# A ROADMAP FOR DECOMPOSITION: ACTIVITIES, THEORIES, AND TOOLS FOR SYSTEM DESIGN

by

**DERRICK TATE**

B.S. in Mechanical Engineering (BSME),
Rice University
May 1992

S.M. in Mechanical Engineering (SM),
M.I.T.
June 1994

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 1999



Signature of Author _____
Department of Mechanical Engineering
December 28, 1998

Certified by _____
Nam P. Suh
The Ralph E. and Eloise F. Cross Professor of Manufacturing and Head
Department of Mechanical Engineering
Thesis Advisor

Accepted by _____
Ain A. Sonin
Professor of Mechanical Engineering and
Chairman of the Departmental Graduate Committee
Department of Mechanical Engineering

# A ROADMAP FOR DECOMPOSITION: ACTIVITIES, THEORIES, AND TOOLS FOR SYSTEM DESIGN

**by Derrick Tate**

Submitted to the Department of Mechanical Engineering
on December 28, 1998
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

## Abstract

Many design theories lack scalability to systems with many elements. They provide guidance to designers about specific facets of a design task but are too cumbersome to apply thoroughly from conceptual to detailed design. Thus the opportunity for rational design is missed.

Axiomatic design (AD) seems ideal for directing the design of large systems because it proposes general principles and a recursive design process. AD provides a fundamental basis for understanding decision making during design. It contains representations for the design object (a hierarchy of functional requirements, design parameters, and design matrices) and the design process (decomposition and zigzagging) combined with rules for decision making (the independence and information axioms). Challenges remain, however, in implementing the theory to large system designs.

The challenge addressed in this thesis is decomposition, the activity through which details of the design emerge. In decomposition, FRs satisfied by conceptual DPs are reduced into sets of sub-FRs. This activity is repeated until the design is completely embodied as a physical system of hardware and/or software. A question asked about decomposition concerns consistency: How can decisions about sub-FRs, and the rest of the decomposed design, be made so that they match the design decisions and representations of the design that were made at higher levels of the design hierarchy? To enable designers to do this, this thesis extends AD by providing a model of the decomposition process that identifies the activities performed and provides guidelines and tools to assist the designers.

The designers' goals and guidelines for achieving them have been generalized and evaluated for each of the decomposition activities: generating sub-FRs, identifying relevant customer needs, integrating sub-DPs, directing progress of the decomposition, dimensioning DPs, layout of DPs, carrying down and refining constraints, and ensuring consistency between levels. The theoretical concepts, the model, guidelines, and tools, have been validated through application to industrial cases including a new reticle management system accommodating customer variety, software algorithms for machine control, system analysis, reuse of design rationale, and software design. The cases examined cover a wide breadth, are from different fields, and have different numbers of designers.

Thesis Supervisor: Nam P. Suh
Title: Ralph E. and Eloise F. Cross Professor of Manufacturing
Head of the Department of Mechanical Engineering

If he had said this to me two weeks ago I would have popped him in the mouth, six weeks ago I would have broken his arm, four months ago and I would be telling this story to my death-row prison mates. Today all I could do was laugh at him.—Eddy L. Harris, *South of Haunted Dreams*, p. 244

On occasions when [Robert E.] Lee and General Francis Smith, Superintendent of VMI, marched together in some joint exercise, Lee consistently marched out of step. A faculty member recalled, "He simply walked along in a natural manner, but although this manner appeared so natural, it seemed to me that he consciously avoided keeping step, so uniformly did he fail to plant his foot simultaneously with General Smith or at the beat of the drum." —Emory M. Thomas, *Robert E. Lee*, p. 396

## Table of contents

# A ROADMAP FOR DECOMPOSITION: ACTIVITIES, THEORIES, AND TOOLS FOR SYSTEM DESIGN

# CHAPTER 1: INTRODUCTION

## *1.1 Motivation*

Why is design theory important? What issues does it cover? Is it useful for real-life applications? This introduction places the research presented in this thesis into the larger context of design theory in general by addressing these questions.

