2001

# Function modeling based on interactions of mass, energy and information.

Bo. Yang
*University of Windsor*

# Function Modeling Based on

# Interactions of Mass, Energy and Information

By

Bo Yang, B. A. Sc.

A Thesis
submitted to the
Faculty of Graduate Studies and Research
through Industrial & Manufacturing Systems Engineering
in partial fulfillment of the requirements for the
Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada
2000

928714

I hereby declare that I am the sole author of this thesis. I authorize the University of Windsor to lend this thesis to other institutions, or individuals, for the purpose of scholarly research.

Bo Yang

I further authorize the University of Windsor to reproduce this thesis by photocopying, or by other means, in total or in part, at the request of other institutions, or individuals, for the purpose of scholarly research.

Bo Yang

The University of Windsor requires the signatures of all persons using or photocopying this thesis. Please sign below, and give an address and a date.

# ABSTRACT

Although the ultimate goal of engineering design is to present a physical form (components and their relationships) of a certain product, function modeling is regarded as the initial and determining stage in the entire design process. While a variety of research efforts have studied and continue to study the role of function modeling in design, the concepts and representations of function among researchers are quite different. This makes it very difficult to utilize research achievements in developing an integrated knowledge-based system (e.g. the AIM-D system established in [1]) that can be used universally throughout an arbitrary engineering design process.

In this thesis, a new function model called Interaction-based Function Model (IFM) which is based on the *basic interactions between mass, energy, and information* is presented. A small set of primitive function types which could represent most (perhaps all) product functions is deduced from this model. The author believes that IFM can provide a widely accepted function-modeling platform that could integrate the current research achievements in this area. Furthermore, by applying IFM to knowledge aided design systems (e.g. ACM system establish by Salustri in [2]), it can be used to shorten the lead-time of product development, motivate creative design, and organize design information system.

An example of implementing IFM to facilitate the conceptual design process is presented. The results demonstrate that this methodology can be used as a design guide to help designers identify key issues in the early design stage without blocking the creativeness of the designers.

# DEDICATION

To

My Family

# ACKNOWLEDGEMENTS

I would like to take this opportunity to deeply thank my supervisor, Dr. Filippo A. Salustri for his inspiration, encouragement, support and great guidance. I would further like to extend my thanks to the committee members, Dr. Peter Frise, and Dr. Michael H. Wang for sharing their helpful suggestions with me. I would also like to express my gratitude to Ms. Jacquie Mummery for her help, and smile, whenever the opportunity arose in my entire graduate study. Finally, I would like to thank everyone that assisted and inspired me throughout, and made this accomplishment complete.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# NOMENCLATURE

**Agent**

Definition in Artificial Intelligence

An entity that resides in environments where it interprets "sensor" data that reflect events in the environment and executes "motor" commands that produce effects in the environment. An agent can be purely software or hardware. In the latter case a considerable amount of software is needed to make the hardware an agent.

Example

Agents can be used in the meeting scheduling. Each person is assigned with an agent that stores his/her private schedule and preference. They negotiate with other agents about making a public schedule by referring user's private schedules and preferences [67].

**Constraint**

Definition in Computer Programming

A constraint is simply a logical relation among several unknowns (or variables), each taking a value in a given domain. A constraint thus restricts the possible values that variables can take, it represents some partial information about the variables of interest.

Example

"The circle is inside the square" relates two objects without precisely specifying their positions, i.e., their coordinates. Now, one may move the square or the circle and he or she is still able to maintain the relation between these two objects. From the user (human) point of view, everything remains absolutely transparent. [68]

**Ontology**

Definition in dictionary

The science of metaphysics which investigates and explains the nature and essential properties and relations of all beings, as such, or the principles and causes of being [49].

Definition in Artificial Intelligence

Ontology is an explicit specification of a conceptualization [51]. It is a set of definitions of content-specific knowledge representation primitives: classes, relations, functions, and object constants [50]. Pragmatically, a common ontology defines the vocabulary with which queries and assertions are exchanged among agents [51].

Example

A bibliography ontology is going to be designed to facilitate automatic translation among existing bibliographic databases. In this ontology, we need to list the definitions of the words such as *references, titles*, documents, *authors*, and *publishers* etc. Furthermore, we also need to define the relationships among them, such as *title* and *document* are two retrieval keys for a *reference,* and *author* and *publisher* are two retrieval keys for *document*. All of these representation primitives consist of a bibliography ontology.

**Teleology**

Definition in dictionary

The study of design or purpose in natural phenomena; the use of ultimate purpose or design as a means of explaining natural phenomena. [49]

Definition in Artificial Intelligence

The cause and effect relationships between activities and constraints; term in other fields

*intent* in functional reasoning; *causal dependency* in model based reasoning [61]

Example

A helicopter flight [activity] causes equipment to be on site [constraint]; the extraction of a tooth [activity] is intended to make the patient more comfortable [constraint];

**Taxonomy**

Definition in dictionary

That division of the natural sciences which treats of the classification of animals and plants; the laws or principles of classification [49].

Definition in Artificial Intelligence

Any organized structure of information; no particular onus exists to have an understanding of the underlying essential characteristics of the domain of interest. Often used, however, as a synonym for ontology [40].

**Mereology**

Dictionary definition: the science of classification, but based on structural aspects of the subject domain rather than general principles.

Artificial Intelligence and Knowledge Representation definition:

A structure used to organize information about a domain of interest based on the various kinds of part-whole relation, rather than (necessarily) on the essential properties of the domain of interest.

**Subsumption**

We define subsumption, saying that F subsumes G, if and only if: [42]

1. The type of F is more general than the type of G

2. If a feature f is defined in F then f is also defined in G such that the value in F subsumes the value in G

3. If two paths are shared in F then they are also shared in G

Example

Engine dysfunction subsumes ignition systems dysfunction, and ignition systems dysfunction subsumes spark dysfunction.

**80/20 Principle**

The 80/20 Principle--that 80 percent of results flow from just 20 percent of our efforts [41]

**AIM-D**: Axiomatic Information Model for Design. It is a formal theory based on axiomatic set theory which concerns representing product information in the design process. [1, pp 1]

**IFM**: Interaction-Based Function Model. A function is any mechanism by which transformations from one or more basic element - mass (M), energy (E), or information (I) - to another(s) occurs in an entity, in response to environmental stimuli. [pp vi]

**ACM**: Artifact-Centered Modeling. It is a general framework to partition the problem of describing design into manageable components. It is shown graphically in Figure 2.6. [2, pp1]

**KBS:** Knowledge-based systems [pp 3]

**FR**: Functional Requirement [pp 22]

**FBS:** function-behavior-structure [pp 25]

**AI:** Artificial Intelligence [pp 27]

**FS:** Function Specification [pp 39]

**FIF**: Function Identification Form [pp 57]

**LFIF**: Leaf Function Identification Form [pp

**NRC**: Node Relation Chart [pp 68]

**AFFT**: Abstract Function Family Tree [pp 71]

**Frame**: Frame is a way that is used to develop knowledge bases. It has frame/slot/facet/value quadruplets to represent different level of details. In frames, slots describe things about the object; but then facets can be thought of as variables whose values describe the slot. [pp 61]

**Facet**: Facet is a level of detail in frame. It is a component of slot. There's usually one special facet in each slot -- called, say, value, that actually contains the value of the slot. Other facets in the same slot are used to contain other useful information about the slot (typically meta-data). Facet can also contain procedures of slot. For example, you can add a procedure to a facet of a slot that will maintain an "inverse relation" automatically whenever a new value is assigned to that slot [pp 61]

# CHAPTER ONE

# INTRODUCTION

Conceptual design is an early phase in the design process, which involves the generation of solution concepts to satisfy the functional requirements of a design problem [32]. Function-based modeling is useful for capturing engineering design intent at an abstract level [33]. It is widely accepted that function-based modeling has the following advantages over structure-based modeling: (a) elimination of unnecessary constraints in the early design stages to motivate creative design, and (b) representation of key design characteristics as indexes when searching design databases. Many researchers have identified the importance and relevance of function modeling, but a well-defined and universally accepted concept of *function* does not exist. Table 1.1 lists some of the current function definitions given by the researchers in this domain.

This thesis builds on a function-modeling framework proposed in [3, 34] by Salustri and Yang. A brief review of this framework will be discussed in Section 1.1. The author proposes that function is an interaction between three fundamental elements – *mass*, *energy*, and *information*. This formulation leads to a small, arguably complete taxonomy of primitive functions, from which arbitrarily complex function descriptions can be developed. The author's approach has yet to be formalized within the general logical framework, such as Salustri's AIM-D [1]. However, the outline presented here clearly defines the range and scope of the representation, and appears to be sufficient to represent many functions.

Table 1.1 Function definitions given by researchers in function-based modeling domain

| Author | The Definition of Function |
|---|---|
| Salustri [3] | How the system achieves its behavior |
| Kitamura and Mizoguchi [4] | A result of interpretation of the behavior under the intended goal |
| Chittaro, Tasso and Toppano [5] | The relation between its behavior and the goals (teleology) assigned to it by the designer |
| Hodges [6] | Function is represented with a schema that describes the initiation and terminating states associated with a process sequence |
| Keuneke [7] | The function of a device is its intended purpose. |
| Sasajima et al [8] | The interpretation of the behavior under a desirable state which the component is expected to achieve |
| Chandrasekaran and Josephson [9] | Let G be a formula defined over properties of interest in an environment E. Let us consider the environment plus an object O. If O (by virtue of certain of its properties) causes G to be true in E, we say that O performs, has, or achieves the function (or role) G |
| Qian and Gero [17] | Function specify what a design does |

## 1.1 Background

This thesis builds on the work presented in [3] and [34] by Salustri and Yang. In those papers, function and behavior are not seen as fundamental characteristics of real-world objects, but as descriptions of those characteristics that are dependent on the reference frame of the user. Let us consider the following statements [3]:

1) The refrigerator keeps food cold

2) The refrigerator preserves food

3) The refrigerator lowers ambient temperature in an enclosed space

Any of these statements in isolation can be considered a behavior of a refrigerator. If we consider statement #1 as a behavioral description and ask *"How is this behavior achieved?"* One possible answer is given by statement #3. So statement #3 is a functional description related to the behavioral description in statement #1. We may also ask "Why does the refrigerator keep food cold?" One answer to this question is statement #2. In this case statement #1 describes a function rather than a behavior and statement #2 describes a behavior with respect to statement #1.

From this observation, it is shown that a function in one context can be a behavior in another context, leading the authors to propose that function and behavior are different views of the same thing, and thus should be uniformly represented in knowledge-based systems (KBS) for design.

Furthermore, [3] and [34] provides the basis of a representational form for so-called predicative descriptions (e.g. function or behavior). A verb-object pair (VOP) is suggested for this purpose, for example *classify documents, relieve stress, transfer heat* etc. The verb part is intended to capture the active, dynamic part of the function, and the

object part is intended to capture some notion of the entity upon which the function acts. It is then shown that abstraction can occur on both the verb and the object parts. While this kind of coupling makes some reasoning tasks (like subsumption) much more complex, it provides a very rich expressive form for function and behavior.

[3] and [34] only sketch the model in preliminary terms. In this research, those ideas are pursued further, with particular emphasis on the identification of fundamental function types.

### 1.2 Statement of thesis

The thesis of this work is that it is possible to develop a new function model that provides a general and interchangeable platform in function related research domain. Such a model will facilitate the engineering designer to improve design efficiency, formalize information flow, and organize design databases in the conceptual design stage.

### 1.3 Organization of this dissertation

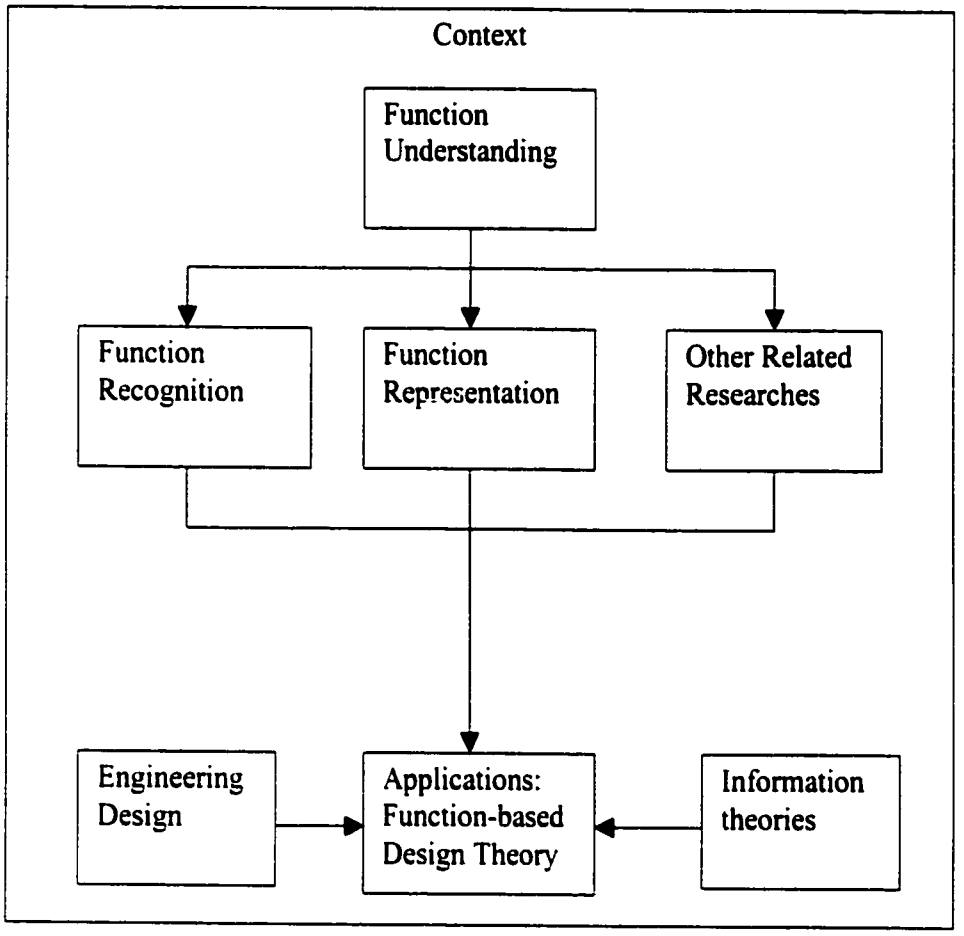
This dissertation is organized as follows: Chapter 2 reviews the current literature in function modeling. Chapter 3 presents the author's new function model based on interactions of mass, energy and information. A proposed example of applying this new function model in design practice is given in Chapter 4. Conclusions are presented in Chapter 5. Two appendixes are attached after the reference section.

# CHAPTER TWO

## LITERATURE REVIEW

Function related research can be classified into 6 categories. Figure 2.1 represents their relationships. The foundation of function related research is *function understanding*, which describes various key concepts of functions. This research branch is discussed in Section 2.1. Three research branches are derived from function understanding. The goal of *function recognition* (Section 2.2) is to identify product functions in existing objects, while *function representation* (Section 2.3) is to find an appropriate form to describe functions. There are also some research efforts in the level between function understanding and applications; the author integrates them into *other related research* branch and discusses them together in Section 2.4. Current *Function-based design theory* (in Section 2.5) could be viewed as an integrated application of its higher-level research domains (Section 2.1- 2.4) and it has the most practical value to real world product development activities. Section 2.6 explains the role of *context* in function related research. The author also believes that information theories are highly related to function modeling, so some of them are discussed in Section 2.7. Since the ultimate goal of the author's function model is to facilitate engineering designers in real world design endeavor, some related engineering design literature are reviewed in Section 2.8.

Figure 2.1 Relationships among function related research areas




*This figure displays the relationships of the function-related research domains.* ***Function understanding*** *is the basis of this research field and* ***function-based design theory*** *has the most practical value to real world design activities. All of these research branches are confined to the environment of "context".*

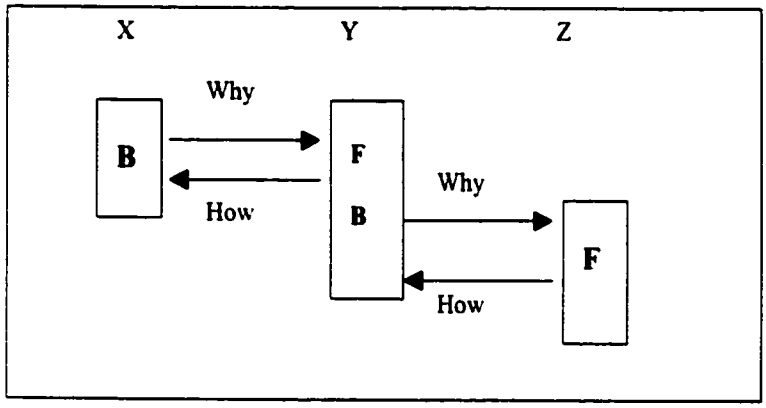
## 2.1 Function understanding

Function understanding is the foundation of all function-related research. The main purpose of this study is to explore the core characteristics of function and then provide a basic framework for other function-related research. *"What is function?"* is the key question has to be answered by each researcher. By reviewing the following papers, the author presents a picture of this research area.

In [3], Salustri presents preliminary results in the development of a KBS component for modeling product function and behavior. The author defines behavior as *the response of a system to stimuli*, and function as *how the system achieves its behavior*. Furthermore, the author uses predicative clauses consisting of verb/object pairs (VOPs), such as *preserve food* and *lower temperature*, to capture the commonality of function and behavior. Functions and behaviors can be further specified by how and why directions. Fig 2.2 shows this how and why relations among KBS components.

The significance of this paper is as follows. First, the author's definitions are quite different to the traditional views in the field. Other researchers use the word *behavior* to represent the meaning of function in his paper and use the word *function* to represent the meaning of behavior. Second, the author indicates that a function in one context can be a behavior in another context. This interesting phenomenon leads the author to propose that function and behavior are different views of the same thing. Third, the author clearly points out the importance of studying the concept of function, which is the foundation of function-based modeling and not much efforts are devoted to until present.

Fig 2.2 *How* and *Why* relations among KBS components



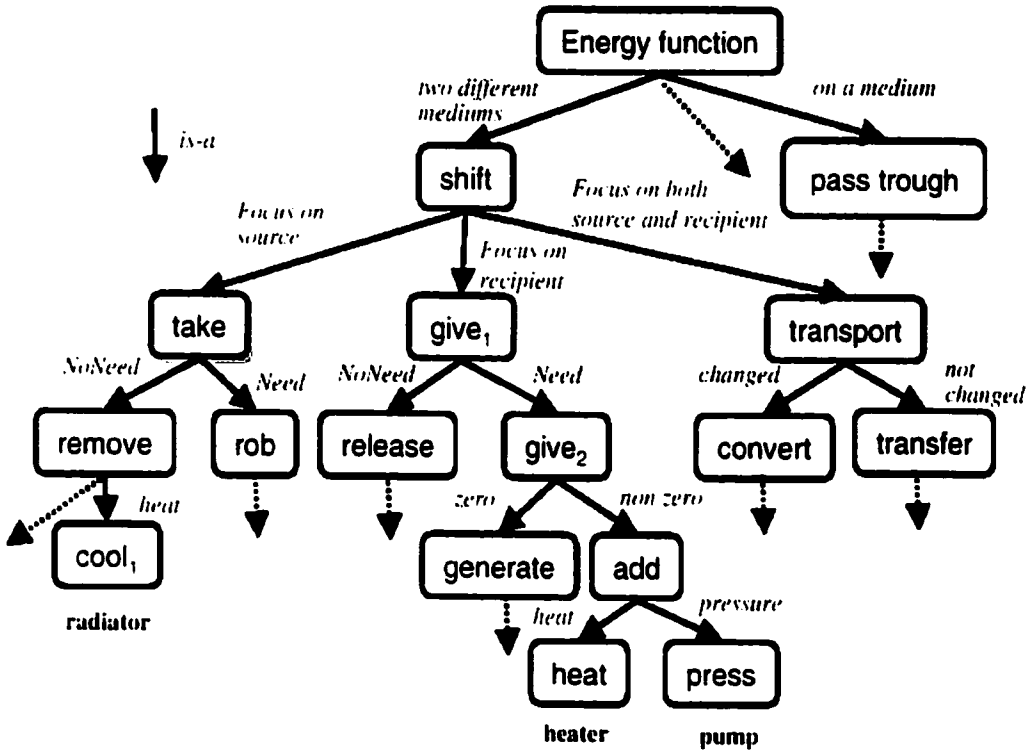

*X, Y, Z are three KBS components. Since Y explains* ***how*** *X achieves and X explains* ***why*** *Y exists, X is treated as a behavior (B) and Y is treated as a function (F) within XY pair. According to the same principle, Y is treated as a behavior and Z is treated as a function within YZ pair. For example, "The refrigerator keeps food cool"(Y) explains why "The refrigerator preserves food" (Z) and "The refrigerator lowers ambient temperature in an enclosed space"(X) explains why "The refrigerator keeps food cool"(Y)*

Kitamura and Mizoguchi propose a framework for functional understanding which aims to solve redesign problem [4]. Their framework tries to identify functional structures of an artifact from its behavioral model. The functional modeling language they used is called FBRL (Function and Behavior Representation Language). It enables them to build an ontology in terms of its mapping primitives between behavior and function. This mapping is called FT (functional toppings which are a set of primitives for mapping between behavioral and functional spaces). The ontology provides a basis for functional reasoning at a comprehensible conceptual level. It enables the functional understanding system to bridge a gap between physical behavioral level and the conceptual functional level by using FTs, which is called behavior-function mapping; and then generate functional hierarchies at the conceptual level, which is called functional hierarchy understanding.

In their paper, *behavior is defined as temporal changes of parameter values and function is defined as a result of interpretation of the behavior under the intended goal.* The authors study the relationships between behavior and function and apply FBRL to decompose the higher-level functions to hierarchy structure. Figure 2.3 illustrates this function decomposition process.

Although this is a clear structure, the roles that guide this decomposition process are not discussed in this paper. In Figure 2.3 the top-level function – *Energy function* is decomposed to *shift* and *pass through* by the number of *mediums (one* and *two)*, but the authors do not state why they use this kind of classification. Since the authors emphasize that the function concepts should be implementation independent, there should be a set of general or exclusive guidelines for this kind of decomposition process. Otherwise if there

Figure 2.3 Function Decomposition Schema [4]




*Applying FBRL (Function and Behavior Representation Language) which is introduced by Kitamura and Mizoguchi to decompose the higher level function, say "Energy Function", to hierarchy structure.*

exist several routes to decompose a function, such as the *Energy function*, then we have to decide which route to apply in a specific circumstance (e.g. structures, behaviors). This indicates that the function decomposition process discussed in this paper is not solid enough to support the authors' *implementation-independent* characteristic of function.

In [5], Chittaro, Tasso and Toppano review the existing approaches to the definitions and the representations of function and classify them into two groups. One is "hybrid" approach and the other is "pure" approach. After discussed the major advantages and limitations of each group, the authors introduce their own function definition – *the function of a system is defined as the relation between its behavior and the goals (teleology) assigned to it by the designer.* The function representation of a system is aimed to describe how the behaviors of individual components contribute to the achievement of the goals assigned to each part of the system. Functional knowledge is then represented through three kinds of models: (a) the model of function roles, (b) the model of process, and (c) the model of phenomena.

The significance of this paper is that it discusses the nature of functional knowledge and disambiguates this type of knowledge from other types, such as teleology, behavior, and structure. The authors also introduced a multi-modeling approach to represent and reason about physical systems. The key idea of this approach is considering the task of reasoning about a physical system as a cooperative activity. It exploits the contribution of several separate models, such as behavioral model, function model, and structure model of the system and connects each individual model to others explicitly and appropriately.

The authors discuss the relations among structural, behavioral, functional, teleological and empirical knowledge and integrate behavior and teleology with function

respectively. But they do not mention the possibility of integrating other types of knowledge with functional knowledge. Furthermore, they do not consider the possibility of integrating more than two types of knowledge together, for example, combining behavior, function and teleology as a whole system.

The current author would like to briefly summarize some points within these papers on function understanding domain:

- All the authors have emphasized the importance of capturing functional aspect of an artifact as well as behavioral or structural aspects.
- All the papers have indicated that function is highly related with behavior.
- There is not a universally accepted concept of function.

It is obvious that building a well-defined and widely applicable function model is expected. Since function ontology is the foundation of all function-related research, the modification of function ontology will affect the entire research scope.

### 2.2 Function recognition

Function recognition is an important research domain derived from function understanding. Given an assumed ontology of function, this research domain seeks to identify functions in existent products. In an integrated knowledge based system, function recognition could be used as a tool to analyze how close two products are in terms of their functions. For example, compact disk and cassette are two different products but they share the same function, say *store electronic information*. Based on this similarity, the data transference between these two products is theoretically and practically possible.

Achinstein recognized three types of object functions: design functions (what it was designed to do), usage functions (what it is used to do), and service functions (what it actually does which serves to benefit something) in [28]. He gives a chair example to explain this idea. "Suppose that magnificent chair was designed as a throne for a king, i.e., it was designed to seat the king. However it is actually used by the king's guard to block a doorway in the palace. But this chair is so beautiful that it draws crowds to the palace to view it and results in a lot of financial contributions."

Achinstein's categorization is rooted in the cognitive roles played by designers and users, but the concepts of functions from user's point of view are quite different from that of designer's and they are highly related with another important concept, *context*. The current author believes that the interaction-based function model that is discussed in Chapter 3 could bridge this gap between design functions and user functions in a relatively easy manner.

Lind divided functions into two types in [29]:

- *Mass and Energy function*: There are six sub-type functions within this catalog. They are "source", "sink", "storage", "balance", "transport" and "barrier".
- *Action function*: There are four sub-type functions within this catalog. They are "maintain," "destroy," "produce," and "suppress."

Lind's classification is convenient from the engineering perspective of view but his approach does not account for the inherent coupling between function and the object(s) upon which the function acts.

Hodges introduces an integrated theory, called FONM (Functional Ontology for Naïve Mechanics) in [6]. It represents device function in terms of both its low-level

structure/behavior and its use in problem-solving context. FONM is composed of three independent layers: (a) device statics which describes the device at rest, (b) device dynamics which describes what will happen to a device when perturbed, precluding transient effects, and (c) device pragmatics which describes how and why the device is used. Within each layer, physical, behavioral, and functional primitives could be identified in different abstract levels.

The author provides an example to show how to use functional recognition to facilitate improvisation design (innovative design in a new context). "Suppose someone left his key in the car, and the only thing he can find is a hanger. Since the hanger has the physical property of length, diameter, stiffness, and ductility, it can be bent into an appropriate shape, slid between the car door and window, and thus used to unlock the car." In this particular context, the hanger is a device that has the function of unlocking.

Hodges believes that his framework is effective to study function recognition, but his method of identifying function is mainly based on identifying object properties and attributes such as shiny, smooth, cold, etc. It seems that this approach relies more on experiences in a specific context. For the hanger example mentioned above, an electrical engineer might use it to connect broken wires. It is hard to establish relations between these two cases of improvisations although the same object – the hanger is used in both situations. Furthermore, Hodges stated that the physical properties and object behaviors or functions are not one-to-one mapping; this leads to an important issue of how to identify essential properties of an object in a certain context. Unfortunately, the author does not address this in his paper. Another problem of this paper is that the author states *function is represented with a schema that describes the initiation and terminating states*

*associated with a process sequence.* This definition may lead to confusion with other concepts such as state or behavior.

By reviewing other papers in this research branch, the current author again realized that without well-defined function ontology, it is difficult to conduct further research.

