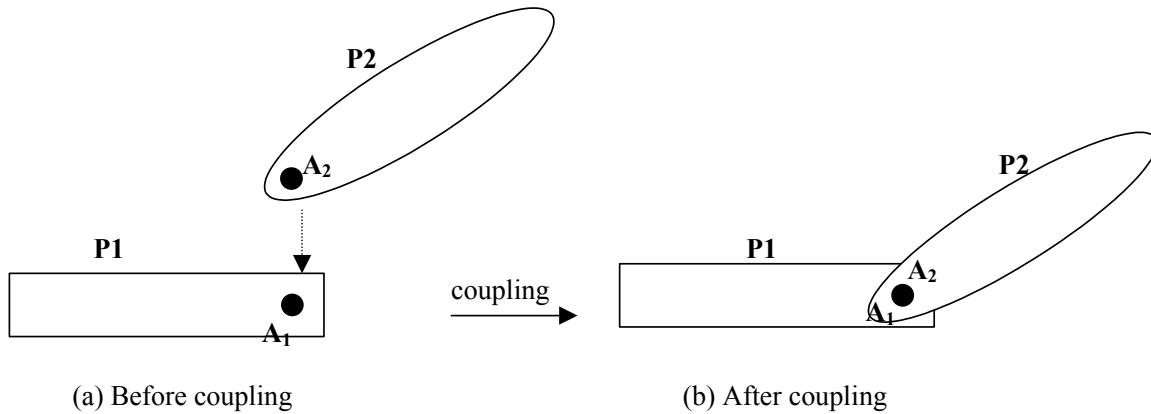


Figure 22 Operand coupling at the functional points, A₁ and A₂

Solid Material operands can be idealized by modeling them as a wire-frame structure (Figure 23) with the regions of particular interest denoted as points on the wire-frame model. When more than two regions (points) are needed as functional point, additional lines may be extended from the existing ones to the new regions of interest. Figure 24 illustrates the concept where four functional points, A, B, C, and M, are identified on the solid operand. The only major constraint imposed on the functional points by the parent solid material operand is that they are part of the same operand and maintain constant spatial relation between one another. To maintain a given relative position between functional points, a dimensional constraint (DC) is imposed on the functional points relative to a reference point known as datum point (M).

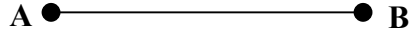


Figure 23 Wire-frame representation of a simple solid material operand

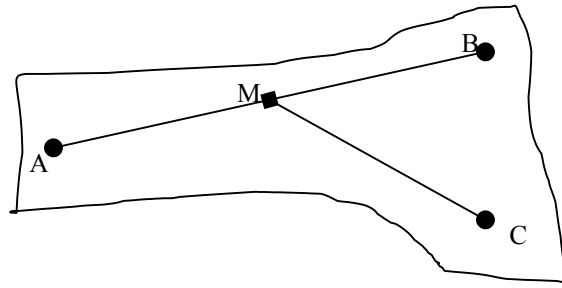


Figure 24 Wire-frame representation of multipoint solid material operand with four functional points A, B, C, and M

In this work, to model functional points in solid material operands, the concept of *markers* is used to describe the position and spatial relations in the functionality model.

Markers

The term “marker” is adopted from Kramer ^[75] and Liu ^[22] with some modification to support modeling of functionality operands. A marker consists of a point (x, y, z) on the operand and two unit vectors (u, v) , which emanate from this point. These markers are used to describe the *geometric constraints* of the *spatial relationships* between their functional marker/features and coupling operands. Examples of markers for the coupling features are shown in Figure 25. It is noted that these two unit vectors (u, v) are defined to be orthogonal in Kramer, ^[75] however that is not necessary in our application as long as these two vectors are not the same. These

markers are generated automatically by the system according to the operand function marker/feature pair. The system chooses or infers from existing functional features on the operands to form the marker.

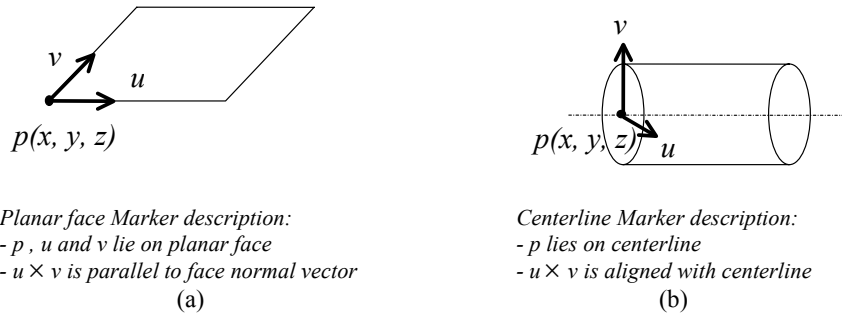


Figure 25 The markers on functional coupling features

A functional marker/feature is defined with markers using the following notation:

M_1 : a point $p_1(x_1, y_1, z_1)$ and a unit vector set (u_1, v_1) .

M_2 : a point $p_2(x_2, y_2, z_2)$ and a unit vector set (u_2, v_2) .

w_i : a unit vector which is obtained by $u_i \times v_i$.

$line(p, w)$: a line through p and parallel to vector w .

$plane(p, w)$: a plane through p and normal to vector w .

$surface_point(\{m: m \in M\})$: the surface fitting (interpolation) through the set of markers, M .

$surface_offset(constr M)$: a revolved surface formed on marker set M .

$intersect_line(M_1, M_2, w_1 \times w_2)$: the intersection line of $plane(p_1, w_1)$ and $plane(p_2, w_2)$, and

$w_1 \times w_2$ is the vector of this line.

From the above definitions, the basic set of functional marker/features of an operand includes the following: a point, a line (including intersect line), and a surface (including a plane as the simple case).

Inter-Functional Marker Constraints

These are geometric constraints that describe the dependencies between functional markers within a solid operand, for example, position of functional point, orientation, feature adjacency, etc. The intra-functional feature constraint describes the geometric *shape*, *dimension*, and *tolerance* of the functional feature. For example, the length of a line, area of a surface, and positional tolerance of a point. The geometric constraint set can then be defined as: (see Figure 26)

- ***on_line*(M_1, M_2, w_2):** p_1 lies on the *line*(p_2, w_2).
- ***on_plane*(M_1, M_2, w_2):** p_1 lies on the *plane*(p_2, w_2).
- ***parallel*(M_1, M_2, w):** w_1 is parallel to w_2 and $(w_1 \bullet w_2) = 1$.
- ***parallel*($M_1, M_2, -w$):** w_1 is parallel to w_2 and $(w_1 \bullet w_2) = -1$.
- ***angle_w*(M_1, M_2, θ):** the angle between w_1 and w_2 is θ .
- ***dist*(M_1, M_2, d):** the distance from p_1 to the projective point *on plane*(p_2, w_2) is d .
- ***dist*(L_1, L_2, d):** the perpendicular distance between line L_1 and line L_2 is d .

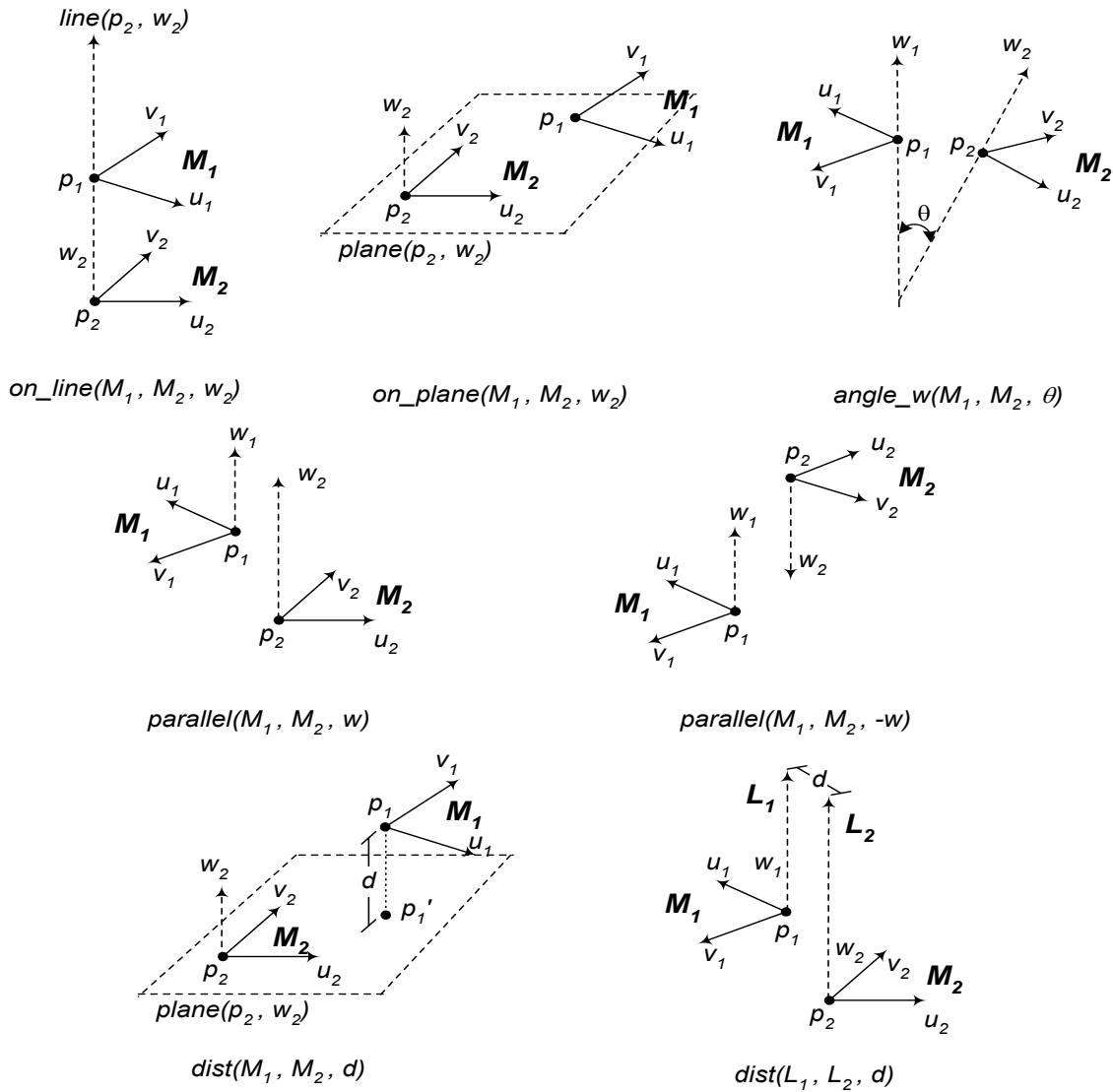


Figure 26 Inter-functional feature constraints representing geometric constraint set

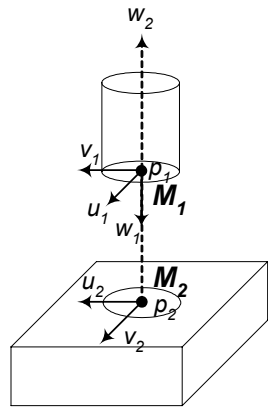
4.1.2 Degree of Freedom: Kinematic Constraints

The *degree of freedom* of a solid material operand defines the allowable motions the operand may undertake in the realization of a functionality operation. In kinematics, a free rigid body has six degrees of freedom. However, the degrees of freedom of an assembled component

are reduced after spatial relationships are imposed. In other words, a spatial relationship can be seen as a constraint used for confining kinematic degrees of freedom of related components. The types of degrees of freedom which can result from spatial relationships are as follows: ^[53, 76]

- *lin_n*: linear translation along n axis.
- *rot_n*: rotation about n axis.
- *cir_n*: translating along a circle with n axis.
- *plane_n, cyl_n, sph*: translating along a planar, cylindrical and spherical surface respectively.
- *roll_lin_n*: rolling along a corner.

The degrees of freedom of a solid 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. In Figure 10, the degrees of the freedom which remained after imposing *aligned* spatial relationship are {*lin_z* :: *rot_z*}. When there is more than one spatial relationship assigned on the component, a rule-based system is applied to determine the intersection of degrees of freedom. A detailed description of these degrees of freedom reduction rules can be found in Liu. ^[53]



$$\begin{aligned} &aligned(F_1, F_2) \\ &= on_line(M_1, M_2, w_2) \cap parallel(M_1, M_2, \pm w) \end{aligned}$$

Figure 27 Aligned spatial relation specification example

4.1.3 Physical Constraint Set

This component of the solid operand includes the desired material physical properties of the operand. This set in effect defines the physical composition of the solid. A knowledge of this set of attribute can be used to infer the type of material suitable for the given functionality task. The magnetic and optical properties of material are not included in this model. These properties form part of the future extensions to this work. This set of attributes include the following:

- **Strength attribute:** measures the stress (*load per unit area* – N/m^2) the operand is able to withstand under operating condition (includes tensile, compressive, torsion, and bending stress). The components of stress are: proportionality limit, elastic limit, yield strength, ultimate strength, and fracture stress. While all the above stress specifications are component of the strength attribute in a realistic design example, most engineering solids are expected to operate well within the *yield limit*.

Design deals with allowable stresses, or reduced value of strength. The allowable normal stress (σ_{all}) and the allowable shear stress (τ_{all}) for ferrous and nonferrous metals for various types of loading may be represented by: [77]

- Tension: $0.45S_y \leq \sigma_{all} \leq 0.60S_y$
- Shear: $\tau_{all} = 0.40S_y$
- Bending: $0.60S_y \leq \sigma_{all} \leq 0.75S_y$
- Bearing: $\sigma_{all} = 0.90S_y$

Where, S_y is the yield strength of the selected material. The above equations may also be applied to polymers and ceramics if the ultimate strength at the break and fracture strength, respectively, are substituted for yield strength in the equations above. Consequently, in this model, the strength attribute refers to the allowable stress limit, which is well within the yield limit.

- **Elasticity attribute:**

- Axial and shear strains. Axial strain is the ratio of change in length to original

length (i.e., $\varepsilon = \frac{l - l_0}{l_0}$).

- Poisson's ratio (ν) relates the transverse strain ($\varepsilon_{t,avg}$) to the axial strain ($\varepsilon_{a,avg}$):

$$\varepsilon_{t,avg} = -\nu\varepsilon_{a,avg}$$

- **Ductility:** measures the degree of plastic deformation sustained at fracture. Specified by

the percentage elongation (%EL) or $(\%EL) = \left(\frac{l_{fr} - l_0}{l_0} \right) \times 100\%$; where l_{fr} is the length

of the operand at fracture and l_0 is the length of operand without load. Thus solids are grouped into two based on their ductility: ductile solids ($\%EL \geq 5\%$) and brittle solids ($\%EL < 5\%$).

- **Density:** this is a measure of mass per unit volume (kilograms per cubic meter – kg/m³)
- **Surface characteristics:** Tribology
 - Surface Friction: force resisting relative movement between surfaces in contact.
 - Force of *static friction* (F_s); Force of dynamic friction (F_k); and Rolling friction (F_r).
 - Surface Roughness: Centerline average or arithmetic average surface roughness R_a is given by: $R_a = \frac{1}{N} \sum_{i=1}^N |z_i|$, where z_i is the height from reference line and N is the number of height measurements taken.
 - Wear rate (W_r): this is the volume of material lost from one surface, per unit distance slid.
- **Thermal characteristics:**
 - Thermal conductivity (K_t), unit - watts per meter-Celsius
 - Thermal expansion coefficient ($\bar{\alpha}$), unit - (°C)⁻¹
 - Specific heat capacity (C_p), unit – J/(kg-°C)
- **Electrical Properties:**

Resistance (R) – Although this is temperature dependant, the temperature effect is neglected in the functionality model developed in this work.

4.1.4 Material Type

Choosing the solid material is an important step in the design of a product. Being able to exploit a material's potential and characteristics is essential to ensuring that the best material is

used for a particular solid operand. Therefore knowing the properties of solid materials is essential. Engineering materials include metals, polymers, composites, ceramics, and wood. For each class of engineering material, wide ranges of types are available to a designer. For example, in the metal class, we have iron and its derivatives, aluminum, copper, etc. Even within a given type of metal, the property still varies depending on the grade of metal and treatment (e.g. thermal treatment) the metal is subjected to. A broad classification of mechanical solid material types is as follows:

- **Metals:**
 - Ferrous metals: carbon, alloy, stainless, and tool and die steels.
 - Nonferrous metals and alloys: aluminum, magnesium, copper, nickel, titanium, super-alloys, refractory metals, beryllium, zirconium, etc.
- **Plastics (Polymers):** thermoplastics, thermosets, and elastomers.
- **Ceramics:** glass ceramics, glasses, graphite, and diamond
- **Composite materials:** reinforced plastics, metal-matrix and ceramic-matrix composites, and honeycomb structures
- **Wood:** laminated, hard wood, etc.

Each type of material will have its own unique value for the following properties: ductility, strength, elasticity, conductivity, etc. as shown in Figure 28.

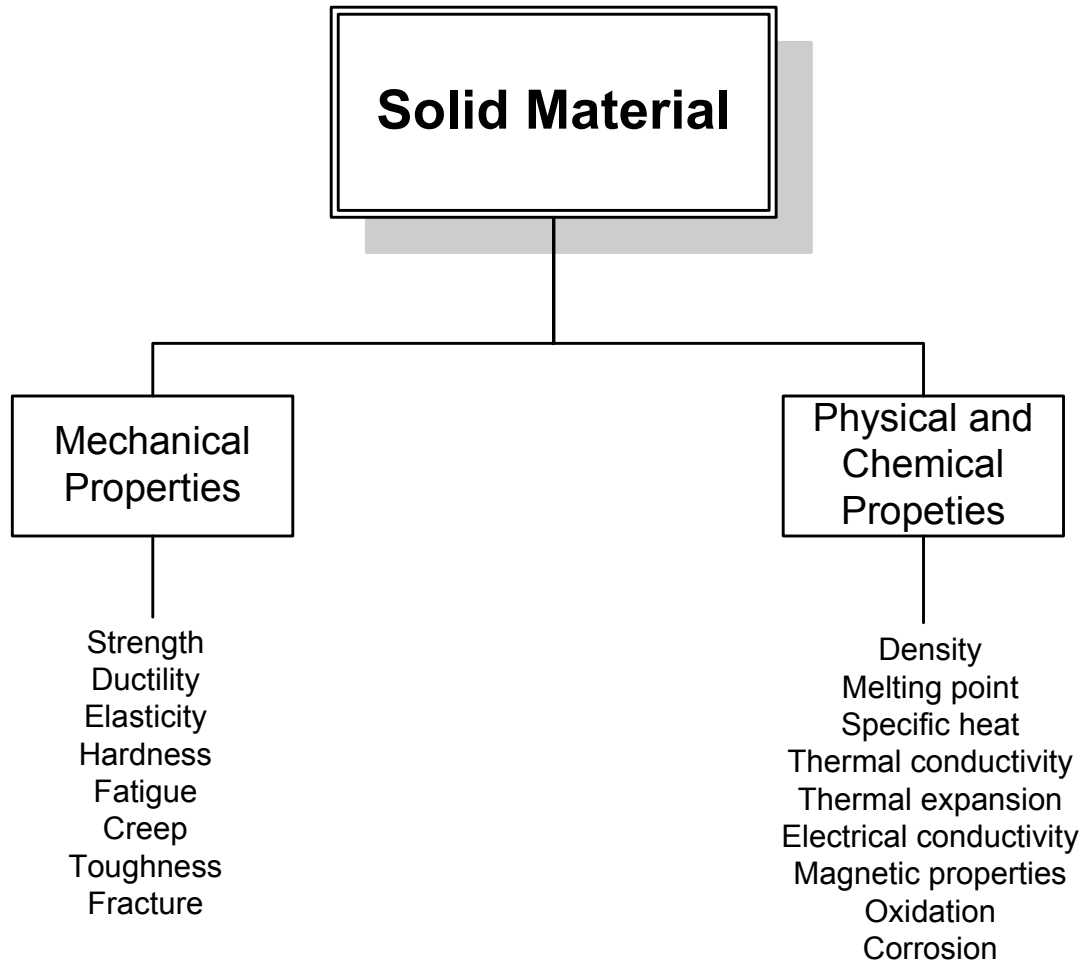


Figure 28 Material properties

The most suitable type of material for a particular solid material operand must satisfy all the property set of the operand. Because of the diverse nature of engineering materials, the functionality model developed in this work only makes provision for inclusion of the name of the material assigned to a given operand. The specific material properties are not included in this model and are expected to be obtained from a third party or engineering material reference library/manuals. Once these properties are obtained, they should be tested against the target operand material properties as defined in the attribute set of the operand.

4.1.5 Mass Properties

- Volume: 3-D Space occupied by solid operand. Computation depends on exact geometry.
- Mass: Density x Volume: In Newton's law of inertia, mass is a measure of a body's inertia – of its resistance to change in motion.
- Area: Computation depends on exact geometry.
- Moment of inertia (I): Computation depends on exact geometry. This is a rotational analog of inertia; it depends not only on mass itself but also on the distribution of mass in relation to the rotation axis. Unit – Kg.m². Rotational inertia depends on the mass of a body and on the distribution of mass about the rotation axis, and is given by:
$$I = \sum m_i r_i^2$$
 for a body consisting of discrete masses, and by $I = \int r^2 dm$ for a continuous distribution of matter.

4.1.6 Topology

During conceptual design, all solid operands assume a functionality bounding geometry consisting of functional points/regions. The functional points are located on the surface of the functionality boundary points. Functional points are equivalent to vertices, edges, or axis of solid components. To define the bounding geometry, the functional points are joined together to form a surface or wireframe of lines and end-points.

In this work, the initial product functionality is modeled using 2-D approach. This approach allows us to demonstrate the concepts developed here, which can then be extended in future to cover functionality modeling in 3-D.

Boundary Modeling (BM) Function:

BM functions are used to add, delete, or modify the lower entities of a solid, such as vertices, edges, and faces, in order to manipulate it directly. Points are first created, then edges are created by connecting the points, and finally surfaces are defined by the bounding edges. Because of the tedious nature of this modeling process, its use in this work is restricted to 2-D, in the X-Y plane.

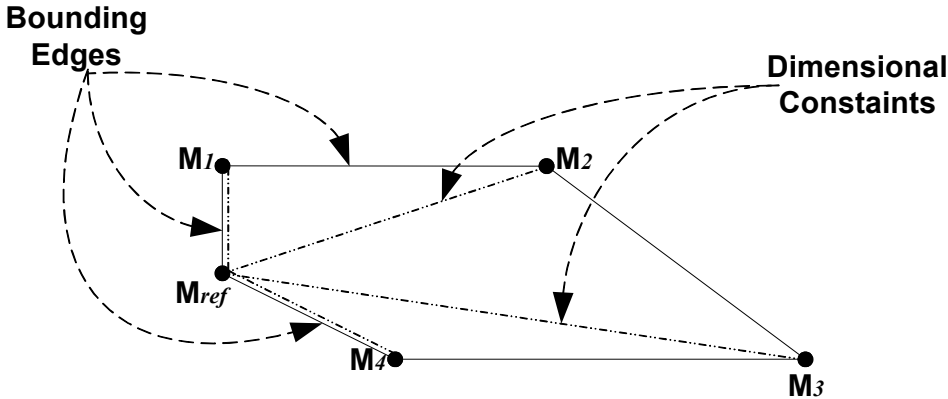


Figure 29 Boundary modeling in product conceptualization with *functional points* only

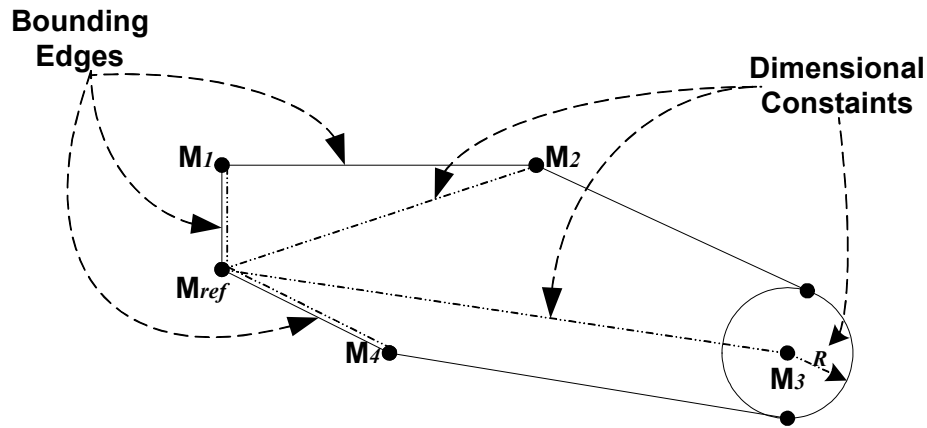


Figure 30 Boundary modeling in product conceptualization illustrating use of *functional points* (M_1 , M_2 , and M_4), *functional region* (M_3), and *reference functional point* (M_{ref})

The creation of 2-D bounding surface of a solid operand is illustrated in Figure 29 and Figure 30. In Figure 29, only functional points are used. These functional points translate to actual vertices of the bounding surface. These functional points are defined for the solid operand by defining the dimensional constraints on the points with respect to the reference point or to other existing functional points. The bounding surface is created by joining all the functional and reference points to form a closed curve (i.e. surface). Instead of functional points, a functionally active region might be used to define functionally relevant regions on the solid operand. For example, in Figure 30, marker M_3 is used to represent a circular region of radius R on the solid operand. Such regions might be used to represent features such as holes, slots, pockets, or some other features that are functionally essential for the proper working of the operand. The bounding surfaces and curves are generated by joining the functional points / regions to form the initial product form as shown in Figure 29 and Figure 30.

4.2 Energy Operand – Mechanical Energy Operand

Energy is the functionality operand that effects some work outcome. Energy may be from mechanical, thermal, electrical, magnetic, acoustic, radioactive, chemical, biological, optical, hydraulic, or pneumatic source. To demonstrate the concepts developed in this work, it will be assumed that all the energy that impact mechanical product functionality is present in mechanical form. For this assumption to hold, all energy sources (other than mechanical energy) are first applied to an energy converter, which transforms that energy into a mechanical form. Thus, energy converter (a specialized transform operation) is used as a primitive operation in this work. (see Figure 31) This assumption makes the system modular and extensible, each energy converter may easily be included in the system (as a functionality component) with all the applicable laws and principles that govern the transformation process.

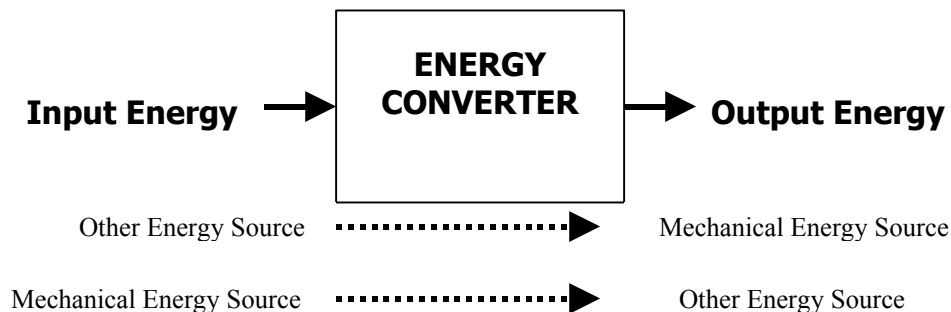


Figure 31 Mechanical energy converter

Some researchers ^[41] have considered energy as being composed of two components: *effort* and its *corollary*. They stated that an effort is any component of energy used to accomplish an intended purpose, while the corollary is a natural consequence of or accompaniment of effort. This model however will not support the view of energy as an operand in which its natural consequence can only be determined by the context under which it operates. That is, other entities with which it interacts with. These other operands together with the relevant relationships determine the context of the application. Hence, the definition of energy operand follows the same pattern used in the material operand, by defining all its attributes and nature. The effects of the operands (relationships) are considered under a different section that discusses the modeling of operand interactions.

Mechanical energy is associated with the spatial displacement of a material operand (machine/component) or the strain energy associated with the loading state of a material operand. For example, application of force at one end of a linkage in a cam mechanism results in a functional path being traced out at the other end of the linkage. Or the energy associated with the rotation of the transmission shaft of a car. Two forms of energy operands are associated with mechanical energy in the operation of mechanical systems. The operands are FORCE and TORQUE operands as illustrated in Figure 32. FORCE is classified as a primary operand as it is the primary energy associated with the functionality of mechanical systems. Hence, FORCE is the primary mechanical energy source in mechanical system. TORQUE operand on the other hand is secondary as it is derived from a FORCE operand. Thus the availability of TORQUE presupposes the existence of a FORCE as the primary source. Hence, TORQUE could be expressed in terms of the source force. The use of TORQUE as an

independent operand is for simplification of some complex interactions involving FORCE, where rotational effect of force is of primary concern.

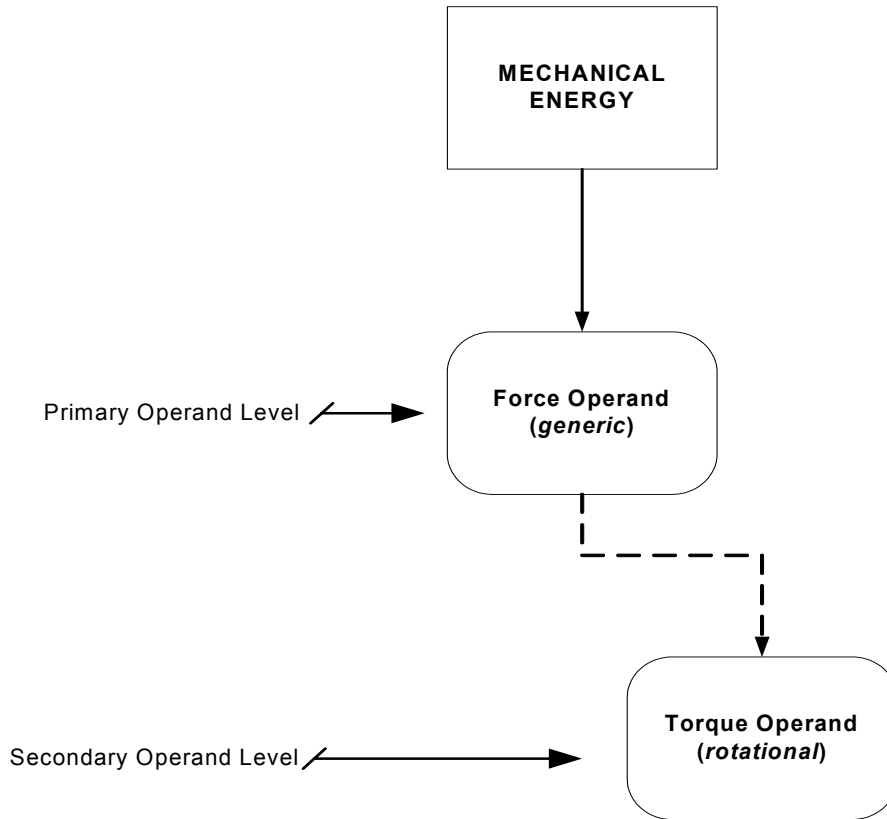


Figure 32 Mechanical energy operands

4.2.1 FORCE Operand

Force causes change in motion and/or change in the spatial relation of internal components of an object (material operands) resulting in stress and strain. In mechanical systems, varieties of forces are always acting on material operands. The individual forces acting on an object is called *interaction forces*, because they always involve other objects interacting with the object in question. For modeling and practical purposes, what matters is the net force

on an object, meaning the vector sum of all the interacting forces on an object. The effect of force relevant to functionality modeling of mechanical devices will be considered in more detail in Chapter 5, Functionality Relationships.

At present, physicists identify three basic forces – the *gravitational force*, the *electroweak force*, and the *color force*. Each of these subsumes other forces once considered distinct. There is still ongoing debate and research on the existence of a common unifying class for all forces. A more detailed discussion on the subject is presented by Wolfson, et al. [78]

- ***Gravitational force***: This attractive force acts between all matters. The most relevant form of this force in mechanical devices is the attraction of all matters (material operands) by earth's gravitational force in the form of the weight of objects. Thus, all components of a mechanical product, having mass exhibit this kind of force.
- ***Electroweak force***: This class of force subsumes electromagnetism and the so-called weak nuclear force. Virtually all the non-gravitational forces we encounter in everyday life are electromagnetic, including contact forces, friction, tension forces, the forces in your muscles, and the forces that hold ordinary matter together at the atomic and molecular level. The weak force is less obvious, but is important in helping determine which atomic nuclei are stable and which are radioactive.
- ***Color force***: Originally called the nuclear force and referred to as the force between individual protons and neutrons. However, protons and neutrons are composed of simpler particles called quarks. The force between the quarks is called the color forces.

The unit of force is Newton. The data structure of the FORCE operand used in this work is as shown in Figure 33. From the figure, it should be noted that a FORCE operand is modeled by: **FORCE {<source | kind> <mag, angle, point> <nature>}**, where source defines the source of force; kind defines the kind of force; nature defines the nature of force, while “**mag**” is the magnitude of force; and angle is the rotation from the Cartesian axis, and point is the 3-D coordinates of the point of application. Hence, the attributes of a FORCE operand is completely defined by **source, kind, mag, angle, point, and nature**. However, at the beginning of the design process (conceptualization) and functional definition, some of this information may not be known and hence incomplete. The model is however defined by noting the unspecified (unknown) attributes and neglecting them (or using default values / approximations) for any intermediate reasoning or computation. As design progress, these attributes are defined.

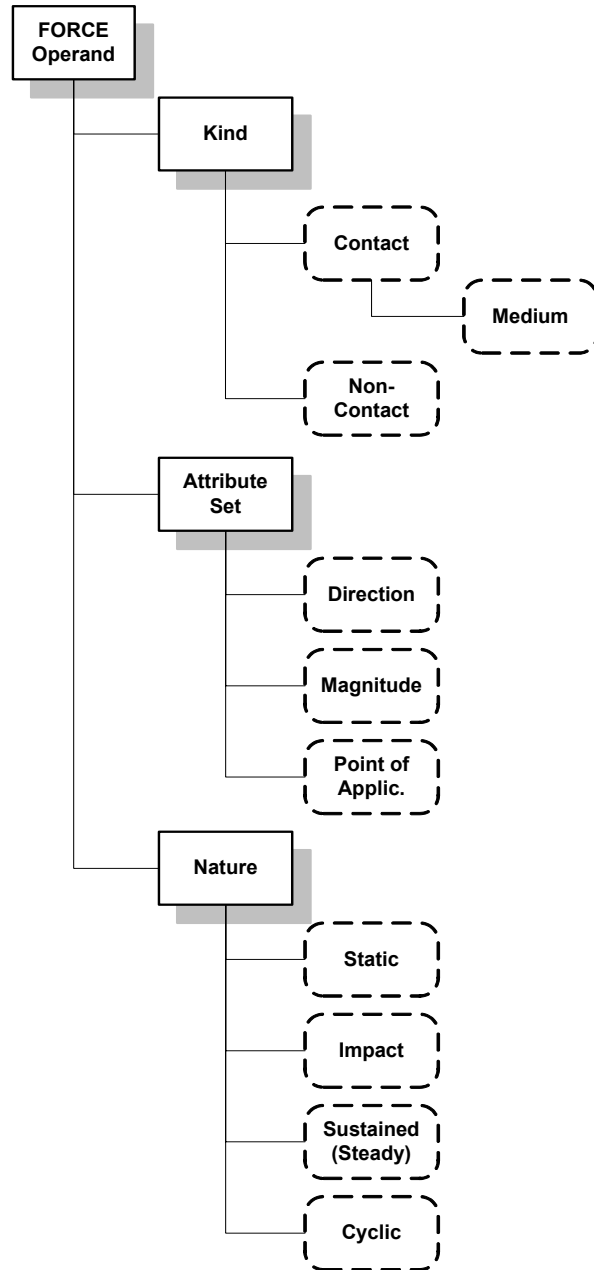


Figure 33 FORCE operand model

a) Kind of Force

Although the three fundamental forces are at the heart of all that happens in the physical world, in our modeling of FORCE as a functionality operand on a macroscopic scale, we do not

consider force at such a fundamental level – with the important exception of gravity. For the functionality model, force is classified into two: *contact force* and *action-at-a-distance force*.

- ***Contact force***: This is a force where one object exerts a force on another through direct contact. Results from direct contact with another body. For example, the upward force of a chair on a human body or the friction between two objects. An important implication of this is the existence of the transmission medium that exerts the force on the object.
- ***Non-Contact (Action-at-a-distance force)***: This is a force where the objects exert the force through seemingly non-contacting objects. Bodies do not have to be in contact for these forces to act. Examples of this class of force include gravitational forces, electric force, and magnetic force.

b) Attribute Set and Vector Representation

This component of the FORCE operand defines the quantitative composition of force. A knowledge of this set of attribute can be used to infer the impact or effect of the force on other operands involved in the realization of a given functionality task. Since force is a vector quantity, its set of attributes includes the following: *direction*, *magnitude*, and *point of application*.

The projection of a vector on a line is called a component of the vector in the direction of the line. In the rectangular coordinate system, these projections onto the major axis is called the rectangular components of the vectors, and are denoted by A_x , A_y , and A_z for the vector \mathbf{A} . The

magnitude of a vector can be written in terms of the vector's rectangular components. The magnitude of vector \mathbf{A} is given by:

$$|\mathbf{A}| = \sqrt{A_x^2 + A_y^2 + A_z^2}$$

Similarly, the direction can be specified in terms of the rectangular components. The angle between a force (\mathbf{A}) and a line parallel to the x-axis is given by, $\theta = \sin^{-1}\left(\frac{A_x}{|\mathbf{A}|}\right)$. The representation of force operand using the unit vector convention enables the functionality model of force operand to relate to the conventional design space in Cartesian coordinate system.

Two vectors defined to be equal if they have the same magnitude and the same direction. This definition is bound to introduce some ambiguity in the modeling of force operand for conceptual design systems. This ambiguity is resolved by also defining the *point of application* (or the spatial relationship) the force operand has with the solid operand. This definition is presented in Section 3.5, functionality relationships.

c) Nature

Any applied force may be classified with respect to time in the following ways:

- **Static force** – Force is gradually applied and equilibrium is reached in relatively short time. The structure experiences no dynamic effects.
- **Sustained force** – Force, such as the weight of a structure, is constant over a long time.
- **Impact force** – Force is rapidly applied. An impact force is usually attributed to an energy imparted to a system.
- **Cyclic force** – Force can vary and even reverse itself in sign and has a characteristic period with respect to time.

4.2.2 TORQUE Operand

This operand is the energy that results from rotation or torsion of a functionality object – solid material. In TORQUE energy operand, a component of the force causing rotation is inclined and at a distance from the axis of rotation. For example, in the design of an automobile, the designer may be interested in the transmission of a rotational energy through the car axle to the wheels. On the other hand, in the design of a power screwdriver, a designer may be interested in the transmission of rotational energy from the electric motor to the head of the screw/bolt through the spindle. For example, in Figure 18, rotational energy is transmitted through the shaft to the disc.