### 1.1.1 Needs of designers in industry

There are several different objectives that companies have in applying design theories and methods to their product development processes, including these:

- reduce use of product development resources (time, money, etc.)
- improve product functionality
- improve product reliability
- reduce product life-cycle costs
- reduce manufacturing cycle time

Better product design processes lead to shorter development times and lower costs as described by a typical *Wall Street Journal* article that discusses the efforts of Japanese and American auto makers to reduce development times [Reitman and Simison (1995) p. B1]:

> Shortening development time has become an increasingly important goal for the world's auto makers. Not only do faster development times cut costs by saving overhead and using engineering and production resources more efficiently, they make it possible for auto makers to get cars to market before trends and customer tastes change.

Companies continually strive to improve their performance and capabilities in product design, and product life cycles are ever shortened. Thus, the goals which companies set are not, nor should they be, fixed for all time. What constitutes a competitive performance in terms of development time today can be out-of-date tomorrow. A company's performance—and survival—are in part determined by its ability to rapidly introduce new products to meet changing market demands.

### 1.1.2 Impact of design theory in industry

To address the needs of industry, academia has been developing the research area of design theory. The areas of fundamental knowledge which are covered within design theory are abstracted as these:[1] the design process, the design object, designers, resources (time, money), and specific field knowledge.

---

[1] The *design process* is the set of activities whereby designers develop and/or select the means to achieve a set of objectives, subject to constraints. The design process may entail the creation of a new solution, the selection of an existing solution, or a combination of the two. A series of activities are performed by which the customers' perception of a design task is transformed into an output—the *design object*, which is any satisfactory solution to this task. The transformation occurs by means of designers working with design tools/methods, with their knowledge of discipline-specific information, and with a set of available resources.

When forming overall questions to be answered by this field, several broad questions can be considered, including the following:

1. How should the products be designed within the company? What activities are required? What organization should be used? What theories, methods, and rules are useful?
2. Who should design the products within or outside the company (vendors, universities, etc.)?
3. What products should the company design, given the context described by the current company goals, resources, and competencies? When should the products be designed?

The scope of this thesis falls within the first question: identifying theories, rules, and methods for specific design activities. This thesis, however, builds on a scientifically based theory of design which can be applied in answering *each* of the above three questions.

Unfortunately, design theory as a whole has been relatively unsuccessful at gaining acceptance within industry[2] or in achieving consensus within academia. One contributing factor is that researchers are not proposing broad, generalizable answers.[3] As noted by Ullman, "There are very few researchers in the United States who are trying to find a basic, philosophical, underlying science of design. This search for an idealized model has been overshadowed by the development of domain and phase dependent [theories] as a result of the interest of the financial sponsors." [Ullman (1991) p. 798]

## *1.2 Background and context*

While some researchers have noted that a research community in design theory does exist, the status of the field of design theory has been termed a *pre-theory* (or *pre-paradigm*) *stage*. For example, Dixon says, "researchers in engineering design theory...constitute a single *goal-directed research community*", yet "[t]here is much yet to be learned and formalized before we may say that a scientific theory and foundation of principles exists for engineering design." Thus, he concludes, the field is "in a pre-theory stage". [Dixon (1987) p. 146, 147] Likewise, in Ullman's opinion, "there is not yet a specific theory to describe. It is still evolving...Design theory research is in the pre-theory stage. There is still a search for the basic vocabulary and building blocks of a theory". [Ullman (1991) p. 794, 800] This is because, as the discussion at a workshop concluded, the field of design theory "lacks a clear agreement about the goals to pursue, a shared research methodology, and a broad theoretical framework to relate the findings of isolated pieces of research to one another." [(1992) p. 213]

### 1.2.1 Paradigm for design theory

The lack of an accepted set of research methods is the crux of the issue. The fragmented nature of design theory research is not due to a disagreement about the overall goal of design

---

[2] For use of design theory in industry see [Araujo, et al. (1996)], [Fulcher and Hills (1996)], [Fredriksson (1994)], and [Fredriksson, et al. (1994)].