### 2.3 Function representation

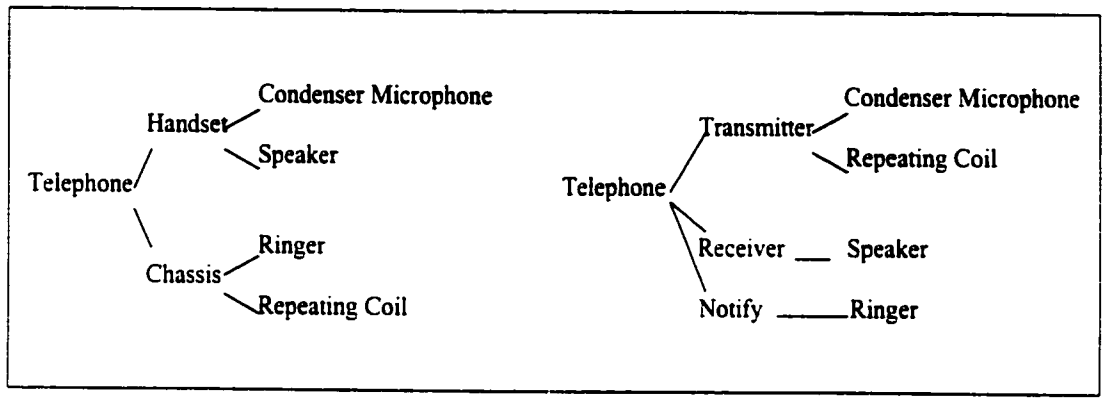
Representing how things work is an important research branch in artificial intelligence. Although many researchers have sought an appropriate representation, there is no real agreement on what should be included in the function representation model.

Keuneke [7] has studied languages for representing device structure and commitments for representing device behavior. She organizes the knowledge of device representation on the basis of functionality. Figure 2.4 demonstrates her idea.

In her article, function and behavior are coupled together. *The function of a device is its intended purpose. Function is what is expected and behaviors are how this expected result is attained.* The most significant part of her paper is that she classified functions into four basic types, which are *To Make, To Maintain, To Prevent, and To Control.* This classification has been widely cited by other researchers.

Keuneke does not give the reasons for her categorization in her paper, and the completeness of this classification still needs to be studied. Though substantially different from others, this approach, like Lind's, does not recognize the coupling between functions and objects too. Furthermore, due to its relatively abstract nature, it is difficult to use inductive inference to argue that the list is complete. For example, Sasajima et al.

Figure 2.4 Structural representation of a telephone [7]




*Representing the structure of a device in different ways. By physical relationships (Left) and by functional relationships (Right)*

Figure 2.5 Template for describing a component [8]

```
Behavior:
   Objects:          /* Objects input to or output from the component */
   SubComponents:    /* Names of subcomponents */
   MP- Relations:    /* Relations among input materials and output
                        products */
   SameClass:        /* Sameness of the class of in/out objects */
   InherentParams:   /* Parameters inherent in the component */
   Ports:            /* Connection with neighbor components */
   QN-Relations:     /* Quantitative relations among parameters */

Functional Topping:
   Goal:             /* Goal of the component */
   FuncType:         /* Function type: To Make, To Prevent, To Control,
                        To Maintain, To Enable */
   O-Focus:          /* Focus on a class of Object */
   S-Focus:          /* Focus on a certain Stream */
   Needs:            /* Necessity of output objects */
```

*The attributes from Object to QN-Relations are used to represent the behavior. The slots, from Goal to Needs, represent necessary information for interpreting the behavior which are called function toppings.*

[8] found it necessary to add a new function type, To Enable, four years after Keuneke's original work.

A well-defined and complete function classification is crucial, because it will facilitate the function, behavior and structure retrieval work in a design knowledge-based system. The current author believes a reasonable argument can be made for the completeness of Keuneke's approach. The reading of the literature suggests that categorizations of function in other research have been constructed in an inductive fashion, on the premise that it is impossible to enumerate *a-priori* all possible functions. The current author applies a deductive approach where all the primitive functions arise out of a very small and stable set of primitive entity types, which are mass, energy, and information. It is hard to imagine a fourth entity type being suddenly discovered. Though this certainly does not mean the current author's model is verifiably complete, it does suggest that its completeness is more likely better than other approaches.

Sasajima et al. proposes a language, called FBRL (Function and Behavior Representation Language), to represent function and behavior with the primitives the authors identified. They also discuss its application to explanation generation. FBRL explicitly represents models of each component in a system in terms of two elements. One is *a necessary and sufficient information for simulation of the component* which they call it *behavior*. The other is *the interpretation of the behavior under a desirable state which the component is expected to achieve*, which they call it *function*. By identifying primitives which are necessary for the interpretation of the behavior in various domains, the authors can capture what function is and represent it by selection and combination of these primitives. The authors also investigate the relation between function and behavior

by the primitives of FBRL. As FBRL can represent concepts at various levels of abstraction, it contributes to explanation generation by providing information for mapping behavior of a component to a term which represents its function.

The format of component description is shown in Figure 2.5. Although this is a well-organized format, some interesting phenomena need to be considered regarding function types. The authors use Keuneke's function classification method and add a new function type, *To Enable*, so the completeness of Keuneke's categorization is challenged. But the authors do not prove or mention the completeness issue of their model either, and this weakens the validation of their categorization and the part of FBRL based on this classification.

Chandrasekaran and Josephson describe a formal framework and definition of device function in [9] that attempts to unify and generalize these intuitions. They seek the minimal sufficient ontological framework for developing a formalism that will support design problem solving process through the use of functionally indexed device libraries. The function is defined in the following form "*Let G be a formula defined over properties of interest in an environment E. Let us consider the environment plus an object O. If O (by virtue of certain of its properties) causes G to be true in E, we say that O performs, has, or achieves the function (or role) G.*" Their central idea is that function of an object is the effect it has on its environment. The authors emphasized that "the definition of a function should not make any reference to the structure of the object that has the function." This definition allows functions to be specified as requirements during the design process, prior to embodiment design.

Chandrasekaran and Josephson state that an item in a device library is associated with generic environmental parameters that are bound to values when the item is instantiated. Let's consider pump as an example. The properties of interest in the environment are the quantities of water, Q1(t0), Q1(tf), Q2(t0) and Q2(tf), at time t0, tf, in location L1 and L2. Let G be the formula corresponding to Q1(t0) – Q2(tf) = Q2(tf) – Q2(t0) = K > 0. That is, a positive quantity of water is moved from L1 to L2 from the initial instant to final instant.

In this way, potential functions are associated with library objects. In the universe of engineered objects, causal dependencies can usually be expressed as relations between the properties of objects and property relations can usually be expressed as mathematical functions. The authors believe they present a definition of function that is sufficiently general to express static and dynamic functions, intended and natural functions, and functions of abstract and physical objects. However, it seems insufficient to capture all types of causality relevant to functional reasoning based only on property relations. For example, Richard states "the effect of engine variables such as speed, load and ignition/injection timing on engine emissions and fuel economy is very complex, and can only be explained qualitatively..." [63]

The authors use physical parameters, such as temperature, voltage, or location to monitor the effect of an object on its environment (function). This is a good method by linking functions with physical components. The proposed shortcoming of this method is that defining functions for objects exhibiting multiple environmental effects, which is often the case, is not treated. In the examples presented in their paper, every component has only one function assigned.

There are also some papers, for example [10-13], offering various function representations, but none of them gives a universally accepted definition of function. It seems that each of these papers has explored one or some core characteristics of function. A well-defined function definition should be able to explain their individual discoveries properly. The current author is trying to move toward this goal in this research.

### 2.4 Other related research

Function modeling is a broad research area, and there are some papers that are hard to be classified into any particular branch. Those papers are discussed here.

[14] is a literature survey paper written by Chakrabarti in 1993. He focuses on exploring the answers of some key questions in function modeling study among various researchers. Those questions include "*What is function?*" "*What is form?*" "*What are the relations between these two?*" Thirteen articles are reviewed in his paper that underscores the present understanding about the function definitions, forms, problems, solutions, and their roles in the design context. He concludes that all of these mentioned above are far from unambiguous and he is expecting a functional reasoning theory that could solve these problems.

Chen and Meng [15] concentrate their interest on identifying the functional requirements (FR) of mechanical products and systems. Their objective is to develop a framework for concurrent engineering that integrates the functional requirements into the various activities of the production cycle to serve as links among design, process planning, manufacturing, and inspection. In their paper, the functional requirements are

Figure 2.6 *Fit* Assembly Relation

Symbol AS1

Function Name FIT (A, B)

Attributes of Related Features

Feature A: Cylindrical pin with parameters d1, l1

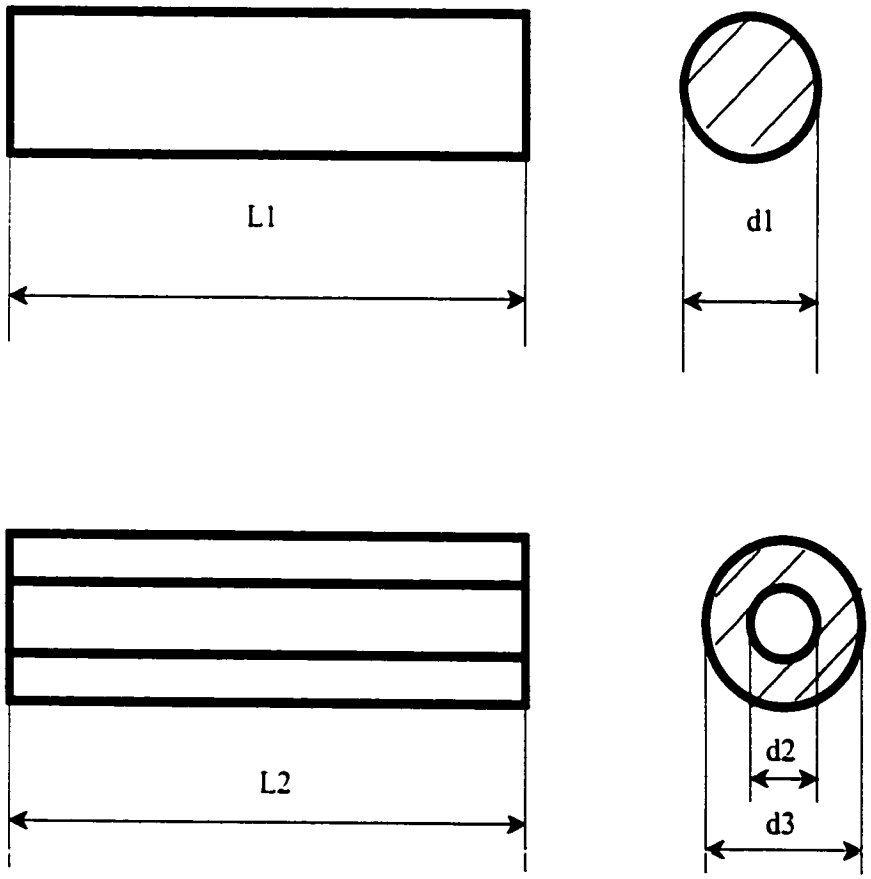
Feature B: Cylindrical sleeves with parameters di, do, l2

Characteristic Set Clearance c and interface pressure P

(1) Requirement: clearance; then $c > 0$ and P is irrelevant;

(2) Requirement: no relative motion; then $c = 0$ and $P = 0$;

(3) Requirement: press fit; then $c < 0$ and $P > 0$.



characterized into three groups: quantitative, abstract, and assembly functional requirements. The hierarchy, decomposition, and dependency check of the functional requirements are identified as three major issues in the proposed framework.

The proposed framework can be used to facilitate various concurrent engineering activities, including functional design, assessment of functionality, manufacturability, and assemblability. For example, there are five types of assembly relations (*Fit, Contact, Attachment, Relative_Position, and Insertion*) which are used to represent the assembly functional requirements. The assembly FR of *Fit* is represented in Figure 2.6.

The current author believes function requirements can be used to help customers specify their real product requirements. On the other hand, it can help designers to identify these requirements as well as their structures and the priorities among them.

Chittaro and Kumar discuss the definitions of structural, behavioral, functional, and teleological knowledge in [16]. The authors emphasize the importance of distinguishing them in representation formalisms. They classify function representations in two major approaches. One is state-based representation wherein function is defined as the abstraction of behavior states. The authors believe that the state-based approach does not provide predefined function primitives, but a language for building user-defined, on-demand functional units. The other approach is a flow-based representation wherein function is defined as a relation between the input and output of energy, matter, and information. The major problem of this approach is that it is hard to list a complete or adequate set of function primitives for a domain.

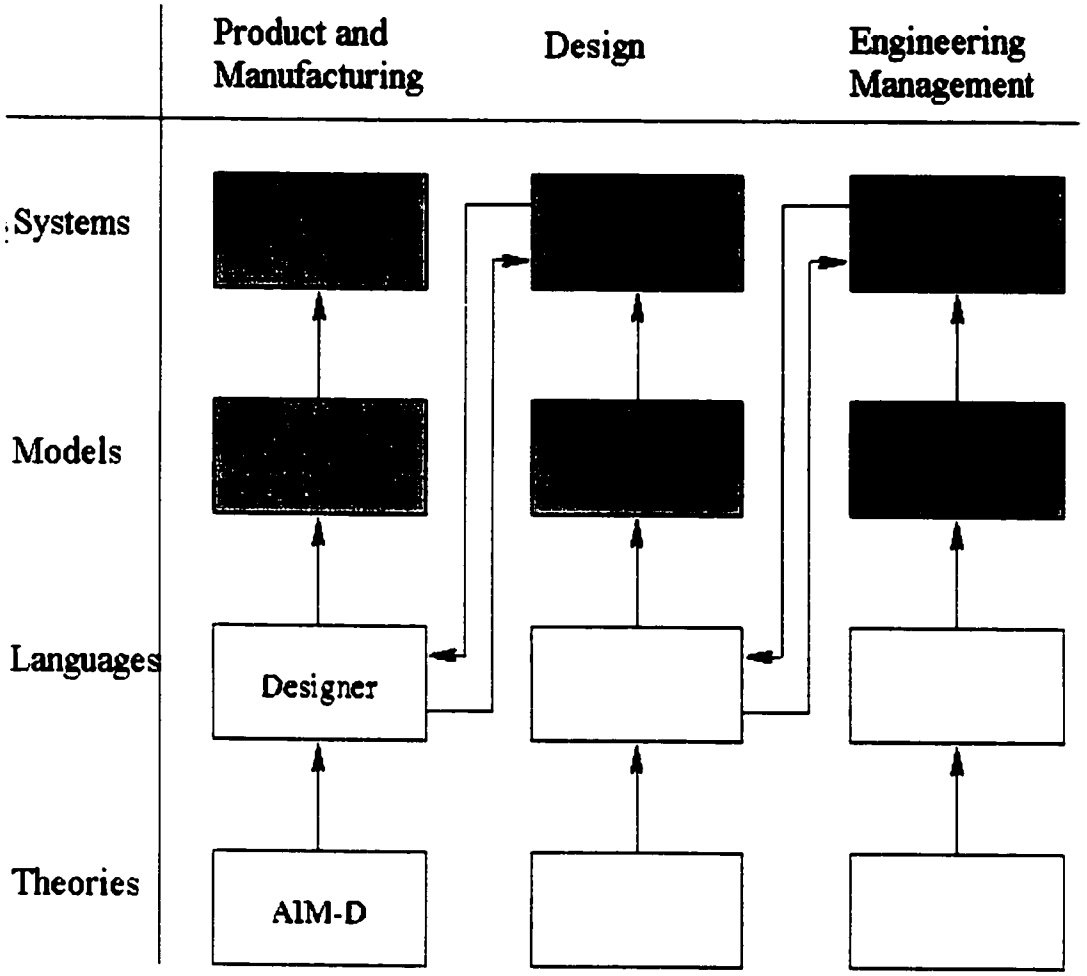
In this thesis, the current author attempts to unify the two approaches by providing a list of complete function primitives and allowing user to define functional units in different abstraction levels easily and appropriately.

## 2.5 Application: Function-based design theory

Salustri establishes a general framework to partition the design endeavor into manageable sections in [2]. This framework is called ACM (Artifact-Centered Modeling) and its overall structure is shown in Figure 2.7. AIM-D (Axiomatic Information Model for Design) is a fundamental part of ACM. It is a formal theory based on axiomatic set theory which concerns representing product information in the design process. The author applies AIM-D in the development of knowledge-based systems for design. This system, called *Designer*, provides the logical rigor of AIM-D within a computerized environment.

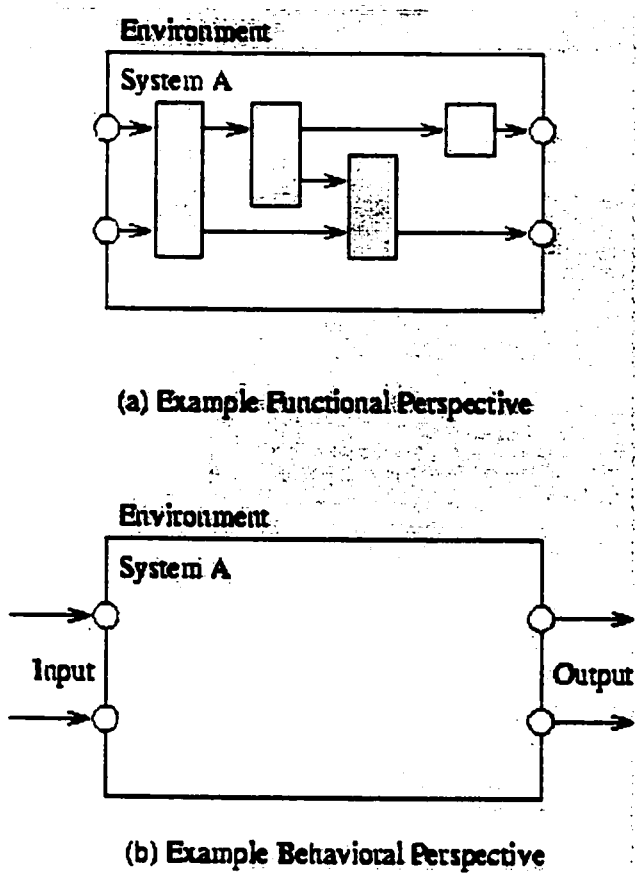
The author assumes three operational perspectives of the elements represented in the ACM matrix which are structural, behavioral, and functional perspectives. He states "Behavioral perspective treats a system as a black box capable of translating a set of inputs into a set of outputs, where the environment in which the black box operates is *transparent*, in the functional perspective, the system's environment is opaque, but its internal structure is *transparent*." In this perspective, the specifics of the transformation from inputs to outputs are described. Behavioral and functional perspectives are highly related to function modeling. Figure 2.8 shows these two perspectives.

Figure 2.7 Artifact-Centered Modeling framework [2]




*The horizontal axis is for function abstraction. Another axis distinguishes four levels of representational abstraction. The relationships between the elements of the ACM matrix are indicated by arrows. For example, the choice of language (manual drawing and 3D modeling) affects the design process [2]. Manual drawing may be faster in initial design stage, but 3D model makes further modification work much easier.*

Figure 2.8 Functional and Behavioral Perspectives in ACM [2]




*In the functional perspective, the system's environment is opaque, but its internal structure is transparent. The behavior perspective treats system as a black box capable of translating a set of inputs into a set of outputs, where the environment in which the black box operates is transparent.*

Qian and Gero [17] present a function-behavior-structure (FBS) framework for analogy design. They state, "*functions specify what a design does and behaviors describe how a design achieves its functions.*" Based on these definitions, analogous design is described as "Two designs are analogous if they have a similar function or similar behavior; they may or may not have similar structures." For example, computer keyboard and computer mice are information-input devices. They have similar function but their physical structures are significantly different.

They believe new ideas of possible structures and associated operations can be obtained from analyzing existing designs with similar function or behavior. Functions and behaviors are treated as retrieval indexes when we search design databases. Since more than 80% of the design work is a routine design process, which means that designers are required to modify the exist product designs to satisfy new customer needs, an analogy-based design model is useful in real world design practice.

Due to its influential roles in determining the product's fundamental features and development costs, the relative significance of conceptual design to basic design or detail design is widely recognized. Although there are some general methodologies dealing with functions in design, virtually no commercial CAD systems can support functional design, in particular the so-called synthetic phase of design [18]. Supporting the synthetic phase of conceptual design is one of the crucial issues of CAD systems with function modeling capabilities. Umeda et al. [18] propose a computer tool called a Function-Behavior-State (FBS) Modeler to support functional design not only in the analytical phase but also in the synthetic phase. To do so, the functional decomposition knowledge and physical features in the knowledge base of the modeler, and a subsystem Qualitative

Process Abduction Systems (QPAS) play crucial roles. The advantages of the FBS Modeler are demonstrated by presenting the examples of how designers use this tool and how the designers design functionally redundant machines.

Each of these models mentioned above attempts to build a practical design theory that could facilitate real world design endeavor. All of them have realized that in order to obtain boarder adaptability, this kind of design theory should reside on functional aspects rather than structural ones.

## 2.6 Context

Akman and Surav indicate that context plays a central role in function modeling, in that (a) it provides terminological information about the definitions of words appearing in the statements, and (b) implies a great deal of information about the operational environment [19].

The role of context is to eliminate certain ambiguities or multiple meanings in a statement. A formal notion of context might be useful in information retrieval because it can increase performance by providing a framework for well-defined queries and intelligent text matching. McCarthy first introduced the notion of context to artificial intelligence in 1986 [20]. Guha finds an essential use for formal contexts in implementing his so-called micro-theories [21]. Buvac and Mason approach context from a mathematical viewpoint [22].

At present, most of the context studies are conducted in the artificial intelligence (AI) area. Although there is some overlap between function modeling and AI research,

there are no specific studies on applying context logic to function modeling. Parabhakar and Goel have emphasized the importance of external environment to the adaptive design of devices in [25], but they do not use context logic in their representations.

The new function model discussed in this dissertation will use the concept of context to implement the mappings among terms, objects and types within function specifications.

## 2.7 Information theory

Information is one of the basic elements in IFM. By reviewing some of the common concepts on information, the current author would like to demonstrate that interactions between information and other two elements, mass and energy, are possible and reasonable.

What is information? Although it is widely used today, information is itself at the center of a theoretical discussion [47]. Information theory is the theory of generating, encoding, transmitting, and receiving information. It has its origin in the great developments of electronic engineering, which has provided essential means of both analog and digital [48]. Table 2.1 lists some of the definitions given by different researchers [44].

Claude Shannon's classic book with Warren Weaver, *The Mathematical Theory of Communication* [46], is a basic source on information theory. His theory was directed toward the problem of determining the maximum capacity of a channel (such as a telephone or telegraph circuit) for transmitting messages. He was particularly interested

Table 2.1 Definitions of Information

| Author | Definition |
|---|---|
| Doede | That which reduces the uncertainties relevant to a purposeful state. That which is common to all representations that are synonymous to the interpreter. |
| Szaniawski | Information concerns the set of possible states in a decision problem... Its value is identified with the highest price (in utility) to be paid for the information... The value of information is thus relative to the method of solving the decision problem. |
| Garner | Information is something we get when a person or machine tells us something we did not know before |
| Shannon and Weaver | Information is a measure of one's freedom of choice when one selects a message. |
| Wiener | Information is a name for the content of what is exchanged with the outer world as we adjust to it, and make our adjustment felt upon it... To live effectively is to live with adequate information.<br>The quantity of Information corresponding to the selection of an event (a message or decision) out of a repertory of 2 equal possible events is called 1 bit |

*There are various definitions of Information among researchers. This table lists some of the popular ideas. Source [44] pp. 277-281*

in the relationship among three factors: the way messages are encoded, the presence of noise (any condition that randomly alters part of the signal), and the capacity of a channel [47]. Nowadays, the fundamental principles of information theory have been extended to system engineering, computer science, human communication, social science etc.

Although the definitions of information are different among researchers, the current author would like to present some common concepts and backgrounds of information.

### 2.7.1 Properties of Information

There are three major properties of information [43]: (a) The contents of the information should be unknown to the receiver before reception; in another words, there must be more than one possible sequence of symbols before reception. (b) In order to receive "true" information, the receiving person can understand the meaning of the message. For example, a message in an unknown language without the key would not constitute information. (c) The information should relevant to the receiving person. That is, the message should have value to the receiver for a decision or for initiation of an action.

### 2.7.2 Levels of Information

In term of the above properties, there are three different aspects or levels of information [43]:

#### 2.7.2.1 Syntactic level

On this level, we are mainly interested in the number of possible symbols, their duration, statistical and deterministic constraints of successive symbols imposed by the

rules of adopted language or coding system. It is a study of information carrying capacities of communication channels and the design of appropriated coding systems for an efficient data transfer with high reliability. Syntactic information theory is of high applicability in data communication and data processing because it is independent of the meaning and significance of the transmitted data.

#### 2.7.2.2 Semantic level

Semantic information is very hard to be defined in exact mathematical terms because it is to a greater dependent on the receiving system (person) than the syntactic information. Context and association of words (symbols) play an important role in semantic problems. Information retrieval is another area of data processing involving semantics.

#### 2.7.2.3 Pragmatic level

On this level of information we are concerned with the value or utility of information. The pragmatic content of information depends strongly on time. The utility of information is a function of the time elapsed from the event to the reception of the report.

### 2.7.3 Information media

On the syntactic level of information, information is stored or transmitted in the medium that can assume more than one distinct state in space and /or time. There are two types of information systems. One is *discrete information system* – the medium where information is stored or transmitted can be divided into discrete units in space or time and

each of these units has a finite number of possible states. The other is *continuous information system* – if such units cannot be uniquely defined [43].

Doede presents the concept of *information vehicles* in [44]. He states, " There is no information without information vehicles. Information vehicles are the carriers of information, the physical material in which the information-for-the-interpreter is encoded. Every information vehicle is a distinguishable form for the interpreter of the information." The Doede's concept of information vehicle provides a theoretical foundation for the current author's function model in that information can be treated as one attributes of its physical material, which is called *node* by the current author, while mass and energy are the other two attributes. Detail discussion can be found in Section 3.3.

## 2.8 Engineering design theory

The current author aims to develop a function model that could facilitate the engineering designers in their product design process. By exploring some current engineering design theories and design procedures, the author could like to make sure that this proposed function model (IFM) has a wide suitability to exist design theories and could be integrated within their design procedures smoothly.

*Engineering Design and Design for Manufacturing* is a widely cited book written by Dixon and Poli [54]. The authors define the term *Design* as "the series of activities by which the information known and recorded about a designed object is added to, refined, modified, or made more or less certain". Based on this definition, they consider engineering design to consist of four stages, which are (1) engineering conceptual design,

(2) configuration design of parts, (3) parametric design, and (4) detail design. A methodology called *guided iteration* is used to solve the problems in each stage. It involves four basic steps: (a) formulating the problem, (b) generating alternative solutions, (c) evaluating alternatives, and if none is acceptable, (d) redesigning, guided by the results of the evaluations.

The first stage of engineering design – Engineering conceptual design has the inherent relations with function modeling. The authors introduce two approaches to do conceptual decomposition. The first one is direct decomposition which decomposes the product directly into its subsidiary sub-assemblies. The second one is function-first decomposition. This approach decides what has to be done purely in terms of functions and how to decompose these functions until simple functions are obtained. Then this approach looks for specific hardware to employ to achieve these functions. Dixon and Poli emphasize, "Function-first decomposition is more promising to generate new ideas and should be applied before direct decomposition. But this approach is highly abstract and not a familiar way to U.S. designers." The current author is constructing a brief pamphlet which aims to facilitate the engineering designers doing this highly abstract function modeling work.

Pahl and Beitz [55] argue that the inputs and outputs to all functional devices or systems, for example, motor, pump, gear, pencil and switch, are one or more of three general types: (a) material flow, (b) energy flow, and (c) information flow. They believe that the behaviors or actions of function devices can also be generalized into a relatively small number of types. Table 2.2 represents the five basic functions to all function devices along with the inputs and outputs.

Table 2.2 Inputs, Generalized Functions, and outputs of Functional Devices

| Input | Functional device | Output |
|---|---|---|
| Material flow ----→<br>And/or<br>Energy flow ----→<br>And/or<br>Information flow ----→ | • Change or Transform<br>• Increase or decrease magnitude<br>• Connect or disconnect<br>• Conduct, channel or block, or move location<br>• Store or take form storage | ----→ Material flow<br>And/or<br>----→ Energy flow<br>And/or<br>----→ Information flow |

*For a functional device, there are five types of functions and three types of inputs and outputs. The combinations of these function types, inputs and outputs result in various functional devices. [55] For example, keyboard transforms mechanical movement (material flow) to information displayed in screen (information flow).*

Pahl and Beitz presented their idea that material, energy, and information are the only three general inputs and outputs for any functional devices in 1984. Until now, there is not a fourth type introduced by any researchers. The current author believes that in the foreseeable future, it is unlikely that a fourth type could be introduced, so mass, energy, and information are regarded as three basic elements in current author's new function model.