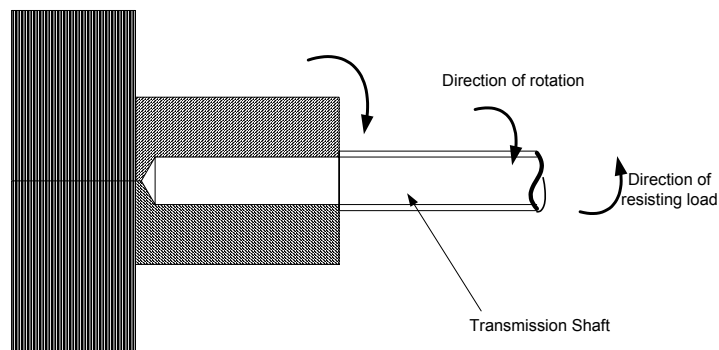


Figure 34 Rotational energy in a rotating shaft

Although Newton's second law, $\mathbf{F} = m\mathbf{a}$, governs all motion, a direct application of the law to every particle in a rotating object would be terribly cumbersome. Instead, an analogous law that deals with rotational quantities is developed to describe rotational effect of force.

Hence, TORQUE, which is a force analogy to rotational motion, is the effectiveness of a force in bringing about changes in rotational motion. Torque depends on the applied force and on how far from the rotation of the axis it's applied. Effectiveness of the force also depends on the direction in which it's applied; application at right angles to the line from the rotation axis to the force application point is most effective.

Based on the above considerations, torque (τ) is defined as the measure of the effectiveness of a given force in producing a change in rotational motion – as the product of the distance r from the rotation axis with the component of the force perpendicular to that axis. This is given as:

$$\tau = r F \sin \theta,$$

θ is the angle between the force vector and the vector r from the rotation axis to the force application point. The unit of torque is N-m. Torque, twisting force, is the rotational analog of force (or Newton's second law) with corresponding law of inertia given as:

$$\tau = I\alpha,$$

Where I is the rotational inertia or moment of inertia and α is the angular acceleration.

The direction of torque (and corresponding angular velocity) is defined with the *right-hand-rule* : if you curl the fingers of your right hand to follow the rotation, then your right thumb points in the direction of the torque and angular velocity. This is illustrated in Figure 35. In vector notation, torque is given by the cross product of the force (\mathbf{F}) and the vector r from the rotation axis of the force application point. Torque is defined by:

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}$$

The result of the equation completely defines torque as a vector. The direction is perpendicular to both vectors \mathbf{F} and \mathbf{r} . In modeling torque, it should be noted that both \mathbf{F} and \mathbf{r} are expressed in the unit vector format in the rectangular coordinate system. Hence, torque $\boldsymbol{\tau}$ is also defined in the unit vector format by $\boldsymbol{\tau} = \tau_x \mathbf{i} + \tau_y \mathbf{j} + \tau_z \mathbf{k}$, where τ_x , τ_y , and τ_z are the x, y, and z, components of $\boldsymbol{\tau}$ respectively. The magnitude of torque is given by $|\boldsymbol{\tau}| = \tau = r F \sin \theta$

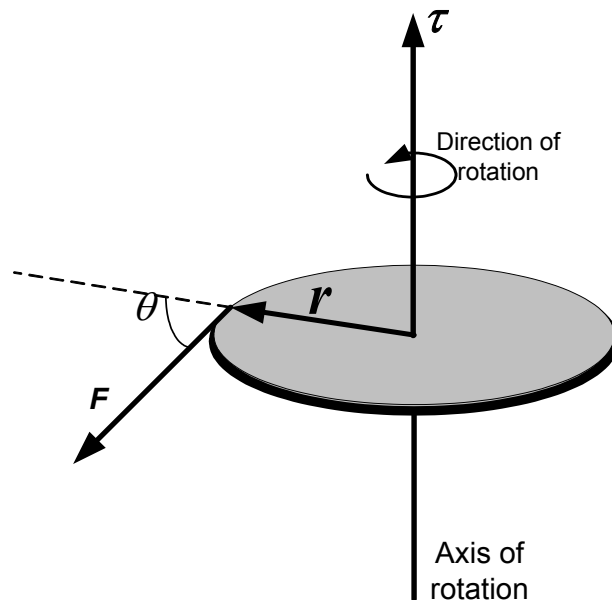


Figure 35 Direction of torque is given by the right hand rule

For modeling purposes, the TORQUE operand has a structure similar to that of FORCE operand. This structure is shown in Figure 36. Thus, the TORQUE operand model is described by specifying the *torque vector*, *source of force*, *axis of rotation*, and *nature of the torque*. The nature of the torque is directly related to the nature of the force that produces the torque. Hence,

the nature of torque may be static, cyclic, steady, impact, or some combination of any of the four.

From the figure, it should be noted that a TORQUE operand is modeled by: **TORQUE** {<**source** | **kind**> <**mag, angle, axis**> <**nature**> }, where *source* defines the source of torque; *kind* defines the kind of torque; *nature* defines the nature of torque, while *mag* is the magnitude of torque; and angle is the rotation from the Cartesian axis, and axis is the axis of rotation of the torque. The *kind* attribute specifies whether the *source force* is a contact or a non-contact force. The attributes of a TORQUE operand is completely defined by **source, kind, mag, angle, axis,** and **nature**. However, at the beginning of the design process (conceptualization) and functional definition, some of these information may not be known and hence incomplete. The model is however defined by noting the unspecified (unknown) attributes and neglecting them (or using default values / approximations) for any intermediate reasoning or computation. As design progress, these attributes are defined.

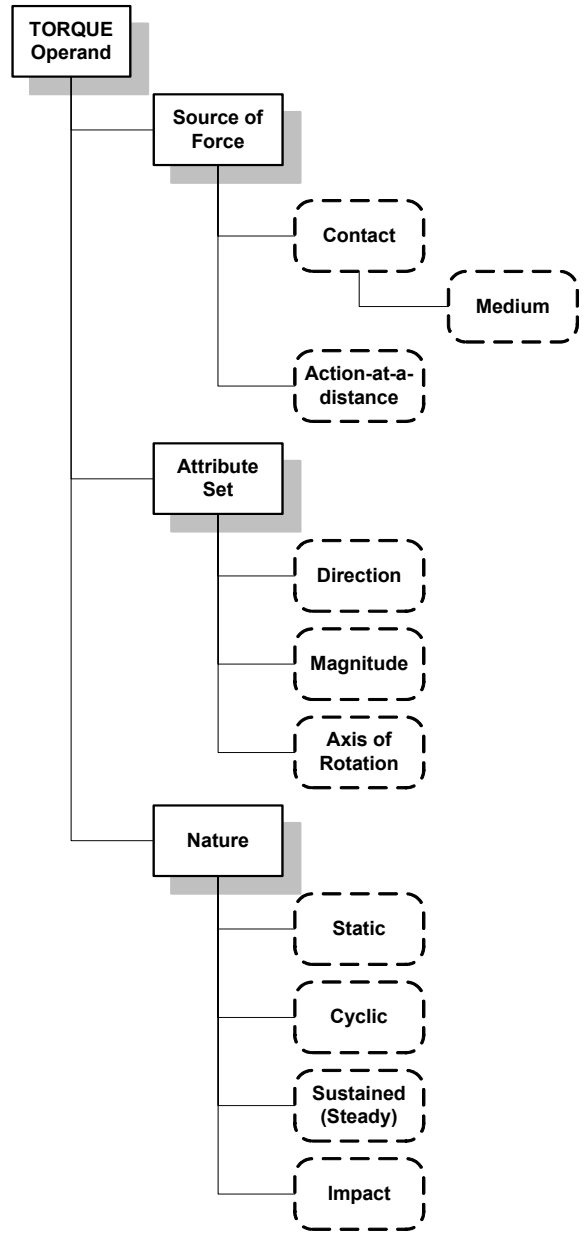


Figure 36 TORQUE operand model

5.0 FUNCTIONALITY RELATIONS AND STATES

5.1 Functionality Relations

To accomplish a mechanical function, a functionality operation must define functionality operand relationships and impose constraints on functionality operands, which may result in some additional implications. The functionality operator accomplishes this. Thus, the operator provides the needed intelligence that leads to the realization of a given task. A functionality operator defines the relationship between operands in the performance of a functionality operation. Hence, the operator defines how the attributes of the functionality operands are related. This is illustrated in Figure 37.

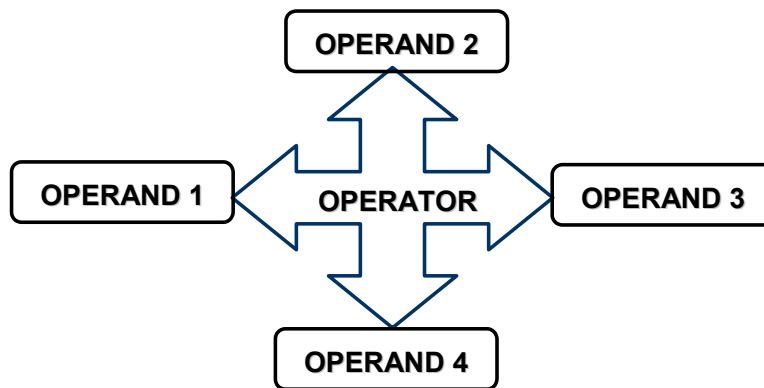


Figure 37 Functionality operator linking four operands

For example, consider a case where two solid material operands are in rubbing contact. The frictional coefficient and the surface wear rate, are some of the operators that related the surface attributes of the two solids in sliding contact. The operands are the two solid materials

in contact: Operand-A, and Operand-B. The functional regions of interest are the two rubbing surfaces. The other attributes that are relevant in this interaction are the surface characteristics (which determines the coefficient of friction, μ) and the mass of the non-fixed operand (Operand-A). The frictional force (F) between the two-sliding/rubbing surfaces is given by:

$$F = \mu \bullet mg,$$

where μ is the coefficient of surface friction that relates the two rubbing surfaces, m is the mass of the movable SOLID operand, and g is the gravitational constant. In this work, the operators are defined by equations in the form of functionality relationships.

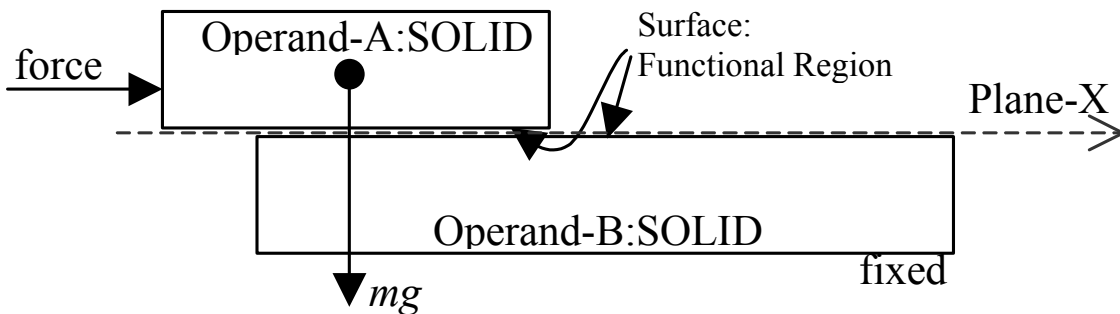


Figure 38 Solid operands in sliding contact

Other operators include spatial linkage operator, that must specify that the two surfaces must maintain an against relationship with B fixed and A allowed to have a plane-x DoF. Hence, the need to consider friction and wear-rate. The operator-operand-attribute listing (in Figure 39) shows the relevant interactions in the example of Figure 38.

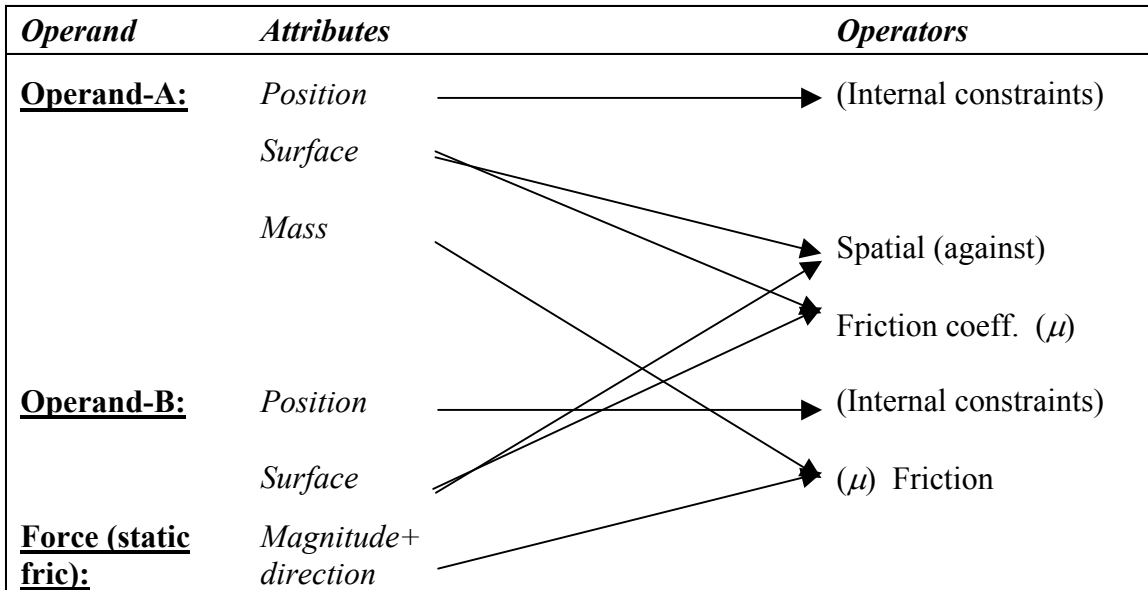


Figure 39 Operator-operand-attribute relation for sliding contact example

One important development in this work is the realization that operands only interact through their attributes. Hence, the attributes are the contact points between operands. A functionality relationship can assume any of the following forms: spatial; physical aspect; technological constraints; or user preferences.

- *Physical aspects*: the physical processes (e.g. energy storage in a compressed spring), interactions (e.g. friction on rubbing surfaces), properties or behavior of something (e.g. plastic deformation or elastic property of materials);
- *Spatial relations*: location of objects in space (e.g. aligned relationship of two solid operands);
- *Technological constraints*: tolerance (e.g. machining tolerance), surface finish, etc; and
- *User preferences*: color, texture, etc.

It should be noted that complex functionalities could be formed by bringing together smaller (basic) functions. This combination is realized by defining relationships between their interacting operands. Sub-functionality operands that are involved in relationship or interaction with other functionalities are called *coupling operands*. Sub-functionalities are also referred to as primitives. They define how functionalities are combined to form compound functionalities. They are equivalent to mating features in feature-based design. In defining the coupling relationships, no causal relationship is assumed. Hence, a more generalized interaction can be described for the coupling operands. Figure 40 depicts a compound functionality that is formed from three primitives, A, B, and C. To form the compound functionality, primitive A is coupled to B and C separately. This coupling is represented by the dashed lines. However, coupling is only possible through functionality operand. This inter-primitive coupling through coupling operands is illustrated in Figure 41. This figure also illustrates an inter-operand coupling within each primitive structure. It is through this maze of coupling operations that it is possible to form more complex mechanical functional structures. This coupling is defined through the inter-operand coupling matrix relations.

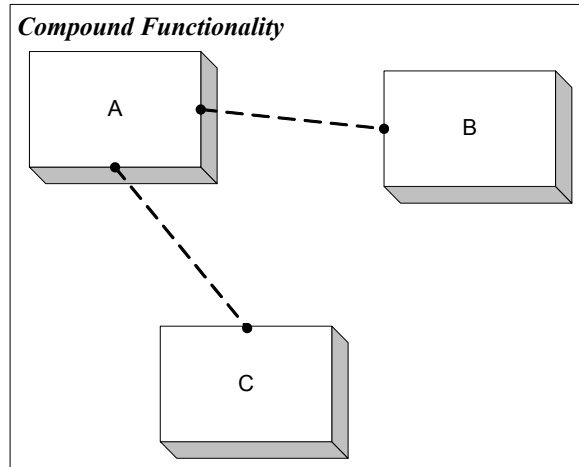


Figure 40 A compound functionality showing inter-primitive interaction

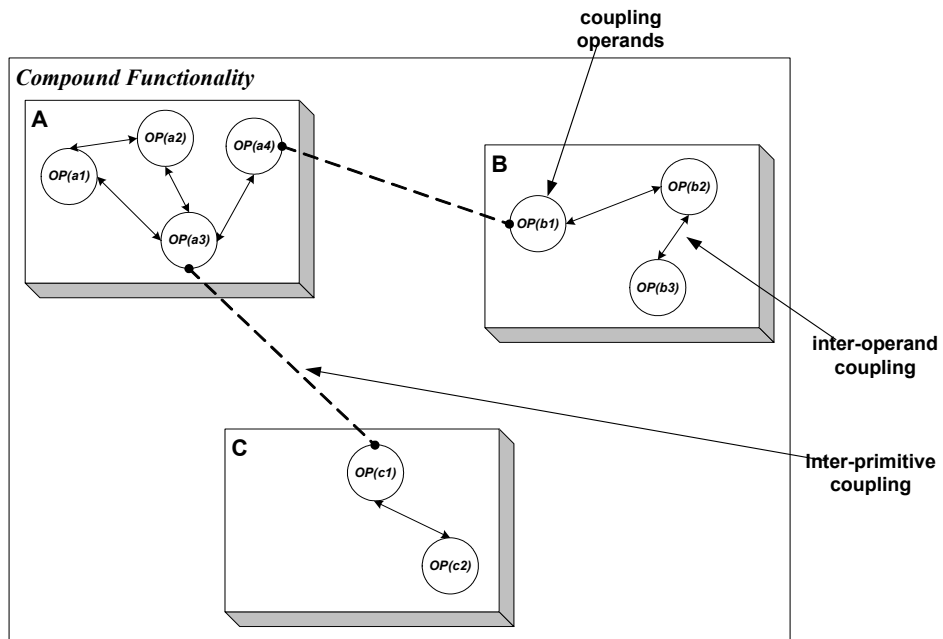


Figure 41 A compound functionality showing inter-primitive interaction between the coupling operands

5.1.1 Coupling Bond (CB)

Engineering relations are constructed by specifying the functionality constraints and relationships between operands, after each operand has been defined with its associated attribute. Functional relationships are built between those attributes (including functional markers/features) on the operands by attaching some sort of linkage. This linkage defines the functionality engineering relations between operand and links separate functionalities together.

Since the primary relations of concern at this step are the coupling relations, the concept of *coupling bonds* is introduced. A *coupling bond* (CB) is created once two coupling attributes on different operands are selected and linked with each other. In addition, the coupling conditions are attached during this linkage. The structure of coupling bonds is shown in Figure 42.

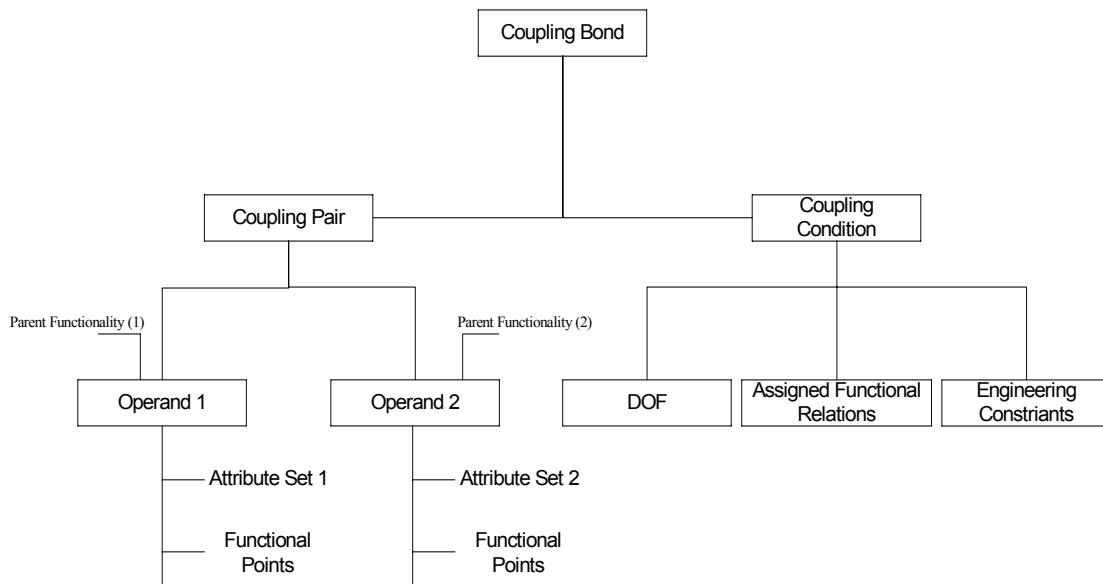


Figure 42 Functionality operand coupling bond

There are two dominant groups in one coupling bond: coupling pair and coupling conditions. Coupling pair contains two coupling operands involved in the relationship. The parental relation is used to record their *parent* functionalities. The functionality hierarchy and the connectivity graph of the whole product can then be created based on these operand-to-operand connection relations. From coupling operands, the system traces back what are their original functionalities and inherits the implied attributes and constraints.

The coupling condition defines the nature of interaction between operands. It includes the type of assigned functional relationships, required degrees of freedom (DoF) and some other engineering constraints.

- The *functional relationship specifies* the required relations that must be maintained between attributes of the interacting operands in order to satisfy the given functionality. For instance, in the conversion of FORCE (F) to TORQUE (T), this relationship is defined by the perpendicular distance (*l*) between the point of application of force and the axis of rotation as defined in the following equation: $T = F \times l$.
- The *DoF* is the desired degree of freedom to be maintained by the involved operands in order to remain functionally correct. The possible types of degree of freedom include:
 - ↳ *lin_n*: linear translation along n axis (e.g. application of FORCE along an axis).

- ↳ *rot_n*: rotation about n axis (e.g. rotation of a car wheel (SOLID operand) about an axle).
 - ↳ *cir_n*: translating along a circle with n axis.
 - ↳ *plane_n, cyl_n, sph*: translating along a planar, cylindrical and spherical surface respectively.
 - ↳ *roll_lin_n*: rolling along a corner.
- *Engineering constraints* are the constraints imposed on the operand attributes and relations to remain functionally correct. For instance, in the case of conversion of FORCE to TORQUE example, if the perpendicular distance is constrained to be within a certain limit (e.g. $3.80 \text{ cm} \leq l \leq 5.25 \text{ cm}$), this will fall within the engineering constraint coupling condition. Engineering constraints could also be applied to attributes of operands. For example, the mass (m) of a solid operand could be required to fall within a given limit (e.g. $15.25 \text{ g} \leq m \leq 20.00 \text{ g}$). Other issues that fall within this category are the tolerance allowance, and physical effect/implication constraints.

The concept of CB is illustrated with the example in Figure 43, showing two blocks in rubbing contact. The DoF on block B is fixed while block A is allowed to have a plane-X degree of freedom. The other components of the coupling bond between the two operands (Block A and Block B) are shown in Figure 44.

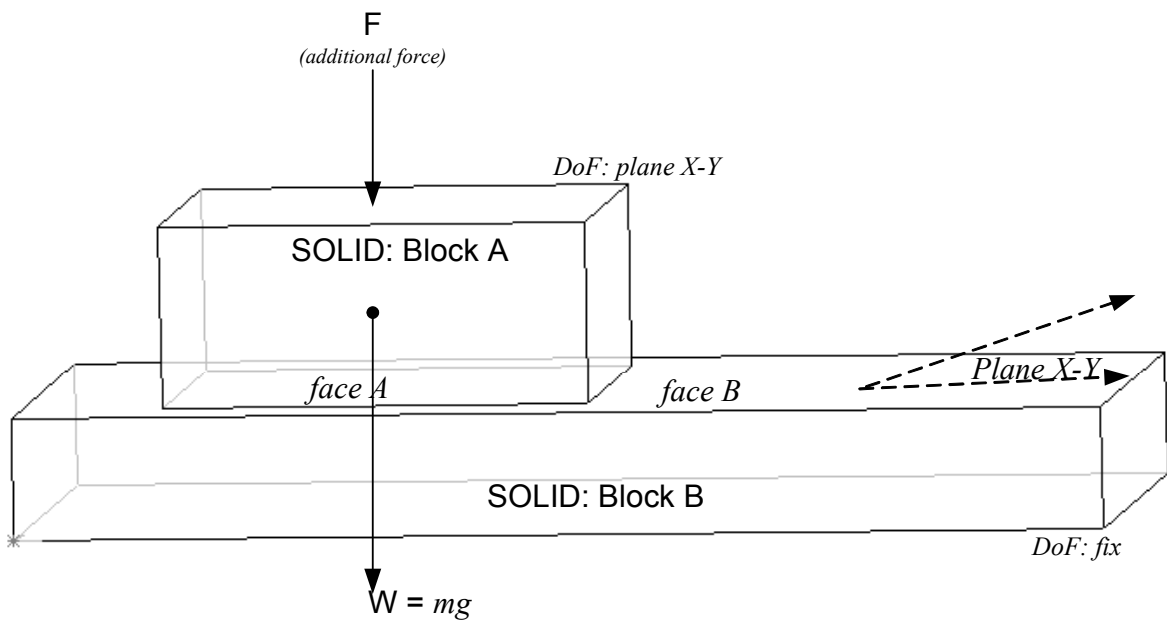


Figure 43 Two solid blocks in sliding contact

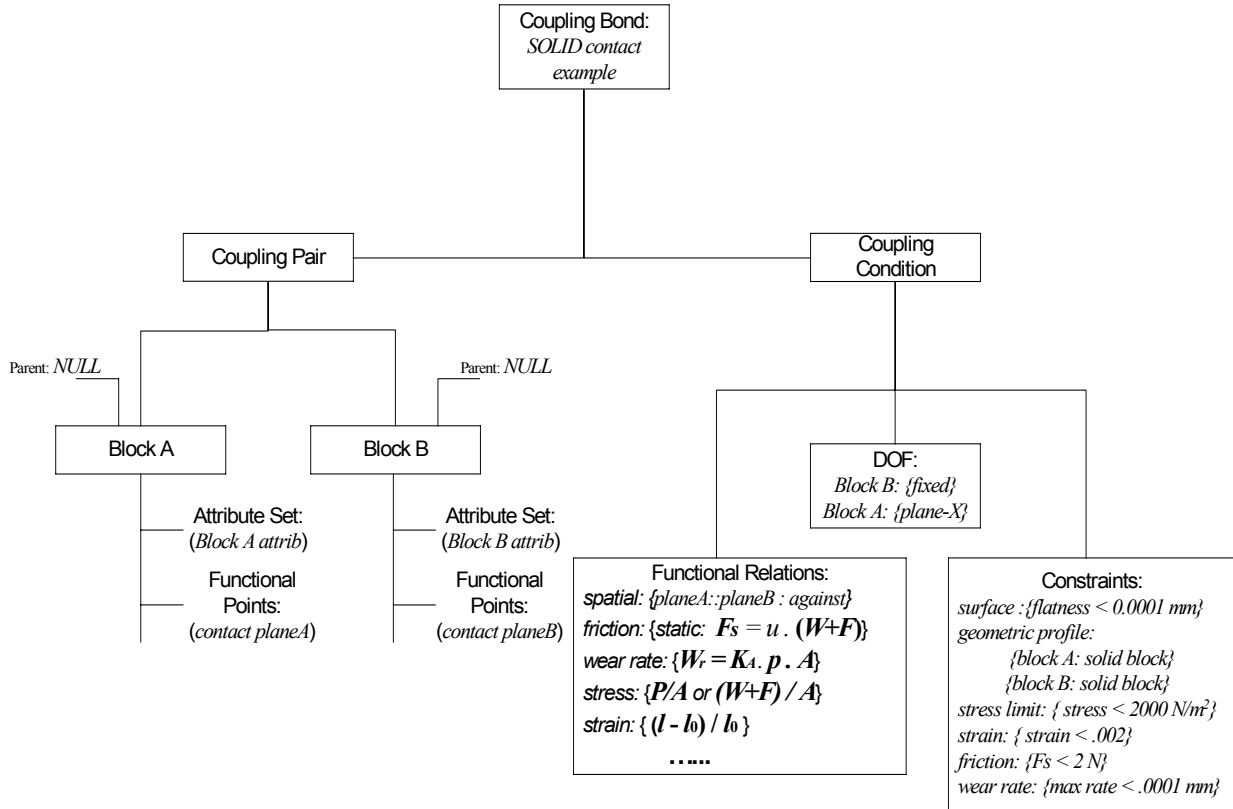


Figure 44 Coupling bond for the two solid blocks in sliding contact

5.1.2 Representation of Functionality Relations

Functionality relations are defined as a set of active interactions between functionality operands. This interaction is realized using coupling bonds. Given a set of operands (O_i) in a 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_{iq} = attribute set for functionality operand o_{iq}

With the above definition of operand set, functionality relationship is defined by a functional relationship set (\mathbf{R}_i) as given in Equation 7.

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

Equation 7

Where

- \mathbf{O}_i = set of operands in functionality operation i .
- r_{ijk} = coupling bond between functionality operands j and k .
- j, k = functionality operand indices

Since the coupling bond (r_{ijk}) defines the active relationships between the functionality operands j and k , \mathbf{R}_i is effectively a set of coupling bonds. The functionality operands (\mathbf{o}_{iq}) are the interacting functionality operands (including material and energy operands and the coupling operands in lower level functionality operation [i.e. sub-functionalities]). An operand can interact with itself in what is defined as a *recursive* interaction/coupling. This interaction defines all the internal of self-constraints imposed on the operand. That is r_{iji} and r_{ikk} are the constraints on the attributes of the operand j and k respectively. This extension allows us to provide a unique means of mapping every relation to downstream design activities.

A matrix called the inter-operand relationship matrix is used to define the possible interactions between each set of operand. The elements of the operand relationship matrices are defined by domain experts and stored in the knowledge base of the designer system or at design time by the designer. Each element r_{ij} is a set of relationships that hold for an interaction

between items on the i th row and j th column. Each column of the operand relationship matrix is called an operand vector and it is used to derive the relationships of inter-item coupling in a functionality operation. The derivation and operation of the operand vector is described below.

Table 4 Inter-operand relationship matrix.

Operand	<i>SOLID</i>	<i>FORCE</i>	<i>TORQUE</i>
<i>SOLID</i>	friction, DoF restriction, force_transmission, spatial relation	motion (acceleration, displacement, impulse, velocity); momentum; friction; stress; strain.	moment; motion (acceleration, displacement, velocity); momentum; friction; stress; torsion.
<i>FORCE</i>		resultant, orientation, transmission, effect, nature.	resultant, orientation, effect, nature.
<i>TORQUE</i>			resultant, orientation, transmission, effect, nature.

5.1.3 Derivation of Functionality Relations

Since functionality operations are accomplished through operand coupling, we can define the set of relationships and constraints for functionality operation by deriving its coupling relationships. To derive the relationships of the operand couplings, we define the inter-operand coupling matrix corresponding to the functionality couplings.

X = inter-operand coupling bond matrix

In the \mathbf{X} matrix, each element in the vector represents a set of valid relationships, constraints, and implications required in the coupling of an operand to other operands. For example, assuming we are transforming force to torque, we might define the relationship between the input force and the output torque as $T = F.l$, where T = torque, F = force, and l = perpendicular distance between the force and the axis (as the relationship between T and F).

In computer modeling, the relationship set is represented by a coupling bond matrix, *functionality coupling bond (CB) matrix*, $\mathbf{X}_{m \times n}$. This functionality CB matrix is given by:

$$X_{m \times n} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

Where,

x_{ij} = is the coupling bond between operands i and j .

Since a combination/interaction need to be defined between an operand and every other operand, the number of columns and number of rows in relationship matrix are equal. Hence, $\mathbf{X}_{m \times n}$ is a square matrix of the functionality coupling bonds. It is also symmetric with $x_{jk} = x_{kj}$, as they represent the same coupling. This is because no causal relationship is assumed in the model, hence the operators are associative. The diagonal ($x_{11}, x_{22}, \dots, x_{jj}, \dots, x_{NN}$) are the functional constraints imposed on the attributes of operands (e.g. maximum strength or maximum mass) The functional (inter-operand) relationship is between the operands within the same functionality operation.

The coupling bond has three components that constitute the coupling conditions: DoF, relations, and constraints. The coupling bond is defined by including these three components in the coupling bond equation as shown in Equation 8.

$$x_{ij} = \{r_{ij}, c_{ij}, d_{ij}\} \otimes y_{ij}$$

Equation 8

Where

r_{ij} = relation between operands i and j .

c_{ij} = constraint on relation between operands i and j .

d_{ij} = degree of freedom on relation between operands i and j .

$$y_{ij} = \begin{cases} 1 & \text{if coupling exist between operands } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

\otimes is an element multiplication, where given two matrices, [A] and [B];

$$[A] \otimes [B] = [a_{ij} \times b_{ij}], \forall_{ij}, a_{ij} \in [A], \text{ and } b_{ij} \in [B].$$

The entire set of coupling bonds in a functionality operation can then be represented by the coupling bond matrix as follows:

$$\mathbf{X} = [x_{ij}], \text{ where } i = 1, 2, 3, \dots, N; \text{ and } j = 1, 2, 3, \dots, N$$

Given as,

$$\mathbf{R} = [r_{ij}] \text{ and } \mathbf{I} = [y_{ij}]$$

subject to, $\mathbf{C} = [c_{ij}]$, $\mathbf{D} = [d_{ij}]$

X is called the coupling bond matrix, I the linkage matrix, R the relations matrix, C the constraint matrix, and D DoF matrix. The coupling bond matrix is modeled by an object representing the coupling bonds between operands.

In a full matrix notation, X is given by:

$$\begin{matrix} & OP_1 & OP_2 & \cdots & OP_N \\ OP_1 & \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \end{bmatrix} \\ OP_2 & \begin{bmatrix} x_{21} & x_{22} & \cdots & x_{2N} \end{bmatrix} \\ \vdots & \begin{bmatrix} \vdots & \vdots & \ddots & \vdots \end{bmatrix} \\ OP_N & \begin{bmatrix} x_{N1} & x_{N2} & \cdots & x_{NN} \end{bmatrix} \end{matrix}$$

Where:

OP_k ($k = 1, 2, \dots, N$) are the functionality operands.

x_{ij} ($i, j = 1, 2, \dots, N$) are the coupling bounds between operand OP_i and operand OP_j .

I is a zero-one square linkage matrix representing the valid linkage interactions for the given coupling operation. The I (linkage matrix) is derived by entering a one for each matrix element where a valid coupling exists between the operands and a zero is entered otherwise. The I matrix is symmetric and hence is modeled in a computer by representing the upper triangle of the matrix by modifying the original definition of I as shown below:

$$I_{ij} = \begin{cases} 1 & \text{if (coupling exist between operands } i \text{ and } j) \text{ and } (i \geq j) \\ 0 & \text{otherwise} \end{cases}$$

Because of the above assumption of symmetry and subsequent modification of I to an upper triangular zero-one matrix, the coupling bond is also an upper triangular matrix as shown below.

$$\begin{matrix} & OP_1 & OP_2 & \cdots & OP_N \\ OP_1 & \left[\begin{array}{cccc} x_{11} & x_{12} & \cdots & x_{1N} \\ 0 & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_{NN} \end{array} \right] \\ OP_2 & & & & \\ \vdots & & & & \\ OP_N & & & & \end{matrix}$$

\mathbf{R} is the operand relations' matrix defining all the valid cross-operand relationships and implications between operands in the functionality operation. The entries in \mathbf{R} matrix are defined a priori (and stored in the system database or knowledge base) or as design progress by the designer with some knowledge on the possible interaction between operands. During coupling, only the relevant entries are selected from the knowledge-base and added to those defined during design to form the relations matrix.

The size of the \mathbf{I} matrix is determined by the number of interacting operands. For example, consider the functionality primitive depicted in Figure 45. Since there are 3 interacting operands, \mathbf{I} is a 3x3 square matrix. The other variables in the equation are derived as follows for the example.

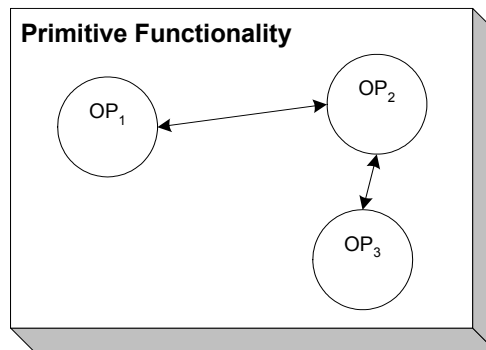


Figure 45 Functionality primitive A

The I matrix is derived by entering a one for each matrix element where a valid coupling exists between the operands and a zero is entered otherwise. That is,

$$I_{ij} = \begin{cases} 1 & \text{if (coupling exist between operands } i \text{ and } j) \text{ and } (i \geq j) \\ 0 & \text{otherwise} \end{cases}$$

Hence, the I matrix is given by:

$$I = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

The R matrix is formed by pulling all the valid relationships in the system involving operands a_1 , a_2 , and a_3 . Thus, R is given by:

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \quad \text{where } R_{ij} \text{ is the valid relationships involving operands } i \text{ and } j; i, j = 1,$$

2, 3.

The C matrix is formed by pulling all the valid constraints in the system involving operands i and j ; $i, j = 1, 2, 3$. Thus, C is given by:

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \quad \text{Where, } C_{ij} \text{ is the valid relationships involving operands } i \text{ and } j; i, j =$$

1, 2, 3.

The D matrix is formed by pulling all the valid DoF restrictions in the system involving operands i and j ; $i, j = 1, 2, 3$. Thus, D is given by:

$$D = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} \quad \text{Where } D_{ij} \text{ is the valid relationships involving operands } i \text{ and } j; i, j =$$

1, 2, 3.

Thus, X matrix is given by:

$$X = \left\{ \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \quad s.t. \quad \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}, \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} \right\} \otimes \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Equation 9

Note: the symbol “ \otimes ” is an element multiplication operator (defined in Equation 8 on Page 116).

$$X = \left\{ \begin{bmatrix} R_{11} & R_{12} & 0 \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix} \text{ s.t. } \begin{bmatrix} C_{11} & C_{12} & 0 \\ 0 & C_{22} & C_{23} \\ 0 & 0 & C_{33} \end{bmatrix}, \begin{bmatrix} D_{11} & D_{12} & 0 \\ 0 & D_{22} & D_{23} \\ 0 & 0 & D_{33} \end{bmatrix} \right\}$$

Note that the matrix X in Equation 9 defines a set of coupling bonds that are involved in the functionality operation. Thus, it gives the set of relationships, constraints, and DoF that must hold in the inter-operand coupling for the functionality to achieve the specified task.

5.2 Functionality States

In accomplishing a given task, a functionality operation might be expected to transition from one state to another. A state is an instant manifestation of the functionality object's condition in process of achieving the functional objective. This manifestation is described in terms of the instantaneous values of the operand attributes and the instantaneous coupling condition present in the functionality operation. As an illustration, the possible states of a brake rotor are (assuming discrete states) shown in Figure 46. In the example, the brake rotor is defined by the four distinct states they assume: brake idle; brake idle and wheel in motion; brake engaged and wheel stationary; and brake engaged and wheel in motion.