The current author also agrees with Pahl and Beitz that there exists a small set of basic functions among various functional devices, but the completeness of their classification methodology along with their five generalized functions are arguable. For example, the function of a wristwatch can be expressed as *indicate time*, it is hard to categorize the verb *indicate* to any of the five generalized functions. The current author tries to solve this complete classification problem in his new function model.

Dym [52] states that "the key element of design is representation" and the *engineering design* is "the systematic, intelligent generation and evaluation of specifications for artifacts whose form and function achieve stated objectives and satisfy specified constraints." Having offered his own definition of *engineering design*, the author goes on study the design as an activity, that is, the process of design. He dissects the design process in seven steps which are: (1) clarifying the client's requirements, (2) identifying the environment, (3) analyzing or modeling, (4) identifying constraints, (5) testing and evaluating, (6) refining and optimizing, and (7) documenting design.

Dym also emphasizes the importance of *taxonomies of engineering design*. According to the dictionary, *taxonomy* is the result of "studying the general principles of scientific classification". Similarly here, a taxonomy of design might (1) allow us to

classify design problems according to certain characteristics, (2) facilitate the organization of the knowledge, representation and reasoning schemes and (3) increase our ability to understand and model the thought process.

Based on a design taxonomy that asking whether a design is routine or creative, we can get three classes of design: (a) Creative design: It usually leads to completely new products and typically it lacks of both domain knowledge and problem-solving strategy knowledge. (b) Variant design: We understand the sources of our design knowledge but lack a complete understanding of how that knowledge should be applied. (c) Routine design: We can identify the specific design or domain knowledge and we know how to apply that knowledge.

Since design processes are highly related to the current author's proposed function model, some of the most widely cited design processes are reviewed here. Figure 2.9 is given by French in [56] and Figure 2.10 is given by Pahl and Beitz in [55]. By reviewing these books, the current author got an overall picture of the engineering design domain and tries to construct a new function model that could be implemented in most of the engineering design processes discussed in these books. In chapter 3, Hyman's method [38] was selected randomly to demonstrate the how to implement the current author's function model (IFM) to his engineering design process, at the same time, this random selection also demonstrate the wide suitability of IFM.

Figure 2.9 A pictorial view of the design process [56]

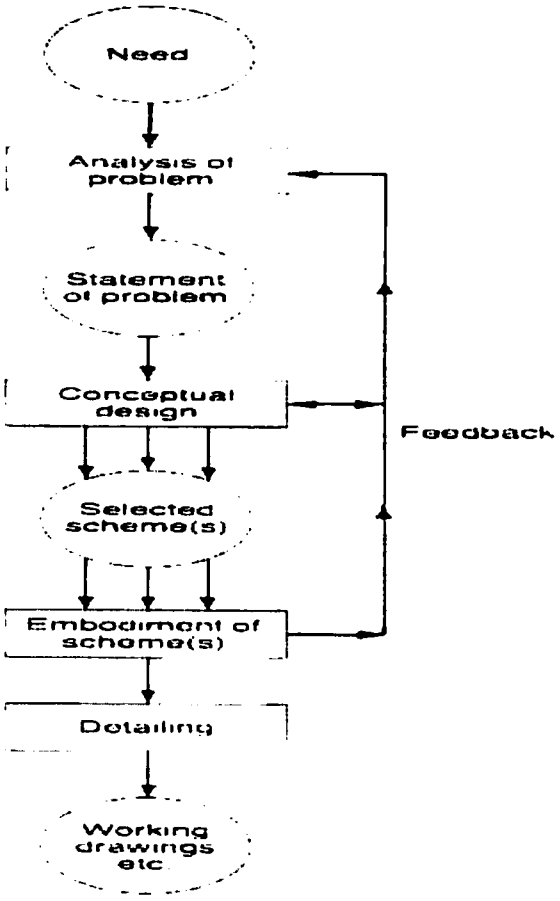
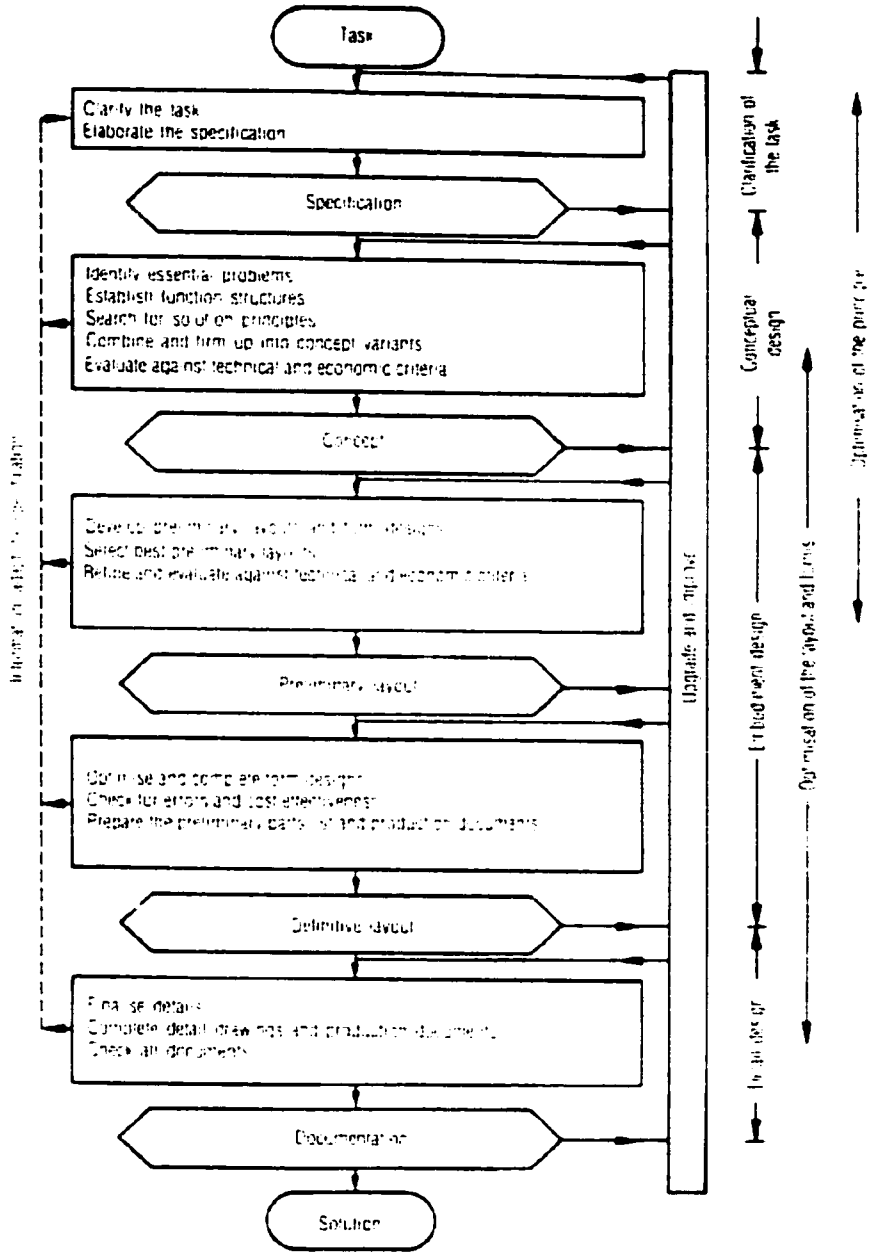


Figure 2.10 A prescriptive model of the design process [55]



# CHAPTER THREE

## A NEW MODEL OF FUNCTION (IFM)

From the literature review section, we can see that many other researchers have identified the importance and relevance of function modeling to the engineering design, but a well-defined and widely accepted concept of function does not yet exist. In this dissertation, the author proposes a new function model based on the interactions of mass, energy and information. This formulation leads to a small, arguably complete taxonomy of primitive functions, from which arbitrarily complex function descriptions can be developed.

This chapter is organized in nine sections. Section 3.1 discusses various function definitions in dictionaries and research papers, and then introduces IFM. Section 3.2 shows how a set of small and arguably complete function types can be derived from IFM. In Section 3.3, the concept of *node* and *node relation chart* are introduced. Function Identification form is used to represent function. It is discussed in Section 3.4. To manage various functions in a certain project (object), the author introduces Abstract Function Family Tree in Section 3.5. Section 3.6 outlines the framework of how to use IFM to facilitate engineering design in its early stage. In Section 3.7, some potential reasoning tasks that IFM could support are explored. Function modeling relates to other concepts such as *structure*, *context*, *mereology*, *purpose/intention* etc. They are discussed in Section 3.8. In Section 3.9 the author explains other researchers' function definitions via IFM.

### 3.1 New function definition

There are various, often conflicting, definitions of *function* in the literature; no widely accepted definition is currently known. A brief review of some of these definitions is relevant to the later sections of this dissertation. Section 3.1.1 lists the function definitions in the dictionaries while Section 3.1.2 lists the function definitions from research literatures. The author presents a new function definition in Section 3.1.3.

#### 3.1.1 Definitions from the dictionary

Dictionaries can provide naïve, but useful, definitions. With respect to products, function is defined as (1) "a natural or usual purpose (of a thing)" and (2) "the way in which something works" [26]. A purpose implies some intention or goal on the part of a designer or user; some researchers (e.g. [25]) have used this aspect in their work. The second definition suggests an explanation of a product's operation; though rare in the function modeling literature, this too has been used (e.g. [3]). Though insufficient for our research purposes, even a dictionary definition can provide some clues of what is most commonly held to be the meaning of function.

#### 3.1.2 Definitions from the technical literature

Table 3.1 lists some of the function definitions the current author collected from the literature reviewed. Most of them can be grouped coarsely into three categories.

The first category typified by [7, 25] has already been mentioned: function deals with the purpose of a product as intended by the designer. The problem here is that such definitions are cognitive rather than physical. While work in the cognitive aspects of

design is very important, the current author is more interested in establishing a basis for function rooted in reality rather than in our perceptions of it.

The second category of work treats product function as an effect on the environment of the product (e.g. [9, 10]). And the third category, exemplified by [17], defines *function* as relationships between inputs and outputs of energy, mass, and information, or as changes in the fluxes thereof. This category is most closely related to the definition used by the current author.

Clearly, each of these definitions has some aspects of worth, yet none are comprehensive enough to capture the fullness of definition that is desired.

### 3.1.3 A new Interaction-based Function Model (IFM)

The author's research is working towards a more comprehensive and formal concept of function and its impacts on engineering design domain. The current working definition is as follows:

> *A function is any mechanism by which transformations from one or more basic element - mass (M), energy (E), or information (I) - to another(s) occur in an entity, in response to environmental stimuli.*

The function of a product describes all possible interactions between a product and its operational environment. Aspects of other definitions are included here and a detail discussion is given in Section 3.9. The notion of identifying information, mass, and energy as basic elements has been very popular since it was first proposed [27], and has withstood great scrutiny by many researchers. Its stability in this regard makes it very attractive to the author, especially as the foundational layer of this emerging structure.

Table 3.1 Function definitions given by researchers in function-based modeling domain

| Author | The Definition of *Function* |
|---|---|
| Salustri [3] | How the system achieves its behavior |
| Kitamura and Mizoguchi [4] | A result of interpretation of the behavior under the intended goal |
| Chittaro, Tasso and Toppano [5] | The relation between its behavior and the goals (teleology) assigned to it by the designer |
| Hodges [6] | Function is represented with a schema that describes the initiation and terminating states associated with a process sequence |
| Keuneke [7] | The function of a device is its intended purpose. |
| Sasajima et al [8] | The interpretation of the behavior under a desirable state which the component is expected to achieve |
| Chandrasekaran and Josephson [9] | Let G be a formula defined over properties of interest in an environment E. Let us consider the environment plus an object O. If O (by virtue of certain of its properties) causes G to be true in E, we say that O performs, has, or achieves the function (or role) G |
| Qian and Gero [17] | Function specify what a design does |

*Various function definitions given by different researchers*

### 3.2 Function types derived from IFM

To represent the functions based on the new definition, the author constructs a general form,

$$\boldsymbol{BE1 > BE2} \tag{3.1}$$

Where BE1 represents all the Basic Element(s) involved before a certain transformation and BE2 represents all the Basic Element(s) involved after that transformation. The symbol ">" between BE1 and BE2 represents the transformation process. For example, $E > E$ means an energy type element transforms to another energy type element and $EM > M$ means an energy type element and a mass type element transforms into a mass type element.

Furthermore, functions can be clustered into different categories according to the number of elements involved in BE1 and BE2. These two numbers are called *slots* by the author. For example, functions with single elements in both BE1 and BE2 are in the same function category with E > E; functions with two slots in BE1 and one slot in BE2 are in the same category with EM > M. In general, suppose there are X slots in BE1 and Y slots in BE2, all the functions in this category can be characterized as an X-to-Y mapping between BE1 and BE2.

Table 3.2 gives the total function types within a certain X-to-Y mapping category, detail mathematical derivation is given in Appendix A. The author would like to discuss the simplest 1-to-1 mapping first, then goes to more complex mappings step by step. A general comment about X-to-Y mapping is given at the end of this section.

Table 3.2 Total number of function types in a certain category

| Slots in BE2 / Slots in BE1 | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 9 | 18 | 30 |
| 2 | 18 | 36 | 60 |
| 3 | 30 | 60 | 100 |

*According to the Slots in BE1 and BE2, the total number of function types in a certain category is determined. For example, if there are two slots in BE1 and one slot in BE2, then the number of function types in 2-to-1 mapping category is 18. Detail mathematical derivation is given in Appendix A.*

### 3.2.1 1-to-1 mapping

There are nine basic types of sub-functions in this 1-to-1 category. These types are enumerated in Table 3.3, where each row represents an initial state, each column represents a final state, and a cell describes a typical transforming function type. Each type is identified with a descriptor (in bold) and an example. Many different functions may fit into one cell. For example, the function in the M > M cell could represent the movement of a mass, but M > M functions also include those that change the amount of mass in an entity (e.g. to *fill* or to *empty* or, more generally, to *store*).

The organization of each function type in Table 3.3 is a matter of on-going research. The selection of descriptors is itself problematic; for example, *to store* could be a valid descriptor for an M > M type function, as described above, or for an I > I type function such as storing information in a program. A more fundamental open issue in this regard is whether the use of the same descriptor for functions of different type (such as *to store*) indicates a true similarity of the functions or rather a cognitive or even linguistic artifact.

How to label sub-functions in a certain category is another issue need to be addressed here. In this thesis, the author uses two methods to label functions. The first one is *Infinitive Verbs*, such as *to power, to regulate, to move* which are used in Table 3.3. The other is *Verb/Object Pair* (VOP) introduced by Salustri in [3], such as *transfer data, edit article* etc. The author would like to view infinitive verb as a general form of a set of VOPs that share a common verb. For example, *transfer data, transfer heat* and *transfer power* can be grouped together by *To transfer*.

Table 3.3: Sub product function types in 1-to-1 mapping category

| | | FINAL STATE / OUPUTS | | |
|---|---|---|---|---|
| | | Mass | Energy | Information |
| INITIAL SATE / INPUTS | Mass | **To move**: motion of one gear causes another gear to move. | **To power**: burning fuel gives off energy. | **To activate** : closing a switch sends a signal. |
| | Energy | **To energize**: the volume of a heated fluid increases. | **To convert**: a wire carrying a current radiates heat. | **To detect**: a photocell responds to light with a signal. |
| | Information | **To actuate**: controller signals a robot to move. | **To regulate**: Amplifier output is controlled by received signal. | **To transfer**: digital to analog conversion. |

*Nine basic product function types in 1-to-1 mapping category. Each row represents an initial state, each column represents a final state, and a cell describes a typical transforming function type. For example, the I > I type function, To transfer, its initial state is a kind of digital information, after the conversion, its final state is a kind of analog information.*

### 3.2.2 2-to-1 or 1-to-2 mapping

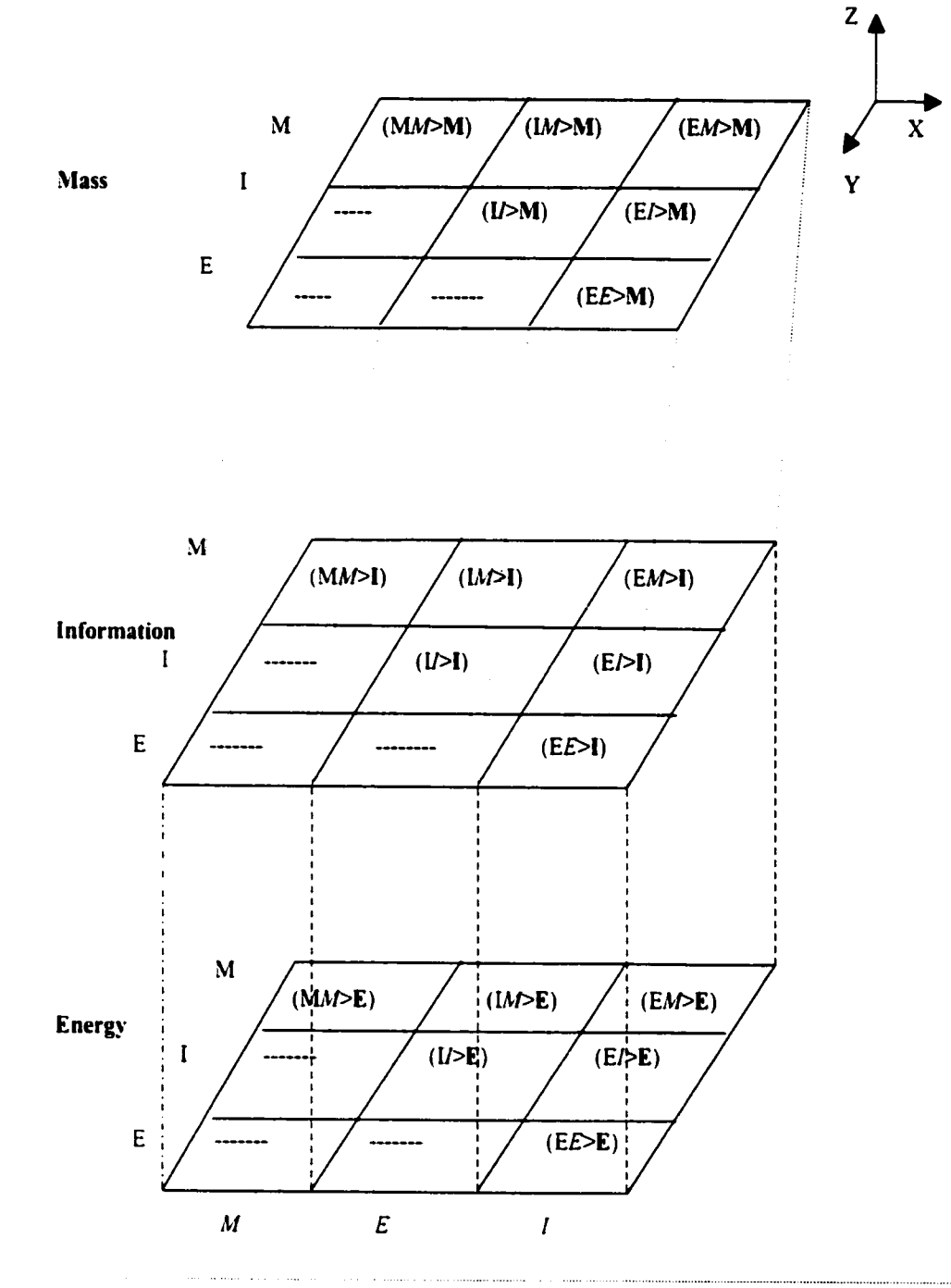
There are 18 sub-function types in each of these 2-to-1 and 1-to-2 categories. Figure 3.1 and Figure 3.2 demonstrate the detail of these somewhat complex mappings. For example, *melting ore* is an example of EM > M cell in Figure 3.1. Originally the ore (mass) is in a solid form. When a huge amount of heat (energy) is applied to the ore and it melts to liquid form (mass) which can be processed. *Digital communications* (e.g. TV program or cell phone call) can be viewed as an IE > I type function. A program is an information flow. To let the customers receive this information, we need to provide huge amount of electric power. *Lighting bulb* is a typical E > EE type function. The electricity power is converted into light and heat which are two other forms of energy.

There are more sub-functions in 1-to-2 and 2-to-1 mapping categories. As stated in Section 3.2.1, we can use VOPs to label functions in a more specific manner. Same element combination and labeling methods could be applied to more complex X-to-Y mapping when it sub-functions are expanding.

### 3.2.3 General comments to X-to-Y mapping

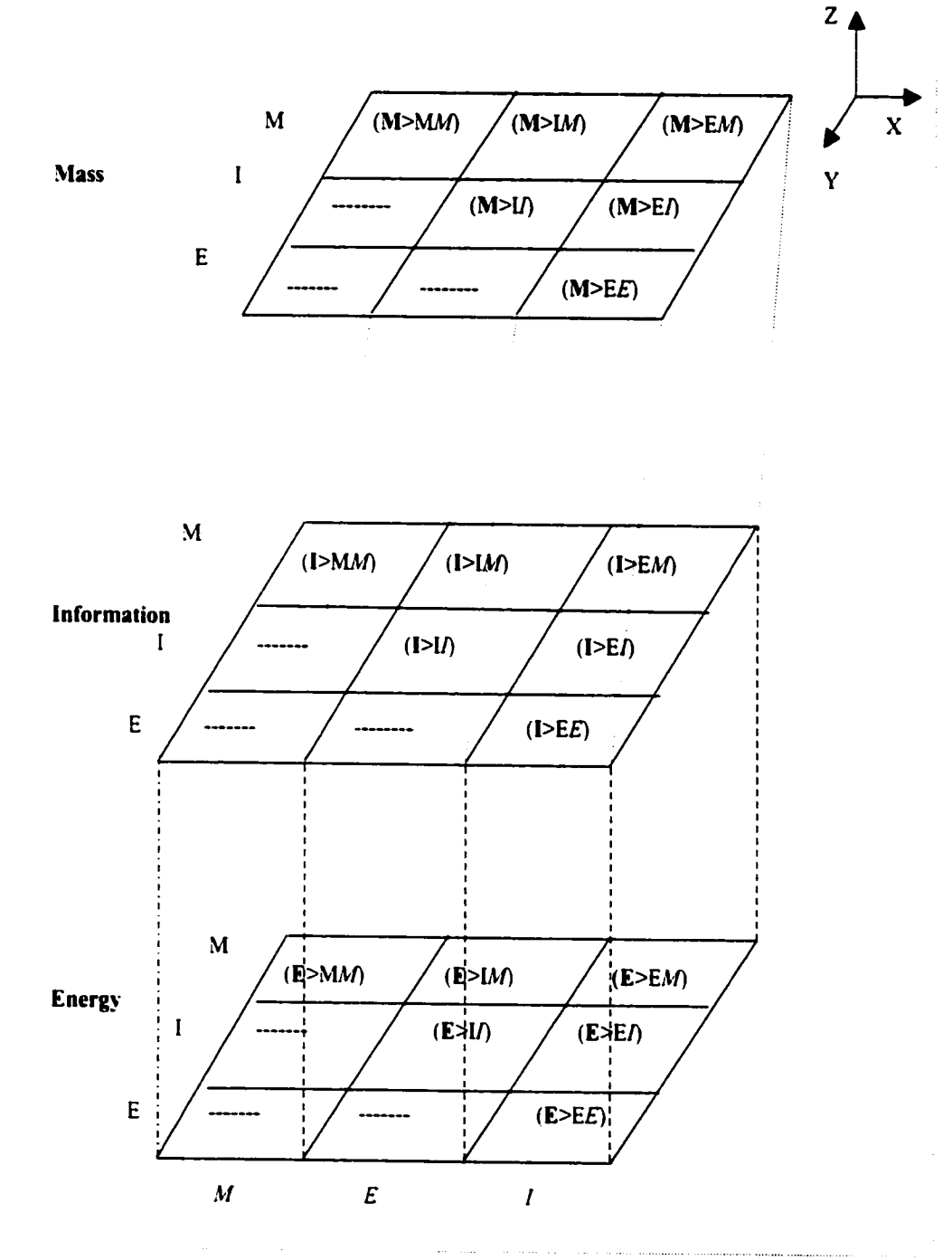
Section 3.2.1 and Section 3.2.2 demonstrate the author's general function classification method in a certain category. From Table 3.1 we can find out that the function types in an X-to-Y mapping category increases rapidly with more basic elements involved either in BE1 or BE2. Following comments about this function taxonomy are made by the author.

Figure 3.1 2-to-1 mapping



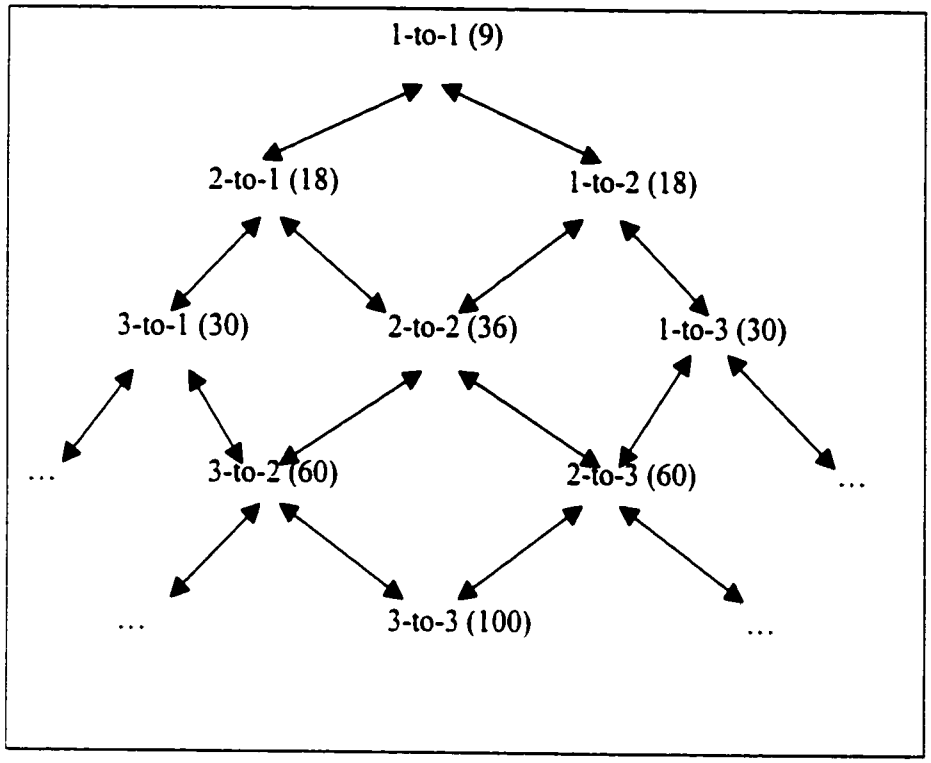

*X and Y axis represent the basic elements in BE1 and Z-axis represents elements in BE2.*

Figure 3.2 1-to-2 mapping




*X and Y axis represent the basic elements in BE2 and Z-axis represents elements in BE1.*

Figure 3.3 Relationships between X-to-Y mapping




*1-to-1 mapping is the root of this hierarchy. Others can be viewed as its descendants which more basic elements are added either in BE1 or BE2. For example, the simplest form of engine function can be viewed as an M > E (fuel > heat). Suppose we need to monitor the engine speed, a sensor is introduced to provide this information. So the original M > E engine function expanded to M > EI function which falls in 1-to-2 mapping category.*

#### 3.2.3.1 1-to-1 mapping is the base of the function classification

1-to-1 is the simplest mapping with only 9 sub-function types. By adding more basic elements in BE1 or BE2, more complex mappings are created; by eliminating some basic elements, simpler mappings are generated. Figure 3.3 represents the relationship among X-to-Y mappings.

According to this relation, it is recommended that in functional decomposition stage designers should consider as simple mappings as possible. There are two possible ways to achieve this goal

First, omitting minor factors. The 80/20 law [41] indicates that a few elements account for the major performance of the system. If we can omit minor factors in a certain situation, the problem can be greatly simplified. For example, burning fuel gives off energy so it is an M > E type function in 1-to-1 category. But we all know that without ignition energy, this chemical process cannot occur which means if we do not omit this minor factor, we need to add an energy type element in BE1 and this function should be modified into EM > E type.

Second, adding layers to reduce complexity: A multi-element mapping can be decomposed into several layers. In each layer, functions are much simpler. For example, *reduce cost in engine production system* is a very complex function where hundreds of elements may be involved. Suppose we decompose this top-function into a hierarchy and one of its lower level functions is "prevent fuel leaking". This lower level function involves far fewer elements and an M > M type function implementation may work. (e.g. add an O-ring to seal the fuel pump and the fuel supply pipe)

### 3.2.3.2 X-to-Y mapping taxonomy makes all functions independent of each other

Many researchers (Dixon and Poli [54], Suh [36]) emphasize that it is highly expected that product/object functions could be mutually independent. For the new interaction-based function definition, suppose C1 and C2 represent the content of elements in functions F1 and F2 respectively, where C1 = {c11, c12, ... c1n} and C2 = {c21, c22, ... c2n}. Two functions in a same level are independent if and only if

$$\{(c1, c2) \mid c1 \neq c2, \forall c1 \in C1, \forall c2 \in C2\} \quad (3.2)$$

This formula means each element of function 1 is different from that of function 2.

For example, fuel injection systems control the ratio of air to fuel to form the proper mixture. We can interpret this process as an MM > M type of function. Furthermore, we can decompose this function into two sub-functions which are M (fuel) > M (mixture) and M (air) > M (mixture). Although these two sub-functions share the same form of M > M, but the contents of the two input elements are different, so these two sub-functions are independent which means any modification in fuel pump systems or air inhale systems will not affect the other in a function point of view.

According to the above definition, the current author believes that this X-to-Y mapping taxonomy can meet this independent criterion in that (a) three basic elements are independent of each other, (b) the function decomposition process derived from the IFM (Detail discuss can be found in Section 3.4.2) ensure that the sub-functions of a certain high level function can not share the same element. The current author's function model can be applied to the following area.

- Although many researchers suggest that function-first decomposition in engineering conceptual design stage could motivate creativity, but none until now discusses how to formalize or direct this function decomposition process. This means that most function-first decompositions are done in an experienced-based manner. By applying the current author's taxonomy, it is suggested that designers could decompose product functions into a 1-to1 mapping first, then add one element at a time, if it can not be achieved, and try 2-to-1, 3-to-1, and 2-to-2, ... step by step.
- This taxonomy can be used as a "key word" to search design databases. For example, CD-ROM driver is a device that implements a top-level function M > I. (the information stored in CD is interpreted). Suppose this CD-ROM design is stored in a certain category of design database labeled by M > I. When DVD driver is going to be designed, we can use its top-level function, M > I, to search the database and locate the M > I category. Designers can get some valuable references and inspirations by reviewing previous designs that implemented the same top-level function. In this example, even if the information process mechanisms of DVD and CD-ROM are totally different, the shape and frame dimensions of the CD-ROM driver still have high reference value in that both drivers are confined in the same 3D space according to the shape of standard computer case tower.

## 3.3 Node, Node Relation Chart (NRC) and Function Decomposition

### 3.3.1 Brief introduction of *Frame*

Salustri [40] discusses the advantages of using *frames* to represent knowledge in KBS over traditional object-oriented method. Most of the material mentioned below is taken from [40].

There are a variety of ways to develop knowledge bases; *frame* is one of them. Since it is similar to *object,* it is helpful to review some facts about object systems first.

- An object (as in C++) is a container for variables, the values of which all describe aspects of a single thing. This notion of objects having variables that have values is often referred to as object/attribute/value triplets.
- Objects are grouped into classes that are descriptions of whole sets of objects.
- Object classes can "inherit" the behavior of other object classes, and pass that behavior on to all its instances.

Frames, on the other hand, have an extra level of detail, called *facets* that are components of attributes (called slots). While object systems have object/attribute/value triplets, frames have frame/slot/facet/value quadruplets. In objects, attributes are variables that describe things about the object. In frames, slots serve this purpose; then facets can be thought of as variables whose values describe the slot.

A very interesting notion that arises from having facets is that users do not need object classes (necessarily). Instead, they can use *prototypes* or *exemplars.* An exemplar is a typical object that can stand for any of a large set of similar objects.

Inheritance is also possible with frames, just as with objects, except that the lack of classes in frame systems requires the use of a different instantiation mechanism. One popular technique is cloning. In cloning, a duplicate of an exemplar is made, which can then be altered.

Another advantage of frames is that procedures can be added to slot facets. Some of these procedures are known to the frame-based system, and are automatically triggered by the system when certain event occurs.

### 3.3.2 Node

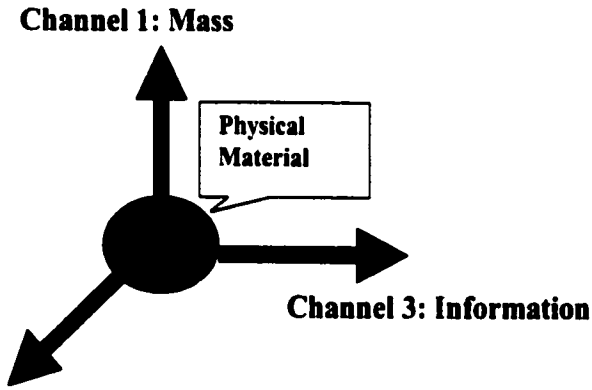
Having briefly introduced the concepts about frames, the current author would like to use frames to represent the basic design element in the form of *nodes*. Mr. Li Hongzhi [39] points out that "every substance in the universe has not only its material existence but also the existence of its characteristic as well." The structure of a node is given in Figure 3.4. The oval in the middle represents the physical material (substance) of the node while three arrows represent three basic characteristics (communication *channels* to the outside) of any substance, which are mass, energy and information. Two or more substances interact with each other(s) via their three basic channels – mass, energy and information. In a particular interaction, if a substance uses it mass, energy or information channel to communicate, we call this substance mass, energy or information element respectively.

Node names are used to indicate the substance which is involved in a particular function and uses three slots – mass, energy, and information to indicate the basic interaction channels of this substance. There are several points that the author would like to mention about the *node*:

1. For the innovative design, we know neither the name nor the facet(s) of a particular node, but only some expected properties (values) of the node. The hardest task of innovative design is to find out the proper physical material or facet(s) of the node to satisfy these expected properties. For example, *High Energy Density, Fast Recharge*

*Capability, Long Cycle Life* and *High Power Delivery* are four expectations for batteries installed in electric vehicles. GM Ovonic found a special material – *NiMH*, to meet above requirements [66].

Figure 3.4 Structure of *Node*




*A node has three channels to communicate outside which are information, mass and energy .In different context, one of these three attributes appears dominant. For example,* ***light*** *is a node. When it is used to direct traffic flows, its information channel is dominant. When we enjoy the comfortable warm feeling under the sun on beach, it is obvious that the energy channel of the light is working. When physicists are trying to study the speed of the light in different mediums, they concern more about its mass channel. Everything in the universe could be treated as a node, and the nodes are interacting with each other via its three channels*

Figure 3.5 Facet explorations for *transfer heat* function

| | |
|---|---|
| **Node Name:** | **Device D** |
| **Mass Slot:** | |
| **Facet 1:** | available space for D (L200 × W70 × H50) |
| **Facet 2:** | weight (< 150 g) |
| **Energy Slot:** | |
| **Facet 1:** | temperature on input and output sides of D (not available now) |
| **Facet 2:** | power required to drive D (< 30 w) |
| **Information Slot:** | |
| **Facet 1:** | control signals (when to start, when to shut down) |
| **Facet 2:** | feedback parameters (not available now) |

*A proposed function facet exploration process. The phsical is unknown even the process is over but the facets of each slots are somewhat clear. (Source:*http://www.kingwin.com/fan-p3.htm*)*

2. For routine design and redesign work, designers knew the physical material of the node, but they need to modify the values of one or some slots to meet the new requirements. For example, Kingwin Inc made CPU cooler (node) for Pentium III serials with the heat sink dimensions of L137 x W48 x H32, but for the AMD K7 serials, heat sink dimensions are adjusted to L124 x W56 x H32.
3. (a) Verb/Object Pairs (VOPs) are used to name functions and these functions can be classified into two groups. *Inject fuel* is in the first group that the object part of VOP describes the physical material of the node explicitly but does not indicate related slot(s). This/these slot(s) and its/their facet(s) are the major concerns of designers instead of that physical material itself. For the *inject fuel* function, designers are more interested in the fuel's volume, chemical components (facets of mass slot), temperature, energy density (facets of energy slot) etc. If an alternative, say natural gas, could satisfy all of the facets mentioned above along with a cost competitive, designers would not hesitate to substitute *inject natural gas* for *inject fuel*.

   (b) *Transfer heat* is in the second group that the object part of VOP does not describe the physical material of the node explicitly but indicate related slot(s) or facet(s) of slot(s). Although seeking a certain physical material that could meet these requirements is a major task of the kind function implementation, the author suggests postponing this effort as late as possible in that the later the physical material is determined, the broader the space for the designers to work out creative design. On

the other hand, designers are suggested to explore the facets of the slots instead. Let us think about CPU cooler again. Before it is designed, what designers know at beginning is that *a device* (D) is needed whose function is *transfer heat*. Instead of thinking about the structure of D, designers are suggested to explore the slots of D first. Figure 3.5 gives a proposed facet exploration for *transfer heat* function. With the exploration of slot facets, a new node D emerges.

4. The author believes that *Conceptualizing alternative approaches* can be achieve in two ways. First, modifying the values of exist nodes (as stated in point 3(a)); second, seeking new nodes (as stated in point 3(b)). At this stage, it is also possible to fill three items in FIF assigned to this step. *Input and output descriptors* are the related slots of input and output node(s) respectively. According to the knowledge representation mechanisms of frame, they can be in the form of *fuel.mass.chemical_components*, *fuel.energy.temperature*, *CPU_cooler.mass.weight* etc. *Function types* can be got by putting all related slots of input nodes (input slots) in BE1 and all slots of output nodes (output slots) in BE2. For example, A.mass, A.energy and B.information are input slots and C.mass and D.energy are output slots, then this is an MMI > ME type function.

5. There are seven basic physical quantities that are listed in Table 3.4. We can see that the *length and mass* describes mass characters of a substance, *time* and *amount of substance* can be viewed as information characters while the last three are obviously energy characters. More expanded physical quantities such as area, current density, speed, stress, power electric resistance etc are listed in [64]. The author believes that these physical quantities provide a rich facet set for slot exploration. By examining

these facets, designers could formalize functional design process which is commonly conducted in an ad hoc manner and treated as a highly experience dependent practice nowadays.

Table 3.4. International System of Units (SI) base units [64]

| Base quantity | SI Base Unit | |
|---|---|---|
| | Name | Symbol |
| Length | meter | m |
| Mass | kilogram | kg |
| Amount of substance | mole | mol |
| Time | second | s |
| Electric current | ampere | A |
| Thermodynamic temperature | kelvin | K |
| Luminous intensity | candela | cd |

*The author believes that these physical quantities provide a rich facet set for slot exploration. By examining these facets, designers could formalize functional design process which is commonly conducted in an ad hoc manner and treated as a highly experience dependent practice nowadays.*

### 3.3.3 Draw Node Relation Chart (NRC) and Decompose Function

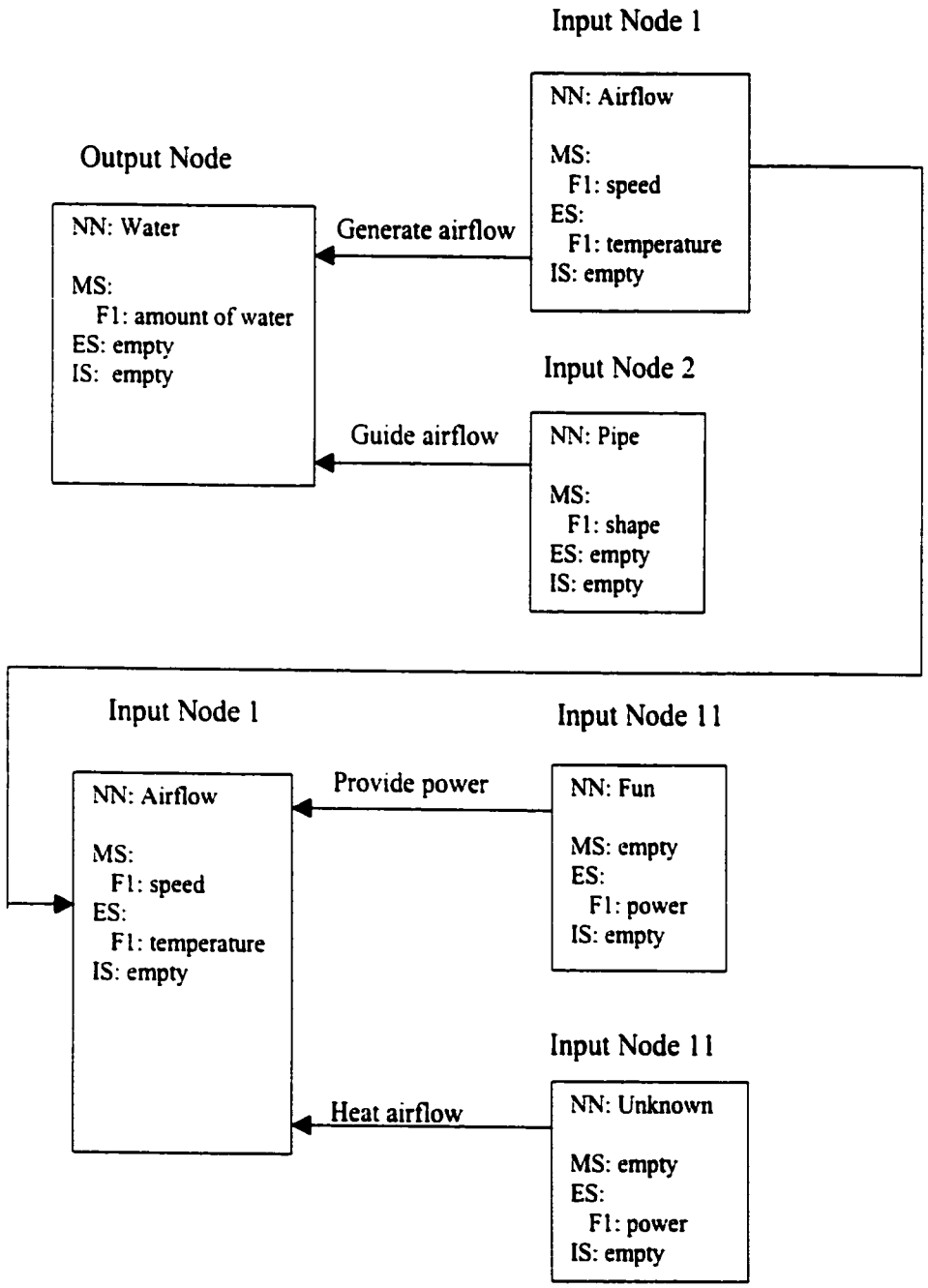
Figure 3.6 shows the node relation of a hand dryer in each level. The output node (ON) is derived from VOP, which is generated from needs analysis. First, we need to explore facets of the output node. This has been discussed in previous section. Then construct and deploy corresponding facets of input nodes (IN) in the causal order or space order according the facets of the ON. This is a highly context sensitive and experience related step where knowledge-based systems (e.g. AIM-D [1]) can be of great help.

Suppose N is a node, in node relation chart all nodes on N's right and point to N are the input node(s) of N. Let N (slot1, slot2 ...) represents the non-empty slot(s) of N. All of N's non-empty slots compose BE2 and all non-empty slots of N's input nodes compose BE1. For example, in Figure B.3, input node 1 (M, E) and input node 2 (M) are input nodes of output node (M). So the type of top-level function is MME > M. Similarly, the type of second level function (input node 11 and input node 12 are the input nodes of input node 1) is EE > ME.

Functional analysis commonly proceeds by decomposing coarse-grained functional requirements, usually provided at the outset of a design process, into more fine-grained functional specifications. According to IFM, functions occur among basic elements via interaction. In NRC the arrows between output nodes and input nodes represent the interactions, so they are the sub-functions of the output node. When corresponding VOPs are assigned to these arrows, sub-functions are generated. Continuously, the input nodes

in this level are treated as the output nodes in the next level and the function decomposition process goes on until a direct implementation is available.

Figure 3.6 Node Relation Chart



*NN – Node Name; MS – Mass Slot; ES – Energy Slot; IS – Information Slot. shows the node relation of a hand dryer in each level. The output node (ON) is derived from VOP, which is generated from needs analysis.*

### 3.4 Representing product function by Function Identification Form (FIF)

The author is working toward building a product function model that could be integrated to knowledge-based systems (KBS) for engineering design. To that end, one possible knowledge-based scheme for this function model is considered. A function specification (FS) has six attributes:

- *Function descriptor* – a specification of the type of function;
- *Input descriptor* – the name of an object or type of object, typical of the input used by the function or the initial state necessary for the function to occur;
- *Output descriptor* – similar to the input descriptor, but typical of the output or final state;
- *How link* – a reference to another FS describing the sub-functions needed to obtain the given one.
- *Why link* – the inverse of the *how* link.
- *Value* – the magnitude and direction of the function.

Each descriptor is represented by a term denoting a knowledge item. The terms' denotations are retrievable by an appropriate query engine. The FS itself would be named by a term. This scheme is easily implemented in many kinds of systems, including object-oriented systems like Java and description logic languages like CLASSIC [35].

The *how* and *why* links are used to connect functional specifications into a

function/behavior tree. This aspect is taken directly from [3].

The knowledge items denoted by terms would be arranged to form a specialization hierarchy. In order to remain consistent with our model, per Table 3.3, that taxonomy would have to contain "primitives" for mass, energy, and information, that represent the most general descriptions of things that provide functions.

The terms denoted by the function descriptors would be arranged in a specialization hierarchy also. The author suggests a two-layer naming mechanism. The first layer is highly abstract; in this layer designers could only determine what and how many elements are involved in BE1 and BE2. An *Infinitive Verb* pair is suggested to describe this group of candidate functions (e.g. To move, To activate, To transfer). The second layer is less abstract; the target object is known to the designer, so a verb/object pair can be used to describe the functions (e.g. input data, prevent leaking, reduce speed).

Function Identification Form is constructed to represent the concepts discussed above. An empty FIF is shown in Table 3.5 is an empty FIF, how to implement FIF in engineering design is discussed in Section 3.6 and Chapter 4. The representation shown here would couple the hierarchies of objects/things and actions/verbs, providing a linkage for reasoning engines to perform a variety of tasks, some of which are discussed in Section 3.7.

### 3.5 Manage various FIFs in a project by *Abstract Function Family Tree*

A single project may contain many functions in different levels. Tamai [57] introduced *Function Family Trees* to organize functions in 1967. A survey made by Tanaka [58] in 1984 showed that 66% of 123 Japanese companies use function family

trees as a function analysis technique. The current author would like to adapt the concept of function family trees and steps further to construct *Abstract Function Family Tree* (AFFT) for use in IFM. Figure 3.7 is a sample function family tree given by Akiyama in

Table 3.5: Function identification form (FIF)

| **Attributes** | **Content** |
|---|---|
| Function type | Empty |
| Function name | Empty |
| Input descriptor | Empty |
| Output descriptor | Empty |
| How link | Empty |
| Why link | Empty |
| Value | Empty |

*An empty FIF contains six attributes. By filling each items of FIF, engineering designers can organize their function design concepts in a clear manner. A detail example of how to use FIF is given in Chapter 4.*

[59]. A Japanese foodstuffs company uses it to improve its sliced ham manufacturing process. Figure 3.8 shows the structure of an AFFT introduced by the current author. It can be used as a general framework to analysis, represent and organize functions.

### 3.5.1 Graph Theoretic model for AFFT

*Basic definitions*

- All of the functions in the AFFT tree are represented with *nodes*, for example in Figure 3.8 A to L are all nodes;
- The top-level function is the *root* of the entire tree. A is the root of this AFFT;
- Functions with no sub-functions are called *leaves*, for example C, E, F, H, I, J, K, L are leaves;
- A higher-level function, such as G, is called the *parent node* of its sub-functions, such as J, K, L. On the other hand, J, K and L are called *children nodes* in terms of G.

*Graph Theoretic representation for AFFT*

- The general Graph Theoretic form for a node is shown in equation 3.3

$$F(X_1, X_2, \ldots, X_j),\ \forall j \in Z^+,\ \forall X_j \in N \tag{3.3}$$

Where j indicates the level that a function resides and $X_j = 1$ to $N_j$ which distinguishes the nodes in $j^{th}$ level. When $j = 0$, we get the form of top-level function, F (). Table 3.6 lists the graph theoretic forms for all nodes in Figure 3.8.

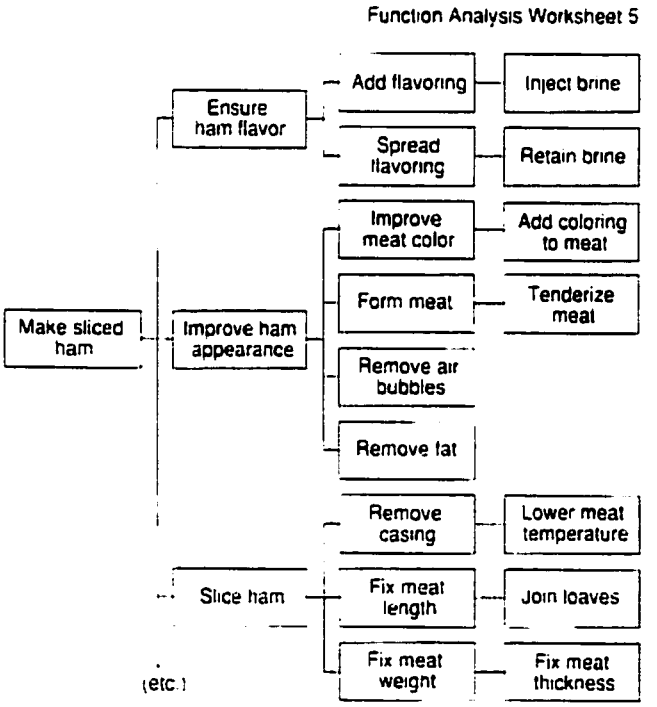
- The relation between a parent node and its children nodes is expressed in equation 3.4

$$F(X_1, X_2, \ldots, X_j) = \cup\{ F(Y_1, Y_2, \ldots, Y_j, Y_{j+1}) \mid \forall j \in Z^+, X_j = Y_j \}$$

(3.4)

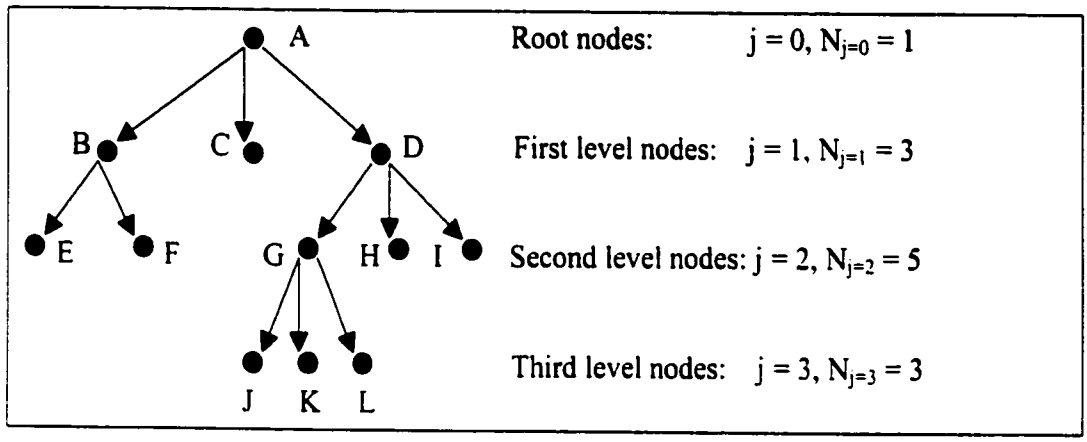
This equation can be interpreted as following: A node (function) *F* in $j^{th}$ level is the set of $j+1^{th}$ level nodes whose first j parameters are equal to those of *F*. From Table 3.6 we can see it clearly that G = {J, K, L}, because the form of G is F (3,1) and J, K, and L share the same first two parameters with G.

Figure 3.7 A sample function family tree [59]




*A Japanese foodstuffs company uses this function family tree to analysis its production system. A sliced ham manufacturing process is chosen to get higher yields and lower costs.*

Figure 3.8. Abstract Function Family Tree (AFFT)




*AFFT represents the relations among the nodes by set theory, so it can be easily implemented in a computerized knowledge-based system. The parameter j represents the level of the functions in AFFT and $N_j$ represents the total nodes at a certain level of AFFT.*

Table 3.6. Graph Theoretic (GH) form for all nodes in Figure 3.8

| Node name | GH form | Node name | GH form |
|---|---|---|---|
| A | F ( ) | G | F (3,1) |
| B | F (1) | H | F (3,2) |
| C | F (2) | I | F (3,3) |
| D | F (3) | J | F (3,1,1) |

| E | F (1,1) | K | F (3,1,2) |
|---|---|---|---|
| F | F (1,2) | L | F (3,1,3) |

*F() represents root function because there is no parameters in its parentheses. F(3,1) is the second level function because there are two parameters in its parentheses. The first parameter "3" indicates that its parent node is node #3 in the first level, its second parameter "1" indicates that it is the first child node of its parent node F(3).*

*Basic AFFT operations*

- Assign value operation

The value of a node (function) is a VOP. For example in Figure 3.7, F ( ) = *make sliced ham*, F (2) = *ensure ham appearance*, and F (2,2) = *form meat*. When a VOP is assigned to a function, a knowledge-based system can retrieve functions in AFFT by it. Exact grammar and syntax of VOP is a larger issue in AIM-D [1] and is not treated directly herein.

- Leaf operation

If we can work out a direct implementation for a specific function in AFFT, there is no further function decomposition effort needed. This function will be marked as leaf function and a possible implementation behavior will be recorded on Implementable function form (IFF). A sample IFF is shown in Table 3.7.

- Children check operation

In Figure 3.7, we can see that *fix meat thickness* is the only sub-function of *fix meat weight*. This situation is not allowed in AFFT. In another word, a parent node should have at least two distinct children nodes. If a function has only one sub-function in certain situation, there are two possible reasons: (a) the higher-level function is not defined properly, (b) the lower level function may be a behavior that implements its

higher-level function. For the first circumstance, the children check operation will simply substitute the higher-level function with the lower level one. For the second circumstance, a leaf operation will be applied to the higher-level function and its only sub-function will be recorded in the *proposed implementation* attribute in IFF.

- Equality check operation

Table 3.7 Implementable function form (IFF)

Label:_____ Date:_____

| ***Attributes*** | ***Content*** |
|---|---|
| *Function type* | *Empty* |
| *Function name* | *Empty* |
| *Input descriptor* | *Empty* |
| *Output descriptor* | *Empty* |
| *Proposed implementation* | *Empty* |
| *Why link* | *Empty* |
| *Value* | *Empty* |

*The major difference between IFF and FIF is that the fourth attribute is changed* ***to proposed implementation*** *instead of* ***how link****, because no further function decomposition is needed.*

A particular function may serve more than one higher-level functions. In Figure 3.9, we can see that the function – *reduce inventory*, supports three higher level functions which are *reduce purchasing cost*, *reduce fixed cost*, and *reduce transportation cost*. So the functions in different levels are not confined to one-to-one mapping. A few researchers mention this phenomenon in the literature but only on a conceptual level. The current author introduces *equality check operation* in AFFT to solve this problem.
To avoid function cross-citation in AFFT and keep all functions in simple parent-children relation, the author modifies the function hierarchy in Figure 3.9 to Figure 3.10. The difference is that each higher-level function has its own *reduce inventory* function instead of sharing a common *reduce inventory* function. The operation of *equality check* is to compare the value (VOP) of each node within a certain level and compute *weight* for each function.