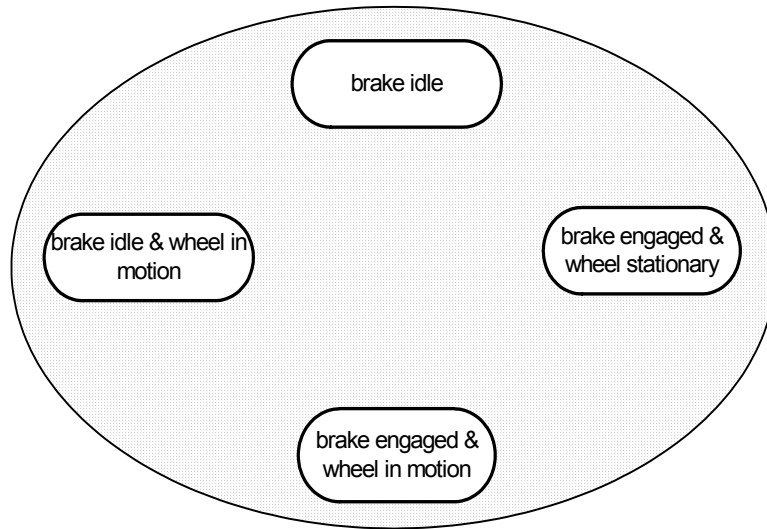


Figure 46 Brake rotor states

Functionality states are important because they may be necessary in determining the exact nature of coupling relations between operands. For instance, consider the pulley system shown in Figure 47. The functionality of the system is accomplished when the support changes from NO_LOAD condition to MAX_LOAD condition corresponding to the load resting wholly on the support beam. This change in state is accomplished by having a spatial change in the position of the loading component. In the case of MAX_LOAD condition, the functionality operation has to contend with the stress-strain values imposed on the supporting beam. At the same time, the tension (tug) in the rope is decreased.

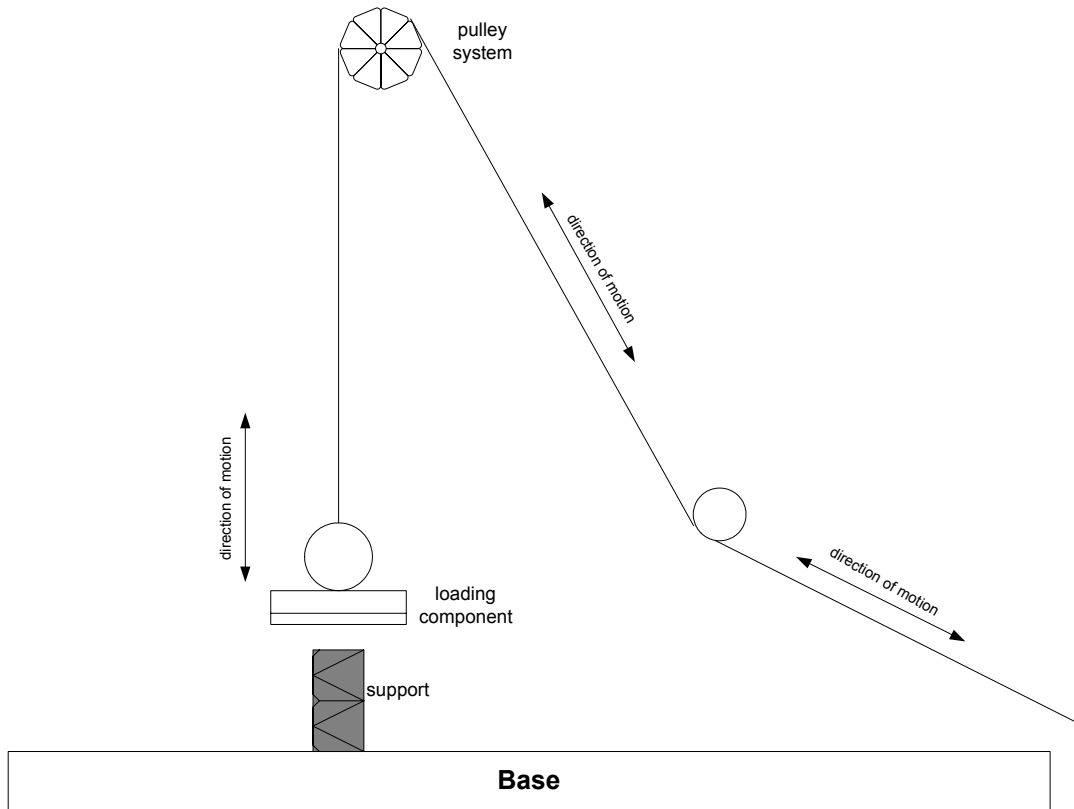


Figure 47 A pulley system

The state of a system may either be considered as *continuous state* or *discrete state* depending on the nature of the transition path. In this work and in most engineering systems, discrete approximations is assumed. This is necessary because of the complexity of some continuous systems. Usually a numerical approximation technique is used for this approximation. A discrete system generally approaches the continuous state if the discretization interval (δ) is successively decreased, reaching continuous state at the limit $\delta \rightarrow 0$ (where δ is the discretization interval).

The state of a functionality object is defined by considering the various possible values its time-varying attributes might assume. For instance, the possible states of a load-bearing

structure (such as support beam in Figure 47) could range from *NO_LOAD* to *MAX_LOAD* condition. It is important to note that the system state is defined by the instantaneous values of attributes of its constituent components (operands). Consequently, the functionality state is defined by considering the states of each individual operand. Hence, the state of a functionality operand, q can be represented mathematically by a state set, S_i as shown in Equation 10.

$$S_i = \{s_{iq} \mid s_{iq} \in S\}$$

Equation 10

Where i is the functionality operation index and s_{iq} is the state of operand q in functionality operation i . Hence, operands may have more than one functionality state. The functionality operand state is given by Equation 11.

$$s_{iq} = \{(a_{iqs}, v_{iqs}) \mid a_{iqs} \in A_{iq}, v_{iqs} \in V_{[aiqs]}\}$$

Equation 11

Where

A_{iq} = attribute set for functionality operand o_{iq}

$V_{[aiqs]}$ = set of possible values (range) of attribute a_{iqs}

In this work, attribute states are assumed discrete finite points including only upper and lower values. Hence, each attribute can have only two states: upper bound and lower bound. Consequently, for each operand, an array of value sets corresponding to each attribute is defined during the design process. These states are used as evaluation parameters during the design analysis and verification phases of product development.

In this work, the particular sequence or state transitions are neglected. Emphasis is placed on a combination of states instead of the path (transition) taken to attain the state or events that trigger the attainment of such state. Hence, states are modeled separately with transitions (i.e. causes of change of state) neglected.

Consider a functionality operation consisting of three operands (A, B, and C) each with only one state variant attribute. See Figure 48. If each operand can assume only two states: UPPER and LOWER states, the total number of possible states in the functionality operation is given by:

$$\text{Number of states} = \#S_A * \#S_B * \#S_C$$

Where

$\#S_i$ is the number of states in operand i

Hence,

$$\begin{aligned} \text{Number of functionality states} &= 2 * 2 * 2 \\ &= 8 \text{ distinct states} \end{aligned}$$

The distinct states are listed in Table 5.

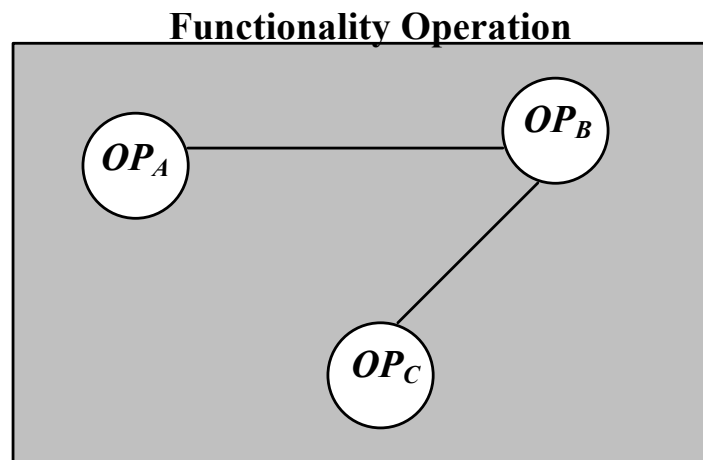


Figure 48 A three-operand functionality operation

Table 5 Listing of functionality operation states

	Operand A	Operand B	Operand C
State 1:	L	L	L
State 2:	L	L	U
State 3:	L	U	L
State 4:	L	U	U
State 5:	U	L	L
State 6:	U	L	U
State 7:	U	U	L
State 8:	U	U	U

*L = lower bound state and U = upper bound state

Generally, in a functionality operation with operands having binary state (UPPER and LOWER limits) attribute value set, the number of possible system states is:

$$\text{Number of states} = 2^{A_1} * 2^{A_2} * \dots * 2^{A_n}$$

$$= \prod_{i=1}^n 2^{A_i} \quad \text{where, } A_i = \text{attribute } i, i \text{ is the attribute index.}$$

A more general expression for the number of functionality states in a system with an arbitrary number of states per attribute is given by:

Number of states = $\prod_{i=1}^n S_{A_i}$ where, i is the attribute index and A_i is attribute i , S_{A_i} is

the number of states of attribute A_i .

In designing a product, each operand must have at least one or more states defined. However, in this work, a maximum of two states is allowed for each operand attribute.

5.3 Functionality Modeling Repository

To demonstrate the concept of knowledge reuse, a set of commonly used mechanical functionalities are described. These mechanical functionalities are referred to as functionality primitives (primitives for short). The primitives are BASIC mechanical functionalities located at the lowest level abstraction in the hierarchical decomposition of complex mechanical functionality operations. Each primitive performs a specific mechanical function that contributes directly or indirectly (as in the case of sub-functions) towards the realization of the overall product function.

While feature-based design is widely used in the industry and offers the possibility of design re-use and creation of design libraries, it presupposes that the designer has already accomplished the conceptual design phase and has already mapped certain functionality to features. This assumption makes feature-based design inadequate for conceptual design (as it assumes that the designer understands the relationship between function and form). To use the

tremendous advantage offered by modern computer tools to aid conceptual design, a more promising approach of functionality-based design (FbD) is proposed.

The advantage of the approach developed in this research is that it offers a convenient way for design reuse. This is because design intents can be modeled as functionality and stored in the design libraries for future reuse. In addition, functionality operation may impose relevant downstream design consideration upfront during the design process. Consequently, some benefits of concurrent engineering, design for manufacturing / assembly (DFM/A), and design for X (DFX) may be incorporated during the design conceptualization. For example, based on the available machining centers or company standards, one can impose a certain level of tolerance on the dimension of certain classes of operands such as SOLIDs.

The functionality-based design approach intends to achieve a goal similar to that of feature-based design approach. A set of desirable attributes can be offered by use of functionality primitive and functionality modeling knowledge reuse. It offers the following convenient properties:

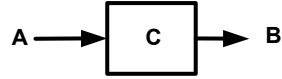
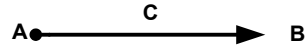
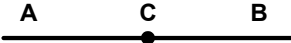
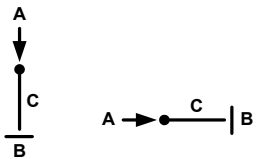
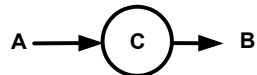
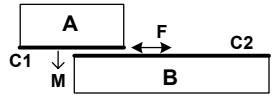
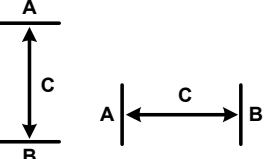
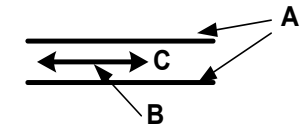
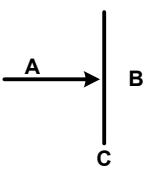
- Recurring product functionalities can be modeled as a primitive, and used as a repository of reusable product knowledge that may be related to a particular form or geometric pattern.
- Specific products can be modeled through their constituent functionality primitives, providing a more natural basis of interaction with the designer than mere geometric models by providing a direct relationship between form and function.

- Design and operational knowledge can be associated with functionality primitives, and accessed to determine the feasibility and producibility of functionality and the associated designed object or for planning its actual deployment.

Nine commonly used mechanical functions are described as functionality primitives that may be deployed in a design repository system. These primitives are fundamental in nature and may be used to build other complex functionality objects for mechanical devices. These primitives are defined as “*atomic mechanical functional element that operate on basic mechanical concepts without the invocation of any other functionality operation*”. The **nine** functionality primitives described in this research are: *transformation, transmission, joint, load bearing, energy converter, frictional, offset, channel, and block*. The definitions of these primitives are summarized in Table 6. A more detailed description of the nine primitives is given in Appendix B.

A subsequent combination of the functionality primitives and definition of their unique relationships and interactions yields a more complex functionality operation that may be used to construct any mechanical function. As an illustration (see Figure 49), a combination of *offset* and *load bearing* primitives will yield the *support* function in a trolley. Additional refinement of this support functionality might then include joint functionality to hold the corresponding functionality artifacts together. Similarly, motion sub-functionality may be achieved by combining force transmission and force-to-displacement transformation. A subsequent combination of support and motion sub-functions will then yield a more complex function – transportation function.

Table 6 Functionality models of commonly used mechanical product functionality

Primitive Operation	Definition (Basic Form)	Description
Transformation		Converts one functionality operand to another, e.g., force to torque or deformation. A: Source Operand B: Target Operand C: Transform Operator: A-B interaction
Transmission		Conveys an operand from one point to another, e.g., force transmission along a beam and torque transmission of a pump spindle. A: Source Operand B: Target Operand C: Transmission Operator: A-B interaction
Joint		Retains an absolute or relative position of functional elements achieved by exerting a retentive force on entity or/and resisting dissociating forces. A: First Operand B: Second Operand C: Joint Operator: A-B interaction
Load bearing		Bear loads exerted by mechanical operands to maintain the equilibrium/stability of the mechanical elements. A: Source Operand (Load) B: Target Operand (Base) C: Load Bearer Operator: A-B interaction
Energy Converter		Stores or transforms energy from one form to another. A: Source Operand B: Target Operand C: Converter Operator: A-B interaction
Frictional		Describes frictional elements in mechanical systems. A: First Operand $\leftarrow C_1$ B: Second Operand $\leftarrow C_2$ C: Friction Operator: A-B-C ₁ -C ₂ -M-F interaction
Offset		Maintain a specified offset between entities by providing obstruction to disallowed position. A: First Operand B: Second Operand C: Offset Operator: A-B interaction
Channel		Provides a mechanical assistance to ensure that an entity assumes a desired spatial location. Constrains displacement along a pre-defined path, e.g., guide rails, chamfers for assembly. A is the channel, guide, or path enforcer. B is the guided or channeled operand. C is the channel Operator defining A-B interaction.
Block		<i>Block</i> : disallows non-permissible and expected entrance/motions, e.g., provision of insulation in electrical systems, guards in fans etc. A: First Operand B: Second Operand C: Block Operator: A-B interaction

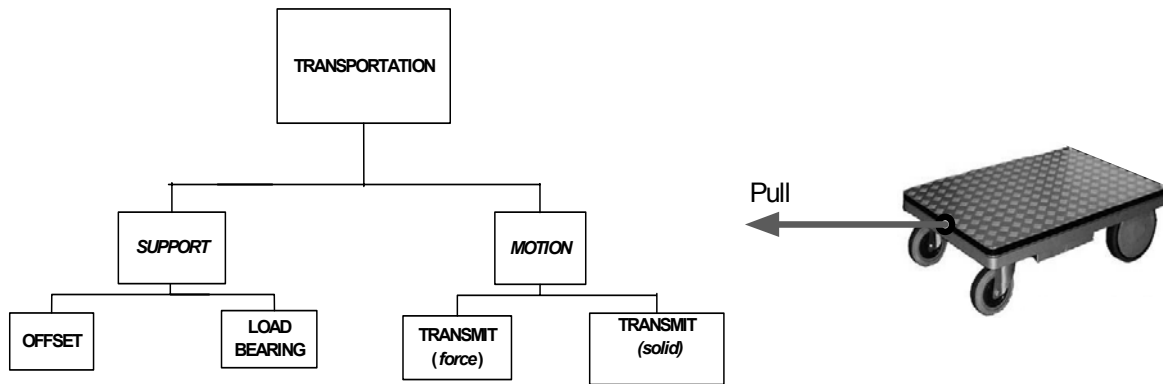


Figure 49 Example of compound functionality in a cart

The idea of functionality design is that form need not be known at the start of design. Only functions are needed. The product in Figure 49 is included to help the reader understand the concept of compound functionality operation and functionality decomposition.

6.0 IMPLEMENTATION AND TESTING

Since the scope of this work is to develop a methodology to support functionality-based design (FbD) in CAD systems, it is appropriate that a computer model is implemented to validate and demonstrate the concepts proposed in this work. However, since the objective of this work is to develop a model that is applicable to mechanical systems, it is necessary to ensure that the computer implementation demonstrates all aspects of the functionality model as is applicable to mechanical systems.

The implementation and testing is achieved by developing a functionality data model and the associated computer tools that are then applied to the design of a real product – space-frame sub-assembly. The following computer tools were developed in this work:

- Functionality modeling and definition tool.
- Functionality constraint imposition tool.
- Functionality data structure.
- Functionality data propagation mechanism.

6.1 Functionality Object Model

The physical realization of product functionality is hierarchical in nature. As illustrated in Figure 50, product functionality can normally be realized by bringing together separate entities of sub-functionalities and operands. Each sub-functionality accomplishes a specific

function that contributes towards the accomplishment of the overall function. The sub-functionalities are in turn sub-divided into sub-functionalities and operands. The operand is the lowest level of this hierarchical decomposition. Each operand has a specific functional purpose that contributes directly or indirectly (as in the case of sub-functionalities) toward the realization of the overall function. This hierarchical organization gives the functionality-modeling problem a robust property that makes it possible to use object-oriented approach in functionality representation.

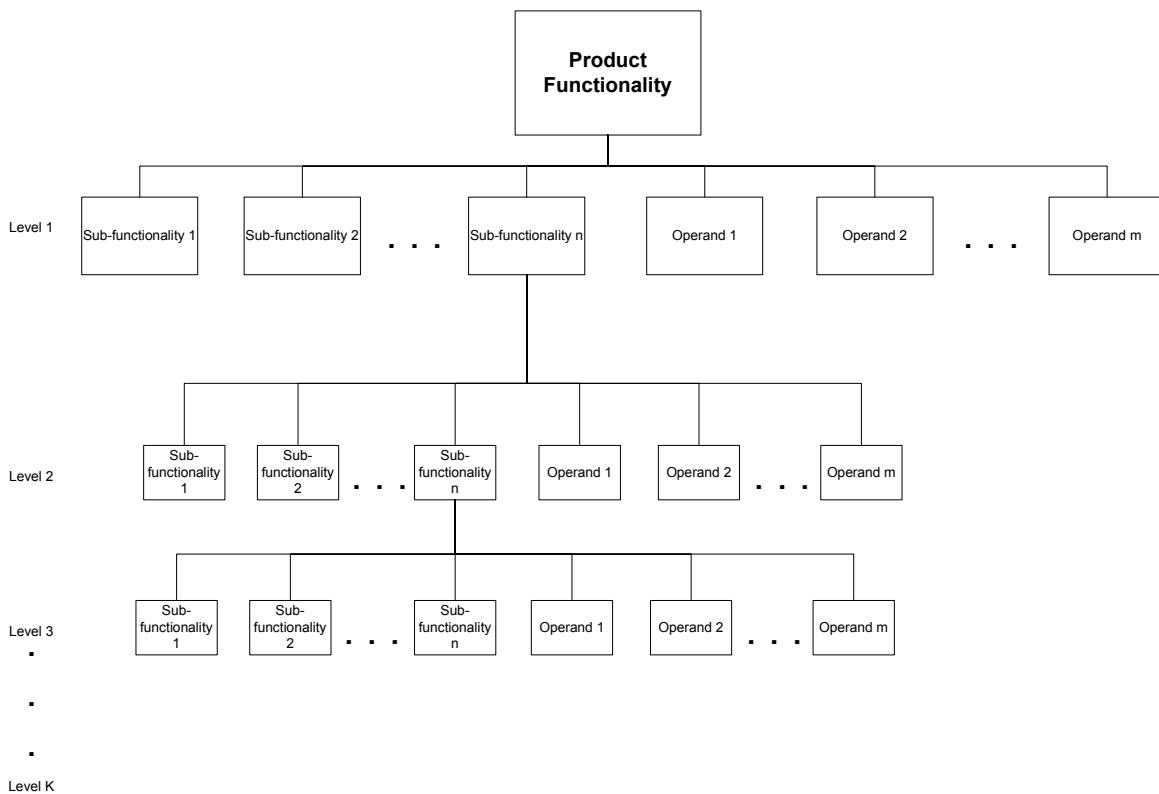


Figure 50 Product functionality decomposition

In this work, *unified modeling language* ^[79] (UML) is used to model product functionalities. This method employs object-oriented approach to describe product functionality as a collection of discrete objects that incorporate both data structure and behavior. The major four aspects of object-oriented approach are: identity, classification, polymorphism, and inheritance. Identity means that data is quantized into discrete, distinguishable entities called objects. Each object has its own inherent identity. This implies that two objects are distinct even if all attribute values are identical. Classification means that objects with the same data structure (attributes) and behavior (operation) are grouped into a class. Each object is said to be an instance of a class. Each instance of a class has its own values for each attribute but share the attribute names and operations with other instances of the class. Polymorphism means that the same operation may behave differently on different classes. A specific implementation of an operation by a certain class is called a method. Inheritance is the sharing of attributes and operations among classes based on a hierarchical relationship. A class can be defined broadly and then refined into successively finer subclasses. Each subclass incorporates, or inherits, all of the properties of its super-class and adds its own unique properties.

In this research, object models are used to describe the static structure of functionality objects (operations) in a system and their relationships. The object model is graphically represented with the object diagram. An object diagram is a graph whose nodes are object classes and whose arcs are relationships among classes.

The class diagram of the functionality object model developed in this work is shown in Figure 51. This object diagram represents the product-level functionality model. The components of the functionality object model include functionality operation, form, operand set,

coupling bond set, and state. Taking this functionality modeling approach, a generic modeling engine is developed to represent product functionality.

A quick comparison of this model with that of the product functionality decomposition (Figure 50) will help to illustrate how the inheritance property of the Object Oriented Programming System (OOPs) may be used. In mechanical systems design, assemblies typically follow the same structure with components as objects. Each assembly inherits all the properties of the components / sub-assemblies. Lack of a uniform progression has been one of the major problems facing previous attempts by other researchers to model functionality. The components of the functionality object model are described below.

6.1.1 Functionality Operation

The *functionality operation* class defines the mechanical task (function) to be realized. Each functionality object model includes this class in its implementation. It acts like an object header that provides a reference point for all operations. The attributes of the operation class are: *name*, *id*, *type*, unit of measure, and functionality description.

Name and ID

The *name* attribute is the name of the function. It could be user assigned based on practice of the specific industry or company. To enhance the re-use of product functionality knowledge, the naming convention should follow a well-defined convention in the industry/company. For instance, in functionality primitive definition, the keyword is the functionality operator followed by the associated operands. This is also a good place to apply some of the naming conventions proposed by previous researcher⁽²⁴⁾ for example use of words formed from *noun+verb*. In the functionality design repository, this convention is followed where the verb is the operator and the noun is composed of the key operands. For instance, consider the function named *force_torque_transform*. This performs a *transformation* function that operates on *force* to produce *torque*.

The *id* attribute is a unique identifier for the functionality operation. This is the handle used to reference an operation by other entities (such as solid models and analysis models) in a CAD design model. Every functionality operation in a product must have its own unique *id*.

Type

The *type* attribute is used to denote the functionality category that the function belongs to. It defines a super-class for the functionality. As an illustration, the functionality primitives

belong to the following types: *transformation, transmission, joint, load bearing, energy converter, frictional, offset, channel, and block*. A functionality that is defined as an object of a class inherits all its attributes hence makes modeling a lot easier.

This is used to define as a new class that extends or uses (duplicates) the capability and attributes of an existing class. It invokes the use of an existing functionality object model to modify its attributes and/or behavior by adding new operands, new sub-operations, or modification of their existing attributes. The selection of a functionality type invokes and constrains some of the functionality attributes and behaviors. Functions with similar attributes invoke similar constraints and considerations. For instance, the selection of a transmission functionality class will invoke and constrain all identifiers as described in the transmission primitive object.

The type notation is used to build reusable functionality models. For examples, it could form a repository of re-usable design knowledge base. A system user can create custom functionality classes that are derived from the existing functionality. Thus, an extension of the existing behavior and attribute of the functionality library is possible. If no *type* is provided in a functionality operation object, this attribute is blank (empty) and implies that a new or generative operation is being defined.

6.1.2 Operand Set

This is a set of all the functionality operands that are used in the functionality model. These operands are distinguished from other operands that do not have a child-parent relationship with the functionality operation. These other categories of operands include the

coupling operands that are part of other sub-functionality operations. An operand set is given by:

OPERANDS: {A<attrib A>, B<attrib B>, C<attrib C>, ... , K<attrib K>}

Where,

A, B, C, ..., K are individual operands; and

<attrib i> is the attribute set of operand *i*.

Each operand provides a reference index attribute (*form_tag*) that is used to link it to specific geometric entities in the CAD solid model. This tag is included in the XML data that is propagated to downstream design activities and used for design verification and functionality analysis.

Based on the scope ^[*] of this work, operands are grouped into two categories: *material* (*SOLID*) and *energy* (*FORCE* and *TORQUE*).

$$\text{Operand Class} \begin{cases} \textit{material} \{ \textit{SOLID} \\ \textit{mechENERGY} \begin{cases} \textit{FORCE} \\ \textit{TORQUE} \end{cases} \end{cases}$$

The additional operand-specific attributes as described in Chapter 4 are:

* The scope of the operand modeling developed in this work is discussed in Chapter 4.

SOLID {FM, DoF, PyC, MP, MT},

Where

FM is the set of functional positional markers;

DoF is the required degree of freedom of operand;

PyC is the set of physical constraints on the operands;

MP is the set of mass properties on the operands; and

MT is the material type constraint.

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

Where

source defines the source of force;

kind defines the kind of force;

nature defines the nature of force,

mag, angle, and **point,** are the magnitude of force; the rotation from the Cartesian axis and the 3-D coordinates of the point of application of the force respectively.

Similarly, TORQUE is modeled by:

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

Where

source defines the source of force;

kind defines the kind of force;

nature defines the nature of force,

mag, angle, and **axis,** are the magnitude of force; the rotation from the Cartesian axis, and the axis of rotation of the torque respectively.

6.1.3 Sub-Functionality Set

The sub-functionality set defines the collection of other related functionality operations that interact with the function. This is used to link functionality cases where a function will invoke other cooperating functions in order to accomplish its objective. For example, a *friction reduction functionality* object might be invoked by a motion functionality operation to reduce friction between moving parts / solids. This linkage cooperation is used to realize the hierarchical structure of a functionality object as shown in Figure 50.

Interaction specifies the cooperation among the entities while the functional invocation is used to actually instantiate the inclusion of the desired functionality object together with all its attributes. The coupling operand is used to define the actual sub-functionality operand that interacts (or interfaces) to the main functionality operation. The coupling operand includes the attributes and states that are used in the coupling bond.

6.1.4 Form

This describes any prior form or concept associated with a given functionality model. This approach is used to store information on physical embodiment or conceptual form associated with a predefined functionality operation. The *formIndex* attribute is the index of the form while *linkedGeometry* is the actual linkage to the instance of the linked form.

6.1.5 Coupling Bond (CB) Set

The coupling bond class defines the set of operators involved in the functionality operation. This is a collection of all the active relations and constraints imposed on the functionality operands (including the coupling operands of the sub-functionality objects). The CB class defines the context of operation and models the what-if and casual effects (input-outputs) between operands.

6.1.6 States

This is a class of all the allowable distinct time-variant operand attribute combinations in the functionality operation. The state class represents the time-variant attributes of the functionality object. This class associates each time-variant attribute to an operand and defines the feasible value range. This feasible value range is defined by the *valueSet* attribute.

6.2 Computer Implementation

The modeling and reasoning concepts proposed in this research are implemented for *Pegasus*,^[80,81] an e-Product design and realization platform. Pegasus system is a collaborative product design environment that allows for customers, designers, engineers, and other stakeholders to participate in product design. It provides a scalable, flexible, and efficient collaborative product design platform, which enables different stakeholders of design to work on product development concurrently. Various computational engineering tools make certain

services available to other design participants in a network-based distributed environment. These tools are called service providers. The services that are provided by different engineering tools are published by a service manager, and are available within this distributed environment. The inclusion of functionality model in Pegasus will allow remote collaborators in a product design to remotely specify and propagate their functionality requirements to other remote collaborators and downstream design activities / tools.

An interactive *graphic user interface* (GUI) is developed in this research to serve as the designer's interface during product design. This interface ensures that a human oriented description of the function (specifications – preferences, constraints, and needs) is translated into the appropriate functionality data structure for reasoning and propagation in a computer system.

The functionality-based design interface tool is implemented in computer system using Microsoft Visio 2002 graphics engine. ^[82] The implementation uses an object-oriented data structure for the representation and propagation of functionality as design constraints. This data structure serves as an interface during the functional requirement definition phase of design. It is used to capture the designer's intent (functional requirements) during the conceptual design phase.

The modeling scope of this computer implementation is limited to functionality operations that can be modeled and conceptualized in two-dimensional coordinate system. The 2-D models developed in this functionality engine are converted to 3-D solid models in CAD systems. The computational tools developed in this work include:

- Functionality Definition Interface and Tool Kits
- Functionality Object Model
- Data Representation in XML

A more detailed description of the various computational tools is provided in the following sub-sections.

6.2.1 Functionality-Based Design Procedure

The implementation of functionality-based design procedure in a CAD environment involves four distinct steps. The flow chart for this functionality-based design procedure is shown in Figure 52. A more detailed description of the activities involved in each step is described in the remainder of this section.

1) Problem definition and specification

This is the design task identification step. It involves the identification of the need and design task that a product is to accomplish. Hence, the inputs to the system include the perceived needs in the form of problem statements, constraints, and preferences. It is important that the designer identify all the needs that should be realized by the system. This is necessary because any need that is neglected at this stage cannot be captured by the functionality model and hence will not be realized by the final design. Example of the problem identification might be the identification of need for a means of transforming a linear motion into rotational motion as in a slider-crank mechanism of an internal combustion engine of a car.

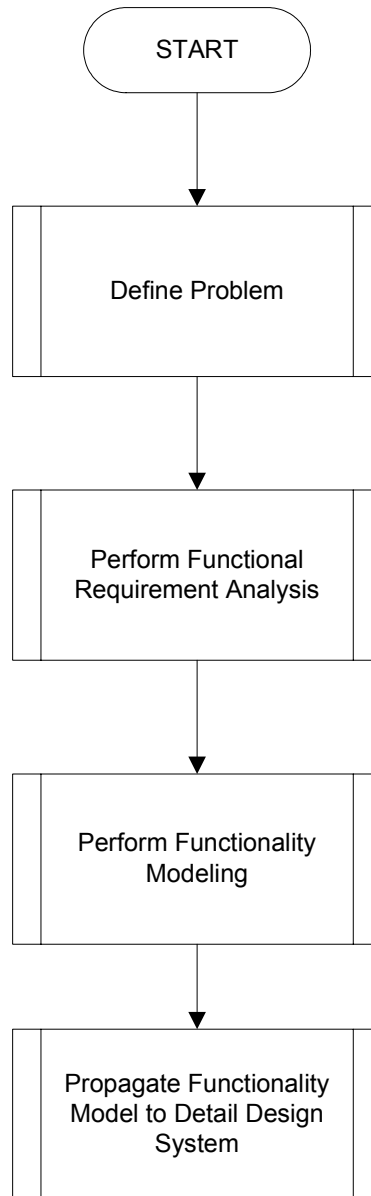


Figure 52 Functionality-based design flow chart

2) Functional requirement analysis

At this stage, the design tasks and needs identified in step 1 are mapped into functional requirements that the designed product must have to perform successfully. This mapping is illustrated in Figure 53.

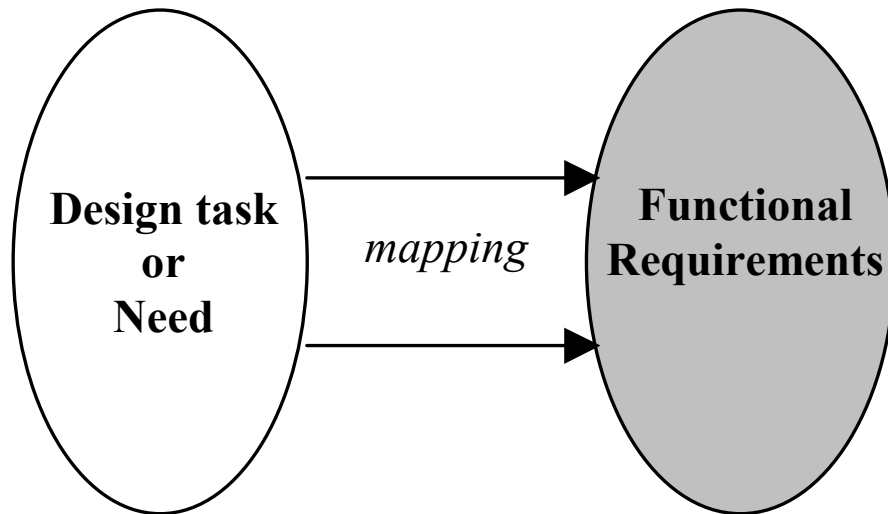


Figure 53 Mapping from design task to functional requirement

The designer should at this stage ensure that all the known factors that might influence the product are clearly identified. Functional requirement analysis leads to the identification of the functional elements necessary for the realization of the design task. That is this step identifies the functional composition necessary for the task. The specific items identified at this stage include:

- The functionality structure and hierarchy.
- Required functionality operands and their corresponding attributes including functional markers.
- Any required sub-functionality operations.
- The nature of interaction between operands.
- Functional constraints.

3) Functionality Modeling

In the modeling step, the functional requirements identified during the analysis phase are translated into object instances in the functionality model. Hence, the identified operands and their corresponding attributes are modeled in a computer system. The inter-operand interactions are defined as relations using operand-coupling bonds. In building the functionality model, the designer is made to consider some factors (such as operand attribute) that may have been neglected during functional requirement analysis. The mapping from “functional requirement” to functionality model is illustrated in Figure 54.

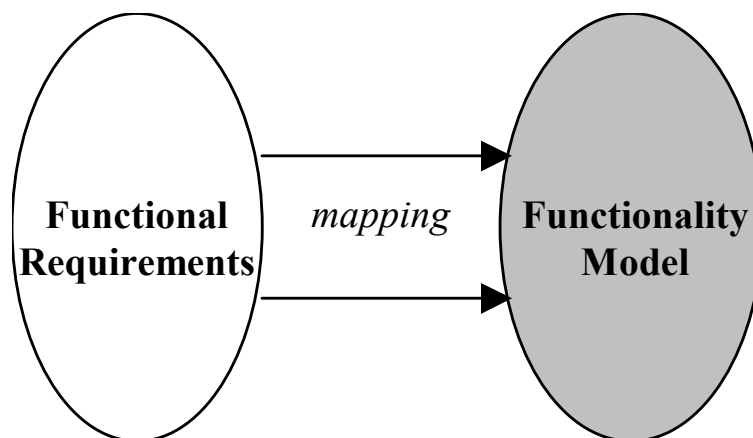


Figure 54 Mapping from functional requirement to functionality model

4) Propagation of functionality model

The model developed in step 3 is now represented in an XML data file (forming what is called the functionality signature). This XML file is then propagated to the detailed design

phase, where the complete solid CAD model is developed and then evaluated against the functionality constraints embedded in the XML data file.

6.2.2 Functionality-Based Design (FbD) Architecture

The functionality flow diagram for the evolution of design in an FbD design environment is shown in Figure 55. The input to the system is functional specification in the form of problem statement, constraints, and preferences. These inputs are used in the functional definition and modeling of the product functionality. At the functionality modeling stage, the designer is allowed to define all the functionally relevant attributes of the product. Such description involves the specification of the known operands, operators, and coupling bonds involved in the functionality operation.

The output of the functionality modeling stage of the FbD system is the *functionality signature*. The functionality signature consists of: *operand set* and *coupling bond set* together with all the *constraints, relations, and required degrees of freedom*. The signature also contains a pointer or a link to any proposed *geometric form* defining or/and satisfying the given functionality constraints or corresponding to specific operands in the model. These geometric links are embedded in the form of an XML tag pointing to the CAD model represented by the functionality component. Additional *extrinsic* constraints may also be imposed on the functionality model by other factors external to the functionality object. These constraints might include some technological and physical restrictions on possible values a functionality parameter might assume.