The weight of a function is the number of times that this function appears in a certain level. For example, the weight of each function in Figure 3.9 is listed in Table 3.8. The *weights* of a set of functions can be used to rank their priorities. The bigger the weight, the more concern should be paid to this function during design process. From

Table 3.8, we can identify that the function – *reduce inventory (with weight 3),* is the key factor that contributes most to the top-level function – *reduce cost.*

The equality check operations are useful in redesign work, such as product improvement. When we analysis the function hierarchy of the exist systems. It is very common that we could not get the final function hierarchy at the first trial. Some modification work and relocation efforts are inevitable. By applying the equality check operations, we can simply input the draft AFFT and some relations among nodes, and let computers work out the final AFFT by itself.

### 3.5.2 Implementation of this Graph Theoretic model

*Organize Function Identification Forms*

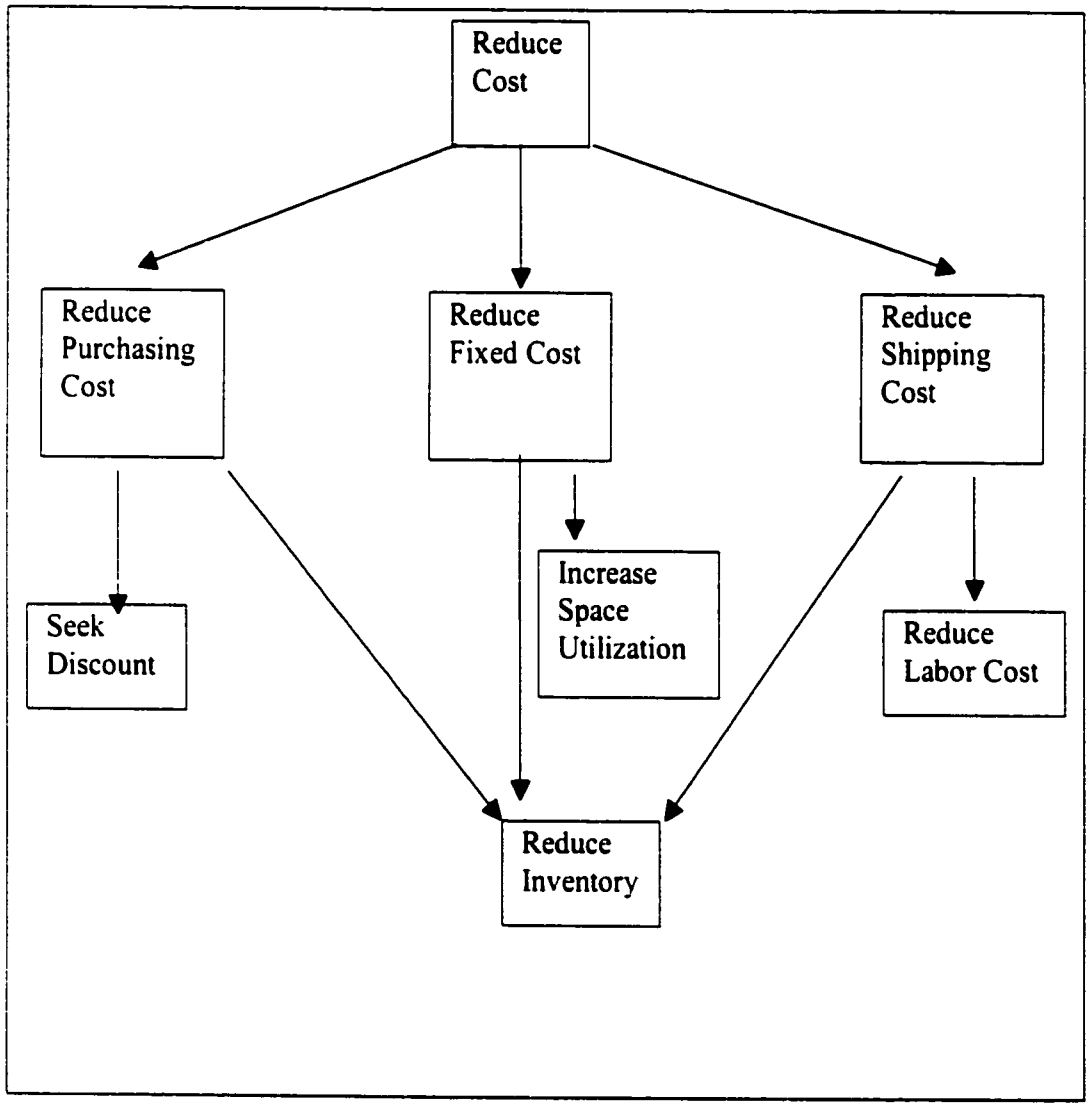
The AFFT mathematics model can be used to organize various functions in a single project. Since each function in AFFT has a unique mathematics form, such as F(2,2), it can be used as a label for each function. The author has introduced the Function Identification Form (FIF) to capture the core characteristics of individual functions. Now we can add this label to each FIF, and they will be organized according to the structure of AFFT. Table 3.9 shows a modified FIF.

*Monitor design process*

AFFT along with a set of colors (Table 3.10) can be used to track design process. At the initial stage, all nodes in AFFT are marked in white. When a leaf node is designed (implemented), it is marked in red. According to the weights of functions in a certain level, we can work out the percentage that the completion of certain function to its parent function, its grandparent function and so on until the root function. Let us suppose *seek*

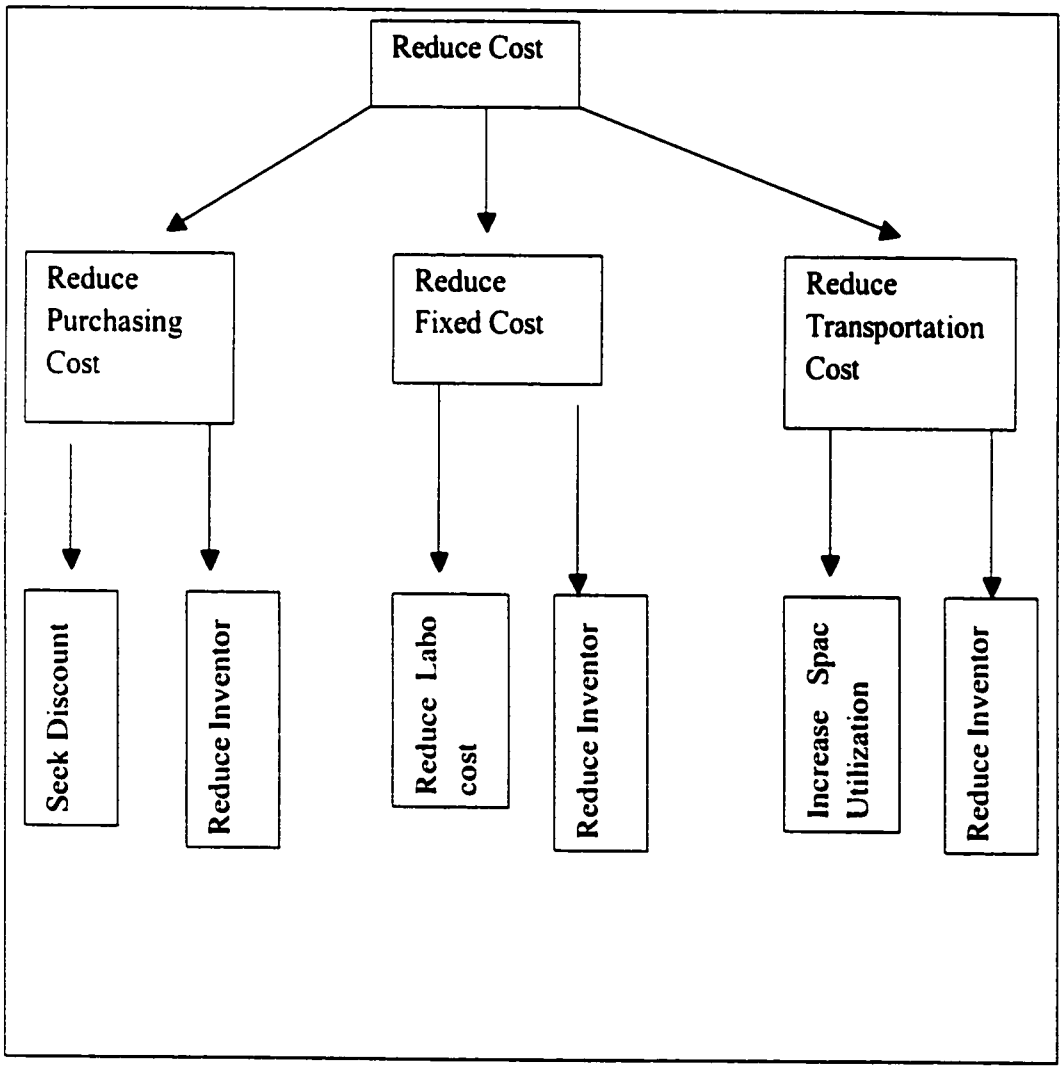
*discount* function in Figure 3.10 is implemented. It contributes 1/2 to it parent function *reduce purchasing cost* and *reduce purchasing* contributes 1/3 to the root function. So the implementation of *seek discount* function contribute to 1/6 (1/2 × 1/3) of the root function. According to Table 3.10, *reduce purchasing cost* is marked in GREEN and *reduce cost* is marked in LIGHT GREY. The color of the root node changes gradually when design progresses. It can be used as a visual indicator to the design process.

Figure 3.9 Initial function hierarchy before equality check



***Reduce inventory*** *function supports three higher level functions which are* ***reduce purchasing cost, reduce fixed cost,*** *and* ***reduce transportation cost.***

Figure 3.10 Final function hierarchy after equality check



***Reduce inventory*** *function supports three higher level functions which are* ***reduce purchasing cost, reduce fixed cost,*** *and* ***reduce transportation cost.***

Table 3.8 Weights of Third Level functions in Figure 3.10

| | Function Name | Weight |
|---|---|---|
| Second Level | Reduce Purchasing Cost | 1 |
| | Reduce Fixed Cost | 1 |
| | Reduce Transportation Cost | 1 |
| Total Weights in Second Level | | **3** |
| Third Level | Seek Discount | 1 |
| | Reduce Inventory | 3 |
| | Reduce Labor Cost | 1 |
| | Increase Space Utilization | 1 |
| Total Weights in Third Level | | **6** |

*Weights of functions in Figure 3.10. It is obvious that reduce inventory has the biggest weight, in another word, it is the most important function in this level.*

Table 3.9 New Function Identification Form (with label)

Label:_____ Date:_____

| **Attributes** | **Content** |
|---|---|
| Function type | Empty |
| Function name | Empty |
| Input descriptor | Empty |
| Output descriptor | Empty |
| How link | Empty |
| Why link | Empty |
| Value | Empty |

*Label and Date are added to an empty FIF for the sake of function organization.*

Table 3.10 Node color and its meaning

| Color | Percentage of Completion |
|---|---|
| WHITE | 0 |

| | |
|---|---|
| PINK | 1/6 |
| ORANGE | 1/3 |
| GREEN | 1/2 |
| BLUE | 2/3 |
| LIGHT GREY | 5/6 |
| RED | 1 |

### 3.6 Implementing IFM to engineering design

The author is implementing this new function model as a guide to design procedure. The conceptualization of this implementation is described here. In Chapter 4, an example is presented to demonstrate how IFM helps a designer to synthesize the customer requirements and generate potential design solutions.

There are many references on engineering design theory [e.g. 36 – 38]. In these books, each author gives a set of formal steps of engineering design and they are similar. The current author believes that the *function identification form* (FIF), which is derived from IFM and presented in Table 3.5, can be used as a guide for designers to formalize their design activities within any of these design processes. To demonstrate this, the current author arbitrarily selected a design process introduced by Hyman in [38].

Hyman uses a nine-step design process which begins with *recognizing the need* and ends with *implementing the preferred design*. The current author found that most steps in Hyman's model could be mapped into the function identification form. This means that FIF can be used as a progress indicator during the entire design process. Initially, the

form is blank. With the design process proceeds, empty items in the form are filled in. When this form is fully filled, the functional design stage is finished.

It must be noted that the function identification form could be used in conjunction with virtually any design process reviewed by the author. The particular design process used here was chosen only for demonstration purposes.

**Step 1: Recognizing the need**

Hyman states, "this step establishes the ultimate purpose of the project via a general statement of the client's dissatisfaction with a current situation." According to IFM, the corresponding item to this design step is *why link* item in Table 3.5. The why link attribute of the top function indicates the purpose of this function and it matches the explanation of the first design step – establishing the ultimate purpose. If the current function is derived from other function(s), the why link indicates its nearest "super" function, which means that the purpose of this function is to provide the necessary support to make its parent function(s) accomplishable. This attribute builds hierarchy structures within a group of functions.

**Step 2: Defining the problem**

This step is to translate the statement of need into one that addresses how we propose to satisfy the need. It is composed of three relevant components which are *goals, objects and constraints*.

Goals: The goal is defined as a brief, general, and ideal response to the need statement. In IFM, the author uses a verb/object pair to represent the goal. For example,

*detect temperature, transfer data, absorb heat* etc. The correspondent item of the goal in FIF is the function name item.

Objects: The objective(s) are quantifiable expectations of performance. The customers and designers can use objective(s) to decide how well a design meets the client's exceptions. For example, one object of a new CPU may be stated as following, *the kernel speed of this CPU should be no less than 400MHz.*

Constraints: Constraints defines the permissible range of the design and performance parameters. If a design does not satisfy a constraint, it will be rejected no matter how well it performs with respect to one or more of the objectives. As for the CPU example mentioned above, one of its constraints might be that its size cannot exceed 100 square millimeters.

Since the difference between objects and constraints are slight, it is convenient for us to combine these two components into *value* item in the FIF.

**Step 3: Planning the project**

**Step 4: Gathering information**

These two steps treat management affairs of the design process. Since the current author is concerned mainly with technical affairs in design process, there are no corresponding items in FIF for them.

**Step 5: Conceptualizing alternative approaches**

This step is where possible design solutions are first envisioned. Since this is the core design step, the current author arranges three correspondent items in FIF to

represent, which are function type, input descriptor, and output descriptor. Detail example can be found in Chapter 4.

**Step 6: Evaluating the Alternatives**

**Step 7: Selecting the Best Alternatives**

**Step 8: Communicating the Design**

These three steps are important but they deal with meta-level or management level of design affairs. Since the author concerns mainly on function modeling, a technical aspect of an engineering design, it is reasonable to leave these affairs to some higher-level design systems of a knowledge based system, for example ACM [2].

**Step 9: Implementing a Preferred Design**

This step is also called *detailed design* or *embodiment design*. When the best alternative is selected in step 7, designers need to think about how to realize this function. There are two possible situations. First, the function is directly implementable. What designers need to do is working out the physical structures of the input/output nodes and their relations. For example, when the parallel port of the computer is identified and connection cable is given, *connect printer to computer* is a directly implementable function. The second situation is that there is no direct implementation of this function. What designers need to do is decomposing this function into sub-functions and list them in *how link* item in FIF. This effort continues until all sub-functions are directly implementable. In Chapter 4, a detailed example of how this process works is presented.

Suh [36] states, "In the absence of principles or axioms that could be used as absolute foundations or referents, design decisions can only be made on an ad hoc or empirical basis." By applying the FIF to this engineering design process, the current author constructs a method to guide designers through the entire design procedure from a functional point of view. Since there are no restrictions in this method, it will not block the creativity of designers in conceptual design stage and provides a formalized guidance at the same time.

### 3.7 Potentially supported reasoning tasks

The goal of any knowledge representation is to enable and facilitate automated and semi-automated reasoning processes. In this section, the author introduces some of the reasoning tasks that could be eventually implemented using IFM. This is not an inclusive list, but it does indicate the potentially broad application of IFM. For the sake of this presentation, the author assumes there exists an ontology of functions and objects for some particular application domain. Though the development of such an ontology is problematic, it is beyond the scope of the author's current work.

#### 3.7.1 Function retrieval

Given some product that has been described in structural terms (e.g. in terms of parts and properties), an automated system would have the ability to "extract" functions. The system would match patterns of connectivity of parts in the structural description to the ontology relating object terms in the function specifications. It would then extract the

functions provided by those patterns and suggest them as possible functions of the initial product. Such a system could be used to extract function information from "legacy" product models, or to verify that a product provides all the functions it is supposed to provide.

### 3.7.2 Case-based reasoning

An ontology of function specifications provides the foundation for creating a design case library for a function-oriented case-based reasoning engine. This engine would be able to advise designers about potential structural descriptions of functionally stated design problems, based on similarities to cases in the library. This kind of system could be used during the early, upstream stages of a design process to suggest design concepts based on past experience of an enterprise.

This is of particular importance for routine or variant design tasks. The cases matched by the reasoning engine could represent alternative variations of existing designs, where the variations are developed by attempting to reconcile differences between function specifications of the given design problem with respect to cases in the library. The DVD design example in Section 3.2.3.2 demonstrates the core procedures of this case-based reasoning which is:

- Determine the type of top-level function
- Search design database for the previous cases in the same top-level function type category for references and inspirations
- Determine how link item of Function Specification (FS)

### 3.8 Other relevant concepts

Function modeling is highly related to other design elements, such as *context, purpose/intention, structure, mereology* etc. It is therefore necessary to examine the interrelations between IFM and other aspects of product modeling.

#### 3.8.1 Context

"The main motivation for studying formal contexts is to resolve the problem of generality in AI. Context eliminates certain ambiguities or multiple meanings in the message." [Akman and Surav, 1996].

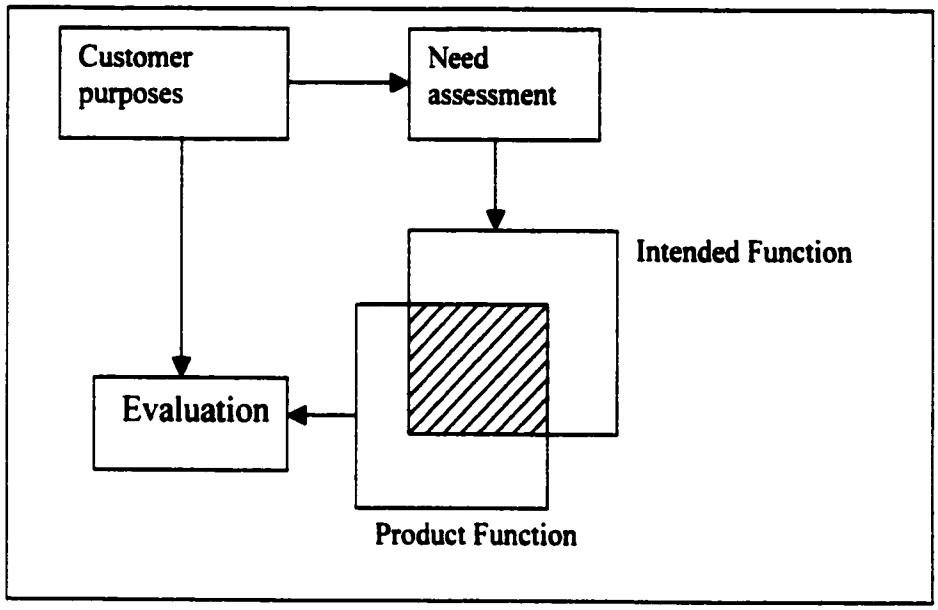
When many designers work on the same design, there will inevitably be mismatches between what they know and the names they used to identify those items of knowledge. The author believes that the *nodes* in IFM can be used as the context in function modeling in that (a) all functions occur among nodes via one or more channels of node, (b) all nodes are represented in the form of frame, (c) nodes have the ability to represent something that is not physically exist yet (by just list the expected properties in certain facets or slots).

#### 3.8.2 Purpose/Intention

##### 3.8.2.1 Concepts of purpose, intended function and product function

Purpose is (a) an intention or plan (b) a person's reason for an action [26]. There are two types of purpose in engineering domain. One is the purpose of the users; the other is the purpose of the designers. The first type is also called *customer requirements* or *customer needs* and the second type is called *intended functions* by the author. *Product*

Figure 3.11 Relations among purpose, intended function and product function




*Intended functions are derived from customer requirements via need assessment procedure. When a product comes into being, it manifests its own function "set" and overlaps part of intended functions. An evaluation procedure is used to judge how well this specific product meets the customer requirement.*

*functions*, in terms of IFM, is a set of all possible interactions between a certain product and its operational environment.

### 3.8.2.2 Relationships among purpose, intended function and product function

Figure 3.11 represents the relationships among purpose, intended function and product function. Intended functions are derived from customer requirements via need assessment procedure [65]. When a product comes into being, it manifests its own function "set" and overlaps part of intended functions. An evaluation procedure is used to judge how well this specific product meets the customer requirement.

Li states [39] "The universe is in motion...Physicists hold that the motion of matter follows certain laws. The changes of our entire universe also follow laws." According to his statement, all nodes and their channels in IFM are in motion guided by the "natural laws". The current author calls them *natural interactions*. A product is an entity that facilitates the preferred interactions and prevents the unwanted ones according to designers' intentions. For example, thermos bottle is designed to retard the interaction between two nodes – hot water and its environment. Without this product, the heat transfer via these two nodes' energy channels would be too fast to keep the water warm in a certain period of time.

### 3.8.3 Structure

Structure describes which components constitute the system and how they are connected to each other [5]. Various methodologies are discussed in the literature for mapping functions to structures. For example, Keuneke advocates a two-pass process [7].

In the first pass, the result of function decomposition is used to identify a component for each lowest-level function to be provided. In the second pass, these components are reorganized and integrated into the parts of the product. Based on the literature, the author believes IFM is suitable to most of these methodologies in that (a) IFM only deal with functional issues and there is not any structural restrictions in IFM, (b) the node name in IFM is highly associated with structural components, (c) structural relationships among components can be derived from AFFT.

### 3.8.4 Mereology

Mereology is the study of part-whole relations. Insofar as many engineered products can be regarded as compositions of parts in one fashion or another, mereology is fundamental to achieving a deeper understanding of design [30]. Function and mereology are tightly coupled in that (a) function decomposition requires a mereological relationship (b) function defines the role that an object plays in a larger system as well as the way that the physical structure of the object responds to external stimuli which means function relates an object's physical structure (parts) to systems that contain it (wholes). Salustri and Lockledge [30] are currently working towards a formalization of mereology that will be consistent with the current work herein on function modeling.

## 3.9 How IFM explains other researchers' ideas

Table 1.1 lists various function definitions given by researchers in function modeling domain. It can be argued that the current author's interaction based function model could explain most of their ideas without conflict.

Kitamura and Mizoguchi [4], Chittaro, Tasso and Toppano [5], Keuneke [7] define function according to designer's purpose or intent. As the current author has pointed out in Section 3.8.2, a product does manifest part of the designer's purpose but it is not complete to treat function solely as purpose or intention. Why a coffee cup can be used as a paperweight that is not a design purpose of the designer? If we treat function as a natural interaction among nodes, it is easy to explain this phenomenon in that the facets of coffee cup's mass slot (weight, shape) interacts with the facet of paper's mass slot (shape) in a certain context. Designers only guide or block some of the transformation mechanisms among basic elements according to their design intention or purpose, and their efforts are inevitably manifested in a product as the dominant characteristics. This is the reason why some researchers treat functions mainly as purposes or intentions according to IFM.

Salustri [3], Sasajima et al [8], Qian and Gero [17] treat function as the abstraction or interoperation of behavior. According to IFM, function is a transformation mechanism among basic elements. This definition has indicated that function itself is a kind of behavior, a transformation or interaction. The difference is that this kind of behavior is in element level instead of in entity level. It is commonly stated as the "reason" for an entity level behavior or in another words abstraction or interoperation. Salustri [3] has found that behavior and function are interchangeable under different contexts and both of them can be represented by the same VOP form. His observation demonstrates that function and behavior can be integrated.

Hodges's definition [6] represents a group of researchers' idea that function is related to the change of the states. This can be also well explained by IFM in that

function is a transformation mechanism between stages. A set of basic elements changes to another set from initial stage to final stage.

Chandrasekaran and Josephson [9] define function as a formula and relate function with its environment. What is environment? Environment is the external node of a certain product/object. What is a formula? According to Chandrasekaran and Josephson, a formula describes the initial and final states of an object according *natural laws* or *principles*. So a formula describes a set of special behaviors that must occur if certain pre-conditions are met. Since it also describes the characteristics of the behaviors, it can be fitted into IFM as the current author has explained above.

To summarize, it seems that IFM could explain many different function definitions given by different researchers. So it is possible that IFM can be used as a platform to integrate or interchange these different function concepts and related function models.

# CHAPTER FOUR

## AN EXAMPLE FOR IMPLEMENTING IFM TO DESIGN PROCESS

***Problem:*** *BeiNei Group is the largest engine manufacturer in China. One of its engine test workshops has 22 platforms. 44 engines can be tested in this workshop simultaneously. The peak sound intensity of this workshop was over 100 dB. A noise reduction project is assigned to a designer. It is required that the sound intensity should not be over 65 dB.*

The conceptual design task can be achieved by following the method introduced in Appendix B step by step.

Hyman [38] states, "need is an expression of dissatisfaction with the current situation and it is the ultimate goal of an engineering design." The current author treats customer needs as the top-level function requirement.

The "real" need of the customer may hide in the problem statement. Designers should excavate them. By doing the customer survey, the following root causes that may lead to this noise reduction problem are explored.

1. The production plan of the engine test workshop is not even. When all of 44 platforms are used, the peak noise is generated. Sometimes, the entire workshop stays quietly because no engines are fed from assembly workshop for test.
2. This engine workshop runs 24 hours a day. The residents around the workshop complain that the noise is so high during the night that they cannot fall asleep.

3. The union claims that the noise is too high to the workers' health. The employer must do something to protect the worker's health.

By applying needs analysis, it is obvious that the needs of the customers are greatly different although the problem statement is the same. This leads to totally different implementations. In another word, the innovative conceptual design solutions are generated at the earliest stage.

Table 4.1 records these three different needs of the customer in the form of a VOP. We note that the first case explores a production planning problem. It is believed that if the production would be distributed evenly, the peak noise-intensity could be reduced because the maximum number of engines running simultaneously is reduced. For the second case, the residents only need isolate the noise at night around their residences. For the third case, it is totally different, the union cares about the noise IN the workshop because the workers' health are its major concern.

To demonstrate how IFM could be used to assistant engineering conceptual design in a wide scope, the author would like to further develop all of these three proposed cases according to Appendix B.

## 4.1 Case #1: Minimize production variation

### 4.1.1 First Design Loop

#### 4.1.1.1 Needs Analysis

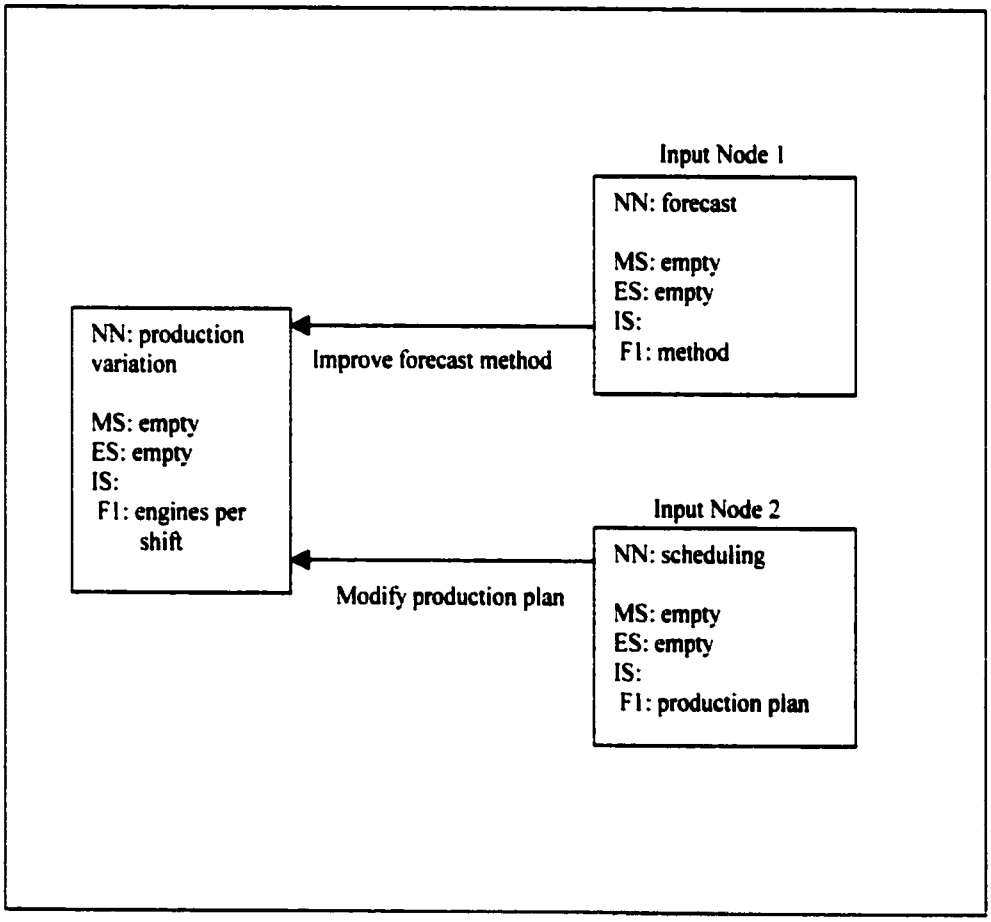
In the first design loop , the needs of customers are the top-level functions. As we have recorded in Table 4.1, *minimize production variation* is the need of the customer in this case.

Table 4.1 Three proposed product functions

| Proposed case | Needs of the customer (VOP) |
|---|---|
| 1 | Minimize production variation |
| 2 | Isolate noise |
| 3 | Protect health |

*For the same problem stated at the beginning of this chapter, there are different needs in different situations.*

Figure 4.1 Node relation chart for *minimize production variation*




*Production variation (I) is the output. Scheduling (I) and forecasting (I) are two input nodes.*

4.1.1.2 Function Evaluation

Let us suppose that there is no direct implementation in either design knowledge database or designer's own experience. Further work is needed.

4.1.1.3 Draw node relation chart

Figure 4.1 is a proposed NRC for *minimize production variation* function. The *production variation* of the engine test workshop is the output node. Information (engines per shift) is the only non-empty slot of this node. There are two input nodes of *production variation*. The first one is *forecast*. Its information slot – *method* affects the performance of the forecast accuracy. The second one is *scheduling*. Its information slot – *production plan* explicitly determines the number of engines tested per shift.

4.1.1.4 Select nodes and generate sub-functions

Let us select all input nodes, so the *minimize production variation* is an II > I type function. Table 4.2 lists the related slots of input nodes, output node and the corresponding sub-functions.

4.1.1.5 Fill FIF for *minimize production variation*

Now we can organize and document all information we have got about the *minimize production variation* by filling FIF. It is shown in Table 4.3. Four sub-functions are going to be evaluated in the next functional design loop and further developed.

4.1.2.1 Improve forecast method

4.1.2.1.1 Function Evaluation

Currently, sales department forecasts expected demands for production department. The figures are mainly based on their experiences. A statistical method – *Simple*

Table 4.2 Corresponding slots and sub-function for *minimize production variation*

| Slot of ON | Slot of IN | Sub-function name | Sub-function type |
|---|---|---|---|
| Production variation (I) | Forecast (I) | Improve forecast method | I > I |
| Production variation (I) | Scheduling (I) | Modify production plan | I > I |

Table 4.3: FIF for *minimize production variation*

Label: minimize production variation Date: February 2, 2000

| Attributes | Content |
|---|---|
| Why link | Minimize production variation |
| Function name | Minimize production variation |
| Input node(s) | Forecast (I), Scheduling (I) |
| Output node(s) | Production variation (I) |
| Function type | II > I |
| Value | Previous demand data, current production plan |
| How link | Improve forecast method (I > I),<br>Modify production plan (I > I) |

*Exponential Smoothing* for short-term forecasting is applied to improve this kind of experience-based manner. Detail discussion about this method can be found in [24]. Simple Exponential Smoothing is the direction implementation for function – *improve forecast method.*

4.1.2.1.2 Fill IFF for Improve forecast method

Table 4.4 is the Implementable function form for function – *Improve forecast method.*

4.1.2.2 Modify production plan

4.1.2.2.1 Function Evaluation

There are 22 platforms in engine test workshop. Each platform contains 2 slots and each slot can be used to test one engine. Figure 4.2 is the current process chart, the process lead-time is 60 min. Figure 4.3 is the modified one, the proposed lead-time is 75 min. Suppose the total available time per shift is 360 min, then the maximum capacity for the current process is 264 engines per shift (360/60*44) and the capacity of modified process is 210 engines per shift (360/75*44). According to the customer demand, the maximum shift production is no more than 132. So this modification will not affect the ability to satisfy customer requirements at present. The difference is that the number of engines testing simultaneously on the platform reduced from 2 to 1. It is reasonable to expect that the peak sound intensity would be reduced.

4.1.2.2.2 Fill IFF for Modify production plan

Table 4.5 is the IFF for function – *Modify production plan.*

4.1.3 Draw AFFT for *Minimize production variation*

Figure 4.4 is an AFFT for function - *Minimize production variation.*

Table 4.4: Implementable function form for *Improve forecast method*

Label: Improve forecast method Date: February 3, 2000

| Attributes | Content |
|---|---|
| Why link | Minimize production variation |
| Function name | Reduce variability |
| Input node(s) | Forecast (I) |
| Output node(s) | Production variation (I) |
| Function type | I > I |
| Value | Maximum Forecast difference < 30% |
| How link | Apply simple exponential smoothing method to forecast |

Table 4.5: Implementable function form for *Change production plan*

Label: Change production plan Date: February 3, 2000

| Attributes | Content |
|---|---|
| Why link | Minimize production variation |
| Function name | Change production plan |
| Input node(s) | Production plan (I) |
| Output node(s) | Production variation (I) |
| Function type | I > I |
| Value | Number of engines testing simultaneously = 1 per platform |
| How link | See Figure 4.3 |

Figure 4.2 Current test process in every platform

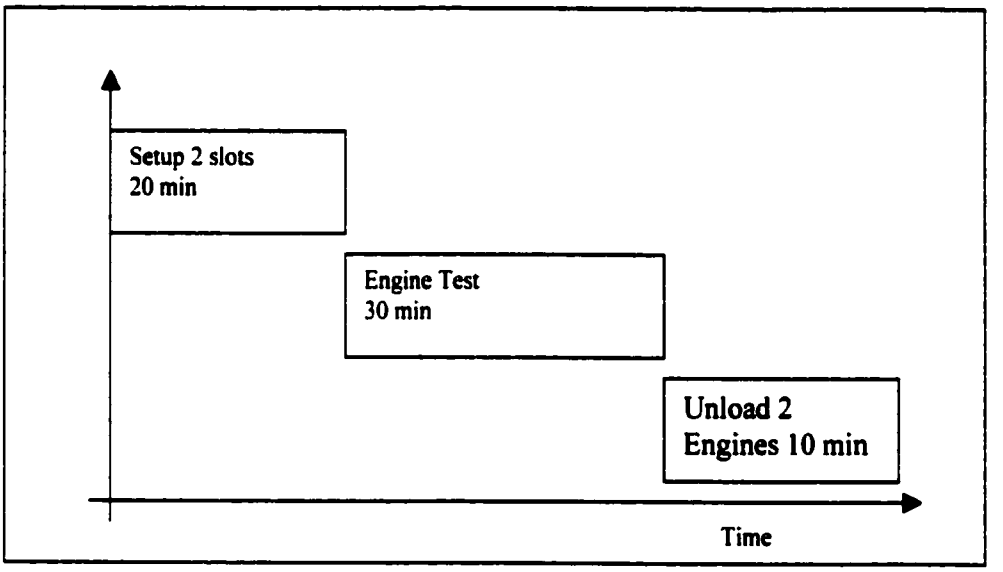



Figure 4.3 Modified test process in every platform

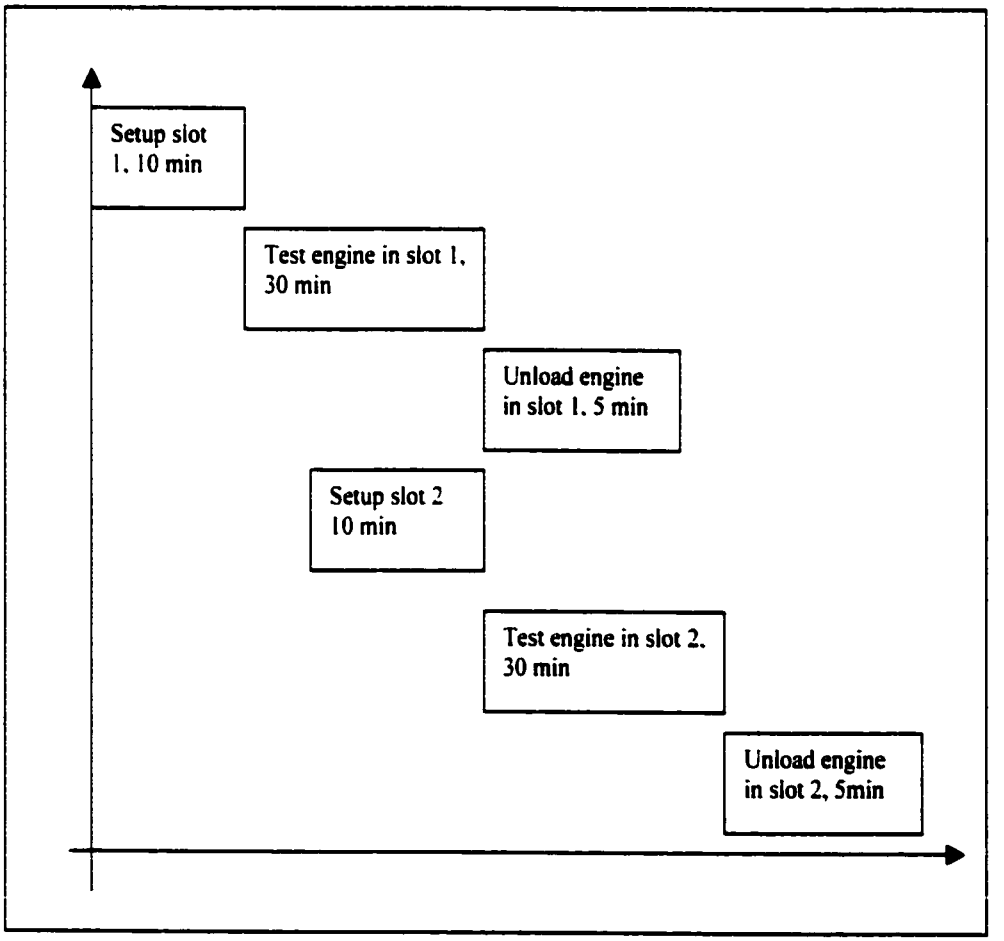
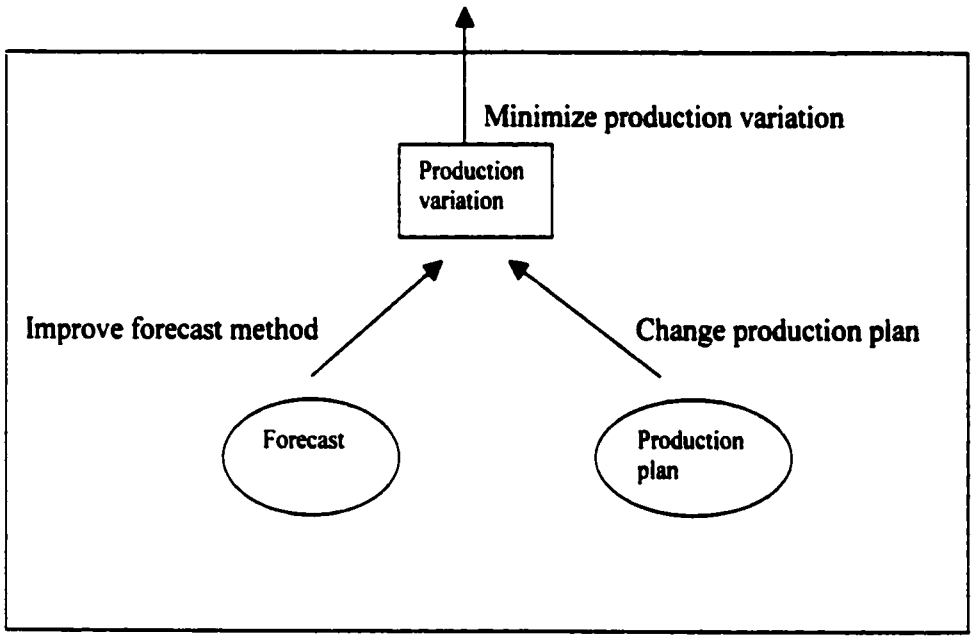


Figure 4.4 AFFT for *Minimize production variation*




*Node names are recorded in rectangles and ovals where rectangle node represents parent node and the oval nodes represents leaf nodes. Functions occur among nodes are recorded beside corresponding arrows.*

## 4.2 Case #2: Isolate Noise

### 4.2.1 First Design Loop

#### 4.2.1.1 Needs Analysis

In the first design loop, the needs of customers are the top-level functions. As we have recorded in Table 4.1, *isolate noise* is the need of the customer in this case.

#### 4.2.1.2 Function Evaluation

According to appendix B, let us suppose that there is no direct implementation in either design knowledge database or designer's own experience. Further work is needed.

#### 4.2.1.3 Draw node relation chart

Figure 4.5 is a proposed NRC for *isolate noise* function. The *noise* in the residence is the output node. Energy (sound intensity) and information (rest schedule) are two non-empty slots of this node. There are three input nodes of *noise*. The first one is *environment*. Its mass slot – sound transmission route affect the performance of the sound transmission process. The second one is *engine*. Its energy slot – sound intensity is the noise source of the entire systems. The third one is *workshop*. Its information slot – production schedule affects where the peak sound intensity located in the timetable.

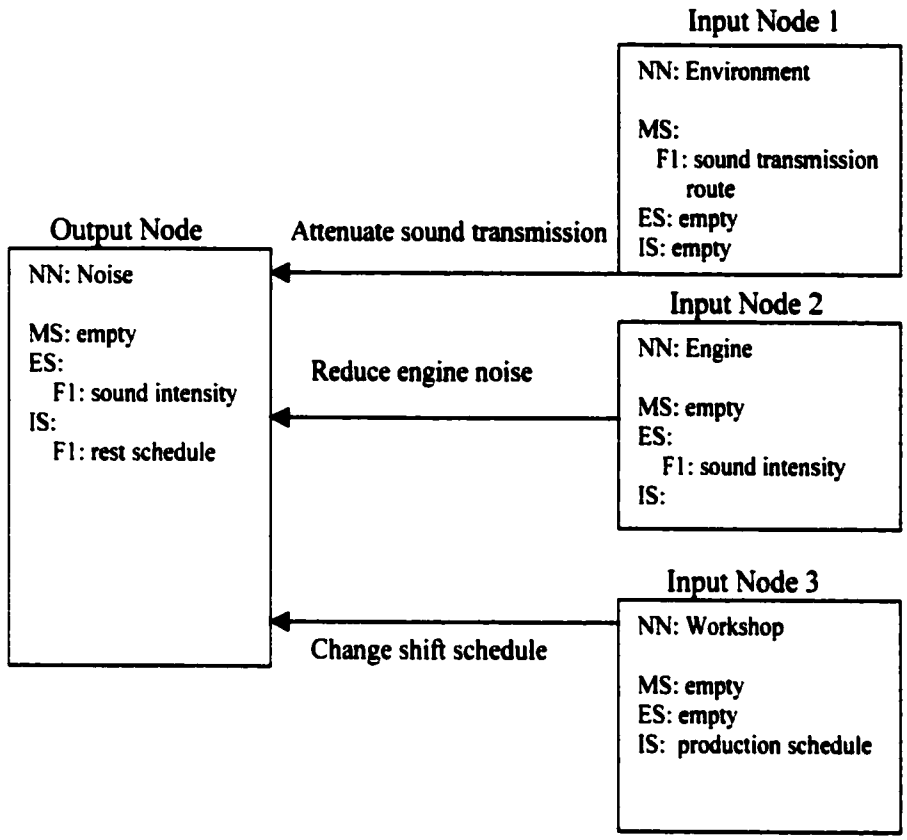
#### 4.2.1.4 Select nodes and generate sub-functions

Let us select all input nodes, so the isolate noise is a MEI > EI type function. Table 4.6 lists the slots of input and output nodes and the corresponding sub-functions.

#### 4.2.1.5 Fill FIF for *isolate noise*

Now we can organize and document all information we have got about the *isolate noise* by filling FIF. It is shown in Table 4.7. Three sub-functions are going to be evaluated in the next functional design loop and further developed.

Figure 4.5 Node Relation Chart for *isolate noise*




*Noise (E, I) is the output node and environment (M), engine (E) and workshop ( I) are three output nodes.*

Table 4.6 Corresponding slots and sub-function for *isolate noise*

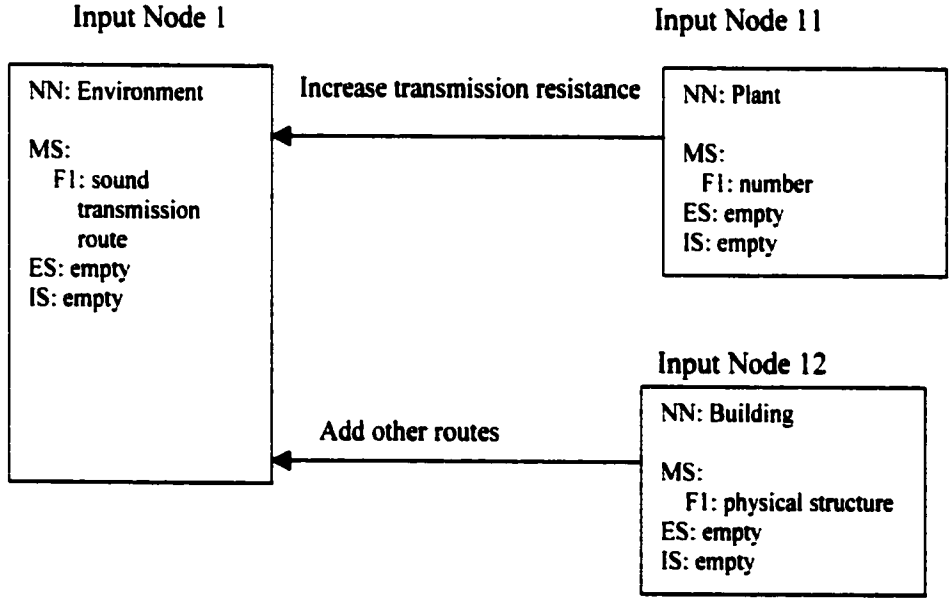
| Slot of ON | Slot of IN | Sub-function name | Sub-function type |
|---|---|---|---|
| Noise (E) | Environment (M) | Attenuate sound transmission | M > E |
| Noise (E) | Engine (E) | Reduce engine noise | E > E |
| Noise (I) | Workshop (I) | Change production schedule | I > I |

Table 4.7: Function Identification Form for *isolate noise*

Label: Isolate noise Date: February 1, 2000

| Attributes | Content |
|---|---|
| Why link | Isolate noise |
| Function name | Isolate noise |
| Input node(s) | Workshop (I), Environment (M), Engine (E) |
| Output node(s) | Noise (E, I) |
| Function type | M EI > EI |
| Value | Sound intensity < 65 dB, lead time < 1 month, cost < $3,000 |
| How link | M > E, M > E, E > E, I > I |

Figure 4.6 Node Relation Chart for *Attenuate Sound Transmission*




*Environment (MI) is the output node. Plant (M) and Building (M) are two output nodes.*

### 4.2.2 Second Design Loop

#### 4.2.2.1 Attenuate Sound Transmission

##### 4.2.2.1.1 Function evaluation

Let us also suppose there is no direct implementation in either design knowledge database or designer's own experience. Further work is needed.

##### 4.2.2.1.2 Draw node relation chart

Figure 4.6 is a proposed NRC for *Attenuate Sound Transmission* function. The *environment* between residence and workshop is the output node. Mass (sound transmission route) is its only non-empty slot. There are two input nodes of *environment*. The first one is *plant*. Its mass slot – number of plants affects the resistance of the sound transmission route because plants can absorb great amount of noise. The second one is *building*. Its mass slot – physical structure reflects sound to other directions which means adding other sound transmission routes to the environment.

##### 4.2.2.1.3 Select nodes and generate sub-functions

All of these two input nodes are selected, so the *Attenuate Sound Transmission* is a MM > M type function. Table 4.8 lists the related slots of input nodes and output node and the corresponding sub-functions.

##### 4.2.2.1.4 Fill FIF for *Attenuate Sound Transmission*

Table 4.9 organizes and documents all information about the *Attenuate Sound Transmission* function. Two sub-functions are generated and sent to the next loop.

Table 4.8 Corresponding slots and sub-function for *Attenuate Sound Transmission*

| Slot of ON | Slot of IN | Sub-function name | Sub-function type |
|---|---|---|---|
| Environment (M) | Plant (M) | Increase transmission resistance | M > M |
| Environment (M) | Building (M) | Add other routes | M > M |

Table 4.9 FIF for *Attenuate Sound Transmission*

Label: Attenuate Sound Transmission Date: February 1, 2000

| Attributes | Content |
|---|---|
| Why link | Isolate noise |
| Function name | Attenuate Sound Transmission |
| Input node(s) | Plant (M), Building (M) |
| Output node(s) | Environment (M) |
| Function type | MM > M |
| Value | Number of plants < 20, height of the building < 3 m |
| How link | M > M and M > M |

Table 4.10 Corresponding slots and sub-function for *Reduce Engine noise*

| Slot of ON | Slot of IN | Sub-function name | Sub-function type |
|---|---|---|---|
| Engine (E) | Platform (M) | Reduce resonance | M > E |
| Engine (E) | Down Pipe (M) | Attenuate noise | M > E |
| Engine (E) | Lubricate Oil (M) | Reduce wear and tear | M > E |

4.2.2.2 Reduce Engine noise

4.2.2.2.1 Function evaluation

Let us suppose that there is no direct implementation in either design knowledge database or designer's own experience. Further work is needed.

4.2.2.2.2 Draw node relation chart

Figure 4.7 is a proposed NRC for *Reduce Engine noise* function. The *engine* is the output node. Energy (sound intensity) is its only non-empty slot. There are three input nodes for *engine*. The first one is *platform*. Its mass slot – fixture clamps the engine firmly to reduce resonance. The second one is *down pipe*. Its mass slot –physical structure affects the performance of sound interference which results in sound attenuation. The third one is *lubricate oil*. Its mass slot – chemical ingredients are highly related to the wear and tear of the motion parts which are the major source of mechanical noise in engine.

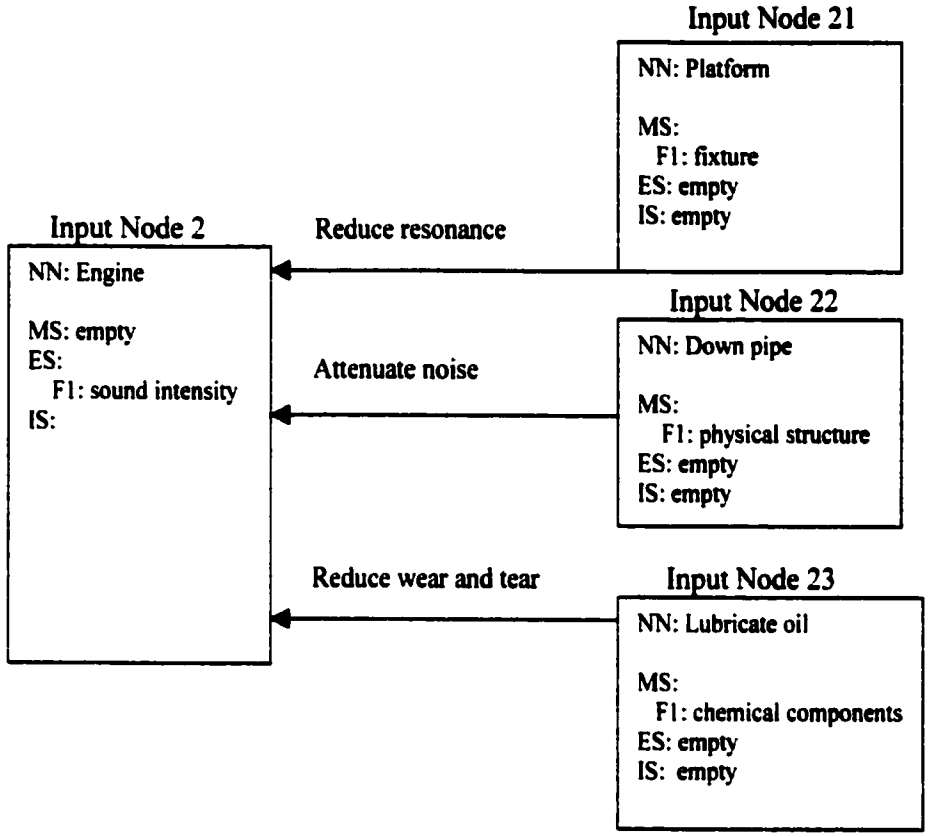
4.2.2.2.3 Select nodes and generate sub-functions

All of these three input nodes are selected, so the *Reduce Engine noise* is a MMM > E type function. Table 4.10 lists the related slots of input nodes and output node and the corresponding sub-functions.

4.2.2.2.4 Fill FIF for *Reduce Engine noise*

Table 4.11 organizes and documents all information about the *Reduce Engine noise* function. Three sub-functions are generated and sent to the next loop.

Figure 4.7 Node Relation Chart for *reduce engine noise*




*Noise (E, I) is the output node and environment (M), engine (E) and workshop (M, I) are three output nodes.*

Table 4.11 FIF for *Reduce Engine noise*

Label: Reduce Engine noise Date: February 1, 2000

| Attributes | Content |
|---|---|
| Why link | Isolate noise |
| Function name | Reduce Engine noise |
| Input node(s) | Platform (M), Down pipe (M) and Lubricate Oil (M) |
| Output node(s) | Engine (M) |
| Function type | MMM > M |
| Value | Oil consumption < 2L, the length of the down pipe > 1.5 m |
| How link | M > M, M > M and M > M |

Table 4.12 Current daily production plan

| Shift | Time period | Production |
|---|---|---|
| Mid-night | 0:00 – 6:00 | 44 ± 22 |
| Morning | 6:00 – 12:00 | 88 ± 44 |
| Afternoon | 12:00 – 18:00 | 88 ± 44 |
| Evening | 18:00 – 24:00 | 44 ± 22 |

4.2.2.3 Change shift schedule

4.2.2.3.1 Function evaluation

Current production plan is shown in Table 4.12. There are 4 shifts per day and the maximum capacity per shift is 132. Obviously, the production capacity per shift is not fully utilized. A new production plan is constructed to eliminate the mid-night shift which is the major reason for resident's complain. According to [23], the average production and variation per shift are recorded in Table 4.13. This is the implementation for *change shift schedule* function.

4.2.2.4.2 Fill IFF for change shift schedule

Table 4.14 is a IFF for function – change shift schedule.