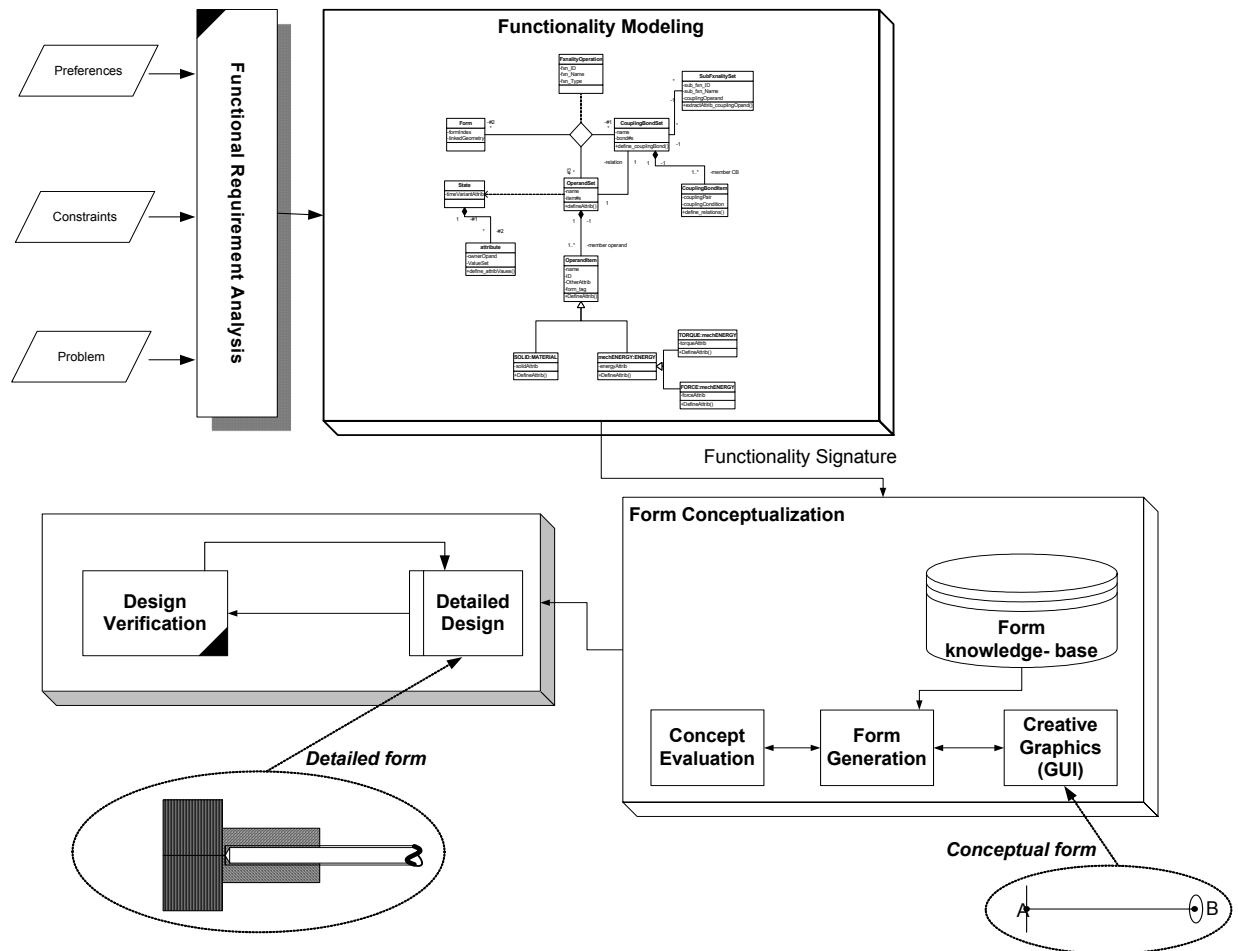


Figure 55 Functionality-based design flow diagram for the evolution of design

The linked concepts and artifact describes any prior form or concept associated with a given functionality description. This approach is used to store information on physical embodiment or conceptual form associated with a predefined functionality operation. When new products are designed using functionality-based approach, the design (conceptual form / artifact) and the associated functionality description are stored in the functionality knowledge base. These descriptions form part of the design repository of reusable designs that may be used in future functionality reasoning.

The functionality signature is propagated to the form conceptualization and detailed design phases. This signature serves as our design objective and is a standard against which all design proposals must be evaluated. Such functionality model may now be used in the conceptualization of design parameters to satisfy the functionality constraints. In the detailed design phase, the constraints are used to evaluate design proposals to ensure that they satisfy the original product functionality. Hence, an automated and transparent functionality verification of designs is possible.

In an intelligent design environment, the functionality signature is used to retrieve acceptable design structures from the knowledge base during the form generation stage. It presents a template for matching functions to stored knowledge when available. Otherwise, they serve as a trigger flags signaling the geometric models that need to be defined or associated to functionality entities. They may also be used in the transformation of functionality representation to a *creative graphic* (wire-frame) representation to aid the creativity of the designer. This concept is illustrated in Figure 56 for a shaft transmission example Figure 56(a). Figure 56(b) illustrates the use of wire-frame model to represent the transmission shaft example. In the figure, points A and B are the functional markers while the line A-B represents the transmission axis.

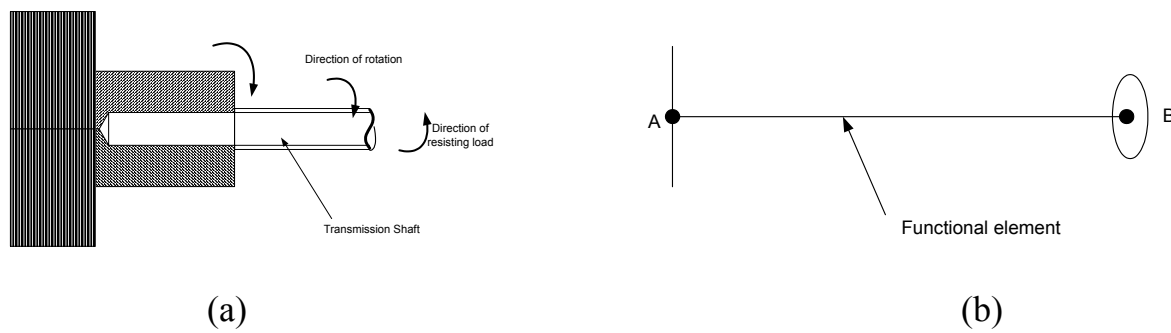


Figure 56 Sample transmission design

6.2.3 XML Functionality Data Format

XML stands for EXtensible Markup Language. Tags enclosed in “<” and “>” characters are used to define the structure and data elements of an XML text or string. These tags are not predefined in XML. Hence, one is required to define custom tags for new implementations. XML uses a Document Type Definition (DTD) or a Schema to describe the data. A DTD or Schema is designed to be self-descriptive.

The primary and sole purpose of XML is to carry data. XML was designed to describe data and to focus on what data is. It is created to structure, store, and to exchange information. It is a cross-platform, software and hardware independent tool for transmitting information. This makes it particularly applicable to represent functionality data that may be exchanged between different CAD platforms and systems.

The following example is an XML description of an operand attribute, mass.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<attribute>
  <name>mass</name>
  <unit>kg</unit>
  <value>37.25</value>
</attribute>
```

With XML, data can be stored in separate XML files and exchanged as texts between incompatible systems. Since XML data is stored in plain text format, it provides a software- and hardware-independent way of sharing data. This makes it much easier to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers. In the CAD world, designer

packages contain data in incompatible formats. One of the most time-consuming challenges for developers has been to exchange data between such systems. The use of XML data format in functionality representation and propagation can greatly reduce this complexity and create data that can be read by many different types of applications. Hence, help to overcome interoperability problems associated with traditional CAD systems.

Plain text files can be used to store data XML formatted functionality information in databases and also be used in a collaborative design environment where data is transmitted to distributed design participants at remote locations.

a) XML Syntax

The syntax rules of XML are very simple and very strict. XML documents use a self-describing and simple syntax. The first line in the document - the XML declaration - defines the XML version and the character encoding used in the document. In the above example on attribute representation, 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. In the above example, the next line describes the root element of the document (like it was saying: "this document is an attribute"): `<attribute>`. All XML documents must contain a single tag pair to define the root element. All other elements must be nested within the root element. All elements can have sub elements (children). Sub elements must be correctly nested within their parent element. The next four lines describe four child elements of the root (name, unit, and value). In XML, all elements must have a closing tag. In the example, the last line defines the end of the root element: `</attribute >`

An XML Parser is used to read and update - create and manipulate - an XML document. Loading an XML files into the parser extracts the data embedded in the XML file. A function (code) written in VBA is used to accomplish the parser function. This parser function is used to perform both read from and write to file operations.

b) Functionality XML data format

The XML schema for the functionality model is listed below: A brief description of each tag is given below.

```
<?xml version= "1.0" ?>
<!-- Pegasus Functionality Definition !-->
<functionality-operation>
    <info></info>
    <opand-set></opand-set>
    <state></state>
    <coupling-bond-set></coupling-bond-set>
    <sub-fxn-set></sub-fxn-set>
    <form></form>
</functionality-operation>
```

The XML declaration

The first line in the document (<?xml version= "1.0" ?>) is the XML declaration. It defines the XML version used in the document. In the functionality model above, the document conforms to the 1.0 specification of XML. The statement enclosed within “<!--“ and “!-->” are comments.

The root tag

The first tag (<functionality-operation>) is the root tag. It describes the root element of the document (like it was saying: "this document is a functionality operation"). It begins the definition of an instance of the functionality model in XML. The last line defines the end of the

root element: `</functionality-operation>`. It marks the end of the XML data of the functionality operation. All the other information concerning the functionality operation must be enclosed within the opening and closing tags.

The info tag

This tag (`<info></info>`) contains the general information about the functionality operation. This information consists of: id, name, type, unit, and description of the functionality operation. The schema for this information is shown below.

Functionality information XML Schema:

```
<info>
  <id></id>
  <name></name>
  <type></type>
  <unit></unit>
  <description></description>
</info>
```

The opand-set tag

This tag (`<opand-set></opand-set>`) contains the set of operands included in the functionality model. Information about each operand is contained inside its own operand opening and closing tags: `<opand></opand>`. Operand specific information include: opand-id; opand-name; opand-type; opand-form-tag; opand-description; and attrib-set (for solid, this includes functional markers, physical property attributes, mass property, DoF, and mass properties). The schema for this information is shown below.

Functionality operand set XML Schema:

```
<operand-set>
  <operand>
    <operand-id></operand-id>
    <operand-name></operand-name>
    <operand-type></operand-type>
    <operand-form-tag></operand-form-tag>
    <operand-description></operand-description>
    <attrib-set>
      <attrib-name></attrib-name>
      <attrib-value></attrib-value>
    </attrib-set>
  </operand>
  .....
  <operand>
  </operand>
</operand-set>
```

The state tag

This tag (<state></state>) contains information on the possible state of the functionality operands.

Functionality state XML Schema:

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

The coupling-bond-set

This tag (<coupling-bond-set></coupling-bond-set>) contains information on the coupling bond set of the functionality operation.

Functionality coupling bond set XML Schema:

```
<coupling-bond-set>
  <coupling-bond>
    <id></id>
    <name></name>
    <type></type>
    <form-tag></form-tag>
    <description></description>
    <pair>
      <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>
    </pair>
    <condition>
      <fxnal-rel-set>
        <f-relation>
          <id></id>
          <factor-1></factor-1>
          <rel></rel>
          <factor-2></factor-2>
        </f-relation>
      </fxnal-rel-set>
      <fxnal-constr-set>
        <f-constr>
          <id></id>
          <factor></factor>
          <rel></rel>
          <const></const>
        </f-constr>
      </fxnal-constr-set>
    </condition>
    <dof-set>
      <dof>
```

```

<id></id>
<type></type>
<ref-frame></ref-frame>
</dof>
</dof-set>
</condition>
</coupling-bond>

<coupling-bond>
</coupling-bond>
</coupling-bond-set>

```

The XML representation of the functionality markers (POINT, LINE, and ARC) is shown below.

Functional Markers XML Schema:

```

<fxnal-marker-set>
  <f-point>
    <id></id>
    <name></name>
    <x></x><y></y><z></z>
  </f-point>
  <f-line>
    <id></id>
    <name></name>
    <type> fixLen or infLen </type>
    <begin>
      <x></x><y></y><z></z>
    </begin>
    <end>
      <x></x><y></y><z></z>
    </end>
    <side>
      <IS1></IS1>
      <IS2></IS2>
    </side>
  </f-line>
  <f-arc>
    <id></id>
    <name></name>

```

```

    <type> circ or elsp </type>
    <begin>
        <x></x><y></y><z></z>
    </begin>
    <end>
        <x></x><y></y><z></z>
    </end>
    <bow-dim></bow-dim>
    <side>
        <aS1></aS1>
        <aS2></aS2>
    </side>
</f-arc>
</fxnal-marker-set>

```

The sub-fxn-set tag

This tag (<sub-fxn-set></sub-fxn-set>) contains information on the sub-functionality operations included in the functionality operation.

Functionality sub-functionality operation set XML Schema:

```

<sub-fxn-set>
  <sub-fxn>
    <fxn-id> </fxn-id>
    <fxn-name> </fxn-name>
    <coupling-opand>
      <opand-id> </opand-id>
      <opand-name> </opand-name>
      <opand-type> </opand-type>
      <opand-form-tag> </opand-form-tag>
      <opand-description> </opand-description>
      <opand-attributes>
        <attrib-name> </attrib-name>
        <attrib-value> </attrib-value>
      </opand-attributes>
    </coupling-opand>
  </sub-fxn>
  .....
  <sub-fxn>
  </sub-fxn>
</sub-fxn-set>

```


The form tag

This tag (<form></form>) contains information that links the functionality operation model to a specific CAD geometric model.

Functionality form XML Schema:

```
<form>  
  <tag></tag>  
  <geometric-index></geometric-index>  
</form>
```

6.2.4 Graphic User Interface and General Capability

The Graphic User Interface (GUI) is implemented in a customized Microsoft Visio 2002 environment. A customized functionality drawing template was created as a model drawing for functionality objects. This template includes the necessary functionality stencil containing master shapes for the various components of the functionality-based design process. Stencils were developed to automate the various activities involved in definition of functionality model. Most of the drawing activities were automated by programs written in Microsoft Visual Basic for Application (VBA).

A stencil is like a software library in which one can collect the shapes you build for later reuse. Masters are customized shapes stored in a stencil for future reuse. A master is a shape, multiple shapes, a group, or an object from another application that is saved on a stencil, which can be opened in other drawings. The master represents basic functionality elements such as operands and coupling bond. Connections and functional marker shapes are also saved as

masters. When a designer drags a master from the stencil onto a drawing page, the Visio engine creates a copy of that master on the drawing's document stencil and creates an instance of the master on the drawing page. An instance is linked to the copy of its master on the document stencil and inherits its behavior and appearance from that master. Because a stencil contains the shapes from which users of the functionality solution will construct a drawing, it is a primary user interface element in this drawing solution.

A template opens particular functionality stencils and specifies page settings, layer information, style, shapes, pre-drawn elements and functionality macros, which makes it simple to deliver a custom solution to users.

To develop a functionality model, the designer opens the customized Visio functionality template, which brings up the basic functionality stencil containing all the basic modeling objects as master shapes. These set of masters in the stencil are called the functionality toolkit as they provide the basic tools needed to model product functionality. On startup of the functionality package, a user interface containing the initial toolset similar to that shown in Figure 57 is displayed on the screen. In the case of a new design, the dialog box shown in Figure 58 is also displayed to collect function –level information such as name, type, description, unit of measure, and an internally generated functionality operation ID. The components of the various functionality stencils are described below.

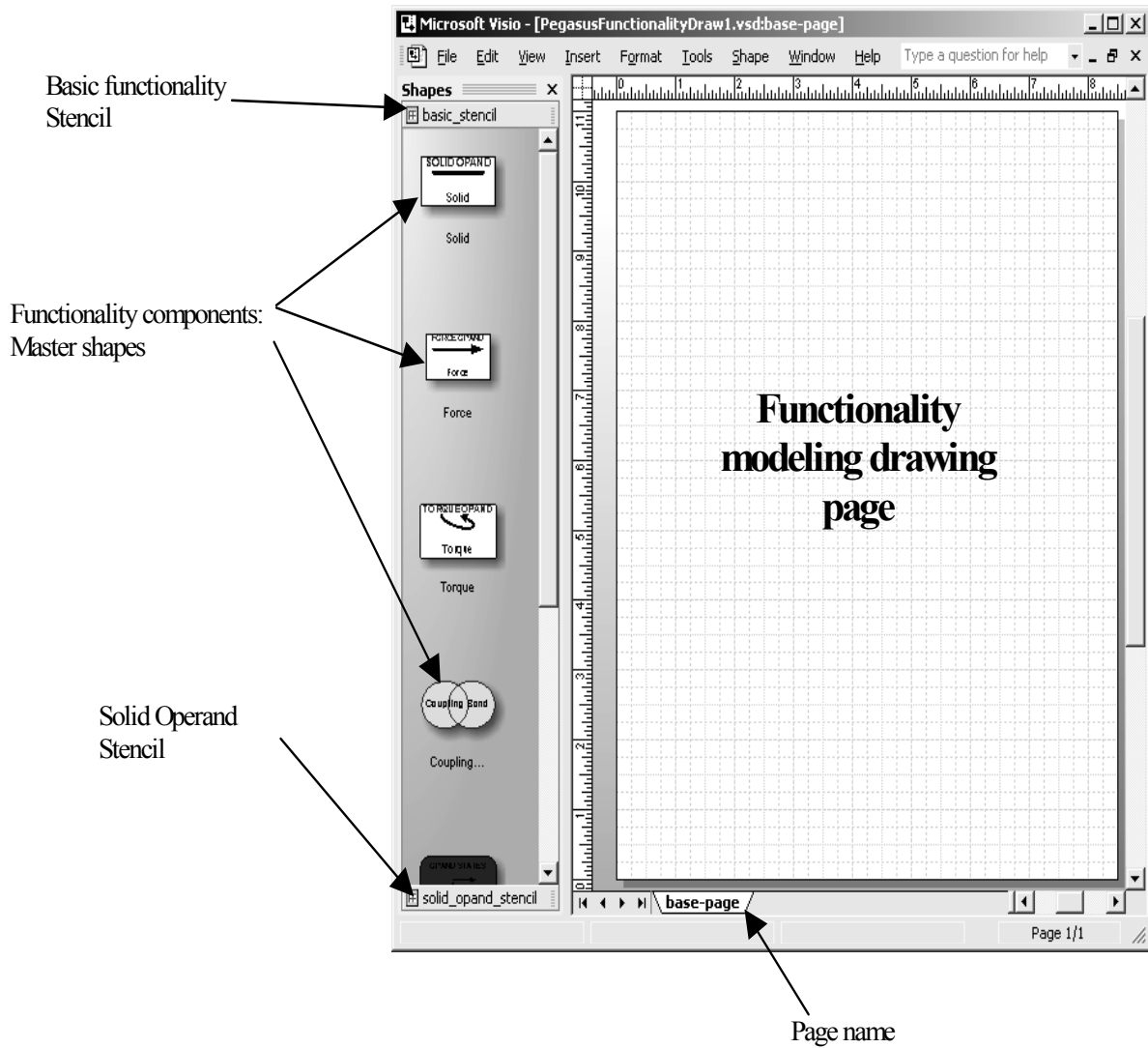


Figure 57 Functionality modeling interface using Microsoft Visio

Figure 58 Functionality operation start-up definition dialog box

a) Basic Functionality Stencil

The basic functionality stencil is used to define the operands, coupling bonds, states, and sub-functionalities that make up a functionality operation. The behavior of each of the component is determined by the VBA codes embedded in the basic functionality stencil. For instance, double clicking or dropping a master from the stencil will instantiate an action in the modeling of the functionality operation. Some of the masters such as operands also trigger events that results in the addition or/and opening of a detailing page for the clicked operand. Each of the masters in this stencil also has its own unique user form to enable customization of the properties of the corresponding functionality component. A brief description of each of the basic functionality operand is listed in Table 7.

Table 7 Summary of the master shapes in the basic functionality stencil

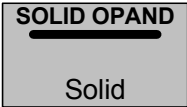
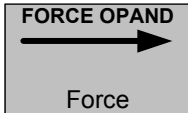

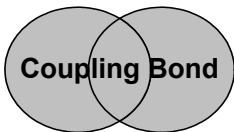
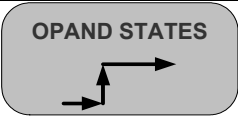
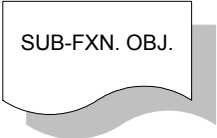

Master	Symbolic Shape	Description
Solid master Operand		<ul style="list-style-type: none"> • Represents an instance solid of operand in the functionality model. • Double clicking this object brings up the operand detail page corresponding to this solid. It also opens the appropriate toolkit (solid operand stencils) needed for the editing. • The definition dialog box (shown in Figure 64) is also opened to guide the user in the creation of the solid operand.
Force master operand		<ul style="list-style-type: none"> • Represents an instance of force operand in the functionality model. • Double clicking this object brings up the operand detail page corresponding to this force. It also opens the appropriate toolkit (energy operand stencils) needed for the editing. • The definition dialog box (shown in Figure 65) is also opened to guide the user in the creation of the force operand.
Torque master operand		<ul style="list-style-type: none"> • Represents an instance of torque operand in the functionality model. • Double clicking this object brings up the operand detail page corresponding to this torque. It also opens the appropriate toolkit (energy operand stencils) needed for the editing. • The definition dialog box (shown in Figure 66) is also opened to guide the user in the creation of the torque operand.
Coupling master operand		<ul style="list-style-type: none"> • Represents an instance of the coupling bond in the functionality model. • Double clicking this object brings up the coupling detail page corresponding to this CB. • The definition dialog box (shown in Figure 67) is also opened to guide the user in the creation of the coupling bond. • The two operands to be coupled are then selected in the CB user form interface. • In addition, all the coupling attributes, relations, constraints, and DoF are defined in this interface.

Table 7 (continued).

<p>Functionality State master</p>		<ul style="list-style-type: none"> • Represents an instance of operand state in the functionality model. • Double clicking this objects brings up the definition dialog box is also opened to guide the user in the creation of the operand states. • The user selects operands and coupling bond in which state is a relevant factor. The selected operands together with their respective attributes are then defined as time variant functionality components. • These time variant components attributes define the different states the system can assume and hence are considered during the design verification / analysis phase.
<p>Sub-functionality master</p>		<ul style="list-style-type: none"> • Represents an instance of a sub-functionality operation included in the model. • Double clicking this object brings up the corresponding operation detail page together with the user form with which the designer selects the operands that are used in coupling bond with the other operands in the super-functionality operation.
<p>Connector master</p>		<ul style="list-style-type: none"> • Represents an instance of component connector in the functionality model drawing. • It links components (operands and CBs) together. • It performs no action and is only used to pictorially link components together.

b) Solid Operand Stencil

The solid operand stencil is used to define the attributes and functional markers of a given solid operand. It is invoked whenever a solid operand detail page is opened or a solid operand is double clicked for editing purposes. The opening of the solid operand page also invokes the “solid operand user form” shown in Figure 64. This form is used to guide the user in the definition of the attributes of the solid operand.

The actual Visio implementation of the functional markers is limited to points, line, and circular arcs. The point is defined by the coordinates of its points in two dimensions: x and y coordinates. The line on the other hand is defined by the begin and end points together with the sides (side 1 and side 2) as shown in Figure 59.

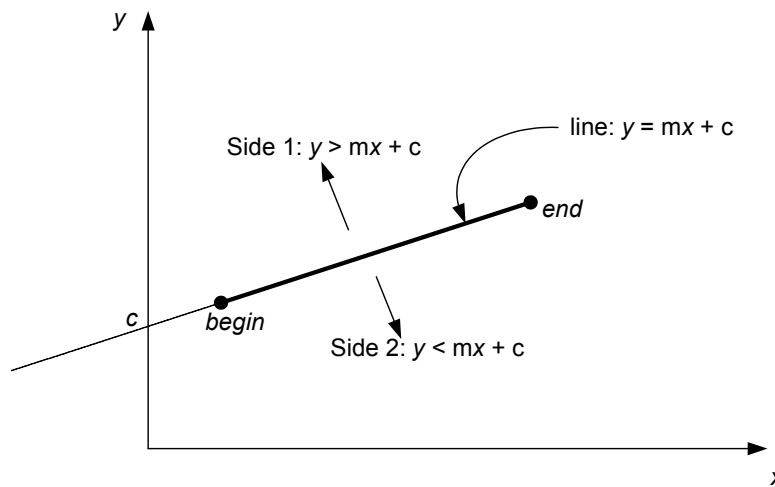


Figure 59 Modeling of functional line in Visio

The circular arc is defined by begin and end points of the arc together with the bow of the arc (see Figure 60). The magnitude of the bow is the distance from the midpoint of the chord to the midpoint of the arc. The bow's value is positive if the arc is drawn in the counterclockwise direction; otherwise, it is negative. The radius or the parent circle is related to the magnitude of the bow as follows:

$$|\mathbf{Bow}| = \mathbf{radius} * (1 - \mathbf{COS}(\mathbf{angle}/2))$$

Where, “radius” is the radius of the arc; and “angle” is the angle that the arc subtends at the center of the circle. If the bow is zero, the arc is a straight line.

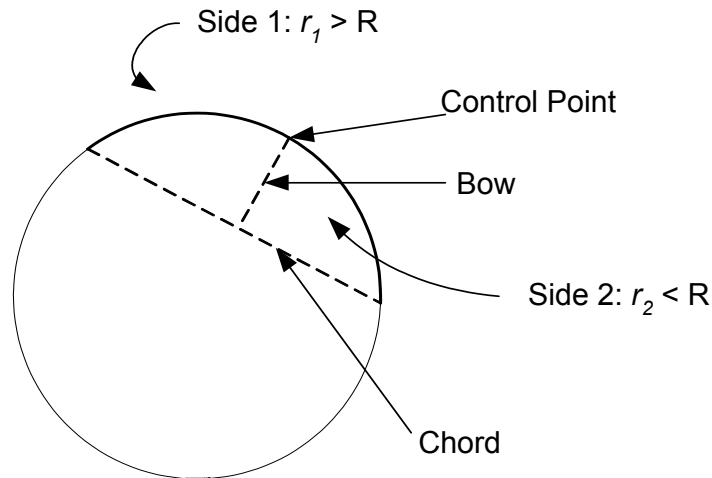





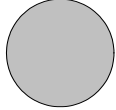

Figure 60 Modeling of functional arc in Visio

The behavior of each of the masters in this stencil is determined by the VBA codes embedded in the solid operand stencil. For instance, double clicking or dropping a master from the stencil will instantiate an action in the modeling of the functionality operation. Each of the masters in this stencil also has its own unique user form to enable customization of the properties of the corresponding functional marker. A brief description of each of the solid operand stencil masters is listed in Table 8.

Table 8 Summary of the master shapes in the solid operand stencil

Master	Symbolic FM Shape	Description
FPoint master	●	<ul style="list-style-type: none"> • Represents a functional point (marker) in a solid operand. • It is placed on the detail page of a solid operand to denote a point that is functionally important for the operand. • Double clicking this object brings up the “FPoint” user form (Figure 61) with which the designer can define the parameters of the functional point.

Table 8 (continued).

Fline master		<ul style="list-style-type: none"> • Represents a functional line (marker) in a solid operand. • It is placed on the detail page of a solid operand to denote a line / axis that is functionally important for the operand. • Double clicking this object brings up the “FLine” user form (Figure 62) with which the designer can define the parameters of the functional line.
FArc master		<ul style="list-style-type: none"> • Represents a functional arc (marker) in a solid operand. • It is placed on the detail page of a solid operand to denote an arc that is functionally important for the operand. • Double clicking this object brings up the “FArc” user form (Figure 63) with which the designer can define the parameters of the functional arc.
FrectArea master		<ul style="list-style-type: none"> • Represents a functional rectangular region (marker) in a solid operand. • It is placed on the detail page of a solid operand to denote a rectangular region that is functionally important for the operand.
FCirArea master		<ul style="list-style-type: none"> • Represents a functional circular region (marker) in a solid operand. • It is placed on the detail page of a solid operand to denote a circular region that is functionally important for the operand.
DimConstraint master		<ul style="list-style-type: none"> • Represents an instance dimensional constraint between the functional markers in a solid operand. • Double clicking this object brings up the coupling definition dialog box to guide the user in the creation of the dimensional constraint. • The two functional markers to be related are then selected in the DimConstraint user form interface. The value of the dimensional constraint is also specified.

Functional Point [X]

System ID: {4BE248D0-FD41-425E-B5...}

Name:

X-Coord.:

Y-Coord.:

Figure 61 Functional point user form

Functional Line [X]

System ID: {E7B9DD05-1051-4A53-94DE-...}

Name:

End Points

Begin Point		End Point	
Begin-X	<input type="text" value="2.875 in."/>	End-X	<input type="text" value="5 in."/>
Begin-Y	<input type="text" value="7.3125 in."/>	End-Y	<input type="text" value="6.25 in."/>

Length: Angle:

Sides

Side1: Side2:

Figure 62 Functional line user form

Functional Arc

System ID: {C3F10ED7-764F-4ADE-B487-}

Name:

End Points

Begin Point		End Point	
Begin-X	<input type="text" value="4.896446609406"/>	End-X	<input type="text" value="6.5"/>
Begin-Y	<input type="text" value="7.625"/>	End-Y	<input type="text" value="7.625"/>

Length: Bow:

Sides

Side1: Side2:

OK Cancel

Figure 63 Functional arc user form

Solid Operand Definition Interface

ID: {26B86282-4EA7-41CE-B93E-BE90F206A5EE}

Name: Type:

FormTag:

Description:

Functional Markers

Physical Constraints

Add Del

Mass Properties

Add Del

Material Type

Coupling Attributes

DoF

Apply OK Cancel


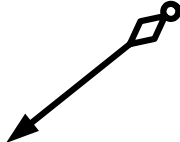
Figure 64 Solid operand definition user form

c) Energy Operand Stencil

The energy operand stencil is used to define the attributes and the functional markers of a given energy operand. It is invoked whenever an energy operand detail page is opened or an energy operand is double clicked for editing purposes. The opening of the energy operand page also invokes the “energy operand user form” shown in Figure 65 (force) and Figure 66 (torque). This form is used to guide the user in the definition of the attributes of the energy operand.

The behavior of each of the masters in this stencil is determined by the VBA codes embedded in the energy operand stencil. For instance, double clicking or dropping a master from the stencil will instantiate an action in the modeling of the functionality operation. Each of the masters in this stencil also has its own unique user form to enable customization of the properties of the corresponding functional marker. A brief description of each of the energy operand stencil masters is listed in Table 9.

Table 9 Summary of the master shapes in the energy operand stencil

Master	Symbolic Shape	Description
EPoint master		<ul style="list-style-type: none"> • Represents an instance of a functional point (marker) in the energy operand. • It is placed on the detail page of an energy operand to denote a point that is functionally important for the operand. • Double clicking this object brings up the “EPoint” user form with which the designer can define the parameters of the functional point.
EVector master		<ul style="list-style-type: none"> • Represents a vector instance of the energy operand. • It is placed on the detail page of an energy operand to denote an energy vector (magnitude and direction) that is functionally important for the operand. • Double clicking this object brings up the “EVector” user form with which the designer can define the parameters of the energy vector.

Force Operand Definition Interface [X]

ID: FX23765XX001

Name: FORCE Kind:

FormTag: XYZ35291F

Nature: Sustained

Description:

Physical Constraints

Magnitude

Lower: 20 N

Upper: 35 N

Direction

Unit Vector Angles

i = 12 x = 38.1731933

j = 5 y = 70.8792018

k = z =

Point of Applic.: <form-tag>FX2453316CB</form-tag> <x>2.3</x> <j>24.5</j>

Coupling Attributes

DoF

Apply OK Cancel

Figure 65 Force operand definition user form

Torque Operand Definition Interface

ID: FX23765XX001

Name: TORQUE Kind: Contact

FormTag: XYZ35291F

Description:

Nature: Cyclic

Physical Constraints

Magnitude

Lower: Upper:

Direction

Unit Vector Angles

i = x =

j = y =

k = z =

Point of Applic.:

Axis of Rot.:

Coupling Attributes DoF

Apply OK Cancel

Figure 66 Torque operand definition user form

d) Coupling Bond Definition

Dropping a master of the coupling bond from the toolkit (stencil) creates an instance of coupling bond master. Double clicking an instance of the coupling bond opens up the coupling bond definition user form (shown in Figure 67). This form is used to define the elements of the coupling bond. It provides a means for the designer to specify the functional relations, degrees

of freedom, and the constraints present at the interaction of the coupling pair (operands). The coupling pair is defined by selecting from the available set of operands in the functionality operation. A form tag number is also specified to link the coupling bond to a solid CAD model.

Operand Coupling Condition [X]

Coupling Bond ID: {5A698935-A372-40EE-AD56-0F6AEE6CCFEB}

Coupling Bond: CouplingBond

Description: [Text Area]

Coupling Bond Form Tag: [Text Field]

Coupling Bond Type: [Text Field]

Coupling Pair: [Text Area] [Add] [Remove]

Engineering Constraints: [List Area] [Add] [Remove]

Degrees of Freedom: [List Area] [Impose] [Remove]

Functional Relations: [List Area] [Define] [Quant] [Spatial] [Delete]

[Apply] [OK] [Cancel]

Figure 67 Coupling bond definition user form

Functional Relations

A general functional relation is given by the expression shown in Equation 12. In the expression, the FACTOR EXPRESSION consists of operators (**opt**) and relational entities [attribute values (**AValue**), and user defined constants (**Const**)]. Note that **Const** must be defined before use by either assigning it a specific value or assigning it to a physical functionality attribute.

$$\mathbf{FACTOR\ EXPRESSION\ A\ <rel>\ FACTOR\ EXPRESSION\ B}$$

Equation 12

The “relator” is represented symbolically by “<rel>” and is defined as the mathematical relationship operator between two functional expressions. Functional relators are classified into two groups: qualitative and quantitative relator.

- Qualitative: in this implementation, the qualitative relator are limited to spatial relations between functional markers of an attribute. This set includes the following relations: against, aligned, coincident, etc.
- Quantitative: the qualitative relator defines a relation between the magnitude of the attributes or FACTORS of operands. The implementation of this is restricted to the following set: equality (=) and inequalities (\geq and \leq).

The functional relation operator co-joins attribute values and constants to form a single functional factor. Operator set includes: multiplication (*), division (/), addition (+), and subtraction (-).

The relational entities are attribute values of operands (**AValue**) or relational constants (**Const**) defined by the designer. Relational constants must be defined before use by either assigning it a specific value or assigning it to a physical functionality attribute. The user form used to define quantitative functional relations is shown in Figure 68, while that used to specify spatial (qualitative relations) is shown in Figure 69.

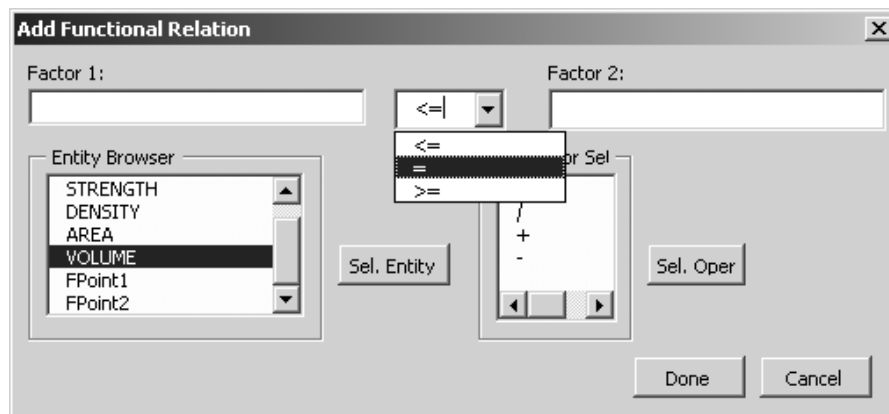
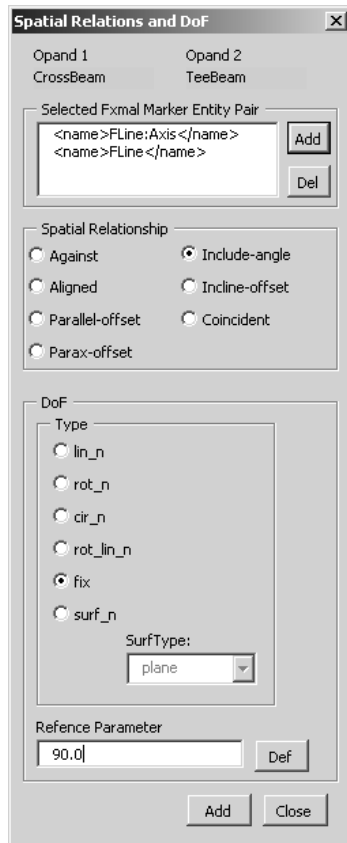
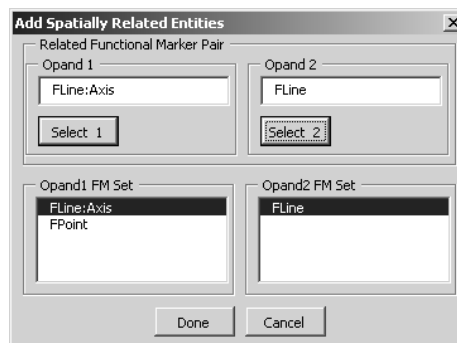


Figure 68 Quantitative functional relations definition user form



(a)

Spatial relations user form



(b)

Entities (functional markers) selection form

Figure 69 Spatial relations definition interface

Constraint representation:

Engineering constraints are special form of functional relations where the right hand side (RHS) is assigned to a constant (**Const**) value. They are used to restrict the limit an operand attribute or functional parameter can assume. Hence, it defines the value range or limits on relational entities (attribute values and constants). For instance, the weight limit of an operand may be given as follows: $mass * gravity \leq maxStrength$, where *mass* is the value of the mass attribute, *gravity* is the earth gravitational constant, and *maxStrength* is a user defined constant

for the maximum weight limit on the object. The maxStrength may be related to the maximum strength of a load bearing operand connected to the operand with the given mass. Unlike in the case of functional relators, the relator set for constraints is limited to quantitative relators: equality (=) and inequalities (\geq and \leq). The operators are also restricted to qualitative operators: multiplication (*), division (/), addition (+), and subtraction (-). The user form used to define engineering constraints is shown in Figure 70.

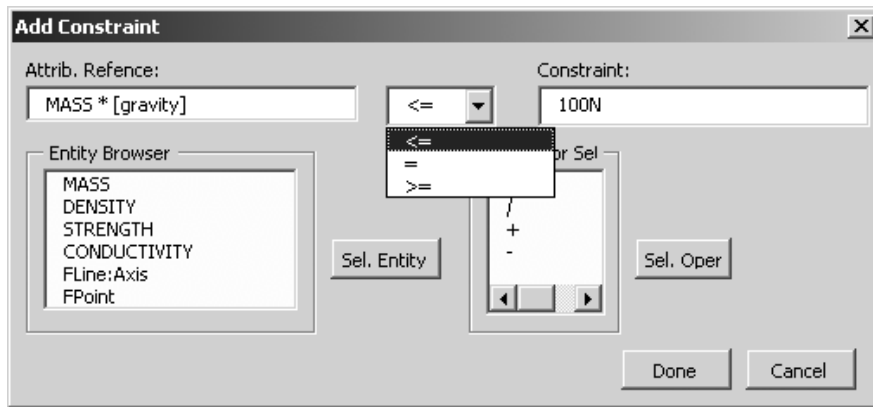


Figure 70 Quantitative constraints definition user form

Degree of Freedom

The degree of freedom (DoF) defines the required restriction on the allowable motions of an operand. The DoF specification applies to operands as an entity not just to a single attribute or operand component. A reference (<ref>) defines a reference axis, marker, or datum used in the DoF specification. It might refer to an axis, plane, or surface. The DoF is defined during the specification of the spatial relations. Hence, the input form used here is the same as that used in spatial relations specification (Figure 69).

6.3 Testing and Validation

The principles developed in this work are tested and validated using a case study: the design of a sub-assembly of an automotive space-frame. The primary activity involved in the validation of the work is answering the question, does the work accomplish the research objectives defined at the beginning of this work. To answer this question, it is important to re-emphasize the objective of a design in general and the objective of functionality modeling in particular. How is a design judged as being good or bad? In this work, a design is considered as being good if it performs successfully the task for which it was designed. The functionality-modeling tool developed in this work is used to specify the desired task (functionality) to be accomplished by a product. Hence, the models developed in this work can be validated by evaluating how efficiently they capture the functional requirements of the product.

To evaluate this modeling efficiency, the following procedure is followed.

- Selection of a case study comprising of a problem that require the use of a mechanical device.
- Definition of the need and design objective.
- Functional analysis of the selected product.
- Development of functionality model using FbD methodology.
- Evaluation of performance in the case, highlighting the benefits and limitations of the methodology.

6.3.1 Functionality-based Design (FbD) Procedure

Before applying the FbD concepts, the general FbD modeling steps identified in Section 6.2.1 is expanded to illustrate the specific design actions that are involved when using the method and computer tools developed in this work. The process flow associated with these steps is illustrated by the flow chart of Figure 71. This FbD algorithm is described below.

1. Identify the needs and specification of the proposed mechanical product.
2. Perform the functional requirement analysis of the problem (need) to identify the relevant:
 - Functionality operands and their corresponding attributes.
 - Functionality relations and constraints including both qualitative and quantitative coupling relations.
3. Start the functionality design toolkit.
4. Instantiate the functionality operation – define functionality NAME, TYPE, UNIT, and DESCRIPTION.
5. Define a new operand:
 - Click and drag from toolkit and drop in the drawing page.
 - Specify known attributes of operand including the functional markers for SOLID operands.
 - Define attribute value set if known.
6. If new operand required:
 - Repeat STEP 5.
7. Else, define coupling bond between operands:

- Select the coupling operand pair.
 - Define coupling constraints
 - Define inter-operand relations (both qualitative and quantitative relations need to be specified)
 - Define the degree of freedom restriction on the coupling pair.
8. If sub-functionalities required:
 - Define a new sub-functionality (from re-usable KB if available)
 - Specify known attributes
 - Specify coupling operands
 - Specify linkage to external functionality operands and operations (coupling bond)
 9. If new sub-functionality required:
 - Repeat sub-functionality inclusion step 9
 10. Generate XML representation of functionality
 11. Map functionality to conceptual form (tag XML to form as the functionality constraint).
 12. Save and propagate XML data file for use in the detail design phase.

The above algorithm is applied to the functionality modeling of the car frame selected as a case study product. The actual FbD modeling steps are described in Section 6.3.2, while the evaluation of the methodology is discussed in Section 6.3.3.

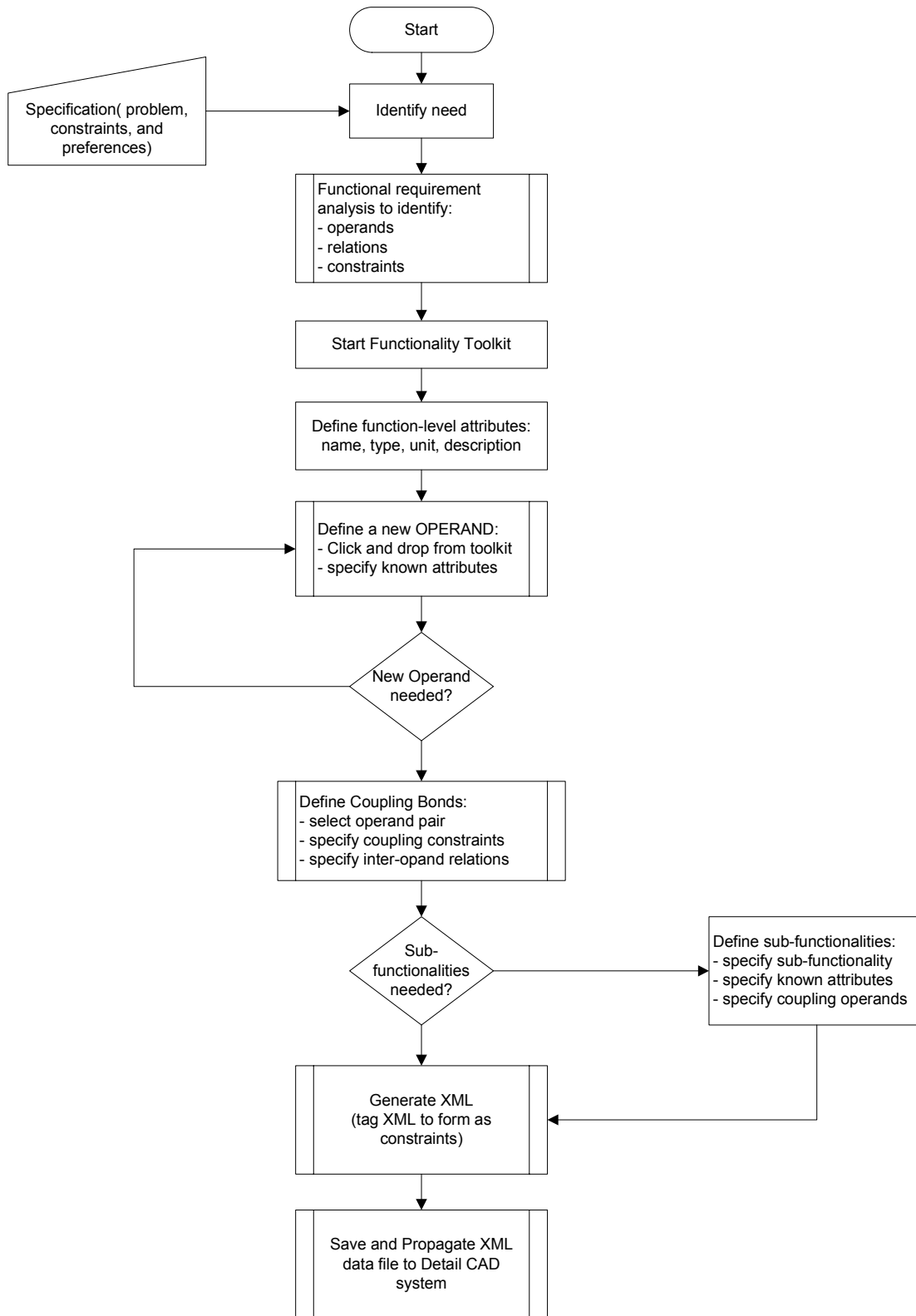


Figure 71 Functionality modeling flow diagram

2-D Approximation of Functional Markers

The demonstration of the FbD concept is restricted to two dimensions. Consequently, the functional markers described in Section 4.1.1 are modified for 2D application as follows. A functional feature is defined with markers using the following notation:

M_1 : a point $p_1(x_1, y_1)$.

M_2 : a point $p_2(x_2, y_2)$.


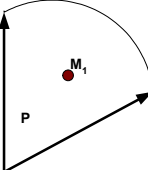
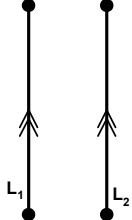
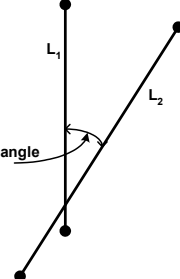
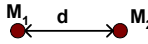
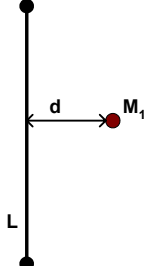
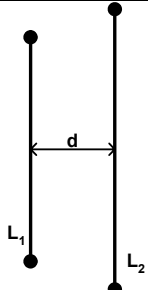
M_3 : a point $p_3(x_3, y_3)$.

L: $line(p_1, p_2)$: a line through p_1 and p_2 .

P: $plane(p_1, p_2, p_3)$: a plane through p_1, p_2 , and p_3 .

From the above definitions, the basic set of functional markers for the 2D operand approximation includes the following: a point, a line, and a plane. *Inter-functional marker constraints* are geometric constraints that describe the dependencies between functional markers within or between solid operands. The functional marker geometric constraint set simplified for 2D modeling application is summarized in Table 10.

Table 10 Inter-functional feature constraints representing geometric constraint set

<i>Constraint</i>	<i>Figure</i>
<i>on_line</i> (M_1, L): p_1 lies on the line L .	
<i>on_plane</i> (M_1, P): p_1 lies on the plane P .	
<i>parallel</i> (L_1, L_2): line L_1 is parallel to line L_2 .	
<i>angle_w</i> (L_1, L_2, θ): the angle between L_1 and L_2 is θ .	
<i>dist</i> (M_1, M_2): the distance from p_1 to p_2 is d .	
<i>dist</i> (M_1, L, d): the distance from p_1 to the projective point on line L is d .	
<i>dist</i> (L_1, L_2, d): the perpendicular distance between line L_1 and line L_2 is d .	

6.3.2 Functionality Modeling of an Automotive Space-Frame Sub-assembly

Space-frames are metal skeletons on which the body panels of a car are hung. A sample space-frame is shown in Figure 72. ^[83] The body panels are hung about the hard points, adding extra strength as well as making the structure aerodynamic and keeping the wind off the driver. Most of the space frame structure is formed from hollow section extrusions with wall thicknesses that vary to distribute stresses evenly. These components are extruded and bent into the required shape, before being assembled. Their exact form depends on how they are used, parts of the roof frame for example are shaped differently from the door pillars.

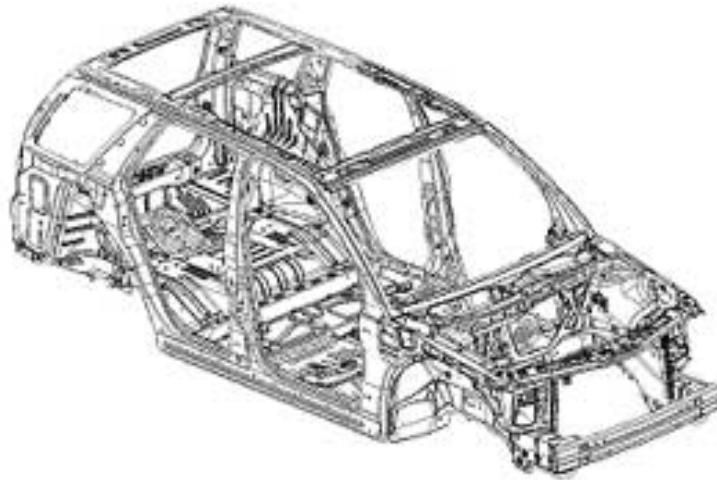


Figure 72 Space-frame of a car

The sample product used to evaluate the principles developed in this work is a sub-assembly of an automotive space-frame. This case study is evaluated by first, developing a functionality model of the frame sub-assembly using the four-step process outlined in Section

6.2.1. The functionality model is then evaluated to illustrate its goodness or deficiencies. This case evaluation is restriction to 2-dimesional analysis as defined in the scope of this research. [*]

For the analysis on the space frame, it is assumed that the exact geometric shape (or form) of the sub-assembly is not known at the beginning of the design process. This assumption makes it possible to demonstrate the design steps involved in evolving a generic product design using FbD methodology.

STEP I: Problem definition

The purpose of this step is to define the problem and the engineering task that the frame needs to accomplish. In a typical design scenario, this is provided by the customer or the target user of the product. Hence, it forms a unique interface between the engineering function of design and the customer. It is important that the proper tasks and or problems are identified at this stage of product development. A poorly defined problem will yield a product that has poor performance as it is ill-conceived with incomplete or even wrong design (engineering) objectives or goals.

The primary engineering tasks of the auto-frame are to:

- ***Maintain the structural integrity*** of the entire automotive assembly (including the body, engine assembly, wheels, transmission system, passenger sits, etc);
- ***Provide support*** for other components of the car (that is, to provide a base to which other components of the car can be mounted to); and
- ***Provide load bearing*** functionality for the car assembly (including weight of other components), its passengers, and cargoes

* The scope of this work is defined in Section 3.1. Computer implementation is restricted to 2-D. Material operands is limited to solids with rigid body.

STEP II: Functional requirement analysis

The identified engineering tasks of step I implies some functional requirements for their satisfaction. The first step in functional requirement analysis is the use of quality function deployment (QFD) ^(84, 85) matrix tool to identify the key functional features of the required product. This tool lists the engineering tasks (identified in step I) as row headings while the column headings are the functional features or attributes of the product. This is illustrated in Table 11. This table does not provide information about detailed engineering specifications, but merely identifies the set of engineering features that correspond to the required product tasks.

In implementing the functionality-based design, the first step is to translate the functionality (engineering task) into functional requirements and subsequently constraints that need to be satisfied. For the space-frame sub-assembly, the functional requirements as shown in Table 11 include the following:

- Operand set;
- Strength requirement;
- Stability requirement;
- Spatial relation of the various components; and
- Material compatibility.

Table 11 Task - functional features QFD for the car frame

	Operand	Strength	Stability	Spatial relation	Material Compatibility
Maintain Structural Integrity	⊕	⊕	⊕	⊕	
Provide Support	⊕	⊕	⊕	⊕	⊕
Provide Load Bearing	⊕	⊕	⊕	⊕	

The “⊕” symbol in Table 11 shows the perceived interrelations between the tasks and the identified functional features. In other to provide the above engineering tasks, the product must satisfy its set of functional requirements. Hence, the next step in the functional requirement analysis is the translation of the functional features into identifiable set of operands and operators (as elements of the coupling bond). The functional requirements need to be translated into functionality constraints. A satisfaction of all the functionality constraints will guarantee that under operation, the final design will meet all the initial functional requirements.

Having identified the functional requirements, the next step is to map these requirements to basic functionality design elements - operands together with their corresponding coupling bonds. The specific operands and their corresponding attributes required to achieve the design tasks are obtained by mapping the functional features to actual functionality elements (operands).

Operands and Spatial Relation:

Given the nature of the task that need to be performed and the context of operation, the identifiable operand are SOLID material and FORCE energy operands. The SOLID operand consists of the actual physical components that make up the product. The FORCE operand on

the other hand, is the set of external forces that represent the resultant impact of other components of the car assembly on the sub-frame.

Material operands:

The context of operation of the frame requires that a topological structure similar to that shown in Figure 73 be evolved for the product. This topology is dictated by the fact that other components of the car need to fit with this frame for it to provide the support and load bearing functionality. The markers (line X and points A, B, C) are used as attachment nodes to other components of the car. As can be seen from the figure, three key points are functionally needed by the context of operation as the point of attachment to other components of the car. Points A, B, and C are at a fixed distance apart and are required to maintain this constant spatial relation. Another requirement for this SOLID operand is that points A and B should be at the end of a straight line, line X.

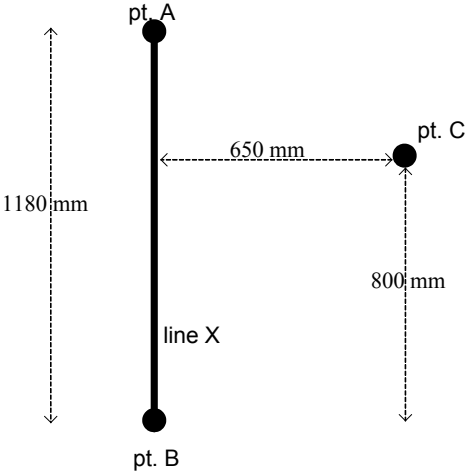


Figure 73 Solid operand features of the frame