4.2.3 Draw AFFT for *isolate noise*

Figure 4.8 is an AFFT for function - *Minimize production variation*. The same functional design method and procedure can be applied in the third design loop. Since the main purpose of this example is to demonstrate the ideas of functional design, the author would like to stop here to avoid too many repeat details.

## 4.3 Case #3: Protect health

4.3.1 First Design Loop

4.3.1.1 Needs Analysis

In the first design loop, the needs of customers are the top-level functions. As we have recorded in Table 4.1, *product health* is the need of the customer in this case.

Table 4.13 Modified daily production plan

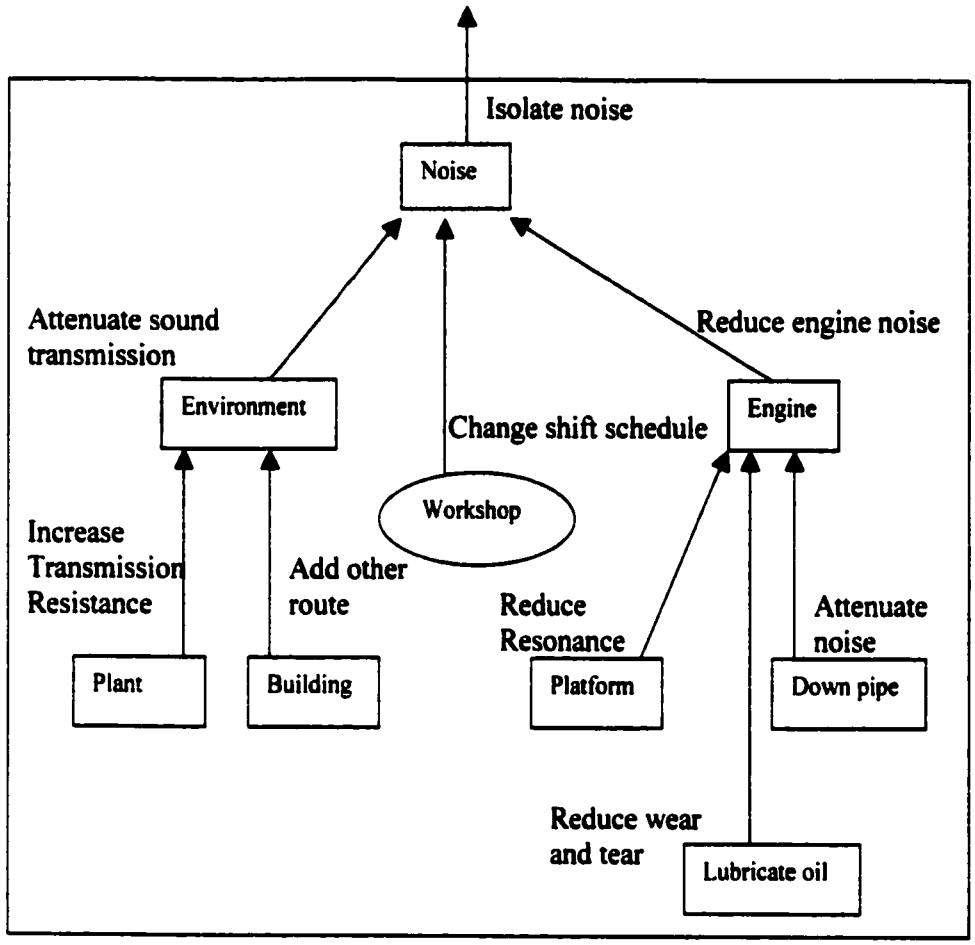
| Shift | Time period | Production |
|---|---|---|
| Mid-night | 0:00 – 6:00 | 0 |
| Morning | 6:00 – 12:00 | 88 ± 44 |
| Afternoon | 12:00 – 18:00 | 88 ± 44 |
| Evening | 18:00 – 24:00 | 88 ± 31 |

Table 4.14 Implementable function form for *Change shift schedule*

Label: Change shift schedule Date: February 3, 2000

| Attributes | Content |
|---|---|
| Why link | Isolate noise |
| Function name | Change shift schedule |
| Input node(s) | Workshop (I) |
| Output node(s) | Noise (I) |
| Function type | I > I |
| Value | Maximum capacity per shift = 132 |
| How link | Table 4.13 |

Figure 4.8 AFFT for *isolate noise*




*Node names are recorded in rectangles and ovals where rectangle node represents parent node and the oval nodes represents leaf nodes. Functions occur among nodes are recorded beside corresponding arrows.*

4.3.1.2 Function Evaluation

According to appendix B, let us suppose that there is no direct implementation in either design knowledge database or designer's own experience. Further work is needed.

4.3.1.3 Draw node relation chart

Figure 4.9 is a proposed NRC for *protect health* function. The *worker* is the output node. Its information (health) slot is the only non-empty slot of this node. There are three input nodes of *worker*. The first one is *air pollution*. Its mass slot – hazard materials is the major treatment to the worker's health. The second one is *engine*. Its energy slot – sound intensity is the noise source of the entire systems. The third one is *workshop*. Its information slot – production schedule determines the peak sound intensity in the workshop.

4.3.1.4 Select nodes and generate sub-functions

Let us select all input nodes, so the *protect health* is a MEI > I type function. Table 4.15 lists the slots of input and output nodes and the corresponding sub-functions.

4.3.1.5 Fill FIF for *protect health*

Now we can organize and document all information we have got about the *protect health* by filling FIF. It is shown in Table 4.16. Three sub-functions are going to be evaluated in the next functional design loop and further developed.

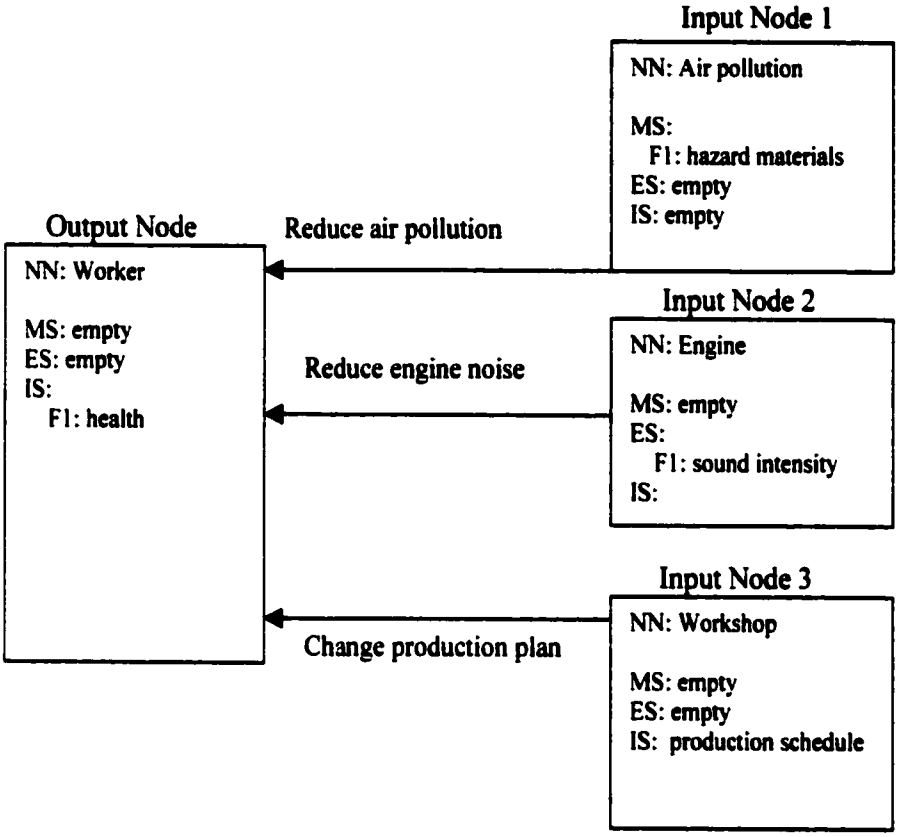
4.3.2 Second Design Loop

4.3.2.1 Reduce Air pollution

4.3.2.1.1 Function evaluation

According to appendix B, let us suppose that there is no direct implementation in either design knowledge database or designer's own experience. Further work is needed.

Figure 4.9 Node Relation Chart for *protect health*




*Noise (E, I) is the output node and environment (M), engine (E) and workshop ( I) are three output nodes.*

Table 4.15 Corresponding slots and sub-function for *protect health*

| Slot of ON | Slot of IN | Sub-function name | Sub-function type |
|---|---|---|---|
| Worker (I) | Air pollution (M) | Reduce air pollution | M > I |
| Worker (I) | Engine (E) | Reduce engine noise | E > I |
| Worker (I) | Workshop (I) | Change production schedule | I > I |

4.3.2.1.2 Draw node relation chart

Figure 4.10 is a proposed NRC for *Reduce air pollution* function. The *air pollution* in workshop is the output node. Mass slot (hazard materials) is its only non-empty slot. There are two input nodes of *air pollution*. The first one is *fuel*. Its mass slot – amount of leaking is a kind of hazard materials in the air. The second one is *emission*. Its mass slot – density is another hazard material in the air.

4.3.2.1.3 Select nodes and generate sub-functions

All of these two input nodes are selected, so the *Reduce air pollution* is a MM > M type function. Table 4.17 lists the related slots of input nodes and output node and the corresponding sub-functions.

4.3.2.1.4 Fill FIF for *Reduce air pollution*

Table 4.18 organizes and documents all information about the *Reduce air pollution* function. Two sub-functions are generated and sent to the next loop.

4.3.2.2 Reduce Engine noise

4.3.2.2.1 Function evaluation

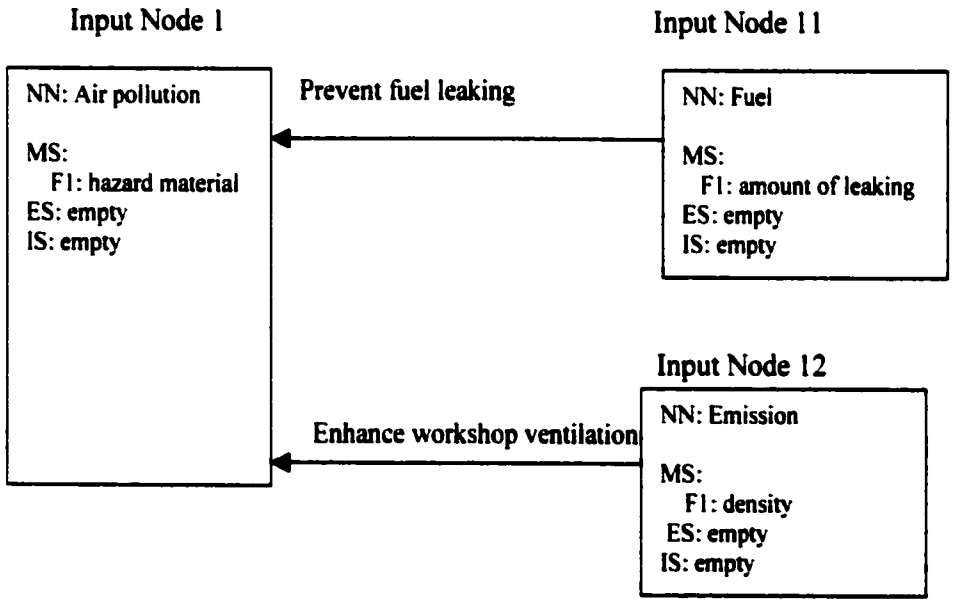
By searching our functional design database, we find that there is another function named reduce engine noise in Section 4.2.2.2. By comparison, the node structures of these two functions are the same and all input nodes and sub-functions for reduce engine noise in Section 4.2.2.2 can be implied to the current one. So we can label these two functions equal and assign weight 2 to reduction engine noise function.

Table 4.16: FIF for *protect health*

Label: Protect health Date: February 4, 2000

| Attributes | Content |
|---|---|
| Why link | Protect health |
| Function name | Protect health |
| Input node(s) | Air pollution (M), Engine (E), Workshop (I) |
| Output node(s) | Worker (I) |
| Function type | M EI > I |
| Value | Sound intensity < 65 dB, HC < 50 ppm, CO < 0.3% |
| How link | M > I, E > I, I > I |

Figure 4.10 Node Relation Chart for *Reduce air pollution*




*Air pollution (MI) is the output node. Fuel (M) and emission (M) are two output nodes.*

Table 4.17 Corresponding slots and sub-function for *Reduce air pollution*

| Slot of ON | Slot of IN | Sub-function name | Sub-function type |
|---|---|---|---|
| Air pollution (M) | Fuel (M) | Prevent fuel leaking | M > M |
| Air pollution (M) | Emission (M) | Enhance workshop ventilation | M > M |

Table 4.18 FIF for *Reduce air pollution*

Label: Reduce air pollution Date: February 4, 2000

| Attributes | Content |
|---|---|
| Why link | Protect health |
| Function name | Reduce air pollution |
| Input node(s) | Fuel (M), Emission (M) |
| Output node(s) | Air pollution (M) |
| Function type | MM > M |
| Value | HC < 50 ppm, CO < 0.3% |
| How link | M > M and M > M |

4.3.2.3 Change production plan

4.2.2.3.1 Function evaluation

By searching our functional design database, we find that there is another function named change production plan in Section 4.1.2.2. By comparison, the node structures of these two functions are the same and all input nodes and sub-functions for change production plan in Section 4.1.2.2 can be implied to the current one. So we can label these two functions equal and assign weight 2 to change production plan function.

4.3.3 Draw AFFT for *protect health*

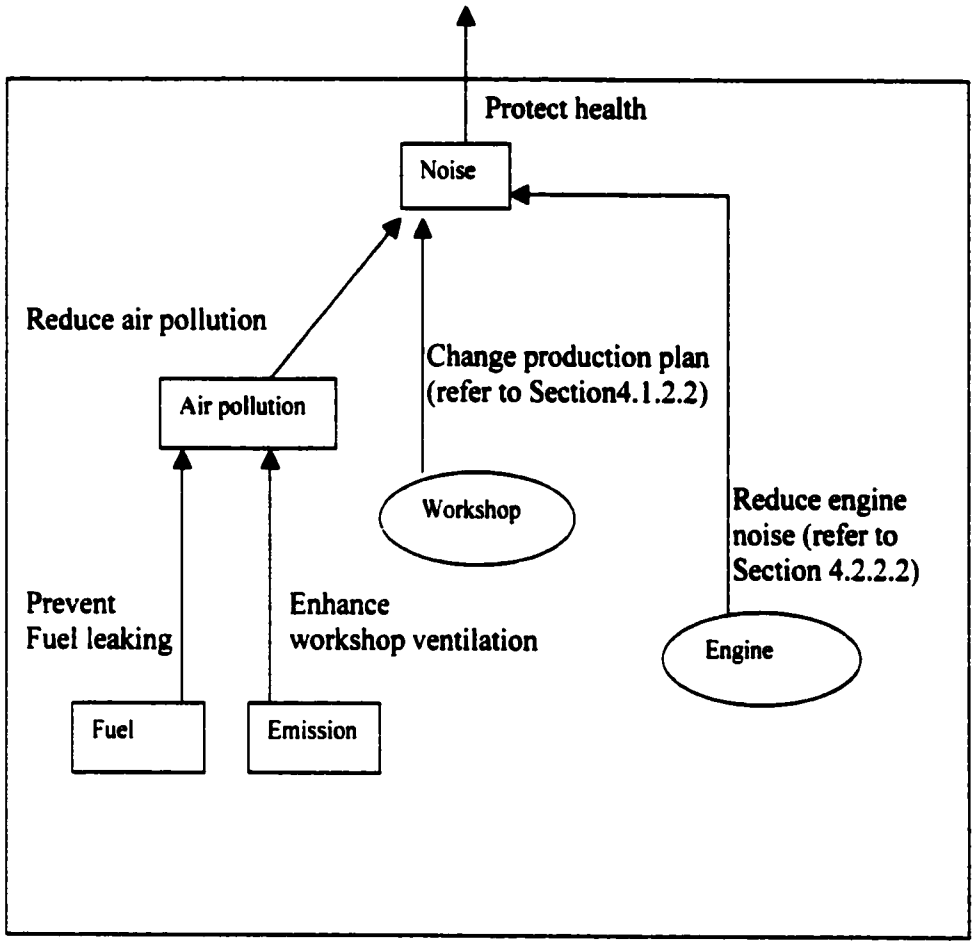
Figure 4.11 is an Abstract function family tree for function – *protect health*.

## 4.4 Summary

This chapter based on an example that the author experienced in his previous work. At that time, a traditional brainstorm method was adopted and ended up with a bulk of engineering design functions, behaviors, intentions, and structures. Without the help of IFM, the author could not clearly distinguish them, organize them, let along work out the hierarchy relationships among them. Finally, this noise reduction project was filed into "proposed improvement opportunity " category.

Now, with IFM in hand, the author not only worked out dozens of design functions step by step, but also organized them in a clear hierarchy structure. Moreover, intentions and structures are eliminated in conceptual design stage; behaviors are coupled with its higher-level functions. According to this example, it is reasonable to believe IFM could help engineering designers handle complex conceptual design problems. Furthermore, some unique characteristics of IFM also explored in this example:

Figure 4.11 AFFT for *protect health*




*Change production plan and reduce engine noise are two direct implementable functions because they have been developed in the previous sections. They can be simply reused here.*

1. Function reusability

From this example, we can see that the functions in Section 4.3.2.1 and 4.3.2.2 reuse previous function design in Section 4.1.2.2 and 4.2.2.2. The IFM has the inherent ability to support function reuse in that (a) functions occur between input and output nodes so they can be treated as the attribute of the node (b) the node is represented in the form of frame which inherits most object-oriented concepts (c) object-oriented concepts are build to support reuse previous designed components. The author believes that reusability can greatly reduce the lead-time in the product design process thus generates significant competitiveness.

2. Explicit function decomposition process

The examples in this chapter clearly demonstrate how to conduct function decomposition (by exploring input nodes of a certain output node). In literatures, only conceptual discussions were presented.

3. Compatible to most current engineering design process

The author pays special attention to compatibility issues when constructing IFM. It is compatible to most current engineering design processes. This makes it easier to learn and adopted by the current engineering designers.

4. AFFT can be used to organize functions and trace root cause

Abstract function family tree clearly shows the relationships among nodes and functions in each level. Furthermore, it also a useful tool to trace root cause of the designed product.

The procedures described in this chapter are summarized in a "pamphlet" form in Appendix B which has been designed for a practicing engineer to use.

# CHAPTER FIVE

# CONCLUSION AND FURTHER RESEARCH WORK

## 5.1 Conclusion

In this thesis, the author is working toward building a new function model that could be used to assistant engineering designers in conceptual design stage. The deliverables of this work are outlined below:

### 5.1.1 A new interaction based function model (Section 3.1 - 3.4)

A new function model (IFM), which is based on the interactions of basic elements – mass, energy and information, is introduced. Its inherent generality can be used as a communication platform that make it possible to integrate seemingly conflicting research achievements developed by current researchers in function modeling domain (details can be found in Section 3.9).

### 5.1.2 A new method to represent design element – node (Section 3. 3)

A node contains a physical material and three communication channels which are mass, energy and information. It is represented in the form of *frame* and named by VOP. Function occurs when two or more nodes interact via their communication channels.

### 5.1.3 A new function organization tool – AFFT (Section 3.4)

Abstract function family tree (AFFT) can be used to organize functions in a design project. It clearly shows the relationships among nodes and functions in each level. Furthermore, it also can be used to trace root cause of the designed product.

### 5.1.4 A new engineering design pamphlet (Appendix B)

The author has also developed a concise pamphlet (details can be found in Appendix B) which is used to guide designers to perform function modeling. The example in chapter 4 demonstrates that IFM has unique aspects comparing to other engineering design method, such as function reusability, explicit function decomposition process, high compatibility.

## 5.2 Further research work

### 5.2.1 Canonicalization of Function Types

Some of the open issues regarding the proposed representation of function types (Table 3.3) have already been discussed in Section 3.2.1. These issues indicate that some of the nine basic types could be derived from others. Determining if this is indeed the case will depend on further study of the domain of interest. Some of the issues include:

1. There are several infinitive phases that could be implemented as one of the nine basic function types. For example, when we record data on a compact disk via CD-writer, *To Write* is another E-I type function rather than *To Detect*. Some of the common characters among those *infinitive* phases within the same function types are worth exploring.
2. Some *infinitive* phases can be implemented in more than one cell. For example, *To Transfer* is suitable for M-M type (transfer goods from one place to another), I-I type (transfer from one computer to another via internet) and E-E type (electric energy transfer from power plant to family). It may be that these common functions represent

an "orthogonal" but not inconsistent organizing mechanism. This is an interesting issue that needs to study.

3. Another point of interest is the difference between function types that lie on the diagonal in Table 3.3 and those that lie off it. The diagonal types appear to be those that affect the characteristics of a thing without affecting its basic identity (e.g. mass remains mass). The function types off the diagonal seem to be those that can change both characteristics and basic identity. The implications of this observation are not clear at this time and further investigations are expected.

### 5.2.2 Relations between Function Types

Relationships exist between the function types in Table 3.3. For example, *move* functions will typically require *power* functions to provide the energy for motion. The categorization of these relationships could be used in the development of tools to support function-based reasoning tasks (e.g. automatic function analysis, case-based reasoning).

Although this function modeling method is not fully matured at present, it does outline a new interaction based function model which aims to facilitate engineering designers in their conceptual design stage. A pamphlet (in Appendix B) has been put forward by the author. It is reasonable to believe that IFM will contribute to improve design efficiency, formalize information flow, and organize design databases in the conceptual design stage when it is developed further.

# References


1. F. A. Salustri, 1995, An Artifact-Centered Framework for Modeling Engineering Design, proceedings of the *1995 International Conference on Engineering Design* (ICED '95), Prague, Czech Republic, pages 74 - 79.

2. F. A. Salustri, 1996, A Formal Theory for Knowledge-Based Product Model Representation, *2nd IFIP WG 5.2 Workshop on Knowledge Intensive CAD,* Carnegie-Mellon University, Sep 16-18, 1996; pages 59-78, Chapman & Hall.

3. F. A. Salustri, 1998, Function Modeling for an Integrated Framework: a Progress Report, Proc. 11th Florida Artificial Intelligence Research Symposium, special track on Reasoning about Function, May17-20, 1998, D. Cook (ed), pages 339-343.

4. Yoshinobu Kitamura and Riichiro Mizoguchi, 1998, Functional Ontology for Functional Understanding, Working Papers of Twelfth International Workshop on Qualitative Reasoning (QR-98), Cape Cod, USA, May 26-29, AAI Technical Report WS-98-01, AAAI Press, pp.77-87, 1998

5. Luca Chittaro, Carlo Tasso and Elio Toppano , 1994, Putting Functional Knowledge on Firmer Ground, *Applied Artificial Intelligence*, 8:239-258

6. Jack Hodges, 1995, Functional and Physical Object Characteristics and Object Recognition in Improvisation, *Computer Vision and Image Understanding*, Vol.62, No. 2, September: 147-163

7. Anne M. Keuneke , 1991, Device Representation, *IEEE Expert*, April: 22-25

8. Munehiko Sasajima, Yoshinobu Kitamura, Mitsuru Ikeda and Riichiro Mizoguchi, 1995, FBRL: Function and Behavior Representation Language, *Proc. Of IJCAI-95*, 1830-1836

9. B. Chandrasekaran and John R. Josephson, Representing Function as Effects: Assigning Functions to Objects in Context and Out, Working Notes of the AAAI-96 Workshop on Modeling and Reasoning with Function, August 4, 1996, Portland, OR.

10. B. Chandrasekaran and H. Kaindl, Representing Functional Requirements and User-System Interactions, AAAI-96 Workshop on Modeling and Reasoning about Function, Portland, OR, August 1996.

11. Yumi Iwasaki, Marcos Vescovi, Richard Fikes, and B. Chandrasekaran , 1995, A Causal Functional Representation Language with Behavior-based Semantics, *Applied Artificial Intelligence* 9: 5 – 31

12. Ashok Goel, 1992, Representation of Design Functions in Experience-Based Design, *Intelligent Computer Aided Design*, D. Brown, M. Waldron, and H. Yoshikawa (editors), pp. 283-308, Amsterdam, Netherlands: North-Holland, 1992.

13. Ashok K. Goel and B. Chandrasekaran, 1989, Functional Representation of Design and Redesign Problem Solving, *Proc. Of IJCAI-89*, 1388-1394

14. Amaresh Chakrabarti, 1993, Towards a Theory for Functional Reasoning in Design, International Conference on Engineering Design, ICED'93, The Hague, August 17-19

15. Baosheng Chen and Chia-Hsiang Menq, 1992, Initial Attempts on the Characterization of Function Requirements of Mechanical Products, *PED-Vol. 59, Concurrent Engineering*, ASME 1992

16. Luca Chittaro and Amruth N. Kumar, 1998, Reasoning about Function and its Applications to Engineering, *AI in Engineering Journal*, in press.

17. Lena Qian and John S. Gero, 1996, Function-behavior-structure Paths and Their Role in Analogy-based design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 10: 289-312

18. Yasushi Umeda, Masaki Ishii, Masaharu Yoshloka, Yoshiki Shimomura, and Tetsuo Tomiyama, 1996, Supporting Conceptual Design Based on the Function-Behavior-State Modeler, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 10: 275-288

19. Varol Akman and Mehmet Surav, 1996, Steps toward Formalizing Context, *AI magazine*, Fall: 55-72

20. McCarthy, J. 1987, Generality in Artificial Intelligence, *Communications of the ACM* 30(12): 1030-1035

21. Guha, R. V. 1993, Context Dependence of Representations in CYC, Technical Report CYC 066-93, Microelectronics and Computer Technology Corporation, Austin, Texas.

22. Buvac, S., and Mason, I. A 1993, Prepositional Logic of Context, In Proceedings of the 11th National Conference on Artificial Intelligence, 412-419. Menlo Part, Calif.

23. John V. Liggett, 1993, Dimensional Variation Management Handbook, A guide for quality, design and manufacturing engineers, Prentice Hall

24. Edward A, Silver, David F, Pyke and rein Peterson, 1998, Inventory Management and Production Planning and Scheduling, John Wiley & Sons.

25. Sattiraju Prabhakar and Ashok Goel, 1997, Addressing Incompleteness of Device Models by Adaptable Function Modeling of Devices for Operating Environments. To appear in *Artificial Intelligence in Engineering*. Elsevier Applied Science.

26. Longman World Publishing Corp. 1987. Longman Dictionary of Contemporary English.

27. Rodenacker, W. 1971. Methodisches Konstruieren. Berlin: Springer-Verlag.

28. Achinstein P, 1983, *The Nature of Explanation*, New York, Oxford, Oxford University Press.

29. Morten Lind, 1994, Modeling Goals and Functions of Complex Industrial Plants, *Applied Artificial Intelligence*, 8: 259-283

30. Salustri, F. A. and Lockledge, J. C. 1999. Towards a Formal Theory of Products Including Mereology. To appear, Proc. 12th Int'l Conf. on Engineering Design.

31. R. Davis, H. Shrobe, and P. Szolovits, What is a Knowledge Representation? AI Magazine, 14(1):17-33, 1993.

32. Amaresh Chakrabarti and Thomas P. Bligh, 1994, An Approach to Functional Synthesis of Solutions in Mechanical Conceptual Design. Part I: Introduction and Knowledge Representation, *Research in Engineering Design* (1994) 6:127-141

33. G. Ermer, E. Goodman, R. Hawkins, J. McDowell, R. Rosenberg, and J. Sticklen, 1993, Steps Toward Integrating Function-based Models and Bond-graphs for Conceptual Design in Engineering, DSC-Vol. 47, Automated Modeling for Design, ASME 1993

34. B. Yang and F. A. Salustri, Function modeling based on interactions of mass, energy and information, Proc. 12th Florida Artificial Intelligence Research Symposium, special track on Reasoning about Function, May 3-5, 1999

35. Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lori Alperin Resnick, LIVING WITH CLASSIC: When and How to Use a KL-ONE-Like Language; Principles of Semantic Networks: Explorations in the Representation of Knowledge, John F. Sowa, ed.; chapter 14, pages 401-456; Morgan Kaufmann Series in Representation and Reasoning, Morgan Kaufmann Publishers, San Mateo, 1991.

36. Ham P. Suh, 1990, The principles of Design, New York Oxford, Oxford University Press.

37. Atila Ertas and Jesse C. Jones, 1996, The Engineering Design Process, 2nd Edition, John Wiley & Sons, Inc

38. Barry Hyman, 1998, Fundamentals of Engineering Design, Prentice-Hall Inc

39. Li Hongzhi, 1996, Zhuan FaLun, FaLun Fo Fa Publishing Co, Hong Kong

40. F. A. Salustri, 1999, Course note for 91-590-07: Knowledge aided design, http://salustri.esxf.uwindsor.ca/~fil/Courses/5:kad/1999w/lectures.html

41. Richard Koch, 1998, The 80/20 Principle : The Secret of Achieving More With Less

42. Bob Carpenter & Gerald Penn, 1994, The Attribute Logic Engine, User's Guide, Version 2.0.1, http://www.ltg.ed.ac.uk/projects/ledtools/ale/guide/guide.html

43. L. P. Hyvarinen, 1968, Information Theory for Systems Engineers, Springer-Verlag Berlin

44. Doede Nauta, Jr., 1972, The Meaning of Information, Mouton & Co. N. V., Publishers, the Hague

45. Stephen W. Littlejohn, 1996, Theories of Human Communication, 5th Edition, Wadsworth Publishing Company

46. Claude Shannon and Warren Weaver, 1949, The Mathematical Theory of Communication, Urbana University of Illinois Press

47. L. David Ritchie, 1991, Communication Concepts 2 – Information, Sage Publications

48. Pieter J. van Heerden, 1968, The Foundation of Empirical Knowledge, N. V. Uitgeverij Wistik – Wassenaar, the Netherlands

49. An electronic field-marked version of Webster's Revised Unabridged Dictionary Version published 1913 by the C. & G. Merriam Co. Springfield, Mass. Under the direction of Noah Porter, D.D., LL.D. This version is copyrighted (C) 1996, 1998 by MICRA, Inc. of Plainfield, NJ.

50. T. R. Gruber. 1992, Ontolingua: A mechanism to support portable ontologies. Stanford University, Knowledge Systems Laboratory, Technical Report KSL-91-66

51. T. R. Gruber.1993, Toward principles for the design of ontologies used for knowledge sharing. In Formal Ontology in Conceptual Analysis and Knowledge Representation, Nicola Guarino and Roberto Poli, editors, Kluwer Academic, in preparation. Original paper presented at the International Workshop on Formal Ontology. Available as Stanford Knowledge Systems Laboratory Report KSL-93-04.

52. Clive L. Dym, 1994, Engineering Design, A synthesis of views, Cambridge University Press

53. Vladimir Hubka and W. Ernst Eder, 1996, Design Science, Springer

54. John R. Dixon and Corrado Poli, 1995, Engineering Design and Design for manufacturing, a structured approach, Field Stone Publishers, Conway, Massachusetts, U.S.A

55. Pahl, G. and Beitz, W., 1984, Engineering Design, Edited by Ken Wallace, the Design Council, London, England

56. M. E. French, 1992, Form, Structure and Mechanism, MacMillan, London

57. Masatosi Tamai, 1967, How to Build Functional Family Trees, *Value Engineering* 8:7

58. Masayoshi Tanaka, 1985, A Survey Concerning Target Costs, VE and Price Estimates in the Product Development and Design Process, Japan VE Association

59. Kaneo Akiyama, Andrew P. Dillon, Translator, 1991, Function Analysis, Systematic Improvement of Quality and Performance, Productivity Press, Inc

60. Glass, G.V & Hopkins, K.D, 1996, 3rd Edition. Statistical Methods in Education & Psychology. Allyn & Bacon

61. Mike Uschold, Martin King, Stuart Moralee and Yannis Zorgios, 1998, The Enterprise Ontology, The Knowledge Engineering Review, Vol. 13, Special Issue on Putting Ontologies to Use. http://www.aiai.ed.ac.uk/%7Eentprise/papers/activity/ontology19.html#topic20

62. A. Asano, K. Itoh, and Y. Ichioka, 1991, Analysis of Nonlinear Filters Using Inductive Inference, Tech. Rep. Osaka Univ. 41, 2056, 221-233

63. Richard Stone, 1993, Introduction to Internal Combustion Engines, Second edition, Society of Automotive Engineers, Inc.

64. Barry N. Taylor, 1995 Edition, Guide for the use of the International System of Units (SI), NIST Special Publication 811

65. Joseph A. Bockerstette and Richard L. Shell, 1993, Time Based Manufacturing, McGraw-Hill, Inc.

66. GM Ovonic web site, http://www.ovonic.com/gmovonic.html

67. Toramatsu Shintani, Takayuki Ito, and Katia Sycara "Multiple Negotiations among Agents for a Distributed Meeting Scheduler" . In Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS'2000)

68. Constraint Programming: In Pursuit of the Holy Grail, Bartak, R., in Proceedings of WDS99 (invited lecture), Prague, June 1999

## Appendix A: The total function types in an X-to-Y mapping

In Section 3.2, the total function types within a certain X-to-Y mapping category are given in table 3.2. In this appendix, a detail mathematical proof is given.

### A.1 The meaning of the symbols

X: The number of slots in BE1

Y: The number of slots in BE2

T (X, Y): Total function types in an X-to-Y mapping

UEC (Z): Unique element combinations in BE1 or BE2 with Z slots available.

ECP: Equivalent Combination Pair, such as (ME, EM)

### A.2 Equation (a)

$$T(X, Y) = \frac{1}{4} * (X+1) * (X+2) * (Y+1) * (Y+2) \qquad (a)$$

### A.3 Proof

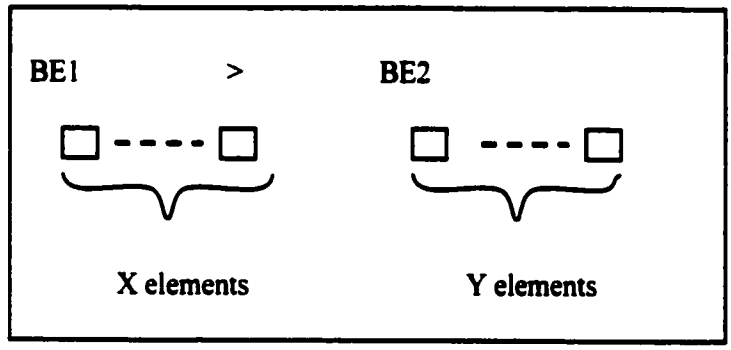
Figure A.1 shows the general representation form of X-to-Y mapping category. There are X slots in BE1 and Y slots in BE2. Each slot should be filled by one of the three basic elements. To calculate the total number of function types in a certain category, we should apply a two-step procedure,

(1) calculate the unique element combinations in BE1 and BE2

(2) apply equation (e).

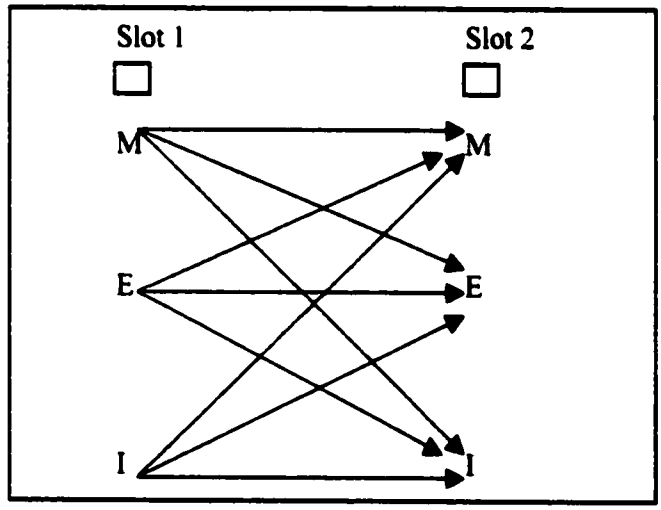
$$T(X, Y) = UEC(X) \times UEC(Y) \qquad (e)$$

Figure A.1 General function representation Form




*There are X slots in BE1 and Y slots in BE2. Each slot should be filled by one of the three basic elements. This is the general representation form for an X-to-Y function category.*

Figure A.2 Two slots in BE1




*There are two slots in BE1. Each slot has three candidates. The total combinations are MM, ME, MI, EM, EE, EI, IM, IE, and II.*

Section A.3.1 explains the meaning of unique element combinations UEC, and Section A.3.2 develops a formula for calculating UEC (Z). The final result is presented in Section A.3.3.

### A.3.1 Unique Element Combinations (UECs)

The author would like to introduce an example to demonstrate the concept of unique element combinations. Suppose there are two slots in BE1, since there are three candidates for each slot, the total number of combinations for these two slots is 9 ($3 \times 3$). These 9 combinations consist of an element combination set. Figure A.2 shows all possible combinations of these two slots.

There are three pairs of combinations which are (ME, EM), (EI, IE), and (MI, IM). Within each pair, the only difference is that the positions of the two elements are reversed. They cannot be treated as the different element combinations according to author's interaction-based function definition. This kind of pairing is called equivalent combination pair (ECP). An ECP can be eliminated if we keep only one copy of these two combinations.

To expand this observation, let N (M), N (E) and N (I) represent the number of element M, E, I in a certain combination respectively. Two element combinations are equivalent if and only if:

$N_1(M) = N_2(M)$, $N_1(E) = N_2(E)$, and $N_1(I) = N_2(I)$;

$N_1(M) + N_1(E) + N_1(I) = N_2(M) + N_2(E) + N_2(I) = Z$; (b)

In a certain element combination set, if there is no ECP, we call these combinations unique element combinations (UECs).

### A.3.2 The formula for calculating UEC

From equation (b), a mathematical algorithm can be developed to calculate the number of UECs, if the number of elements is given. The key concept is to count UEC by the number of each element. For example, if there are two slots in BE1, first of all, we count how many UECs are there for two M in these slots, then count UECs for one M in BE1, and at last zero M. This algorithm is listed below in the form of pseudo code. Within /* and */ pair, there are the annotations.

*Algorithm A.1*

```
UEC (Z) = 0;    /* initiate UEC (Z) = 0 */
For (N (M) = 0; N (M) = N (M) + 1; N (M) < Z)                    /*
outer loop when the number of element M < Z */
For (N (E) = 0; N (E) = N (E) + 1; N (E) < Z - N (M))            /*
inner loop when the number of element E is in the range from 0 to Z - N (M) */
UEC (Z) = UEC (Z) + 1;          /* one unique element combination counts */
End             /* inner loop ends */
End             /* outer loop ends */
```

We can see that the loop of element I is not presented in algorithm A.1, because when N (M) and N (E) are determined, the N (I) is determined automatically by equation (b). In another words, we can pick up any two of the three elements and apply *algorithm A.1*. The results are the same.

The UEC (Z) will be increased by 1 each time the inner loop (line 3 to 5) executes. So UEC (Z) equals the total number of the inner loop executed. For an given N (M), the inner loop runs (Z – N (M) + 1) times, so UEC (Z) can be calculated as below:

$$UEC(Z) = \sum_{N(M)=0}^{Z} (Z - N(M) + 1)$$
$$= \sum_{N(M)=0}^{Z} (Z + 1) - \sum_{N(M)=0}^{Z} N(M)$$
$$= (Z+1)^2 - Z(Z+1)/2 = (Z+1)(Z+2)/2 \qquad \text{(d)}$$

### A.3.3 The final result

According to equation (e),

$T(X, Y) = UEC(X) \times UEC(Y)$

$= (X+1) * (X+2)/2 * (Y+1) * (Y+2)/2$

$= ¼ * (X+1) * (X+2) * (Y+1) * (Y+2)$

So equation (a) has been proven.

## Appendix B: An Engineering Design Pamphlet

This pamphlet is designed to assist engineering designers in conceptual design stage. Figure B.1 is a proposed conceptual design procedure. A step-by-step design guide in each loop is provided below.

- **Apply *needs analysis***

Figure B.2 depicts the widely used "house of quality" matrix [65]. It can help designers relate customer needs to product features. Then these *customer requirements* should be translated into verb/object pairs. For example, the first row of this matrix can be represented by *increase spark voltage.*

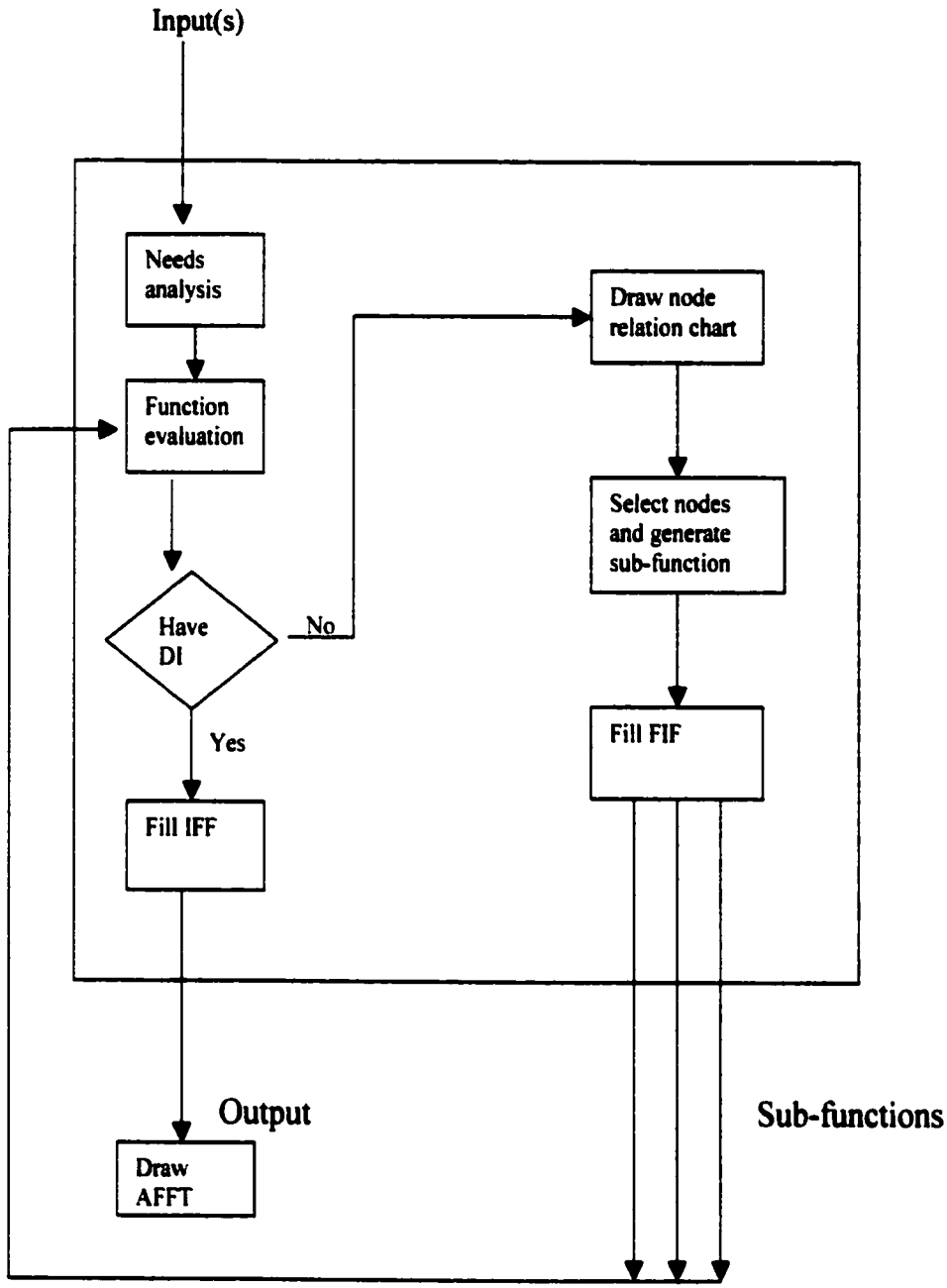
- **Function evaluation**

Functions generated from *needs analysis* should be evaluated to see whether it could be implemented directly. We can search either the design knowledge base by the function name or designer's own experience. If it is implementable, an implementable function form is filled and the functional design process for this particular function is over. If it is not, go to next step.

- **Draw node relation chart (NRC)**

Figure B.3 is a sample *node relation chart* for hand dryer. Output node (ON) is derived from VOP which is generated from needs analysis step. We put ON on the left most of NRC and explore its facets along mass, energy and information directions. According the explored facets of ON, corresponding facets of input nodes (IN) can be constructed and deployed in the causal order or space order.

Figure B.1 Functional design loop




*DI – direct implementation; IFF – implementable function form; FIF – function identification form. Outputs are the sub-functions of $F_x$ whose detail information is recorded in FIF. Inputs are customer requirements or the sub-functions of adjacent higher level that do not have direct implementation.*

Figure B.2 The House of Quality Matrix

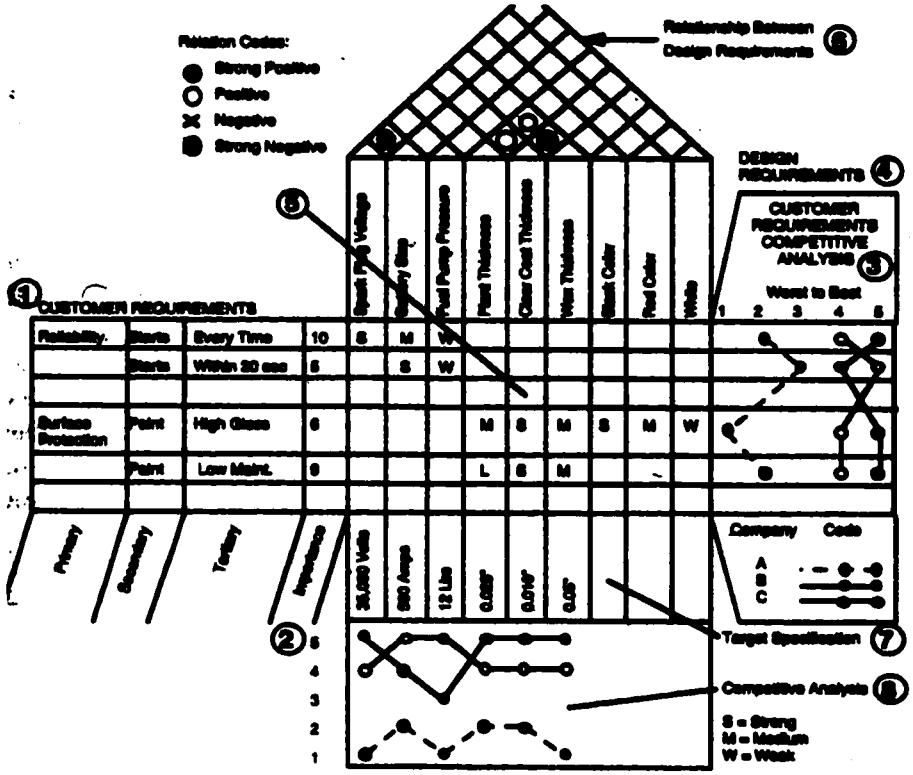



Figure 3-11. House of quality matrix. (American Supplier Institute 1987)

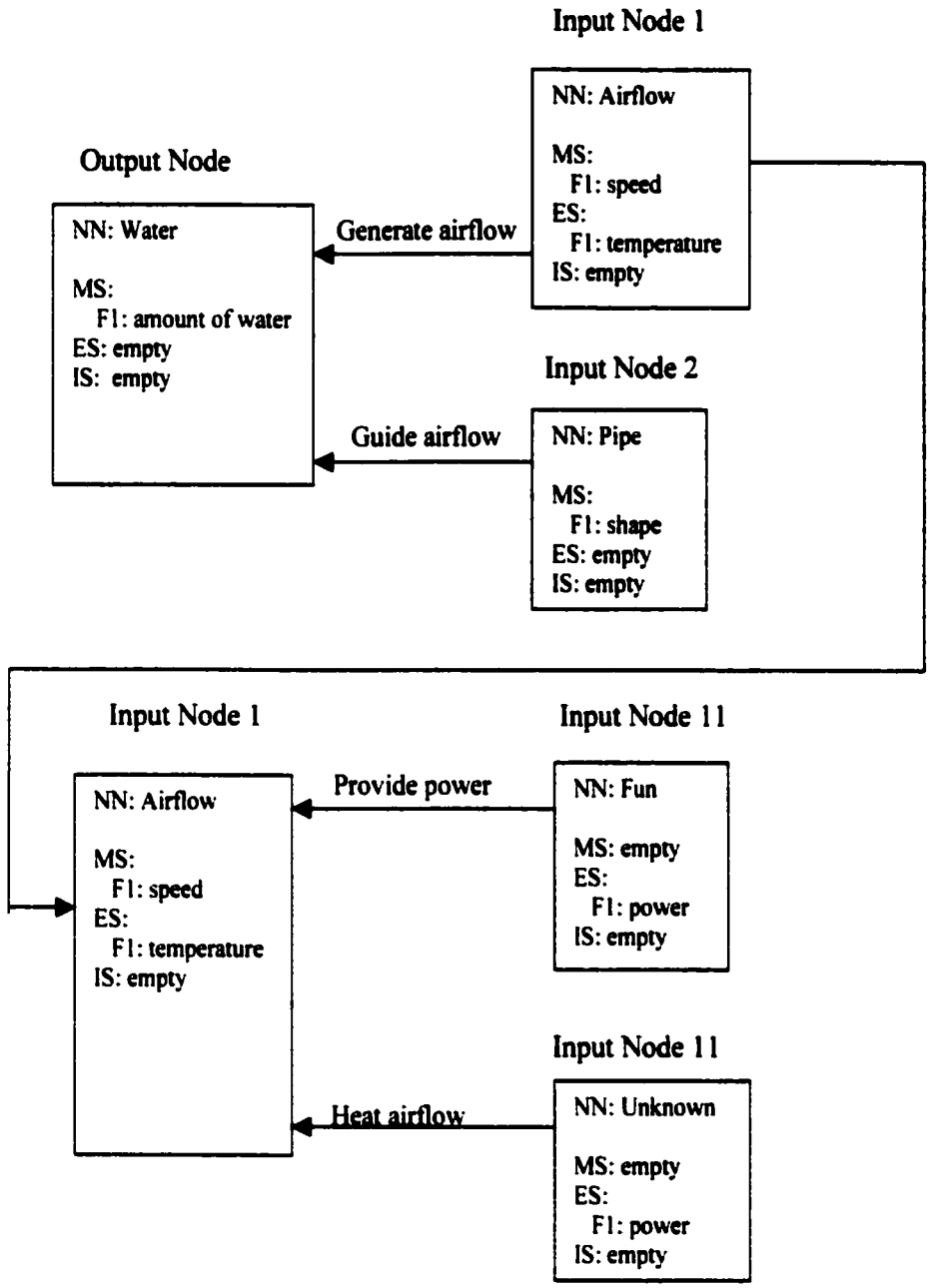
*The "house of quality" matrix [65] can help designers relate customer needs to product features. Then these* customer requirements *could be translated into verb/object pairs. For example, the first row of this matrix can be represented by* ***increase spark voltage.***

Table B.1: FIF for $F_X$

Label:_____ Date:_____

| Attributes | Content |
|---|---|
| Why link | |
| Input descriptor | |
| Output descriptor | |
| Function type | |
| Function name | |
| Value | |
| How link | |

Figure B.3 Node Relation Chart (NRC) for hand dryer




*NN – Node Name; MS – Mass Slot; ES – Energy Slot; IS – Information Slot.*

- **Select nodes and generate sub-functions**

NRC provides a rich set of proposed functions that contribute to satisfy customer needs. Because of the limited resources, designers can only implement some of them. As introduced in "the house of quality matrix" [65], an *importance factor* can be used to select functions.

Sub-functions are generated by exploring interaction facets among input and output nodes then labeling these interactions by VOPs. For example, in Figure B.3, input node 1's M and E facets and input node 2's M facet interact with output node's M facet. So top-level function (dry water) is decomposed into two functions (generate airflow and guide airflow). Similarly, *generate airflow* is decomposed into *provide power* and *heat airflow*.

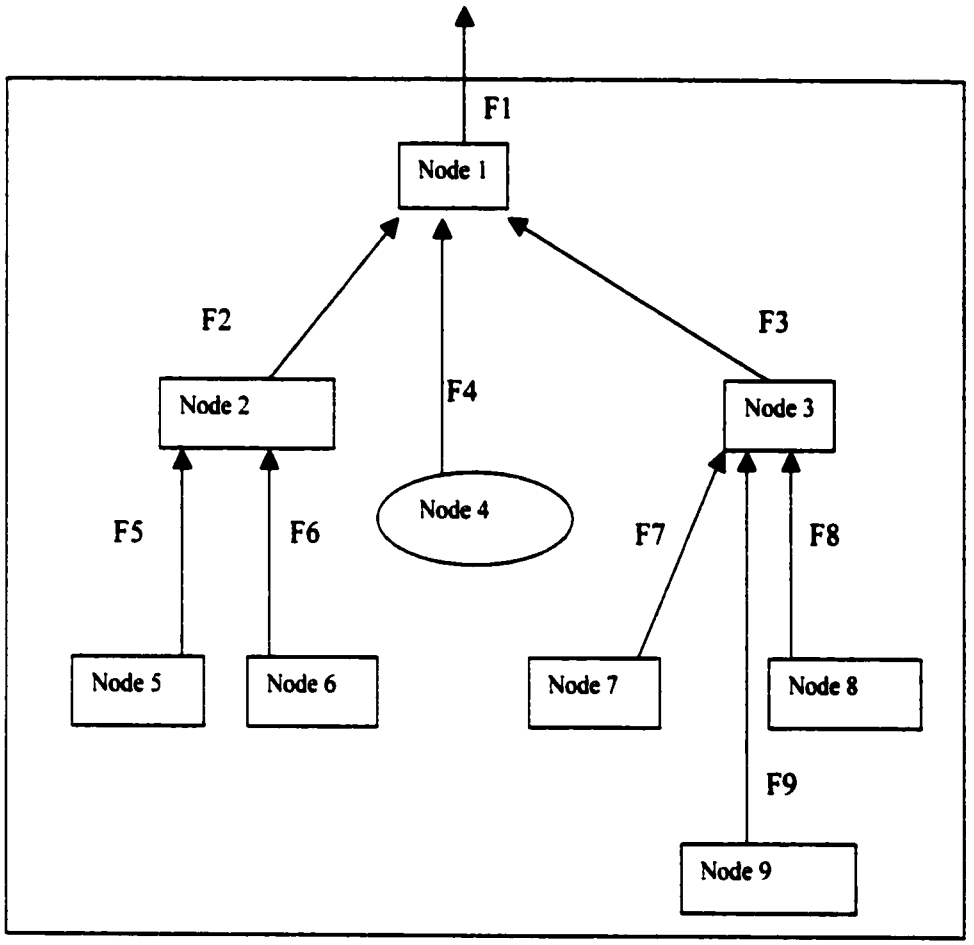
- **Fill FIF**

In this step, some function identification forms need to be filled. They are used to organize and document all information about the functions constructed until now. FIF is also used as the start point of the next design loop. An empty FIF is shown in table B.1.

- **Draw Abstract Function Family Tree**

When the entire functional design procedure is over, an AFFT (Figure B.4) can be organized to represent the overall relationship among nodes. It can be used as the foundation for FMEA (Failure Mode Effect Analysis), which is a widely used and highly concerned part of product design.

Figure B.4 Abstract Function Family Tree




*AFFT represents the entire node relationship of a product. It also can be used as the foundation Failure Mode Effective Analysis.*

# VITA AUCTORIS

| | |
|---|---|
| NAME | Bo Yang |
| PLACE OF BRITH | Beijing, China |
| YEAR OF BRITH | 1971 |
| EDUCATION | Beijing Polytechnic University, Beijing, China<br>1990-1994 B.Sc. |
| | University of Windsor, Windsor, Ontario<br>1998-2000 M.A.Sc. |