With the above requirements in mind, points A, B, and C, and line X could all form one rigid SOLID component or they can be split into separate components and then joined (coupled) together. The SOLID operand is split into two smaller solids: $\{A, B, X\}$ and $\{C\}$. The two solids can assume any specific form provided that they satisfy the requirements of providing features A, B, C, and line X, while maintaining the required spatial relation. It is intuitive to conceptualize $\text{SOLID}\{A, B, \text{and } X\}$ as a beam with points A and B as end points as shown in wireframe model of Figure 74. On the other hand, $\text{SOLID}\{C\}$ may assume any form (option 1, option 2, or option 3) as shown in Figure 74 or it may be formed by the union of all the options to form a planer structure. The exact form of $\text{SOLID}\{C\}$ is determined by other factors such as strength, weight, cost, and designer's preference. For the initial functionality model, $\text{SOLID}\{C\}$ is assumed to be a solid beam with point C as one of its end. The other end (J_C) is connected to the AB beam.

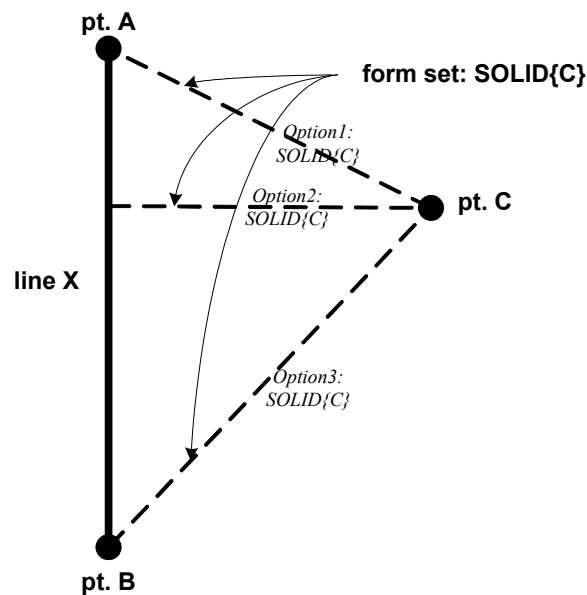


Figure 74 Solid operands: $\text{SOLID}\{A, B, X\}$ and $\text{SOLID}\{C\}$

Consequently, SOLID operands for the space-frame functionality model is composed of two solids:

- SOLID{A, B, X}: referred to as the Cross-beam (CROSSBEAM)
- SOLID{C}: referred to as the Tee-beam (TEEBEAM)

Energy operands:

The second type of operand that is required for this model is the FORCE energy operand. All vehicles are subject to both static and dynamic loads, which cause stresses. The static analysis of the structure gives, on one hand, the internal loads in the structural elements making up the structure (e.g., the direct loads, normal loads, bending moments, and torque) due to the instantaneous external loads, and on the other hand, the internal and external displacements caused by the load system. The following list of static loads, which should be taken as stressing cases, includes maximum dynamic loads, which only occur infrequently.

- Static load of stationary vehicle (including weight of components attached to the frame)
- Braking
- Acceleration
- Cornering
- Torsion
- Maximum load on front axle
- Maximum load on rear axle
- Drawbar loads

The values for the individual load cases are taken from the expected service conditions of the particular vehicle. The worst-case loading conditions (distribution of load) as well as overloading are assumed for static load case. The braking and acceleration cases are determined by the possible driving conditions. The lateral acceleration in cornering will be determined by the tire forces available, or a tilt test may be specified.

In practice, it is sufficient to use a lumped mass approximation, with simple statically equivalent masses concentrated at selected nodal points. ^[86] Discrete attached masses are connected to the structure at the nodal points. Hence, the equivalent assumed maximum loads at the node points are F_1, F_2, \dots, F_6 as shown in Figure 75. The values used for F_1, F_2, \dots, F_6 are approximate worst case forces that might impact the space-frame sub-assembly. It is important to note that the use of lumped masses is only an approximation of the real loading situation, where the loading is distributed over the entire region of the solid operand. However, in such cases, other complex methods of analysis as discussed in [Beermann 1989] ^[86] may be used to analyze the loads.

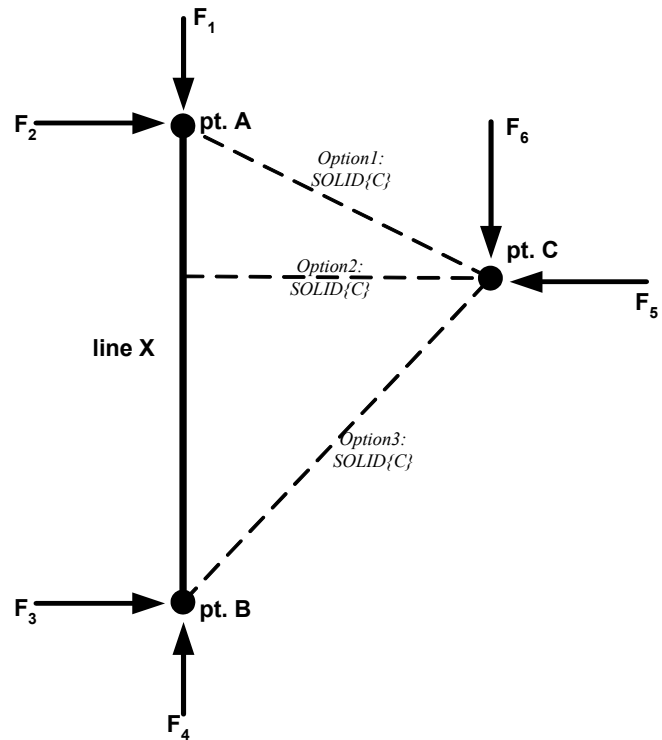


Figure 75 Resultant loading forces on the frame structure

The loading forces determine the strength and optimum spatial configuration for the load bearing components (the SOLID operands).

Strength:

The strength functional feature specifies the maximum deformation and stress that the space-frame should withstand. Under operational condition, a car is usually subjected to different kinds of forces including impact, steady, and random forces as earlier discussed under energy operands. The frame structure should be designed in such a way that it can withstand such forces. Hence, the strength functional feature refers to the following attributes of the SOLID operands: the allowable normal stress and the allowable shear stress. In this work, rigid

body solids are assumed; hence, the strain of the solid operands is neglected. This strength requirement must be specified for the entire solid frame including any joint that is required to maintain a fixed spatial relation between solid operands.

Stability:

The stability feature ensures that the design yields a product that is stable while in operation. It stipulates that the internal forces acting on the product are balanced. The internal forces are because of the SOLID operand's reaction to the external forces ($F_1, F_2, F_3, F_4, F_5, F_6$). For this stability requirement to be satisfied, the strength of the SOLID operand must be sufficiently high to withstand the external loading forces. An extension of this stability requirement is that the effect of the external forces all must cancel out as a necessary condition for equilibrium.

Material Compatibility:

Since other materials need to be attached to this frame, it has to be compatible with those components to avoid any negative compatibility issues. The components to be attached to this frame are all metals – STEEL and ALUMINUM. Hence, any compatible metal or material may be selected for the frame, subject to the satisfaction of other factors such as strength, weight, and cost. In this case study, aluminum appears to satisfy this requirement, hence ALUMINUM is proposed as an initial material type in the functionality model. It should be noted that the designer is at liberty to change the material type at any time during the design process.

STEP III: Functionality modeling

In building a functionality model, the functional features are instantiated as design objects with their corresponding attributes. This case study demonstrates how the generic functionality model discussed in Section 3.3.3 is used to map the functionality description of the car frame into mathematical relationships and constraints. It provides a means for *representing*, *propagating*, and *enforcing* car frame functionality as design constraints. The generic functionality model of the functionality operation (a modeling representation for *frameBOPN*) is defined by the ternary relation, Γ_i , given by Equation 13.

$$\Gamma_i = \{(x, s, f) \mid x \in X_i, s \in S_i, \text{ and } f \in F_i\}$$

Equation 13

Where,

- i = functionality operation index
- X_i = a functionality operator
- S_i = a set of functionality states
- F_i = function-form mapping set

Operands and attributes

Given a set of operands (O_i) in a 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_{iq} = attribute set for functionality operand o_{iq}

As identified in Step II of the FbD procedure, the operands involved in the space-frame functionality operation are listed below:

<i>j</i>	Operand Name	Operand Type
1	Cross-beam	SOLID
2	Tee-beam	SOLID
3	F₁	FORCE
4	F₂	FORCE
5	F₃	FORCE
6	F₄	FORCE
7	F₅	FORCE
8	F₆	FORCE

A detailed description of these operands and their corresponding attributes is shown in Table 12 - Table 19. In the tables, a question mark (?) is used to denote the operand attributes whose “absolute” value is either not known at conceptualization or is determined by other factors during operand coupling. This is especially true for designs in which values of attributes are not known during initial problem definition and functional requirement analysis. This procedure supports incremental decision making in design.

The strength of a solid operand depends on both the type of material, treatment, and geometry of the solid and on the type of loading the operand will experience. This allowable stress constitutes a set of strength constraint on the SOLID operands. By knowing the loading condition (type and magnitude of force) and the required strength (yield strength) on the SOLID, the geometric configuration that satisfies this strength requirement can then be determined from the strength constraint set. In a simple SOLID with uniform cross-sectional area, the stress is given by F/A , where F = applied force and A = cross-sectional area. Given a material with yield

strength, S_y , the allowable stress as discussed in Section 5.1.3 is given by the following constraints.

- Allowable normal stress (N/m^2):

$$0.45S_y \leq S_N \leq 0.60S_y$$

- Allowable shear stress (N/m^2):

$$S_S = 0.40S_y$$

Coupling relation is given as $F_{S_y} \geq F$. This relation implies that the force required to attain the yield stress of S_y must be greater than applied external loading force (F). This force (F_{S_y}) is however dependent on the geometry of the solid.

Table 12 Functionality attributes of cross-beam solid operand

Type	Fline - SOLID operand	Value
Functional markers	• Begin point markers	A
	• End point marker	B
	• Joint point marker	J_{AB}
	Intra-FM coupling	$on_line(AB)$ $dist(A, J_{AB}, d)$
Length	$dist(A, B, d)$	1180 mm
Strength	Normal (Tensile & Compression)	$0.6 S_y$
	Shear	$0.4 S_y$
Mass properties	Mass	?
	Area	?
	Volume	?
Material type	Compatible with Aluminum and Steel	ALUMINUM
DoF		Fixed

Table 13 Functionality attributes of tee-beam solid operand

Type	Fline - SOLID operand	Value
Functional markers	• Begin point markers	C
	• End point marker	J _C
	• Joint point marker	J _C
Length	<i>dist</i> (C, J _C , d)	[650.00, 1030.78] mm
Strength	Normal (Tensile & Compression)	0.6 S _y
	Shear	0.4 S _y
Mass properties	Mass	?
	Area	?
	Volume	?
Material type	Compatible with Aluminum and Steel	ALUMINUM
DoF		Fixed

In computing the value of the external loading forces, the impact forces (as a result of crash) were neglected. The space-frame is subjected to only steady/static loading. This restriction is necessary as the models developed here are based solely on the assumption of rigid bodies. Inclusion of impact forces (say from a crash) will necessitate an extension of the model to consider energy-absorbing beams, subject to deformation. It is believed that this approximation is sufficient to demonstrate the application of the concepts developed in this work. The attributes of the loading forces are shown Table 14 – Table 19. The magnitudes of the forces are only approximate estimations (since the impact forces are excluded) large enough to illustrate application of FbD methodology.

Table 14 Functionality attributes of F_1 force-energy operand

Type	F _{Energy} – ENERGY FORCE operand	
Magnitude	$ F_1 $	50,000 N
Direction	A → B; to FlineAB.	[def@coupling with reference to cross-beam]
Point of contact	Point B on Crossbeam	[def@coupling with reference to cross-beam]
Source	External loads	External loads and components assembled to the frame.
Nature	External forces operands	Steady
DoF		Fixed

Table 15 Functionality attributes of F_4 force-energy operand

Type	F _{Energy} – ENERGY FORCE operand	
Magnitude	$ F_4 $	50,000 N
Direction	A → B; to FlineAB	[def@coupling with reference to cross-beam]
Point of contact	Point B on Crossbeam	[def@coupling with reference to cross-beam]
Source	External loads	External loads and components assembled to the frame.
Nature	External forces operands	Steady
DoF		Fixed

Table 16 Functionality attributes of F_2 force-energy operand

Type	F _{Energy} – ENERGY FORCE operand	
Magnitude	$ F_2 $	50,000 N
Direction	From side 1; ⊥ to FlineAB	[def@coupling with reference to cross-beam]
Point of contact	Point A on Crossbeam	[def@coupling with reference to cross-beam]
Source	External loads	External loads and components assembled to the frame.
Nature	External forces operands	Steady
DoF		Fixed

Table 17 Functionality attributes of F_3 force-energy operand

Type	F _{Energy} – ENERGY FORCE operand	
Magnitude	$ F_3 $	50,000 N
Direction	From side 1; ⊥ to FlineAB.	[def@coupling with reference to cross-beam]
Point of contact	Point B on Crossbeam	[def@coupling with reference to cross-beam]
Source	External loads	External loads and components assembled to the frame.
Nature	External forces operands	Steady
DoF		Fixed

Table 18 Functionality attributes of F_5 force-energy operand

Type	F _{Energy} – ENERGY FORCE operand	
Magnitude	$ F_5 $	50,000 N
Direction	From side 2; ⊥ to FlineAB	[def@coupling with reference to cross-beam and tee-beam]
Point of contact	Point C on Teebeam	[def@coupling with reference to tee-beam]
Source	External loads	External loads and components assembled to the frame.
Nature	External forces operands	Steady
DoF		Fixed

Table 19 Functionality attributes of F_6 force-energy operand

Type	F _{Energy} – ENERGY FORCE operand	
Magnitude	$ F_6 $	50,000 N
Direction	From side 2; ⊥ to FlineAB	[def@coupling with reference to cross-beam and tee-beam]
Point of contact	Point C on Teebeam	[def@coupling with reference to tee-beam]
Source	External loads	External loads and components assembled to the frame.
Nature	External forces operands	Steady
DoF		Fixed

Coupling bonds (CB)

Having identified the operands in the sub-frame, the next task is to define the interaction between them to yield the space-frame functionality identified in Step I. Using the notation presented in Section 4.3, this functionality is symbolically represented as:

$$\mathit{frameBOPN}: \{ \langle o_1 : a_1 \rangle, \langle o_1 : a_2 \rangle, \langle o_i : a_j \rangle, \dots \langle o_8 : a_8 \rangle \}$$

where,

$\mathit{frameBOPN}$ = space-frame functionality operator

$\langle o_j \rangle$ = functionality operand j ,

$\{ a_j \}$ = attributes of operand j

i, j = 1, 2, ..., 8 (operand indices)

The operand indices and corresponding operands are listed below. These indices will be used to refer to each operand in the remainder of this case study.

j	Operand name
1	Cross-beam
2	Tee-beam
3	F₁
4	F₂
5	F₃
6	F₄
7	F₅
8	F₆

With the operand set known, the functionality relationship is defined by a functional relationship set (\mathbf{X}_i) as given in Equation 14.

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

Equation 14

Where

- \mathbf{O}_i = set of operands in functionality operation i .
- \mathbf{x}_{ijk} = coupling bond between functionality operands j and k .
- j, k = functionality operand indices

The coupling bond (\mathbf{x}_{ijk}) defines the active relationships between the functionality operands j and k . It is the coupling condition that describes the nature of interaction between operands.

In computer modeling, the relationship set (\mathbf{X}_i) is represented by a matrix, *functionality coupling bond (CB) matrix*, $\mathbf{X}_{m \times n}$. Since there are eight operands in the space-frame functionality, the dimension of CB matrix is 8 by 8. The coupling bond has three components that constitute the coupling conditions: DoF, relations, and constraints. Thus, it gives the set of relationships, constraints, and DoF that must hold in the inter-operand coupling for the functionality to achieve the specified task. The coupling bond is defined by including these three components in the coupling bond equation as shown in Equation 8.

$$\mathbf{x}_{ij} = \{r_{ij}, c_{ij}, d_{ij}\} \otimes y_{ij}$$

Equation 15

Where

r_{ij} = relation between operands i and j .

c_{ij} = constraint on relation between operands i and j .

d_{ij} = degree of freedom on relation between operands i and j .

$$y_{ij} = \begin{cases} 1 & \text{if (coupling exist between operands } i \text{ and } j) \text{ and } (i \geq j) \\ 0 & \text{otherwise} \end{cases}$$

The entire set of coupling bonds in a functionality operation is represented by the coupling bond matrix as follows:

$$X = [x_{ij}], \text{ where } i = 1, 2, 3, \dots, 8; \text{ and } j = 1, 2, 3, \dots, 8$$

given that,

$$R = [r_{ij}] \text{ and } I = [y_{ij}]$$

subject to,

$$C = [c_{ij}]$$

$$D = [d_{ij}]$$

In a full matrix notation, X is given by:

$$\begin{matrix} & OP_1 & OP_2 & \cdots & OP_8 \\ OP_1 & \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{18} \\ 0 & x_{22} & \cdots & x_{28} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_{88} \end{bmatrix} \end{matrix}$$

Where:

OP_k ($k = 1, 2, \dots, 8$) are the functionality operands.

x_{ij} ($i, j = 1, 2, \dots, 8$) are the coupling bounds between operand OP_i and operand OP_j .

\mathbf{I} is a zero-one square linkage matrix representing the valid linkage interactions for the given coupling operation. The \mathbf{I} (linkage matrix) is derived by entering a one for each matrix element where a valid coupling exists between the operands and a zero is entered otherwise. The size of the \mathbf{I} matrix is determined by the number of interacting operands. For the space-frame, the functionality interaction graph is depicted in Figure 76. Since there are eight interacting operands, \mathbf{I} is a 8x8 square matrix. Hence the \mathbf{I} matrix is given by:

$$I = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

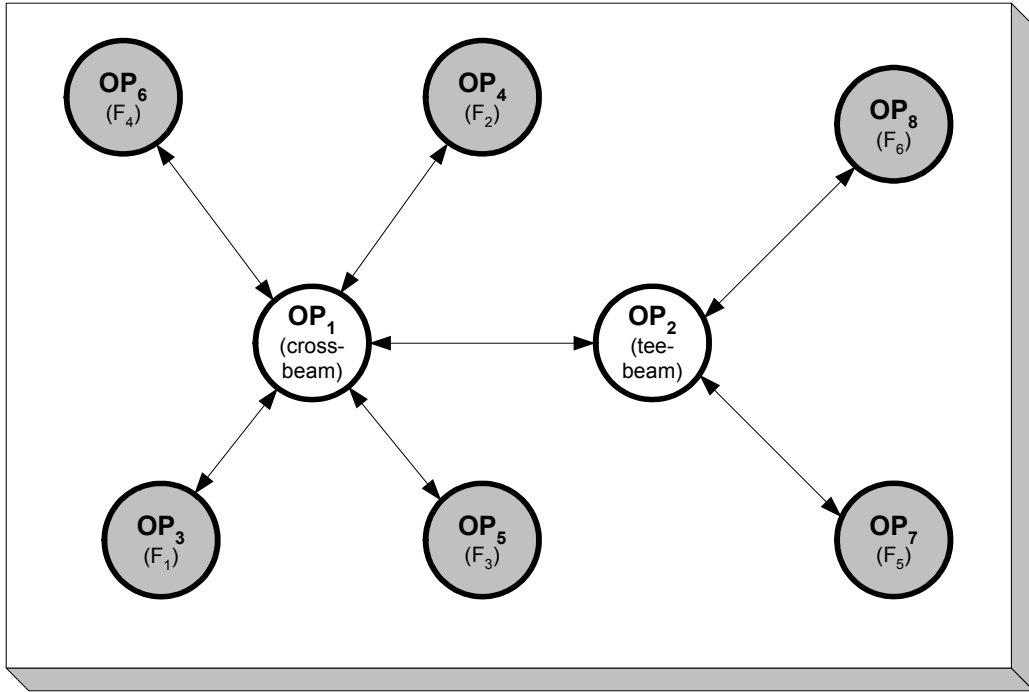


Figure 76 Car frame operand interaction graph

(Note: Shaded FORCE operands are involved in a global coupling to ensure the stability of the system)*

The other variables in the equation are derived as follows for the example. Applying the linkage matrix, **I**, gives the **X** matrix as:

* For equilibrium, sum of forces must equal zero and sum of moments must equal zero.

$$X_{8 \times 8} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & 0 & 0 \\ 0 & x_{22} & 0 & 0 & 0 & 0 & x_{27} & x_{28} \\ 0 & 0 & x_{33} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{44} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_{66} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_{77} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{88} \end{bmatrix}$$

Where, the diagonal elements, x_{11} , x_{22} , x_{33} , x_{44} , x_{55} , x_{66} , x_{77} , and x_{88} , are the intra-operand constraints and relations. They include the constraints on the attributes of the corresponding operands; degree of freedom restriction on the functional markers and attributes of the operands, and finally relations between attributes of the operand. Some of the known attribute values for the space-frame are specified in Table 12 to Table 19 while others that are dependant on other operands are defined during coupling. The meaning of the other non-zero elements of the coupling bond matrix is described in Table 20.

Table 20 Coupling bond elements

CB element				
x_{12}	<i>SOLID-SOLID</i>	Crossbeam – Teebeam	Joint	Table 21
x_{13}	<i>SOLID-FORCE</i>	Crossbeam – F ₁	Compression	Table 22
x_{14}	<i>SOLID-FORCE</i>	Crossbeam – F ₂	Shear	Table 23
x_{15}	<i>SOLID-FORCE</i>	Crossbeam – F ₃	Shear	Table 24
x_{16}	<i>SOLID-FORCE</i>	Crossbeam – F ₄	Compression	Table 25
x_{27}	<i>SOLID-FORCE</i>	Teebeam – F ₅	Compression	Table 26
x_{28}	<i>SOLID-FORCE</i>	Teebeam – F ₆	Shear	Table 27
<i>global</i>	<i>FORCE-FORCE</i>	F ₁ , F ₂ , F ₃ , F ₄ , F ₅ , F ₆	Equilibrium	Table 28

The CB matrix (\mathbf{X}) for the automobile sub-frame can be expanded in terms of \mathbf{R} , \mathbf{C} , and \mathbf{D} as follows. The \mathbf{R} matrix is formed by pulling all the valid relationships in the system involving operands OP_1 to OP_8 . Similarly, \mathbf{C} and \mathbf{D} matrices are formed by pulling all the valid constraints and DoF restrictions respectively in the system involving operands OP_1 to OP_8 . Hence,

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} & 0 & 0 \\ 0 & r_{22} & 0 & 0 & 0 & 0 & r_{27} & r_{28} \\ 0 & 0 & r_{33} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{44} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & r_{66} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{77} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{88} \end{bmatrix}$$

subject to

$$X = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} & 0 & 0 \\ 0 & c_{22} & 0 & 0 & 0 & 0 & c_{27} & c_{28} \\ 0 & 0 & c_{33} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{44} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{66} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{77} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{88} \end{bmatrix}$$

and

$$\begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} & d_{16} & 0 & 0 \\ 0 & d_{22} & 0 & 0 & 0 & 0 & d_{27} & d_{28} \\ 0 & 0 & d_{33} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d_{44} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & d_{66} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d_{77} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{88} \end{bmatrix}$$

A detailed listing of the CB components (x_{ij}) for the car frame is listed in Table 21 to Table 27. Figure 77 and Figure 78 illustrates the effect of coupling bond (CB) relations on the conceptual form of the space-frame. Figure 79(a)-(b) is the three members of the possible form set of the space frame. [*] For evaluation purposes, the form shown in Figure 79 (b) is selected.

Table 21 Joint operation: crossbeam-teebeam coupling bond (x_{12})

Functionality Entity	Entity 1 ($j=1$)	Entity 2 ($k=2$)
Coupling Pair (O_{1q})	O_{11} = SOLID: <i>Crossbeam</i>	O_{12} = SOLID: <i>Teebeam</i>
Attributes (A_{1q})	A_{111} = length, A_{112} = FM set A_{113} = material type	A_{121} = length A_{122} = FM set A_{123} = material type
Coupling Bond Relations (X_l)	Functional Constraints	
	$C_1 = \{dist(A, C, d_1) = 752.93 \text{ mm}\}$ $C_2 = \{dist(B, C, d_2) = 1030.78 \text{ mm}\}$ $C_3 = \{C \text{ on side2}\}$ $C_4 = \{dist(A, J_{AB}, d_3) = fixed\}$ $C_5 = \{strength_joint \geq min_strength(crossbeam, teebeam)\}$	The effect of C1, C2, and C3 is shown in Figure 77.
	Functional Relations	
	$R_1 = \{J_{AB} \text{ coincident } J_C\}$ $R_2 = \{J_{AB} \text{ on_line}(AB)\}$	The effect of C1, C2, and C3 is shown in Figure 78.
	Degree of Freedom	
	$D_1 = \{J_{AB} : fix\}$ $D_2 = \{crossbeam : fix\}$ $D_3 = \{teebeam : fix\}$	
Functionality-form Mapping (F_l)	$F_l = \{(r_{112}, f_{112}) \mid FF_{l=1}, FF_{l=2}, \dots\}, l = 1, 2, 3.$ Member elements shown in Figure 79.	

* It is possible to expand the form set to include form conceptual form by considering the continuous nature of the possible location of marker J_{AB} on the cross-beam.

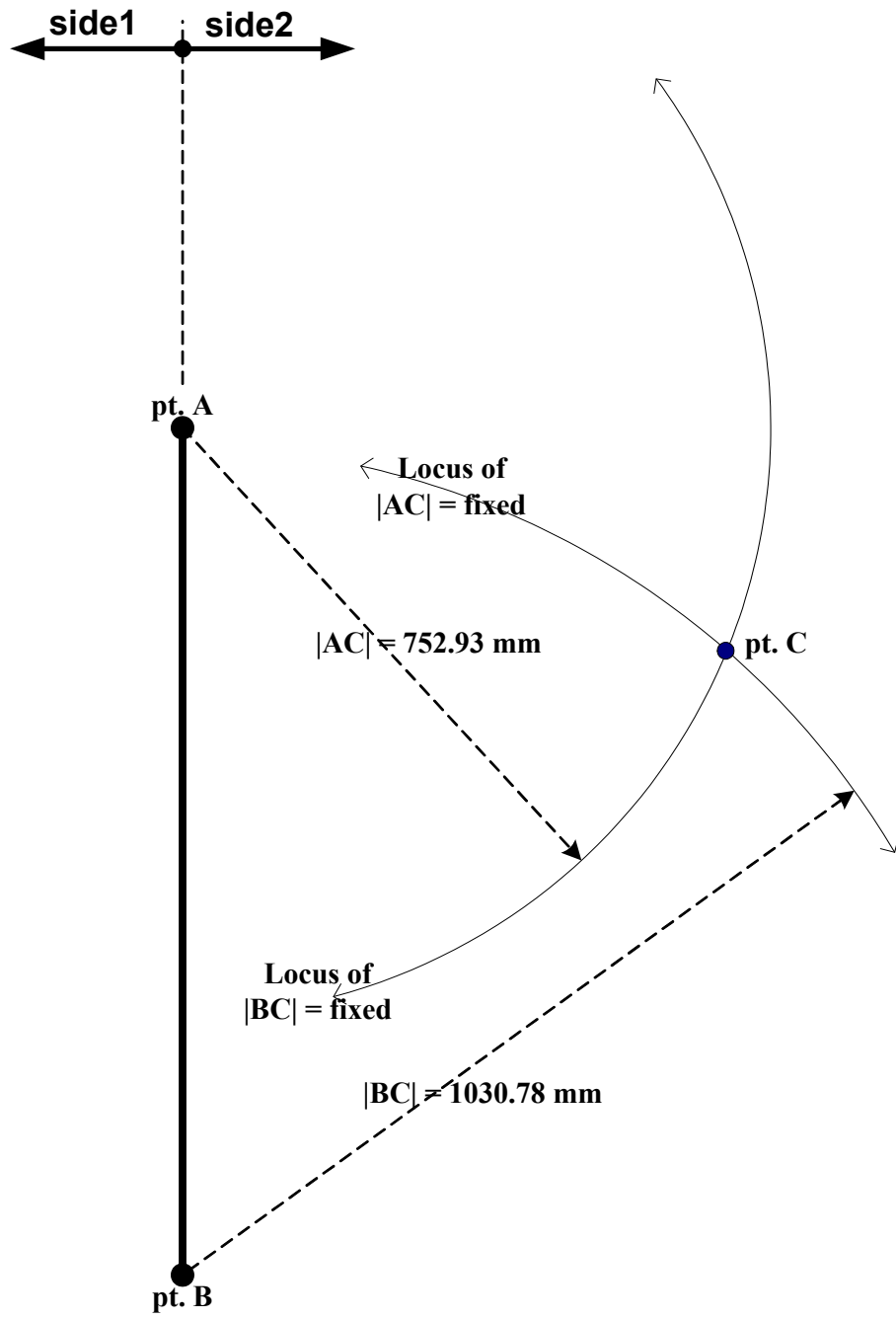


Figure 77 Application of constraints on functional markers A, B, and C

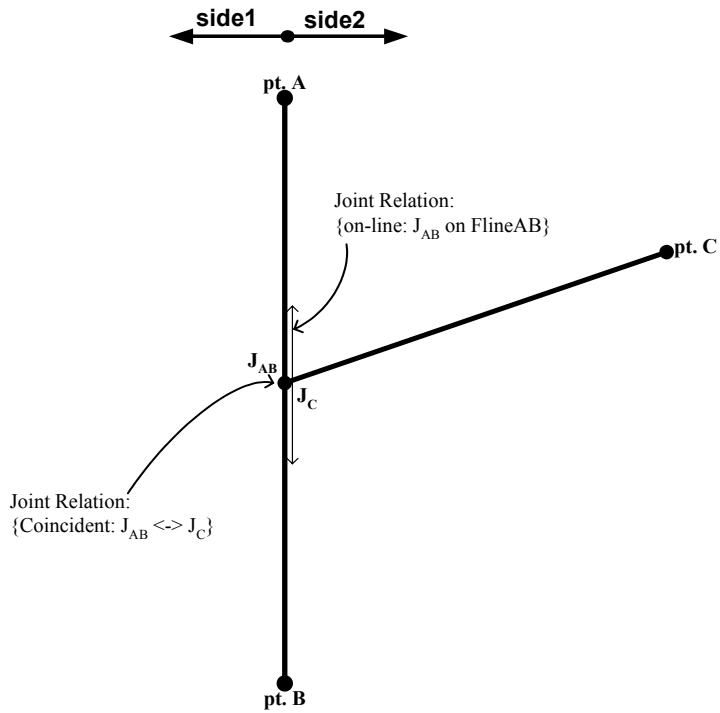


Figure 78 Application of constraint on joint location

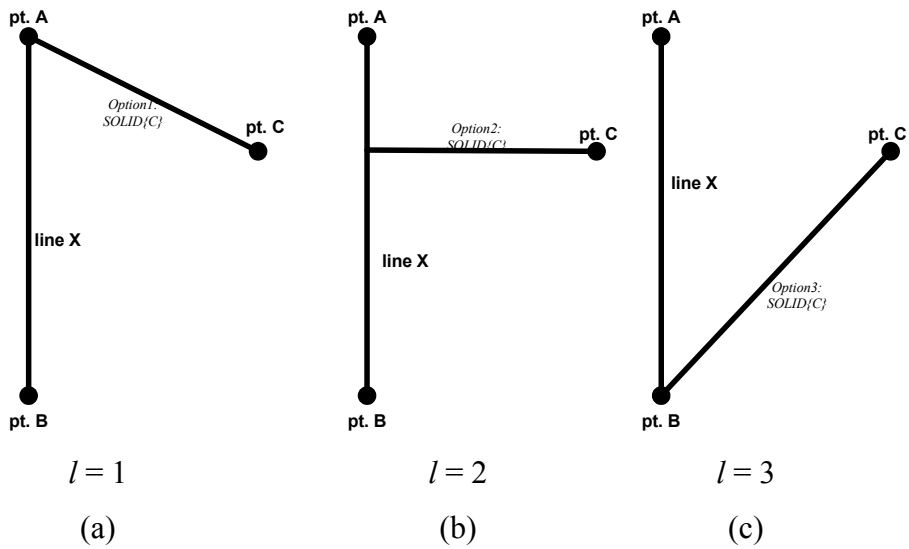


Figure 79 Conceptual form set for joint operation ($l = 1, 2, 3$)

Table 22 to Table 27 shows the coupling bond for the various loading forces on the *cross* and *tee* beams. It should be recalled that stress is given as a ratio of force to cross-sectional area. Since stress depends on cross-sectional area, for a solid operand with varying cross-section, the stress varies along the loading axis. Hence, to ensure the satisfaction of the stress requirement, the instantaneous internal forces (as a result of the stress) along the *operand* must be greater than the applied load.

Let

$\mathbf{F}_{\text{internal}}$ = instantaneous internal force;

$\mathbf{F}_{\text{applied}}$ = loading force;

A = cross-sectional area; and

σ = normal stress.

The *normal strength* requirement of the solid operand require that the following constraint be satisfied:

$$\mathbf{F}_{\text{internal}} \geq \mathbf{F}_{\text{applied}}$$

$$\text{Where, } \mathbf{F}_{\text{internal}} = A \times \sigma$$

For a given material of known yield stress (S_y), the allowable stress $\sigma_{\text{allow}} = 0.65S_y$. Consequently, the maximum allowable internal force (\mathbf{F}_{S_y}) in the solid operand if given by substituting this into the constraints to yields:

$$\mathbf{F}_{S_y} = A \times \sigma_{\text{allow}} \geq \mathbf{F}_{\text{applied}}$$

Hence, the strength coupling constraint between a solid and force operand is given as $\mathbf{F}_{Sy} \geq \mathbf{F}_{\text{applied}}$. This relation implies that the force required to attain the allowable stress of σ_{allow} must be greater than applied external loading force ($\mathbf{F}_{\text{applied}}$). This force (\mathbf{F}_{Sy}) is however dependent on the geometry of the solid.

In the following tables, *normal_strength_operand* refers to the design limit of maximum internal forces (\mathbf{F}_{Sy}) in the solid operands under normal loading condition. Similarly, *shear_strength_operand* refers to the design limit of maximum internal forces in the solid operands under shear loading condition.

Table 22 Load bearing: crossbeam- \mathbf{F}_1 coupling bond (x_{13})

Functionality Entity	Entity 1 ($k = 1$)	Entity 2 ($j=3$)
Coupling Pair (\mathbf{O}_{1q})	$\mathbf{O}_{12} = \text{SOLID: Crossbeam}$	$\mathbf{O}_{11} = \text{FORCE: } \mathbf{F}_1$
Attributes (\mathbf{A}_{1q})	$\mathbf{A}_{121} = \text{length}$ $\mathbf{A}_{122} = \text{normal_strength}$ $\mathbf{A}_{123} = \text{FM: A}$	$\mathbf{A}_{111} = \mathbf{F}_1 $ $\mathbf{A}_{112} = \text{dir}(\mathbf{F}_1)$ $\mathbf{A}_{113} = \text{nature}$ $\mathbf{A}_{114} = \langle \text{applic point} \rangle$
Coupling Bond Relations (\mathbf{X}_j)	Constraints: $C_1 = \{ \text{normal_strength_crossbeam} \geq \mathbf{F}_1 \}$ Relations: $R_1 = \{ \langle \text{applic point} \rangle \text{ coincident A} \}$ $R_2 = \{ \text{dir}(\mathbf{F}_1) \text{ parallel AB} \}$ DoF: $D_1 = \{ \langle \text{A} \rangle : \text{fix} \}$ $D_2 = \{ \text{dir}(\mathbf{F}_1) : \text{fix} \}$	

Table 23 Load bearing: crossbeam-F₂ coupling bond (x₁₄)

Functionality Entity	Entity 1 (j = 1)	Entity 2 (k = 3)
Coupling Pair (O _{1q})	O ₁₂ = SOLID: <i>Crossbeam</i>	O ₁₁ = FORCE: <i>F₂</i>
Attributes (A _{1q})	A ₁₂₁ = <i>length</i> A ₁₂₂ = <i>shear_strength</i> A ₁₂₃ = FM: A	A ₁₁₁ = <i>F₂</i> A ₁₁₂ = dir(<i>F₂</i>) A ₁₁₃ = <i>nature</i> A ₁₁₄ = < <i>applic point</i> >
Coupling Bond Relations (X ₁)	Constraints: $C_1 = \{shear_strength_crossbeam \geq F_2 \}$ Relations: $R_1 = \{<applic\ point> \text{ coincident } A\}$ $R_2 = \{angle_w(dir(F_2), AB, 90)\}$ DoF: $D_1 = \{<A>: fix\}$ $D_2 = \{dir(F_2) : fix\}$	

Table 24 Load bearing: crossbeam-F₃ coupling bond (x₁₅)

Functionality Entity	Entity 1 (j = 1)	Entity 2 (k = 5)
Coupling Pair (O _{1q})	O ₁₂ = SOLID: <i>Crossbeam</i>	O ₁₁ = FORCE: <i>F₃</i>
Attributes (A _{1q})	A ₁₂₁ = <i>length</i> A ₁₂₂ = <i>shear_strength</i> A ₁₂₃ = FM: B	A ₁₁₁ = <i>F₃</i> A ₁₁₂ = dir(<i>F₃</i>) A ₁₁₃ = <i>nature</i> A ₁₁₄ = < <i>applic point</i> >
Coupling Bond Relations (X ₁)	Constraints: $C_1 = \{shear_strength_crossbeam \geq F_3 \}$ Relations: $R_1 = \{<applic\ point> \text{ coincident } B\}$ $R_2 = \{angle_w(dir(F_3), AB, 90)\}$ DoF: $D_1 = \{: fix\}$ $D_2 = \{dir(F_3) : fix\}$	

Table 25 Load bearing: crossbeam-F₄ coupling bond (x₁₆)

Functionality Entity	Entity 1 (j = 1)	Entity 2 (k=6)
Coupling Pair (O _{1q})	O ₁₂ = SOLID: <i>Crossbeam</i>	O ₁₁ = FORCE: <i>F₄</i>
Attributes (A _{1q})	A ₁₂₁ = <i>length</i> A ₁₂₂ = <i>normal_strength</i> A ₁₂₃ = FM: A	A ₁₁₁ = <i>F₄</i> A ₁₁₂ = dir(<i>F₄</i>) A ₁₁₃ = <i>nature</i> A ₁₁₄ = < <i>applic point</i> >
Coupling Bond Relations (X ₁)	Constraints: $C_1 = \{normal_strength_crossbeam \geq F_4 \}$ Relations: $R_1 = \{<applic\ point> \text{ coincident } B\}$ $R_2 = \{dir(F_4) \text{ parallel } AB\}$ DoF: $D_1 = \{: fix\}$ $D_2 = \{dir(F_4) : fix\}$	

Table 26 Load bearing: teebeam-F₅ coupling bond (x₂₇)

Functionality Entity	Entity 1 (j = 2)	Entity 2 (k=4)
Coupling Pair (O _{1q})	O ₁₂ = SOLID: <i>Teebeam</i>	O ₁₁ = FORCE: <i>F₅</i>
Attributes (A _{1q})	A ₁₂₁ = <i>length</i> A ₁₂₂ = <i>normal_strength</i> A ₁₂₃ = <i>shear_strength</i> A ₁₂₄ = FM: C	A ₁₁₁ = <i>F₅</i> A ₁₁₂ = dir(<i>F₅</i>) A ₁₁₃ = <i>nature</i> A ₁₁₄ = < <i>applic point</i> >
Relation (R ₁)	Constraints: $C_1 = \{normal_strength_teebeam \geq F_5 _{normal}\}$ $C_2 = \{shear_strength_teebeam \geq F_5 _{shear}\}$ Relations: $R_1 = \{<applic\ point> \text{ coincident } C\}$ $R_2 = \{angle_w(dir(F_5), AB, 90)\}$ DoF: $D_1 = \{<C>: fix\}$ $D_2 = \{dir(F_5) : fix\}$	

Table 27 Load bearing: teebeam- F_6 coupling bond (x_{28})

Functionality Entity	Entity 1 ($j = 2$)	Entity 2 ($k=5$)
Coupling Pair (O_{ij})	$O_{12} = \text{SOLID: Teebeam}$	$O_{11} = \text{FORCE: } F_6$
Attributes (A_{1q})	$A_{121} = \text{length}$ $A_{122} = \text{normal_strength}$ $A_{123} = \text{shear_strength}$ $A_{124} = \text{FM: C}$	$A_{111} = F_6 $ $A_{112} = \text{dir}(F_6)$ $A_{113} = \text{nature}$ $A_{114} = \langle \text{applic point} \rangle$
Relation (R_I)	Constraints: $C_1 = \{ \text{normal_strength_teebeam} \geq F_6 _{\text{normal}} \}$ $C_2 = \{ \text{shear_strength_teebeam} \geq F_6 _{\text{shear}} \}$ Relations: $R_1 = \{ \langle \text{applic point} \rangle \text{ coincident C} \}$ $R_2 = \{ \text{dir}(F_6) \text{ parallel AB} \}$ DoF: $D_1 = \{ \langle C \rangle : \text{fix} \}$ $D_2 = \{ \text{dir}(F_6) : \text{fix} \}$	

FORCE-FORCE coupling bonds

Force-force coupling is necessary for stability and equilibrium of the frame. However, since these forces are external to the frame, they are related to each other with reference to other coupled operands that serve as source to these forces. The actual sources of these forces are neglected in this coupling.

The aim of a static analysis is to determine the internal loads of a structure when subjected to external loads. The basis for this analysis is that the equilibrium of forces shall be maintained at all points in the structure. The conditions of equilibrium are sufficient for the direct analysis and guarantee of equilibrium of statically determinate structures. The equilibrium conditions for the forces F_k in the coordinate directions and the moments M_k acting about these coordinates give:

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

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

Where,

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

It is preferable to define these forces at the nodal points or the joints. This results in the forces being defined at the ends of the beams. The analysis finds the internal loads at any point along the beam. The internal loads are given as stress distributions in the cross section of the beam.

In the case of the space-frame, the structure breaks up naturally into beam elements with the ends as nodes (Figure 75). At each node, there are two forces and two moments (2D coordinate system assumed). These forces and moments are referred to as the loads at a node. For the k th node the loads are arranged in column matrices:

$$\mathbf{f}_k = \{X_k \ Y_k \ M_{xk} \ M_{yk}\}$$

And for the complete structure

$$\mathbf{f} = \{ \mathbf{f}_1 \ \mathbf{f}_2 \ \dots \ \mathbf{f}_k \ \dots \ }$$

To ensure the stability of the space-frame, all the external loading forces are coupled through a global coupling bond that defines the equilibrium conditions as constraints on the product functionality. This coupling bond is shown in Table 28.

Table 28 Equilibrium constraint: global coupling (F₁, F₂, F₃, F₄, F₅, F₆)

<i>Functionality Entity</i>	
Coupling operands (O_{ij})	O₁₁ = (F ₁); O₁₂ = (F ₂); O₁₃ = (F ₃); O₁₄ = (F ₄); O₁₅ = (F ₅); O₁₆ = (F ₆)
Attributes (A_{iq})	A_{1i1} = F , A_{1i2} = dir, A_{1i4} = applic point
Coupling Bond Relations (X_l)	<p>Force Constraints:</p> $C_1 := \{ \sum_{\forall X} F_k = 0 \mid F_2 + F_3 + F_5 = 0 \}$ $C_2 := \{ \sum_{\forall Y} F_k = 0 \mid F_1 + F_4 + F_6 = 0 \}$ <p>Moment Constraints:</p> $C_3 := \{ \sum M_k = 0 \mid \sum M_A = 0; \sum M_B = 0; \sum M_C = 0 \}$
Functionality-form Mapping (F_l)	

Computer implementation

Screen captures of the computer implementation of the functionality model for the space-frame structure is shown in Figure 80 to Figure 83. Figure 80 is the start-up screen of the Visio interface for inputting the name of the functionality operation. . Figure 81 is a screen capture of a wire frame model of the Crossbeam solid operand. Figure 82 is a screen capture of the operand interaction graph of the space-frame. Figure 83 is a screen capture of a wire frame model of the coupled solid operands.

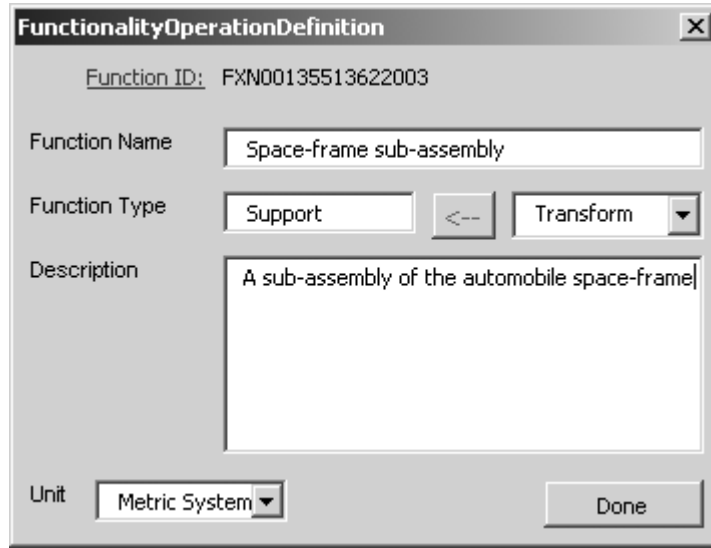


Figure 80 Dialog box showing the start-up screen of the Visio interface

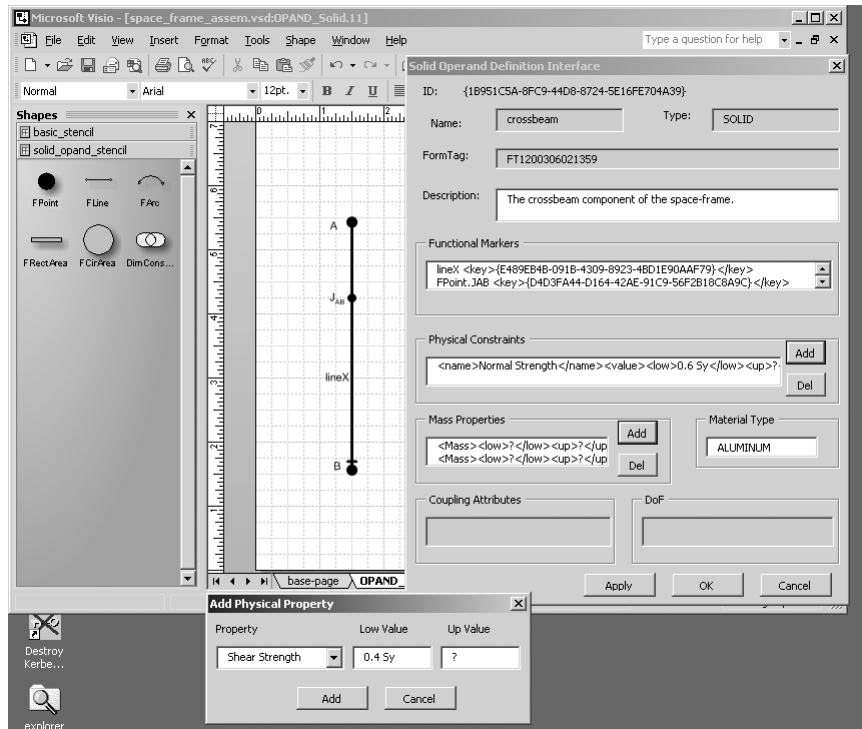


Figure 81 Screen capture of a wireframe model of the crossbeam solid operand

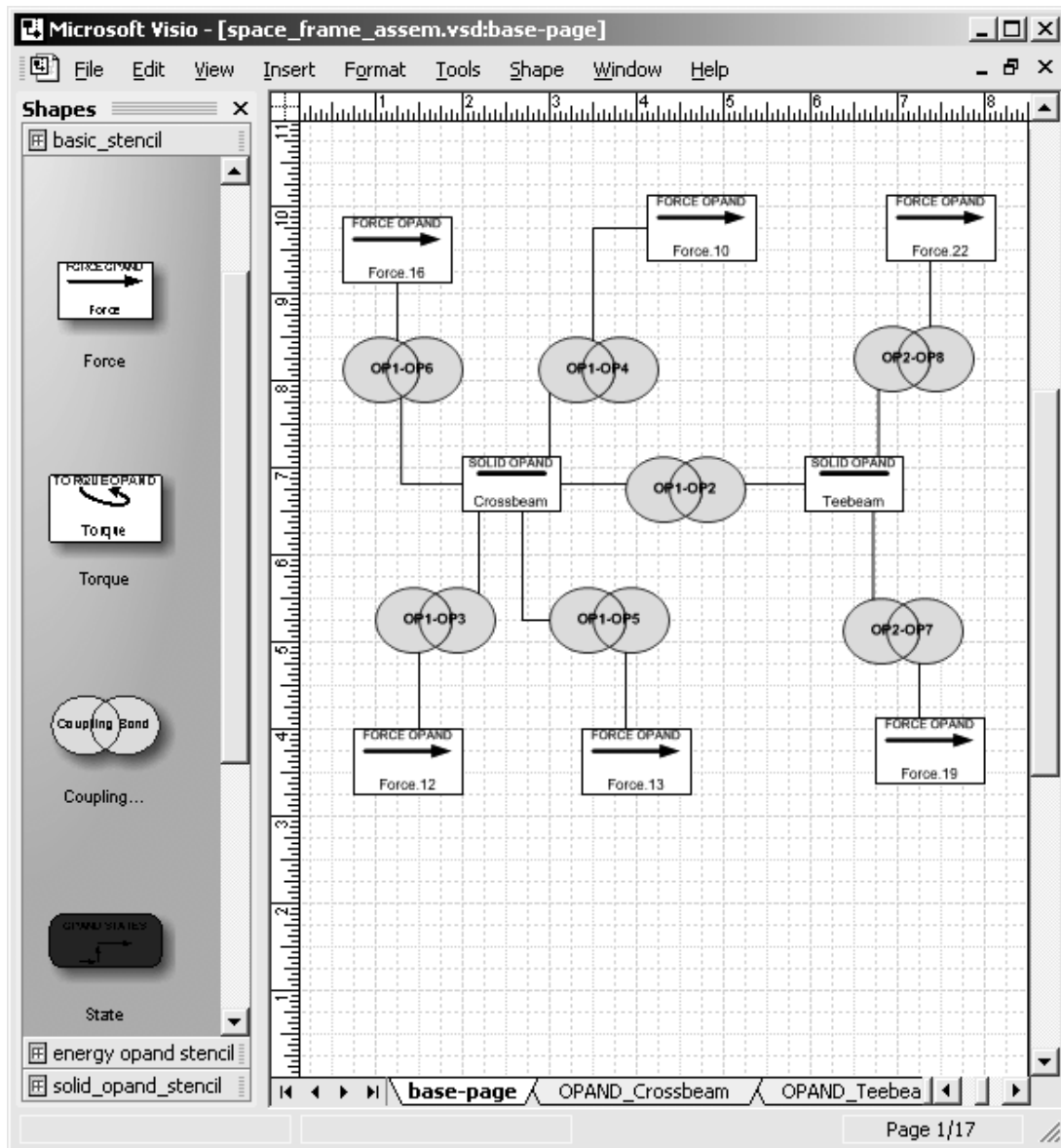


Figure 82 Screen capture of the space-frame operand interaction graph

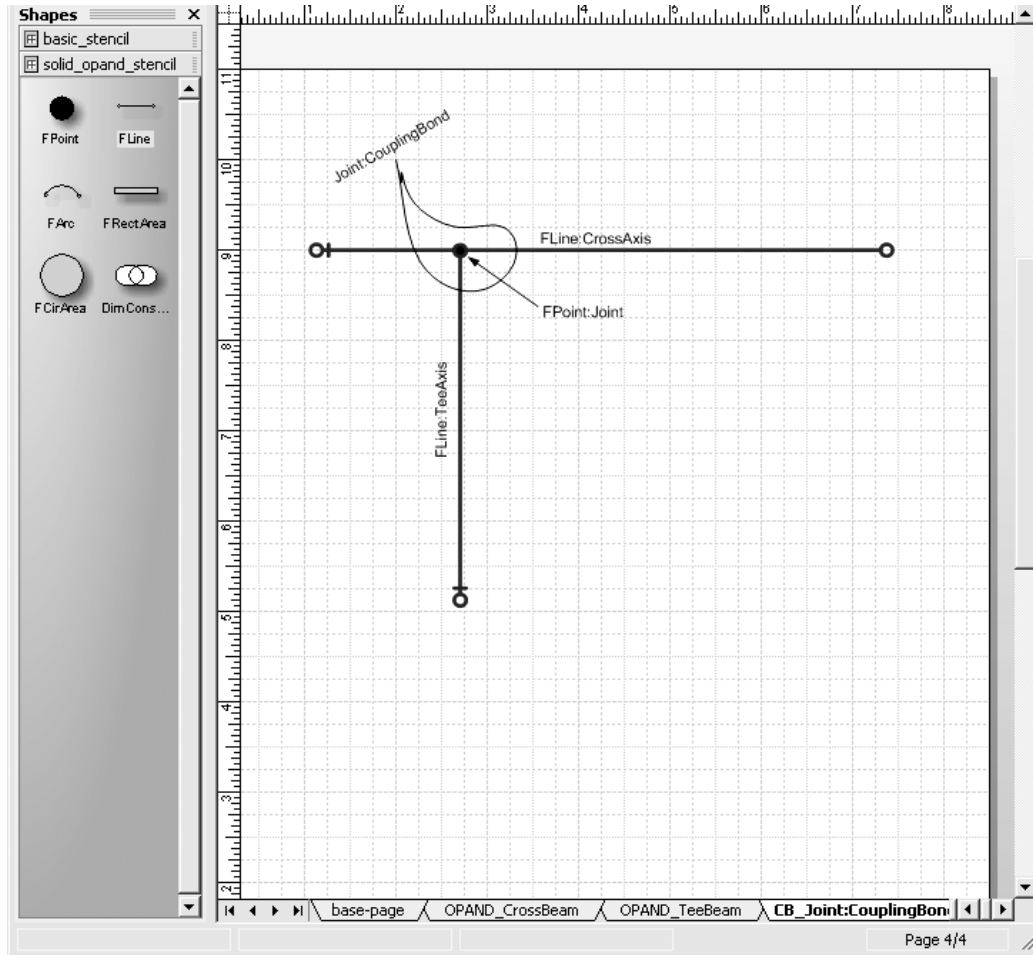


Figure 83 Screen capture of a wireframe model of the coupled solid operands

STEP IV: Propagation of functionality model to detailed design

The model developed in Step III is now represented in an XML data file (forming what is called the functionality signature). This XML file (containing information on the conceptual form and functionality constraints) is propagated to the detail design phase. In the detail design system, where the complete solid CAD model is developed and then evaluated against the functionality constraints embedded in the XML data file.

6.3.3 Evaluation of Application of Methodology

The functionality-based design (FbD) methodology developed in this research is used to specify the desired product task (functionality) to be accomplished by a designed product. These tasks were defined (in terms of required resources (operands) and functionality constraints and engineering relations as shown) in Table 12 through Table 28. The model developed for the space-frame is validated by evaluating how efficiently it captures the original functional objectives of the product. To accomplish this, the set of functionalities of the space-frame is examined. Each of these functionalities is evaluated against the propagated functionality constraints to demonstrate that the satisfaction of the constraints will guarantee the accomplishment of the design goal as captured by the functionality definition.

1. Maintain structural integrity;
2. Provide support; and
3. Provide load bearing.

Maintenance of structural integrity

The structural integrity of the space frame is realized only if the conditions of equilibrium (stability) are satisfied. This condition implies the following functional requirements:

- the beams (solid operands) used in the design should have enough strength to withstand all the external forces that impact the space frame;
- that the sum of all the external forces and moments on the space frame is zero.

To guarantee a satisfaction of the above functional requirements, the following specific constraints were developed during the functionality modeling and then propagated to the detailed design phase through the XML data.

- The various coupling bonds between FORCE and SOLID operands ($x_{13}, x_{14}, x_{15}, x_{16}, x_{27}, x_{28}$) stipulate that the strength of any selected material and the corresponding geometric feature must exceed the required internal forces necessary to overcome the impacting external forces ($\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3, \mathbf{F}_4, \mathbf{F}_5, \mathbf{F}_6$). Since this requirement is modeled as a constraint and propagated to the detail design phase, it is available to the design system in a transparent manner, ensuring that its satisfaction will guarantee the realization of functional requirement #1.
- The *global coupling bond* on the external forces ($\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3, \mathbf{F}_4, \mathbf{F}_5, \mathbf{F}_6$) has as its coupling constraint that the equilibrium conditions are met. This condition states that the equilibrium conditions for the forces \mathbf{F}_X and \mathbf{F}_Y in the coordinate directions and the moments \mathbf{M}_x and \mathbf{M}_y acting about these coordinates give:

$$\sum F_x = 0; \quad \sum F_y = 0;$$

$$\sum M_x = 0; \quad \sum M_y = 0;$$

As these constraints are also propagated to the downstream design stage, their satisfaction will guarantee the realization of the structural integrity functionality objective of the space-frame.

Provision of support as in mounting points

The functionality of providing support for other components of the car is realized is only if acceptable mounting points are provided in the final design. This functionality implies that the space frame is made of a suitable material that is compatible with the attached components and that the mounting points are properly positioned to support the mounts.

To guarantee a satisfaction of the above functional requirements, the following specific constraints were developed during the functionality modeling and then propagated to the detail design phase.

- Two solid operands (cross-beam and tee-beam) coupled are used to provide the mounting points. These operands correspond to actual physical rigid bodies to which the components are attached. Appropriate constraints imposed on the possible forms of the solids are expressed in the form of constraints (Table 12, Table 13, Table 21 – Table 27).
- The mounting points are defined as functional markers in the SOLID operands corresponding constraints imposed on their position and orientation to guarantee that the correct spatial relationships are maintained by the final design for the support of the mounts.
- To ensure that the selected material is compatible to the mounts, a material type constraint is imposed on the solid operands. This constraint restricts the material type to ALUMINIUM.

Since these constraints are also propagated to the downstream design stage, their satisfaction will guarantee the realization of the support functionality objective of the space frame component.

Provision of load bearing task.

The load bearing functionality is realized only if acceptable stress bearing components with appropriate support points are provided in the final design. This functionality implies that the space frame is made of a suitable material that is capable of withstanding any external load it is subjected to. Analysis of the external forces (Table 14 - Table 19) shows that the maximum expected resultant external force ranges from 0 N (F_1) to 50,000 N (F_4).

To guarantee a satisfaction of the above functional requirements, the following specific constraints were developed during the functionality modeling and then propagated to the detailed design phase through the XML data exchange.

- The strength of the various SOLID operands were expressed as constraints for both the normal and shear stresses of the operands. However, since the stress specification of a solid is related to the geometry (cross-sectional area), this requirement is hence defined as a constraint related to the specific geometric form of the product. The internal forces are as a result of reaction to external forces ($F_1, F_2, F_3, F_4, F_5, F_6$) on the solid operands.
- Another constraint that ensures that the load bearing is achieved is the spatial relationship requirement that specifies the orientation and location of the load bearing components. This constraint is expressed as coupling relations imposed on the functional markers and joint coupling operation.

Since these constraints are also propagated to the downstream design stage, there satisfaction will automatically guarantee the realization of the support functionality objective of the space frame component.

An experiment is designed to validate and demonstrate how functionality constraint can be used to verify design parameters in an integrated product development environment. This experiment involves the use of ANSYS analysis package to perform finite element analysis on sample designs to verify that the normal strength constraints are satisfied by the proposed design.

The procedure used for this analysis is to subject two designs (each with different design parameter – beam thickness) to the expected maximum normal loading forces to determine its compliance to the strength functionality constraint. The CAD model of the test model is shown in Figure 84. The first variation of the design is referred to as Design I, while the second design is referred to as Design II. The two beams (cross-beam and tee-beam) have uniform cross-sectional areas. Design I has a beam thickness of 2.5 mm and cross-sectional area of 675 mm². Design II has a beam thickness of 0.75 mm and cross-sectional area of 207.75 mm².

Both beams are designed with aluminum alloy material. The yield strength (S_y) of this material under normal loading is 280.00 N/mm². Using the functionality strength constraint listed in Table 12 and Table 13, the maximum allowable normal stress (S_{allow}) for the two designs is given by:

$$\begin{aligned} S_{allow} &= 0.6 S_y \\ &= 0.6 \times 280 = 168 \text{ N/mm}^2 \end{aligned}$$

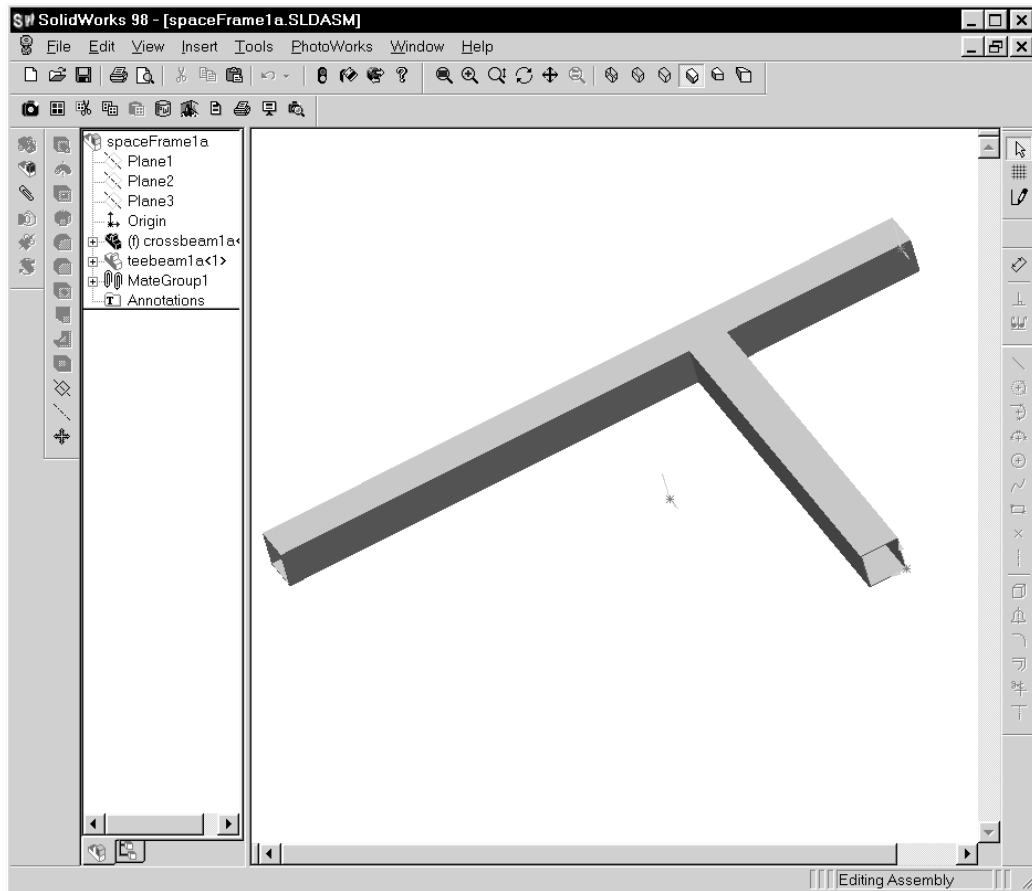


Figure 84 CAD model of test design

Consequently, the normal stress in the beams should not exceed this value (168 N/mm^2) for the design to be functionally acceptable. The ANSYS finite element analysis (FEA) package is used in the analysis of the normal stresses. Each design is tested under three different loading conditions to evaluate the resulting normal stress distribution in the beams. The loading conditions are shown in Appendices A.1 – A.4.

The ANSYS result shown in Table 29 shows the maximum normal stress on the beams. The result of the ANSYS analysis in shows that while Design I was able to sustain the maximum expected external load, Design II exceeds the defined stress limit (might breakdown under this loading condition) under test condition #3. A complete listing of the result of this analysis is shown in Appendix A.

Table 29 ANSYS test result showing maximum normal stress in space-frame

Design	Test condition #1	Test condition #2	Test condition #3
<i>Design 1</i> <i>(max stress – N/mm²)</i>	40.41	39.30	92.41
<i>Design 2</i> <i>(max stress – N/mm²)</i>	124.73	121.11	279.23

While this demonstration is manual, an integration of the functionality-based design procedure developed in this research in to CAD and analysis packages can help provide support for an automated verification of product functionality, which is not presently supported in CAD systems.

Benefits compared to the commercial CAD packages

The commercial CAD systems have evolved into powerful designer aid in the development of mechanical products. The common CAD systems including AutoCad,

Solidworks, ProE, and Catia, are considered as the state-of-the-art in the product design community. These CAD systems are compared to the capabilities of an FbD-based system as outlined in Section 6.3.2 for evaluation and validation of the functionality modeling methodology. The result of this comparison is summarized in Table 30. The tabulation shows that the FbD methodology provides support for the modeling and propagation of product functionality as constraints through its XML file propagation system. This support allows downstream design systems to access product functionality data for better integration and decision-making purposes.

Table 30 FbD capability versus Existing Commercial CAD Systems

Comparison measure	Commercial CAD systems (AutoCAD, Solidworks, ProE, and Catia)	FbD Methodology
Mathematical representation of product function	Not available	Supported
Function to concept bridge	Limited support (QFD, axiomatic design, etc)	Supported
Functionality constraint representation	Limited support (e.g. feature-based design)	Supported
Link between function and form	Not supported	Supported
Propagation of function to detail design	Not supported	Supported
Transition from concept to detail	Limited support (e.g. Pro-D from PTC)	Supported
Provision of mechanism + data for automatic function verification and enforcement.	Not supported	Supported (through XML data)

7.0 CONCLUSION AND FUTURE WORK

7.1 Conclusion

This research provides a methodology to support computer-aided conceptual design. It provides a methodology for mapping product needs in the form of problem statements and preferences through functional requirement analysis into a functionality model describing all the functional aspects of a design that could solve the original problems. 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 were introduced in the modeling of product functionality. The resources required to accomplish a function have been described in the form of functionality operands. The relations and constraints are defined in terms of coupling bonds. A generic model of product functionality has been developed to describe functionality in a mathematical form.

The concepts developed in this work have been demonstrated in a computer system. This demonstration was accomplished by building a customized functionality-modeling engine in a Microsoft Visio platform. Specifically, the following computer tools have been developed to support functionality-based design for mechanically engineered products:

- Functionality modeling toolkit implemented by customizing and automating master shapes, stencils and drawing functions in a Visio graphic design environment.
- Special graphic user interfaces were developed to provide an interactive environment for the modeling of product functionality.

- Functionality constraint modeling tools were developed to extract constraint information from the functionality components.
- Functionality data structure was created using object modeling approach. This data modeling approach provides an easy and flexible means of managing functionality information.
- Functionality data propagation to downstream design activities has been enabled by the use of an XML data representation schema developed in this work for functionality propagation and exchange.

The methodology developed in this research allows CAD systems to capture and model product functionality during conceptualization. Hence, it is possible to impose product functionality as a set of design constraints that may be used during the detailed design and analysis phase of the product. This work will help designers to ensure that the original design intents are maintained throughout the design process. It also serves as an interface tool between the conceptual design phase and detailed design phase of a product.

The integration of the functionality-based design tool with CAD systems (solid modelers) will support the re-use of past design experiences and knowledge. The development of functionality-based design system will significantly reduce the product development time and the associated costs. This is possible because the functionality constraints propagated to detail design phase can be used to evaluate design proposals in a transparent manner, thereby avoiding the need for the usual iterative design and analysis process and the associated overhead cost and time.

Functionality modeling can support product issues such as safety and reliability. The reliability of a product requires that it perform the primary task without failures. Safety, on the other hand, ensures that the operation of the product does not result in an injury to its operators, users, or other people. The inclusion of operational requirements of a product as functionality constraints in the functionality model ensures that a reliable and safe design is produced, as it must satisfy the functionality constraints that guarantee the desired level of reliability and safety. Reliability and safety inclusion in the functional requirement, however, would require an understanding of factors that impact product reliability and safety. These factors are then mapped into functionality elements in terms of operands (and corresponding attributes) and coupling relations (coupling bonds).

The proposed functionality-based design procedure provides a framework that allows a designer to carry out conceptual design with the aid of a computer. It will also serve as an interface tool between the conceptual design phase and detailed design phase of a product. The result of this research will be integrated into the Pegasus^[10] designer platform.

7.2 Future Work

This research provides the basic structure and methodology for functionality-based design. Future research will extend the set of operands developed in this work to include all material and energy operands. It is also possible to extend the result of this work to the design of other engineering products (other than the current restriction to mechanical devices). The proposed functionality verification scheme can be incorporated into solid CAD modelers. This

verification scheme may be extended and used for decision-making in the selection of competing design proposals from bidders (say in a supply chain environment).

The computer implementation has been used to demonstrate the concepts developed in this work. There are obvious limitations on the computer toolkit developed in this work. It is restricted to modeling product functionalities that may be captured in 2-dimensional coordinate modeling environments. Future work on this project may extend this to capture 3-dimensional objects. Other possible extensions include the provision of functionality state modeling tools and the provision of more functional markers (to cover shapes such as rectangles, circles, etc) instead of the current implementation that is restricted to points, lines, and curves.

Functionality states were not included in the computer implementation of this work. A future extension of this work will extend the scope of the operand states covered from two to continuous states. A computer implementation to support functionality states will also be part of the future extension of this work.

Finally, another possible extension of this work is the provision of real design repository in computer system covering the basic mechanical primitives discussed in this work. This extension will actualize the proposed advantages of reusable design knowledge.

7.2.1 Integration with CAD System

The functionality-modeling support may be integrated into commercial CAD systems by the use of attribute XML form tags. These form tags are unique attribute identification number (*attribute ID*) assigned to each functionality model element (including, coupling bonds, operand and associated functional markers). These numbers when propagated to a CAD system informs

the detailed CAD system of the solid components and their corresponding features that need to be instantiated to build a solid model for the propagated functionality model. The detailed CAD system (designer) creates solid models that correspond to each of the propagated attribute ID. Hence, the “*attribute IDs*” are incorporated into the XML data file that is propagated to the CAD.

For instance, the implementation of this in ACIS kernel would require the use of a special customization feature provided in ACIS ^[87] architecture to support definition of user assigned attributes. ACIS is an object-oriented three-dimensional (3D) geometric modeling engine from Spatial Technology Inc. ^[88] It is designed for use as the geometric foundation within virtually any end user 3D modeling application. The ACIS model representation consists of various geometric and topologic entities, as well as attributes that may be attached to the entities. The model is implemented in C++ using a hierarchy of classes. All geometric entities specified in the XML data are linked to solid model. In ACIS solid model, attribute ID is used as a linkage tag. This functionality model’s XML data goes together with geometric data (solid model) in functionality data transitions. It allows functionality information to be persistently captured in a CAD design environment.

Hence, functionality is attached to the solid components as attributes just as geometric information is done. During design verification, the system may simply invoke the attributes and compare them with that of the actual designed product (solid CAD model).

7.2.2 Extension to Commercial Product Level

The work in this research has been focused on the development of modeling concepts necessary for the representation of product functionality in CAD systems. The implementation has been restricted to demonstration of the concepts developed in this research. To advance the functionality-modeling tool to a commercial product level, the following additional improvements need to be performed.

- A special API to handle the use of XML tags as means of propagating functionality information to CAD systems needs to be developed to enable integration with other design tools.
- The FbD tools need to be integrated ^[*] with existing CAD and analysis systems in such a way that transparent functionality verification and propagation are supported. A standard communication protocol needs to be developed to enable interoperability of the design tools.
- Support for functionality states should be implemented in computer system for all types of states ranging from discrete to continuous states.
- The computer implementation of the functionality-modeling tool should be extended to support three-dimensional (3-D) models.
- An extension of the supported operands is necessary to cover operands such as gases, liquids, electrical, chemical, thermal, electrical, magnetic, acoustic, radioactive, chemical, biological, optical, hydraulic, or pneumatic energy source. In addition, the solid operand should be extended to cover deformable solids.

* See discussion on FbD – CAD integration, Section 7.2.1.

- There is a need to develop a verification mechanism to be supported by existing analysis packages such as ANSYS, ^[89] ADINA, ^[90] CFX, ^[91] etc. For this to be accomplished, it is necessary to work in collaboration with the software companies that developed these packages to evolve a standard specification mechanism to allow for transparent analysis of product functions.
- The computer graphic capability needs to be improved by implementing the system as an independent package (that is outside of Visio environment). Alternatively, its capability in Microsoft Visio 2002 may be improved by working in close collaboration with Microsoft Inc. to achieve extensive customization without compromising the processing speed.
- Provision of a design repository to serve as a knowledge-base containing prototype functionality models from which as user may select objects and then customize them for their specific needs.
- Inclusion of decision-making support will enable the system to suggest design alternatives in case of design violation during functionality verification.

APPENDICES

APPENDIX A

ANSYS FINITE ELEMENT ANALYSIS RESULT

Software Used: ANSYS DesignSpace 6.1

A.1 Definition of Aluminum Alloy

Table 31 "Aluminum alloy" properties

Name	Type	Value	Temperature
Modulus of Elasticity	Temperature-Independent	71,000.0 MPa	
Poisson's Ratio	Temperature-Independent	0.33	
Mass Density	Temperature-Independent	2.77×10^{-6} kg/mm ³	
Coefficient of Thermal Expansion	Temperature-Independent	1.7×10^{-5} 1/°C	
Thermal Conductivity	Temperature-Dependent	0.11 W/mm·°C	-100.0 °C
Thermal Conductivity	Temperature-Dependent	0.14 W/mm·°C	0.0 °C
Thermal Conductivity	Temperature-Dependent	0.17 W/mm·°C	100.0 °C
Thermal Conductivity	Temperature-Dependent	0.18 W/mm·°C	200.0 °C

Table 32 "Aluminum alloy" stress limits

Name	Type	Value
Tensile Yield Strength	Temperature-Independent	280.0 MPa
Tensile Ultimate Strength	Temperature-Independent	310.0 MPa
Compressive Yield Strength	Temperature-Independent	280.0 MPa
Compressive Ultimate Strength	Temperature-Independent	0.0 MPa

Description: "6061-T6 aluminum. Fatigue properties come from MIL-HDBK-5H, page 3-277."
"Aluminum Alloy" contains nonlinear data for thermal conductivity.

A.2 Test Condition 1

Table 33 Model : parts

Name	Material	Bounding Box (mm)
"Part 1"	"Aluminum Alloy"	70.0, 70.0, 1,180.0
"Part 2"	"Aluminum Alloy"	650.0, 70.0, 70.0

Table 34 Contact conditions

Name	Behavior	Associated Parts
"Contact Region"	Bonded	"Part 2" and "Part 1"

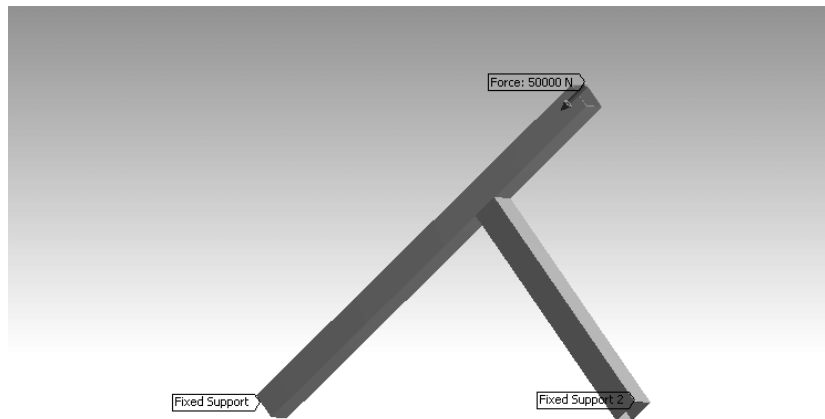


Figure 85 Loading condition for test condition 1

Table 35 Structural loading

Name	Type	Magnitude	Associated Parts
"Force"	Surface Force	50,000.0 N	"Part 1"

Table 36 Structural supports

Name	Type	Associated Parts
"Fixed Support"	Fixed Surface	"Part 1"
"Fixed Support 2"	Fixed Surface	"Part 2"

A.3 Test Condition 2

Table 37 Model : parts

Name	Material	Bounding Box (mm)
"Part 1"	"Aluminum Alloy"	70.0, 70.0, 1,180.0
"Part 2"	"Aluminum Alloy"	650.0, 70.0, 70.0

Table 38 Contact conditions

Name	Behavior	Associated Parts
"Contact Region"	Bonded	"Part 2" and "Part 1"

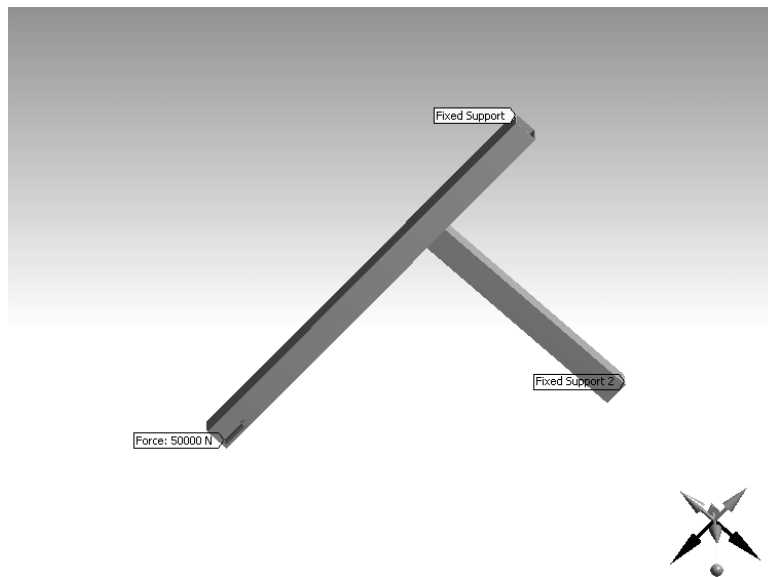


Figure 86 Loading condition for test condition 2

Table 39 Structural loading

Name	Type	Magnitude	Associated Parts
"Force"	Surface Force	50,000.0 N	"Part 1"

Table 40 Structural supports

Name	Type	Associated Parts
"Fixed Support"	Fixed Surface	"Part 1"
"Fixed Support 2"	Fixed Surface	"Part 2"

A.4 Test Condition 3

Table 41 Model : parts

Name	Material	Bounding Box (mm)
"Part 1"	" <u>Aluminum Alloy</u> "	70.0, 70.0, 1,180.0
"Part 2"	" <u>Aluminum Alloy</u> "	650.0, 70.0, 70.0

Table 42 Contact conditions

Name	Behavior	Associated Parts
"Contact Region"	Bonded	"Part 2" and "Part 1"

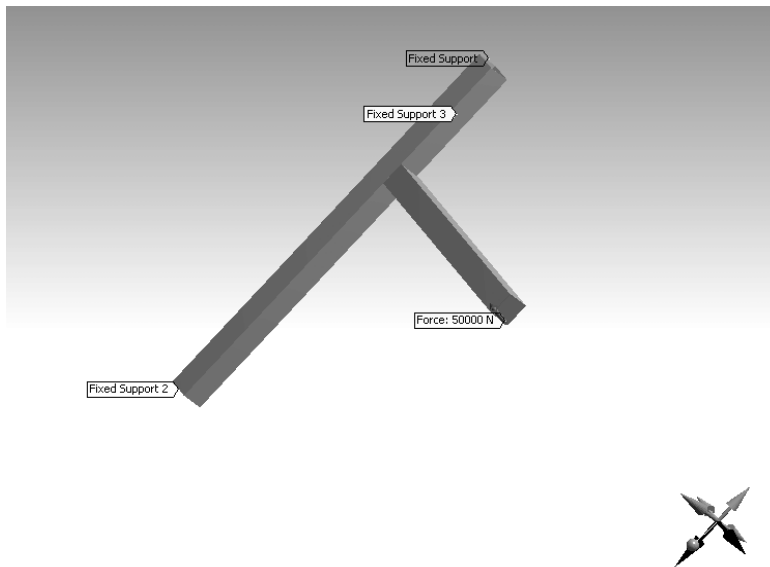


Figure 87 Loading condition for test condition 3

Table 43 Structural loading

Name	Type	Magnitude	Associated Parts
"Force"	Surface Force	50,000.0 N	"Part 2"

Table 44 Structural supports

Name	Type	Associated Parts
"Fixed Support"	Fixed Surface	"Part 1"
"Fixed Support 2"	Fixed Surface	"Part 1"
"Fixed Support 3"	Fixed Surface	"Part 1"

A.5 Result of Design 1, Test condition 1

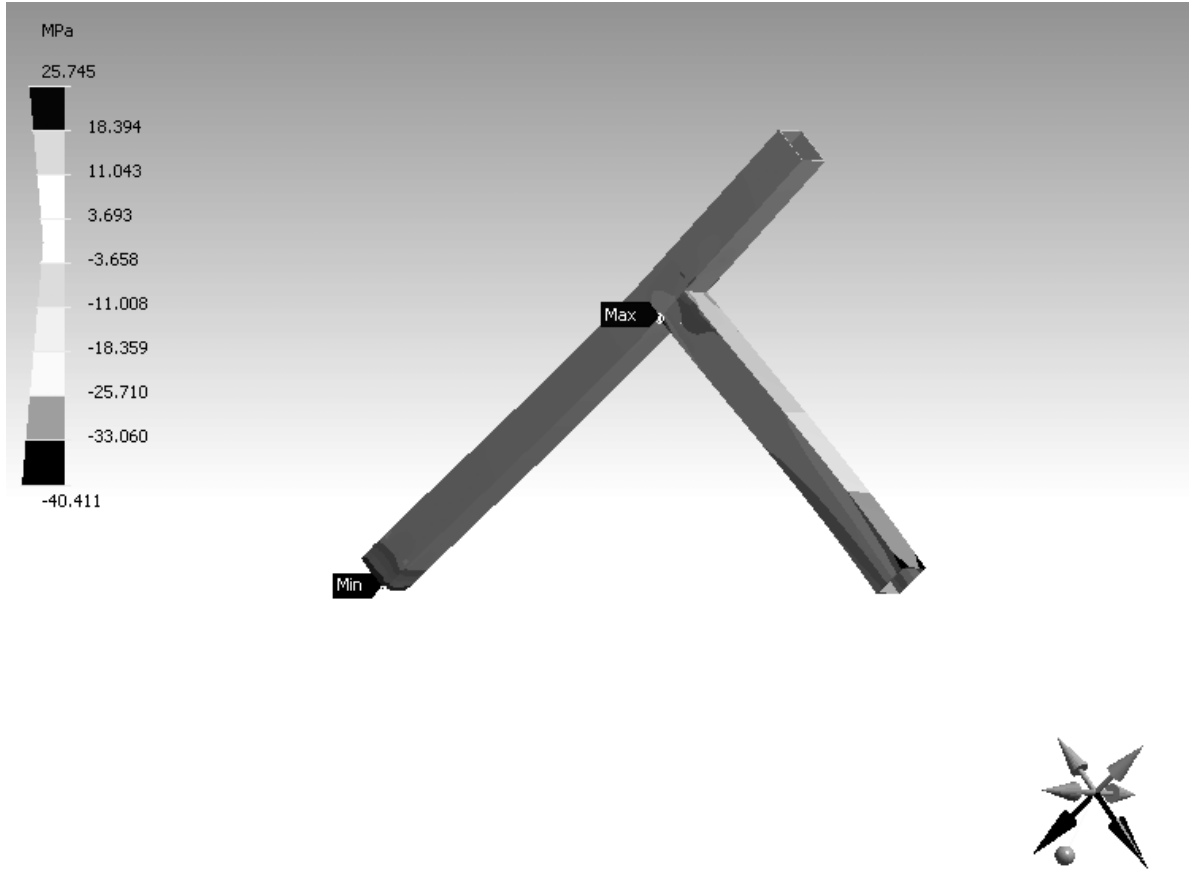


Figure 88 Normal stress distributions for design 1, test condition 1

Table 45 Structural results

Name	Scope	Orientation	Minimum	Maximum	Alert Criteria
"Normal Stress"	All Parts In "Model"	World X Axis	-40.41 MPa	25.74 MPa	None

A.6 Result of Design 1, Test condition 2

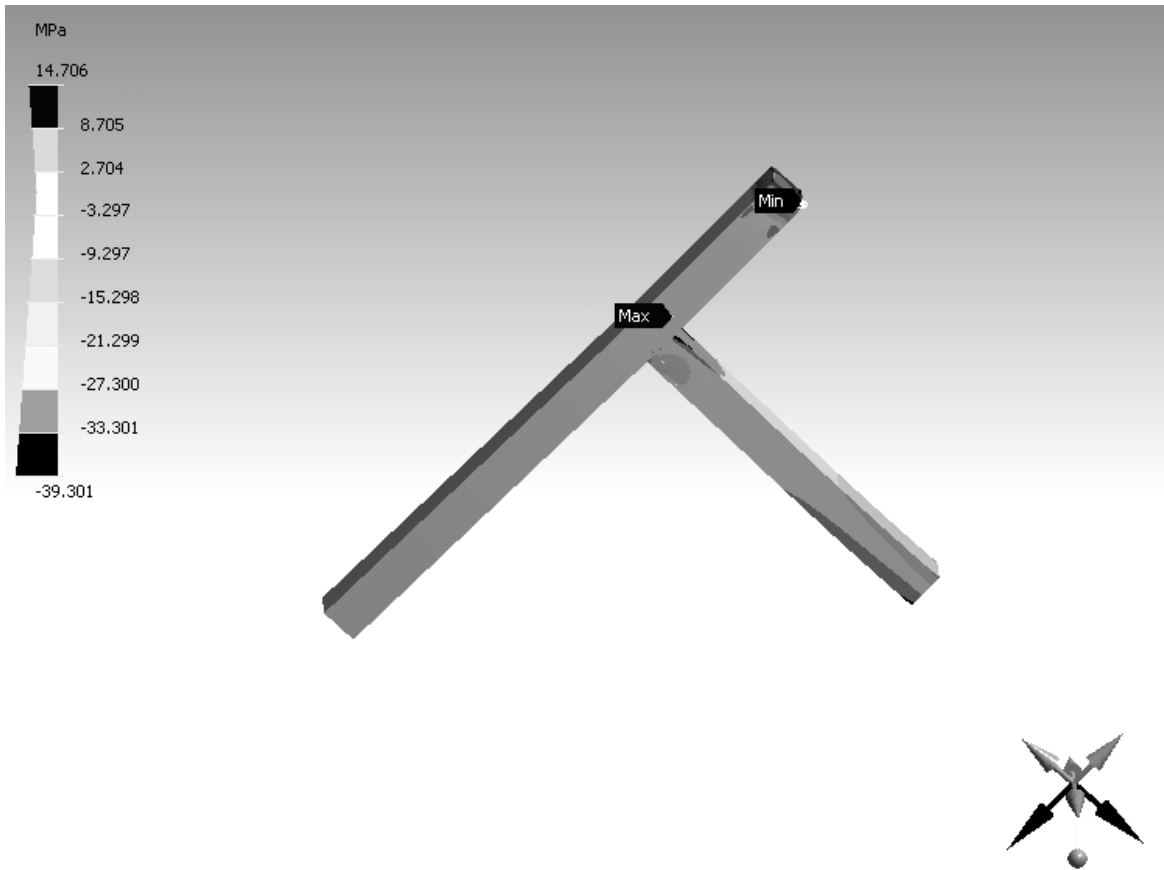


Figure 89 Normal stress distributions for design 1, test condition 2

Table 46 Structural results

Name	Scope	Orientation	Minimum	Maximum	Alert Criteria
"Normal Stress"	All Parts In "Model"	World X Axis	-39.3 MPa	14.71 MPa	None

A.7 Result of Design 1, Test condition 3

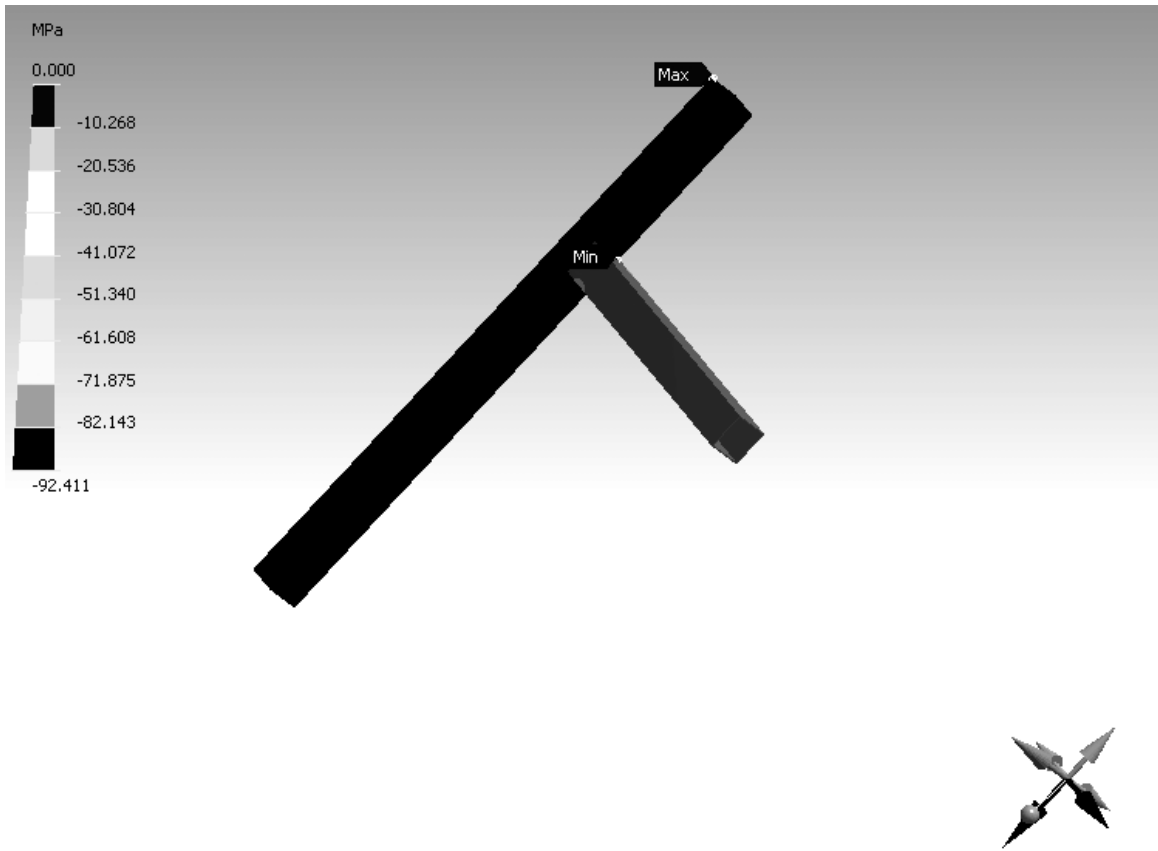


Figure 90 Normal stress distributions for design 1, test condition 3

Table 47 Structural results

Name	Scope	Orientation	Minimum	Maximum	Alert Criteria
"Normal Stress"	All Parts In "Model"	World X Axis	-92.41 MPa	0.0 MPa	None

A.8 Result of Design 2, Test condition 1

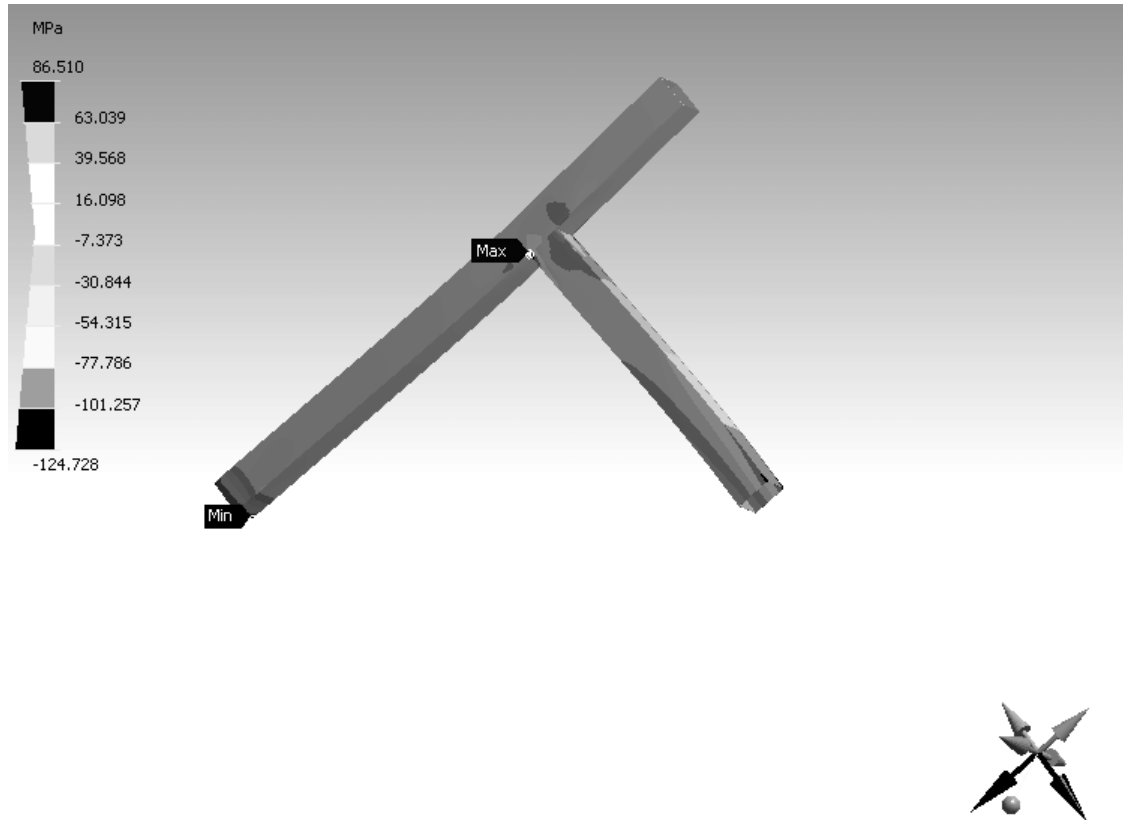


Figure 91 Normal stress distributions for design 2, test condition 1

Table 48 Structural results

Name	Scope	Orientation	Minimum	Maximum	Alert Criteria
"Normal Stress"	All Parts In "Model"	World X Axis	-124.73 MPa	86.51 MPa	None

A.9 Result of Design 2, Test condition 2

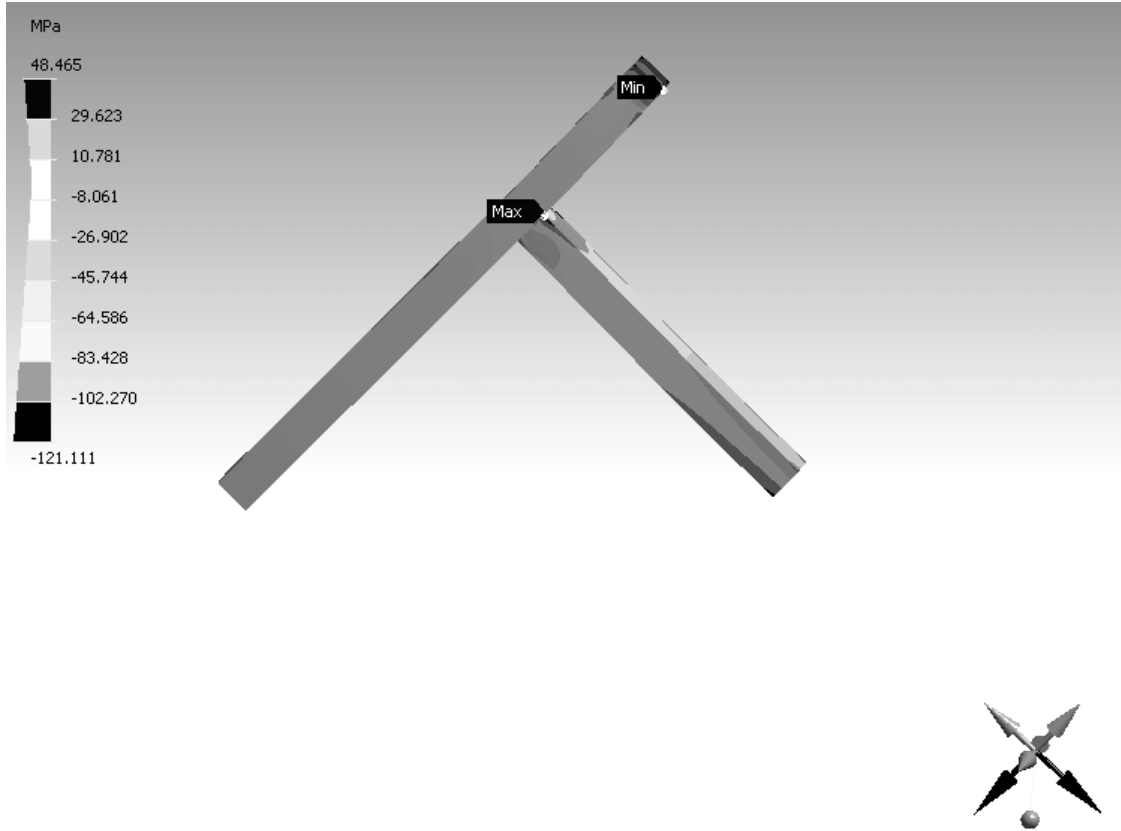


Figure 92 Normal stress distributions for design 2, test condition 2

Table 49 Structural results

Name	Scope	Orientation	Minimum	Maximum	
"Normal Stress"	All Parts In "Model"	World X Axis	-121.11 MPa	48.46 MPa	None

A.10 Result of Design 2, Test condition 3

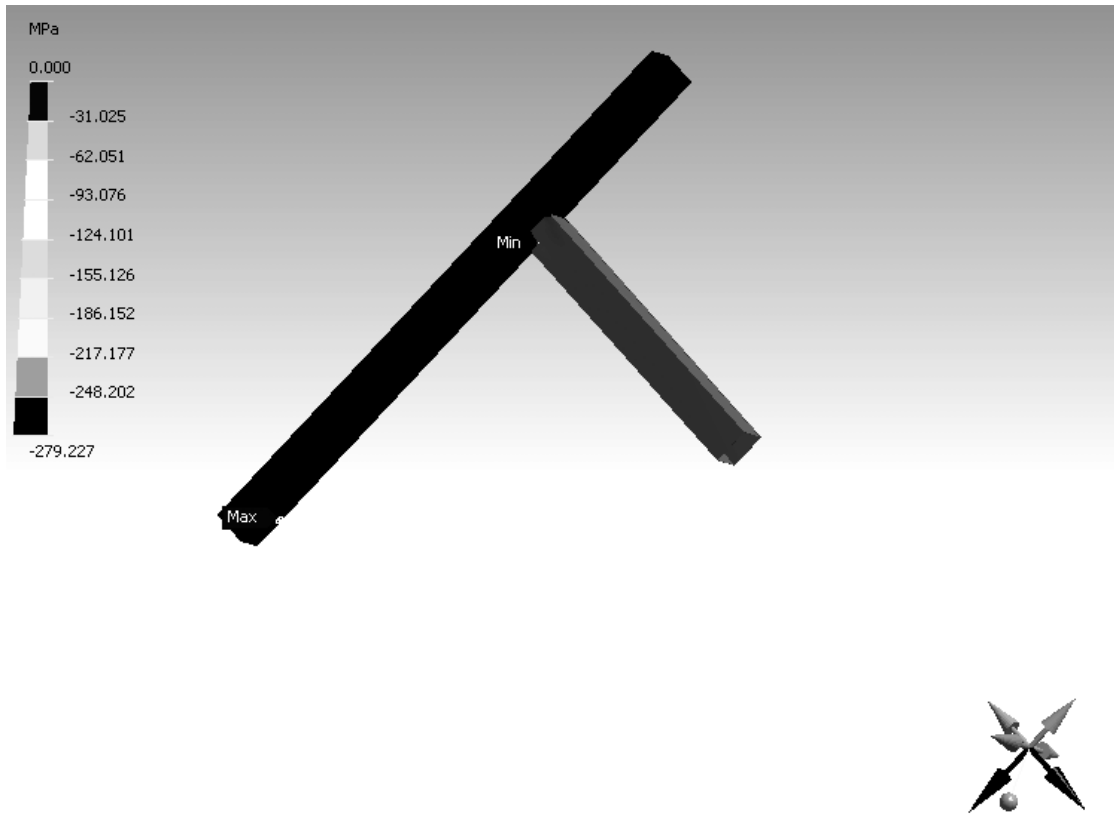


Figure 93 Normal stress distributions for design 2, test condition 3

Table 50 Structural results

Name	Scope	Orientation	Minimum	Maximum	Alert Criteria
"Normal Stress"	All Parts In "Model"	World X Axis	-279.23 MPa	0.0 MPa	None

APPENDIX B

MODELING WITH FUNCTIONALITY PRIMITIVES

B.1 Transformation Operation

This functionality operation transforms or converts one mechanical operand (such as: force, torque, pressure, heat, etc) to another mechanical operand. For example, force to torque, force to deformation/strain (*effect*), force to linear displacement, and torque to angular displacement. The transform operation has three components (A, B, and C) as shown in Figure 94.

- A is the Source Operand
- B is the Target Operand
- C is the Transform Operator defining the nature of A-B interaction

These components and their corresponding attributes are identified during the initial requirement analysis of the function.

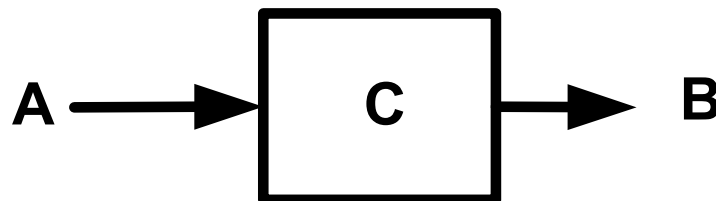


Figure 94 Transform operation

The *source* and *target* operands identified during the functional requirement phase are specified at design time by selecting from any of the functionality operands. Hence, one might select FORCE as the source operand and TORQUE as the target operand. This decision will automatically invoke the attributes corresponding to FORCE and TORQUE operands. This approach thus ensures that the designer is aware of all the relevant factors that might influence the functionality of the design.

The *transform* operator defines the relevant coupling bond necessary to achieve the A-to-B transformation operation. This coupling bond includes the set of DoFs, constraints, and the functional relations necessary to achieve the desired transformation task. The designer is allowed to also define some custom relations involving the attributes and the functional features of the operands.

Additional operands that might be required to achieve the transformation are also defined for the operation. For the case of FORCE to TORQUE transformation, a MATERIAL MEDIUM is necessary to realize this functionality as shown in Figure 95 and Table 51.

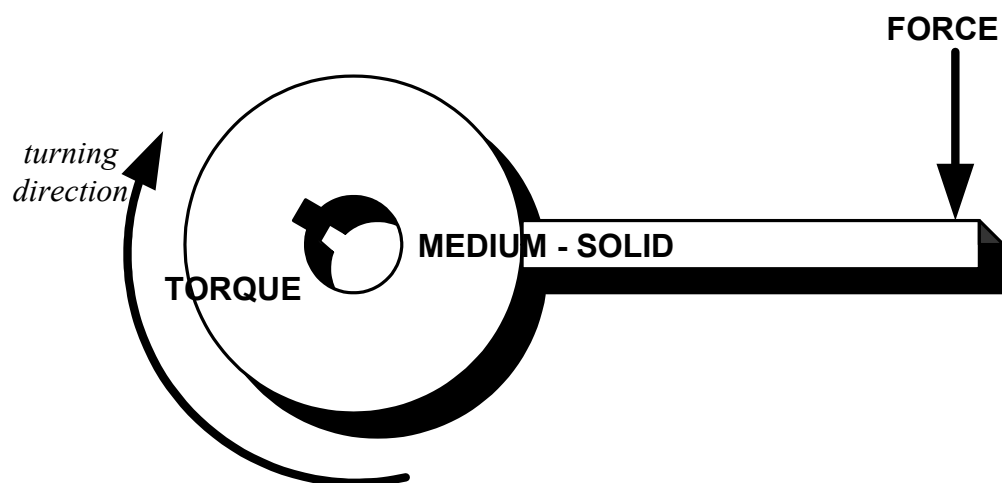


Figure 95 FORCE to TORQUE transformation functionality operation

Table 51 Force to torque transformation functionality operation

<source operand>	FORCE	Attributes of FORCE.
<target operand>	TORQUE	Attributes of TORQUE.
<operator>	<additional operands>	MEDIUM - SOLID
	<relations>	$T = F \times L$; where, L is MEDIUM attribute; Spatial relations between operands.
	<constraint>	{MEDIUM: strength, functional markers, length} {FORCE: magnitude, direction} {TORQUE: magnitude, direction}
	<dof>	Defines movement restrictions on: MEDIUM, FORCE, TORQUE

B.2 Transmission Operation

This functionality operation conveys a mechanical operand from one point to another. This primitive has a positional component associated with it. It does not change the nature of an operand, but merely changes its spatial location in space. Examples include: force transmission along a beam, power transmission of an automobile shaft, torque transmission of a pump spindle, and pressure transmission in a pipe. Other examples that are outside the scope of this work include: electromagnetic transmission, sound transmission, and thermal transmission. The transmission operation has three components (A, B, and C) as shown in Figure 96.

A: Source Operand

B: Target Operand

C: Transmission Operator: A-B interaction

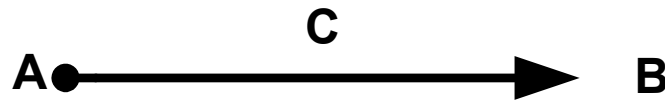


Figure 96 Transmission operation functionality

The components of transmission functionality together with their corresponding attributes are identified during the initial functional requirement analysis of the function. The *source* and *target* operands are specified at design time by selecting from any of the functionality operands. Hence, one might select FORCE as the source operand and FORCE' as the target operand. This decision will automatically invoke the attributes corresponding to FORCE and FORCE' operands. This approach thus ensures that the designer is aware of all the relevant factors that might influence the functionality of the design.

The *transmission* operator defines the relevant coupling bond necessary to achieve the A-to-B transmission operation. The coupling bond includes the set of DoFs, constraints, and the functional relations necessary to achieve the desired transformation task. The designer is allowed to also define some custom relations involving the attributes and the functional features of the operands.

Additional operands that might be required to achieve the transformation are also defined for the operation. For the case of FORCE-to-FORCE' transformation, a MATERIAL MEDIUM is usually necessary to realize this functionality. Force transmission results in *displacement*, *loading*, *stress*, *strain*, *pressure*, etc. For each transmission element, the system invokes the corresponding attributes that may be imposed as constraints. Consequently, for a force

transmission element, we will have attributes such as mode of application, point of application, nature of force (impact, steady, or random loading), and associated transformation.

As an illustration, consider as case of MATERIAL-to-MATERIAL transmission functionality operation. This transmission operation conveys or relocates a MATERIAL operand from one point to another. Thus, it effects the spatial displacement of a material. This material transmission functionality is commonly known as motion function. For example, Figure 97 (and Table 52) shows a SOLID material BLOCK conveyed from point A1 to point A2.



Figure 97 Solid transmission functionality example

Table 52 Solid transmission functionality operation example

<source operand>	SOLID: BLOCK	Attributes of BLOCK solid operand.
<target operand>	SOLID: BLOCK	Attributes of BLOCK solid operand.
<operator>	<additional operands>	- FORCE (weight, friction and applied forces) - MEDIUM / trajectory.
	<relations>	<ul style="list-style-type: none"> • Force and motion equations; • Frictional relations; and • Spatial relations
	<constraint>	{MEDIUM - trajectory: strength, length} {FORCE: magnitude, direction} {BLOCK: solid material attributes}
	<dof>	Defines movement restrictions on: MEDIUM, FORCE, BLOCK

B.3 Joint Operation

Retains/maintains an absolute or relative position of functionality operands (holds two or more entities in relative position). This is usually achieved by exerting a retentive (restricting) force on operand or/and resisting dissociating forces. The joint operation has three components (A, B, and C) as shown in Figure 98.

A: First Operand

B: Second Operand

C: Joint Operator: A-B interaction

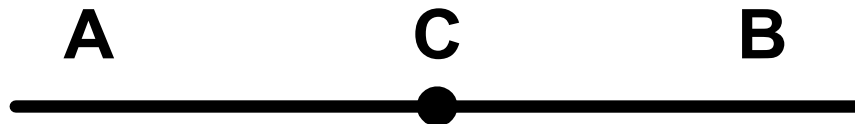


Figure 98 Joint functionality operation

The components and their corresponding attributes are identified during the initial functional requirement analysis phase of the function. The *first* and *second* operands need to be specified at design time by selecting from any of the operands. These operands are usually SOLID material operands that need to maintain a given spatial relation with each other. It is possible to extend the concept of joint functionality beyond MATERIAL operands to include energy operands. However, in this work description of joint operation is limited to SOLID operands. Hence, one might select two solid operands as the A and B components of the joint functionality. The selection of A and B operand will automatically invoke the attributes corresponding to selected solid operands.

The joint operator defines the relevant coupling bond necessary to achieve the A-to-B joint operation. The coupling bond includes the set of DoFs, constraints, and the functional relations necessary to achieve the desired joint task. The designer is allowed to also define some custom relations involving the attributes and the functional features of the operands.

Additional operands that might be required to achieve the joint operation are also defined for the operation. A joint functionality is usually realized by the inclusion of some additional operand or operand mechanisms necessary to spatially maintain the specified DoF.

Some functional considerations necessary for the definition of the joint functionality design are:

- Consideration of the type of loading, such as shear and tension, to which the structure will be subjected and the size and spacing of holes.
- Compatibility of the fastener material with the components to be joined is important. Incompatibility may lead to galvanic corrosion and crevice corrosion.
- Nature of expected dissociating forces: steady, random, vibration, impulsive, etc.
- Resistance force: effect and reaction of force
- Desired spatial relation and the degree of freedom restriction requirements.

In specifying the joint functionality operation, the class or type of joint required is usually determined by the desired spatial relation and degree of freedom of the operands (coupling components). Two broad classes of joints are available: *hard* and *soft* joints.

- *Hard joint* from hard fastening: A fixed DoF is maintained between the coupled operands. Hence, no relative motion is allowed between the fastened parts. Examples of hard joints are formed by welding, screw, glue, and rivets processes.
- *Soft joint* from soft fastening: When a fixed DoF is not necessary between the operands, a soft joint option is an option. Here, operands are joined together so that relative motion between these two parts is consistent. Soft joints are used to create movable joints, such as hinges, sliding mechanisms for drawers and doors, and adjustable components and fixtures. Examples of such joints include revolute joint, cylindrical joint, spherical joint, prismatic joint, helix, and plane joints.

In terms of reversibility of the joint operation, two classes of joints are available:

- *Permanent / Semi-Permanent joint*: welding, glue, rivets, etc.
- *Reversible (fastening) Joint*: Used to join two or more components in such a way that they can be taken apart sometime during the product's service life. Examples: clips, threaded fasteners, rivets, metal stitching, seaming, crimping, and clamping.

As an illustration of joint functionality operation, consider as case where two SOLID operands (steel beams) are brought together to form a tee-joint using the joint functionality operation. This joint operation maintains the two operands in a fixed spatial relation with respect to each other. Thus, it maintains the spatial location of the material operands.

B.4 Load bearing Operation

Bear loads exerted by mechanical operands by resisting displacement/deformation forces without exceeding a maximum deformation limit, thus maintains a state of stability for the mechanical operands. The load bearing functionality operation has three components (A, B, and C) as shown in Figure 99.

A: Source Operand (Load)

B: Target Operand (Base)

C: Load bearer Operator: A-B interaction



Figure 99 Load bearing functionality operation

The source operand (A) is the operand that provides the load that needs to be resisted. Load is expressed in terms of force. This source of loading force could be because of the weight of a solid (material) operand or as a result of application of an external force. Whatever the nature of this force, its attribute is captured by its properties and any associated solid operands. Other issues that need to be considered include:

- Nature of loading: steady, impulse, random, vibration
- Position and orientation of load (vertical, horizontal, or inclined)

The target operand (B) is the operand that provides the resistance to the applied loading. Hence, it provides conditions necessary to maintain stability or equilibrium. In this work, this operand is limited to SOLID operands. A future extension of this set could include the use of energy operands such as the forces provided by electromagnetic systems to bear load. The usual attribute set of SOLID materials needs to be defined for this operand. In particular, the strength and elastic properties should be such that it can withstand the type of loading it is subjected to.

The load-bearing operator defines the relevant coupling bond necessary to achieve the A-to-B load bearing operation. The coupling bond includes the set of DoFs, constraints, and the functional relations necessary to achieve the desired load-bearing task. The designer is allowed to also define some custom relations involving the attributes and the functional features of the operands. Spatial relations need to be defined with respect to operands A and B. This spatial relation determines some other factors such as: nature of the load (tensile, compressive, torsional, and shear loading), bending moments, stress, strain. In addition, the required DoF of the load and its interaction with the base defines the issues related to support and stability.

As an example, consider the design of a shelf that supports stacks of books by providing a load bearing functionality operation. This functionality operation ensures that the books placed in the shelf are retained by resisting the weight (loading force) exerted by the books. In this example, component A is the stack of books, component B is the shelf (SOLID operand) that served as a base, and component C is the interaction (coupling bond) A and B. This load bearing operation maintains the two operands in a fixed spatial relation with respect to each other.

B.5 Energy Converter Operation

This is a special type of transformation functionality operation occasioned by the scope of this work. In this work, it is assumed that all the energy that impact mechanical product functionality is present in mechanical form. For this assumption to hold, all energy sources, other than mechanical energy, are first applied to an energy converter, which transforms that energy into a mechanical form. Thus, energy converter (a specialized transformation operation) is used as a primitive operation in the functionality model.

Consequently, the energy converter transforms energy from one form to another. It has three components (A, B, and C) as shown in Figure 100.

A: Source Operand: Input Energy

B: Target Operand: Output Energy

C: Converter Operator: A-B interaction

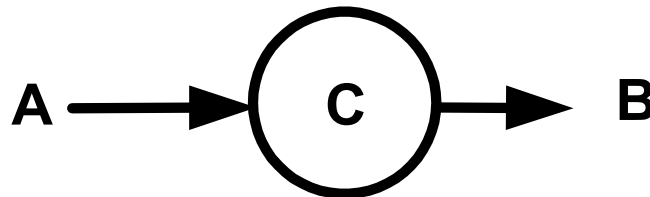


Figure 100 Energy converter functionality operation

Although energy may assume any form (mechanical, electromagnetic, etc), the only form that is modeled in this work is the mechanical energy (FORCE and TORQUE). Although A and B components may assume any form, the scope of this work is such that either A or B (depending on the one directly in contact with the rest of the functionality model) must be in a mechanical form (FORCE or TORQUE).

Converter operator (C): A lot of work has been done by physicists in determining the relation between various forms of energy. In this work, no attempt is made to include the detail of this relation in the model. The designer is at liberty to define the details of such relations. The only relevant factor in this work is an accurate description of the mechanical energy components (FORCE / TORQUE) that is in direct contact with the rest of the functionality model. This description will typically include all the attributes of a FORCE / TORQUE operand (magnitude, direction, and spatial relations).

B.6 Frictional Operation

This is functionality primitive that manages frictional relations between surfaces of operands in mechanical systems. It may reduce, increase or maintain mechanical friction. The joint operation has three components (A, B, and C) as shown in Figure 101.

A: First Surface (C1): functional Region of SOLID operand A

B: Second Surface: (C2): functional Region of SOLID operand B

C: Friction Operator: A-B-C₁-C₂-M-F interaction

Where, F is the frictional force and M is the normal load on surface C2

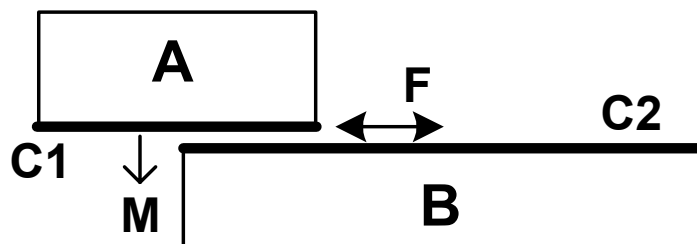


Figure 101 Frictional functionality operation

The First and Second entities (C1 of A and C2 of B) need to be specified at design time by selecting two surfaces (as a functional region) from the functional marker set of any of the solid operands already included in the functionality model. The operands (A and B) are SOLID material operands that involve some sliding motion on the contacting surfaces. It is possible to extend the concept of frictional functionality beyond SOLID MATERIAL operands to include GASEOUS and LIQUID operands. However, in this work, description of frictional operation is limited to SOLID operands. Hence, one might select two SOLID operands as the A and B components of the frictional functionality. The selection of A and B operand will automatically invoke the attributes corresponding to selected solid operands. This approach thus ensures that the designer is aware of all the relevant factors that might influence the functionality of the design.

The friction operator defines the relevant coupling bond necessary to achieve the A-to-B frictional operation. The coupling bond includes the set of DoFs, constraints (including frictional forces), and the functional relations necessary to achieve the desired frictional task. The designer is allowed to also define some custom relations involving the attributes and the functional features of the operands.

Additional operands or tasks (example polishing, ball bearing, lubrication, etc) that might be required to achieve the desired friction level are also defined for the operation. Some additional functional considerations necessary for the definition of the joint functionality design are:

- Nature of expected forces: steady, random, vibration, or impulsive.
- Desired spatial relation and the degree of freedom restriction requirements.

- Frictional Heat: In *sliding friction*, if the frictional force F is collinear with the direction of the velocity V of the point of application, the amount of the friction heat generated while the force was applied from time t_1 to t_2 is:

$$Q = \int_{t_1}^{t_2} F \cdot V dt$$

If the force and velocity remain constant in time,

$$Q = FV(t_2 - t_1) = F\Delta s$$

Where,

Δs is the distance traveled at constant velocity

- Amonton's law demands that friction force is:

$$F = \mu F_n$$

where,

F_n = normal force compressing the two rubbing solids

μ = constant (friction coefficient, depends only on material and surface)

Similarly, if constant friction torque T is applied to a solid rotating at an angular velocity w , during a time interval $t_2 - t_1$ or angle of rotation $\Delta\theta = w(t_2 - t_1)$, the heat generated is:

$$Q = Tw(t_2 - t_1) = T\Delta\theta$$

B.7 Offset Operation

Maintain a specified offset (distance) between entities by providing obstruction to disallowed motion. This might invoke sub-functions such as fastening and load-bearing functionalities. This is usually achieved by exerting a retentive (restricting) force on entity

or/and resisting dissociating forces. The offset operation has three components (A, B, and C) as shown in Figure 102.

A: First Operand

B: Second Operand

C: Offset Operator: A-B interaction

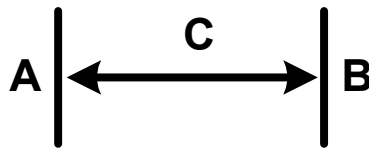


Figure 102 Offset functionality operation

The First and Second operands are specified at design time by selecting from any of the operands already included in the design tool. These operands are usually material operands that need to maintain a given spatial offset relation with each other. It is possible to extend the concept of offset functionality beyond MATERIAL operands to include energy operands. However, in this work, description of offset operation is limited to SOLID operands. Hence, one might select two material operands as the A and B components of the offset functionality. The selection of A and B operand will automatically invoke the attributes corresponding to selected solid operands.

The offset operator defines the relevant coupling bond necessary to achieve the A-to-B offset operation. The coupling bond includes the set of DoFs, constraints, and the functional relations (including spatial relations) necessary to achieve the desired offset task. The designer

is allowed to define some custom relations involving the attributes and the functional features of the operands.

Additional operands that might be required to achieve the offset operation are also defined for the operation. An offset functionality is usually realized by the inclusion of some additional operand or operand mechanisms necessary (as separators) to spatially maintain the necessary offset.

Some functional considerations necessary for the definition of the offset functionality design are:

- Consideration of the type of loading that might be present, such as shear and tension, to which the offsetting operator will be subjected.
- Nature of expected dissociating forces: steady, random, vibration, impulsive, etc.
- Desired spatial relation and the degree of freedom restriction requirements.

As an illustration, consider as case where two SOLID operands (steel components) offset functionality operation. This offset operation maintains the two operands in a fixed spatial relation with respect to each other.

B.8 Channel Operation

Provides a mechanical functionality that ensures that entities assume a desired spatial position. Constrains displacement along a pre-defined path by exerting reaction forces. Examples: guide rails, fixtures and jigs, and chamfers for assembly.

This is usually achieved by exerting a retentive (restricting) force on entity or/and resisting dissociating forces. The channel operation has three components (A, B, and C) as shown in Figure 103.

A is the channel, guide, or path enforcer.

B is the guided or channeled operand.

C is the channel Operator defining A-B interaction.

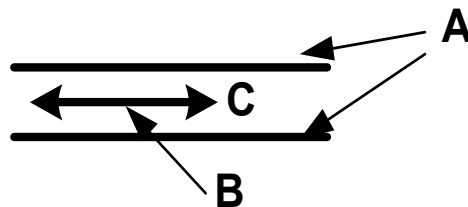


Figure 103 Channel functionality operation

The First and Second operands are specified at design time by selecting from any of the functionality operands. These operands are usually material operands that need to maintain a given spatial relation with each other. It is possible to extend the concept of channel functionality beyond MATERIAL operands to include energy operands (as in optical fibers). However, in this work description of channel operation is limited to SOLID operands. Hence, one might select two components as the A and B components of the channel functionality. The selection of A and B operand will automatically invoke the attributes corresponding to selected solid operands. This approach ensures that the designer is aware of all the relevant factors that might influence the functionality of the design.

The channel operator (C) defines the relevant coupling bond necessary to achieve the A-to-B channel operation. The coupling bond includes the set of DoFs, constraints, and the functional relations (including spatial relations) necessary to achieve the desired channel task. The designer is allowed to also define some custom relations involving the attributes and the functional features of the operands.

Additional operands that might be required to achieve the channel operation are also defined for the operation. Channel functionality is usually realized by the inclusion of some additional operand or operand mechanisms necessary (as separators) to spatially maintain the necessary channel.

B.9 Block Operation

Block: Disallows non-permissible entrance / motions. Examples include provision of insulation in electrical systems, guards in fans, etc. Maintain a specified separation/offset between entities by providing obstruction to disallowed contacts. It functions by providing an operand as a separator. This is usually achieved by exerting a retentive (restricting) force on entity or/and resisting dissociating forces. This operation has three components (A, B, and C) as shown in Figure 104.

A: First Operand

B: Second Operand

C: Block Operator: A-B interaction

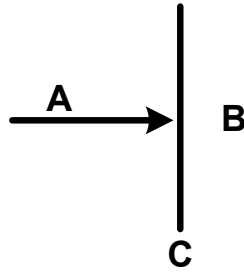


Figure 104 Block functionality operation

The First and Second operands need to be specified at design time by selecting from any of the functionality operands. These operands are usually material operands that need to maintain a given spatial separation and relation with each other. It is possible to extend the concept of block functionality beyond MATERIAL operands to include energy operands. However, in this work description of block operation is limited to SOLID operands. Hence, one might select two components as the A and B components of the block functionality. The selection of A and B operand will automatically invoke the attributes corresponding to selected solid operands. This approach ensures that the designer is aware of all the relevant factors that might influence the functionality of the design.

The block operator defines the relevant coupling bond necessary to achieve the A-to-B block operation. The coupling bond will include the set of DoFs, constraints, and the functional relations (including spatial relations) necessary to achieve the desired block task. The designer is allowed to also define some custom relations involving the attributes and the functional features of the operands.

Additional operands that might be required to achieve the guard-block operation are also defined for the operation. Block functionality is usually realized by the inclusion of some

additional operand or operand mechanisms necessary (as separators) to spatially maintain the blocking.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Dentsoras, A.J., "A Selective, Multi-criteria Method for Handling Constraint Violations in Well-defined Design Problems", Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, Vol. 13, (1999), pp. 205-215.
2. Tien-Lun, L., and Nnaji, B.O., "Realization and Management of Product Design Constraints in CAD Modeling", (To be published in Computer-Aided Design).
3. Mullins, S.H. and Anderson, D.C., "Automatic Identification of Geometric Constraints in Mechanical Assemblies", Computer-Aided Design, Vol. 30, No. 9 (1998), pp. 715-726.
4. Rowe Jeffrey, "Teamwork", MCAD Vision, June 2000, <http://www.mcadcafe.com>.
5. Abrahamson, S, et al., "Integrated Design in a Service Marketplace", Computer-Aided Design, Vol. 32, (2000), pp. 97-107.
6. Oh, V. and Sharpe, J., "Conflict Management in an Interdisciplinary Design Environment", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 9, (1995), pp. 247-258.
7. NSF workshop on *e-Product Design and Realization*, University of Pittsburgh, Pittsburgh, Oct 19-18, 2000.
8. Tong, C. and Sriram, D., ed., Artificial Intelligence in Engineering Design: Design Representation and Models of Routine Design, Volume I, "Introduction, by C. Tong and D. Sriram" (New York: Academic Press Inc, 1992), pp. 1-53.
9. Lotter, B., Manufacturing Assembly Handbook. (Boston: Butterworths, 1986).
10. Nnaji, B., "Pegasus e-Product Designer System" IUCRC workshop on e-Product Design and Realization (Pittsburgh, Pennsylvania: University of Pittsburgh, December, 2000).
11. Huang, G.Q., ed., Design for X; Concurrent Engineering Imperatives, "Design for Manufacture and Assembly: The Boothroyd-Dewhurst Experience, by G. Boothroyd" (New York: Chapman & Hall, 1996), pp. 1-17.

12. Lee, Kunwoo, Principles of CAD / CAM / CAE Systems (Massachusetts: Addison-Wesley, 1999), pp. 101.
13. Chan, B., and Finger, S., "Supporting Conceptual Design: A Model for Reflective-Interactive Design", Knowledge Intensive CAD, 1998, pp. 215-236.
14. Terpenney, J.P., Nnaji, B.O., and Bohn, J.H., "Blending Top-Down and Bottom-Up Approaches in Conceptual Design", Proceedings of the Seventh Industrial Engineering Research Conference, (Banff, Canada: May 9-10, 1998).
15. Gorti, S.R., Sriram, R.D., "From Symbol to Form: a Framework for Conceptual Design", Computer-Aided Design, Vol. 28, No. 11 (1996), pp. 853-870.
16. Mukherjee, A. and Liu, C.R., "Conceptual Design, Manufacturability Evaluation and preliminary Process Planning using Function-Form relations in Stamped Metal Parts" Robotics & Computer-Integrated Manufacturing, Vol. 13, No. 3 (1997), pp. 253-270.
17. Brown, D. C., and Chandrasekaran, B., "Investigating Routine Design Problem Solving", Artificial Intelligence in Engineering Design: Design Representation and Models of Routine Design, Vol. I, (1992), pp. 221-249.
18. Pahl, G, and Beitz, W., Engineering Design: A Systematic Approach, (New York: Springer, 1996), pp. 1-400.
19. Chang, T. C., Expert Process Planning in Manufacturing, (Addison Wesley, 1990).
20. Shah, J., Sreevallasan, P., and Mathew, A., "Survey of CAD/Feature-based Process Planning and NC Programming Techniques", Computer-Aided Engineering Journal, (February, 1991), pp. 25-33.
21. Nnaji, B. O., Kang, T., Yeh, S., and Chen, J. P., "Feature Reasoning for Sheet Metal Components", International Journal of Production Research, Vol. 29, No. 9 (1991), pp. 1867-1896.
22. Tien-Lun, L., "A Coordinated Constraint-Based Modeling and Design Advisory System for Mechanical Components and Assemblies" (unpublished Ph.D. Dissertation, School of Engineering, University of Massachusetts, 1998).
23. Rodenacker, W., Methodishes Konstruieren, (Berlin: Springer, 1971).
24. Tomiyama, T., Umeda, Y., and Yoshikawa, H., "A CAD for Functional Design" Annals of the CIRP, Vol. 42, No.1 (1993), pp. 143-146.
25. Hubka, V., and Eder, W.E, Theory of Technical Systems, (Berlin: Springer, 1988).

26. Proceedings of the Workshop on Universal Design Theory, Karlsruhe, May 1998, "Universal Design Theory", by Grabowski, H., Rude, S., and Grein, G.
27. Grabowski, H., Rude, S., and Huang, M., "Supporting Early Phase of Mechatronic Product Design with Layered Function Models", IEEE Transactions , Vol. 2 (1999), pp. 914-918.
28. Al-Hakim, L., Kusiak, A., and Mathew, J., "A Graph-theoretic Approach to Conceptual Design with Functional Perspectives", Computer-Aided Design, Vol. 32 (2000), pp. 867 – 875.
29. Gorti, S.R. and Sriram, R.D., "From Symbol to Form: a Framework for Design Evolution", Technical Report, No: IESL 94-02, (Intelligent Engineering Systems Laboratory, Dept of Civil and Environmental Engineering Nov 1994).
30. Cole, E.L. (Jr.), "Functional Analysis: A System Conceptual Design Tool", IEEE Transactions on Aerospace and Electronic Systems, Vol. 34, No. 2 (April 1998), pp. 354-365.
31. Cross N., Engineering Design Methods: Strategies for Product Design (New York: Wiley, 1994).
32. Mudge A.E, Value Engineering: A Systematic Approach, (Pittsburgh, PA: J. Pohl Associates, 1989) unpublished.
33. Deng, Y. -M, Britton, G.A., and Tor, S.B., "Constraint-Based Functional Design Verification for Conceptual Design", Computer-Aided Design, Vol. 32 (2000), pp. 889-899.
34. Usher, J.M., Roy, U., and Parsaei, H.R., ed., Integrated Product and Process Development Methods, Tools, and Technologies, "Functional Design, by Tor, S.B., Britton, G.A., Chandrashekar, M., and Ng, K.W." (New York: Wiley, 1998), Chapter 2, pp. 29-58.
35. Deng, Y.-M., Britton, G.A., and Tor, S.B., "A Design Perspective of Mechanical Function and its Object-oriented Representation Scheme", Engineering with Computers, Vol. 11, No. 4 (1998), pp. 309-320.
36. Proceedings of the 12th International Conference on Engineering Design, Munich, Germany, 1999, "A Comprehensive Representation Model for Functional Design of Mechanical Products, by Tor, S.B., Deng, Y.-M., and Britton, G.A." (Germany, 1999), Vol. 3, pp.1929-1932.
37. Proceedings of the Fifth ACM Symposium on Solid Modeling, Ann Arbor, Michigan, USA, 1999, "A Computerized Design Environment for Functional Modeling of Mechanical Products, by Deng, Y.-M., Tor, S.B., Britton, G.A." (Ann Arbor, Michigan, USA, 1999), pp. 1-12.

38. Roy, U., Pramanik, N., Sundarsan, R., Sriram, R.D., and Lyons, K.W., "Function-to-Form Mapping: Model, Representation and Applications in Design Synthesis", Computer-Aided Design, Vol. 33 (2001), pp. 699-719.
39. Szykman, S.; Fenves, S. J.; Keirouz, W., and Shooter, S.B., "A Foundation for Interoperability in Next-generation Product Development Systems", Computer-Aided Design, Vol. 33, No. 7 (2001), pp. 545-559.
40. Proceedings of 1999 ASME Design Eng. Technical Conferences (International Conf. Design Theory and Methodology), "The Representation of Function in Computer-Based Design, by Szykman, S., Racz, J.W., and Sriram, R.D." (New York: American Society of Mechanical Engineers, 1999), DETC99/DTM-8742
41. Szykman, S., Sriram, R.D., Bochenek, C., Racz, J.W., and Senfaute, J., "Design Repositories: Engineering Design's New Knowledge Base", IEEE Intelligent Systems & their applications. Vol. 15, No. 3 (2000), pp. 48-54.
42. Hsu, W, and Woon, I.M.Y, "Current Research in the Conceptual Design of Mechanical Products", Computer-Aided Design, Vol. 30, No. 5 (1998), pp. 377-389.
43. McAdams, D.A., Stone, R.B., and Wood, K.L., "Functional Interdependence and Product Similarity Based on Customer Needs", Research in Engineering Design, Vol. 11 (1999), pp. 1-19.
44. Proceedings of the Tenth FAIM 2000 - Flexible Automation And Intelligent Manufacturing Conference, University of Maryland, College Park, Maryland, June 26-28, 2000, "A Methodology for Knowledge Discovery and Classification, by Terpenney, J.P., Strong, S., and Wang, J." (2000).
45. Yan, Hong-Sen, Creative Design of Mechanical Devices (Singapore: Springer-Verlag, 1998), pp. 9-10.
46. Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., and Tomiyama, T., "Supporting Conceptual Design Based on the Function-Behavior-State Modeler", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 10 (1996), pp. 275-288.
47. Wang, Y. and Nnaji, B., "Functionality-Based Modular Design for Mechanical Product Customization Over the Internet", Journal of Design and Manufacturing Automation, Vol. 1, No. 1 & 2 (2001) pp. 107-121.
48. Ambler, A. P., and Popplestone, R. J., "Inferring the Positions of Bodies from Specified Spatial Relationships", Artificial Intelligence, Vol. 6, No. 2 (1975), pp. 157-174.

49. Popplestone, R. J., Ambler, A.P., and Bellos, I. M., "RAPT: An Interpreter for a Language Describing Assemblies", Artificial Intelligence, Vol. 14 (1978), pp. 79-107.
50. Popplestone, R. J., Ambler, A.P., and Bellos, I. M., "An Efficient and Portable Implementation of RAPT", Proceedings of First ICAA, (Bedford, UK: IFS (Publications), March 1980), pp. 411-422.
51. Popplestone, R. J., "The Edinburgh Designer System as a Framework for Robotics", Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 3 (1987), pp. 1972-1977.
52. Liu, H.S., and Nnaji, B. O., "Design with Spatial Relationships," Journal of Manufacturing Systems, Vol. 10, No. 6 (1991)
53. Liu, H., "Design with Spatial Relationships and Tolerance Specification and Propagation for a Product Modeler" (unpublished Ph.D. Dissertation, School of Engineering, University of Massachusetts, 1992).
54. Kalay, Y. E. ed., Computability of Design, "Designing with Constraints, by Gross, M., Ervin, S., Anderson, J. and Fleisher, A." (New York: John Wiley & Sons, Inc., 1987), pp. 53-83.
55. Tong, C. and Sriram, D., ed., Artificial Intelligence in Engineering Design: Design Representation and Models of Routine Design, Volume I, "Tools and Techniques for Conceptual Design, by Serrano, D. and Gossard, D." (New York: Academic Press Inc, 1992), pp. 71-116.
56. Ullman, D. G., The Mechanical Design Process (2nd edition; New York: McGraw-Hill, Inc. 1997).
57. Ullman, D. G., Dietterich, T. G. and Stauffer, L. A., "A Model of the Mechanical Design Process Based on Empirical Data," Artificial Intelligence for Engineering Design, Analysis & Manufacturing, Vol. 2, No. 1 (1988) pp. 33-52.
58. Sivaloganathan, S. and Andrews, P. T. J., ed, Design for Excellence: Engineering Design Conference, "Invited Specialist Paper: Constraints Modeling in Product Design, by Lin, L. and Chen, L. -C." (London: Professional Engineering Pub., 2000), pp. 151-161.
59. Stauffer, L. A. and Slaughterbeck-Hyde, R. A., "The Nature of Constraints and Their Effect on Quality and Satisfying," Design Theory and Methodology, ASME, DE Vol. 17 (1989), pp. 1-7.
60. Van Hentenryck, P. and Saraswat, V., "Strategic Direction in Constraint Programming", ACM Computing Surveys, Vol. 28, No. 4 (1996), pp. 701-726.

61. Chan, W. T. and Paulson, B. C., "Exploratory Design Using Constraints," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 1, No. 1 (1987), pp. 59-71.
62. Suh, N. P., 1990, The Principles of Design, (New York: Oxford University Press, 1990).
63. Shah, J. J. and Mantyla, M., Parametric and Feature-based CAD/CAM: Concepts, Techniques, and Applications, (New York: John Wiley & Sons, Inc., 1995)
64. Gero J. S. ed., Artificial Intelligence in Engineering: Design, "Constraint Management in MCAE, by Serrano, D. and Gossard D." (New York: Computational Mechanics Publications, 1988).
65. Gu, P. and Kusiak, A., ed., Concurrent Engineering: Methodology and Applications, "Constraint Management in Design Fusion, by Navinchandra, D., Fox, M. S. and Gardner, E. S." (New York: Elsevier Science Publishers, 1993), pp. 1-30.
66. Kusiak, A. and Wang, J., "Decomposition of the Design Process", Journal of Mechanical Design, Vol. 115, No. 4 (1993), pp. 687-695.
67. Bowen, J., O'Grady, P., and Smith, L., "A Constraint Programming Language for Life-Cycle Engineering", Artificial Intelligence in Engineering, Vol. 5, No. 4 (1990), pp. 206-220.
68. Bowen, J. and Bahler, D., "Frames, Quantification, Perspectives, and Negotiation in Constraint Networks for Life-Cycle Engineering", Artificial Intelligence in Engineering, Vol. 7, No. 4 (1992), pp. 119-226.
69. Thornton, A. C. and Johnson, A. L., "CADET: A Software Support Tool for Constraint Processes in Embodiment Design", Research in Engineering Design, Vol. 8, No. 1 (1996), pp. 1-13.
70. Lakmazaheri, S. and Rasdorf, W. J., "Constraint Logic Programming for the Analysis and Partial Synthesis of Truss Structures", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 3, No. 3 (1989), pp. 157-173.
71. Jaffar, J. and Maher, M. J., "Constraint Logic Programming: A survey", Journal of Logic Programming, Vol. 19, (1994), pp. 503-581.
72. Hsu, W. and Woon, I.M.Y., "Current Research in Conceptual Design of Mechanical Products", Computer-Aided Design, Vol. 30, No. 5 (1998), pp. 377-389.
73. Yan, H., Creative Design of Mechanical Devices, (Singapore: Springer-Verlag Ltd., 1998), pp 17-31.
74. Weizsäcker von, C.F., Die Einheit der Nature – Studien, Munchen: Hanser 1971.

75. Kramer, G. A., Solving Geometric Constraint System, a Case Study in Kinematics, (Cambridge, Massachusetts: MIT Press, 1992).
76. Liu, Hsu-Chang, Nnaji, Bartholomew O., "Design with Spatial Relationships", Journal of Manufacturing Systems, Vol.10, No.6 (1991), pp.449-463.
77. Hamrock, J.B., Jacobson, B., and Schmid, S. R., Fundamentals of Machine Elements, (Boston: McGraw-Hill Pub., 1999), pp. 111-112.
78. Wolfson, R. and Pasachoff, J. M., Physics for Scientists and Engineers, (New York: Addison Wesley Longman, Inc., 1999).
79. Watson, M. and Harmon, P., Understanding UML: The Developer's Guide, with a Web-Based Application in Java, (San Francisco: Morgan Kaufmann Pub., Inc., 1998).
80. Nnaji, B. O., Muogboh, O. S., and Ene, P., "Evolving CAD Systems for Collaborative Product Design and Realization", Conference Proceedings, Sixth Africa-USA Intl. Conference on Manufacturing Technology, Abuja, July 2002, (Preprint).
81. Nnaji, B.O., Wang, Y., Kim, K., and Muogboh, O., "Pegasus: A Service-Oriented Product Engineering System over the Internet", Submitted to IIE Transactions on Design.
82. Visio 2002 Product Information, February 2, 2003, <http://www.microsoft.com/office/visio/evaluation/default.asp>.
83. American Iron and Steel Institute, Southfield, Michigan, "Great Designs in Steel Seminar 2002 Launches to an Enthusiastic Crowd of 500 Automotive Engineers", 2002 E-zine article, http://www.steel.org/autosteel/articles/2002_gdis_ezine.htm
84. Guinta, L. R. and Praizler, N. C., The QFD Book: The Team Approach to Solving Problems and Satisfying Customers Through Quality Function Deployment, (New York: Amacom, American Management Association, 1993).
85. Mazur, Glenn H., "QFD for Service Industries: From Voice of the Customer to Task Deployment", The Fifth Symposium on Quality Function Deployment, Novi, Michigan, June 1993.
86. Beermann, H.J., The Analysis of Commercial Vehicle Structures, (London: Mechanical Engineering Publications Limited, 1989).
87. Corney, T., and Theodore Lim, T., 3D Modeling with ACIS, (2nd Edition, Scotland: Saxe-Coburg Publications, 2002).
88. Spatial Inc., "3D Software Development Technologies", 2003, <http://www.spatial.com/>.

89. ANSYS Inc., “ANSYS Software”, 2003, <http://www.ansys.com/ansys/index.htm>.
90. ADINA R & D, Inc., “The Finite Element System for Structures, Heat Transfer, and CFD”, 2003, <http://www.adina.com/>.
91. CFX, “Computational Fluid Dynamics (CFD) Software and Services”, 2003, <http://www.software.aeat.com/cfx/default.asp>.