[3] The reader is referred to [Cross (1993), Evbuomwan, et al. (1996), Hubka and Eder (1992), Karandikar and Shupe (1995), Tate and Nordlund (1995), Wilson (1980)] among others to understand the breadth of research within the field of design theory.

theory or a lack of communication channels, but rather due to differences in the means by which design researchers seek to reach that goal that have led to the current state of the field. For example, one paper asks,

> Is there convergence and agreement internationally on the research results to date?...What is the best methodology for design research?...Is a structure for research topics emerging?...Should design research be carried out in accordance with the discipline and methodology of orthodox engineering science? Or should it conform to research methodologies used in social sciences?...Or should design research stand on its own two feet somewhere in the middle—a new pure form of research or a hybrid? [Fulcher and Hills (1996) p. 184]

A unifying view of design researchers is needed. This view is termed a *paradigm*,[4] consisting of "accepted examples of actual scientific practice—examples which include law, theory, application, and instrumentation together—[that] provide models from which spring coherent traditions of scientific research". [Kuhn (1970) p. 10] Thus, the framework for doing design research includes the identification of important design phenomena to address (that is, identifying which decisions or tasks of designers to aid, for example, how to choose the best concept), the selection of a means to address these phenomena, a history of past successes with other phenomena, and a language for communicating phenomena. Currently there are multiple, competing paradigms, each with its own objectives and research approach, as can be seen by examining the literature.[5]

## 1.2.2 Aim and scope of design theory

The needed view for unifying design must possesses several important characteristics: It must recognize the importance of rational decision making, identify the cognitive aims of designers, and provide rules for achieving these aims. It is argued in this work that a paradigm for design theory *is* available, and the view provided by this research tradition forms a foundation for this thesis. The basis for this view of design is provided by axiomatic design.[6]

### 1.2.2.1 Importance of rational decision making

In design, decision making is most important.[7] This is because designers must make many types of decisions: for example, the choice among various alternatives in order to create or select the best design, the development of a set of suitable requirements, the division of the work among several designers, etc.. Kitcher describes how to answer the question of whether a system is well designed: "We can make this question more precise by specifying

---

[4] *Paradigm* is the terms used by Kuhn, and I argue that this is fundamentally the same concept as Lakatos's *research program* or Laudan's *research tradition*—although none of the three makes the same argument himself.

[5] See [Tate and Nordlund (1995)] or appendix 3 for further discussion of competing paradigms within design theory.

[6] Axiomatic design as a paradigmatic basis for viewing design is discussed in section A1.6 in appendix 1.

[7] This fact is recognized researchers as diverse—and as removed from axiomatic design—as Hazelrigg [Hazelrigg (1997)] and Andreasen [Andreasen (1991b)]. It must be noted, however, that Hazelrigg neglects the creative task of synthesizing the set of options from which the designers must make their choices.

the entity (or entities) whose design we are to investigate, the kinds of goals we bring into consideration, and the standard for good design." [Kitcher (1993) pp. 178-179] Therefore, in making their rational decisions, designers must choose from among the options they create, according to explicit or implicit rules, in order to meet their cognitive ends.

### 1.2.2.2 Cognitive aims of designers and scope of design theory

Different researchers consider the scope of design to be more or less broad. At the narrow end of the range, German researchers study the detailed activity known as *konstruktion*.[8] At the broad end, Clausing views the purpose of design research as synthesizing a "modern way of developing new products that will be competitive in the global economy. It combines the best engineering, the best management, the best strategy, and especially, the best teamwork. The resulting improvements are greatly reduced development time, a reduction in all costs, higher quality, and increased product variety." [Clausing (1994) p. 3] In-between these two extremes, the scope of Suh's axiomatic design has covered the following: "Design, as the epitome of the goal of engineering, facilitates the creation of new products, processes, software, systems, and organizations through which engineering contributes to society by satisfying its needs and aspirations." [Suh (1990) p. 5]

### 1.2.2.3 Rules for achieving the designers' aims

Some researchers believe that, to understand design and to develop theories, a distinction needs to be made between descriptive and prescriptive research.[9] The distinction between prescription and description is said to be that "[s]ome...models [of the design process] simply *describe* the sequences of activities that typically occur in designing; other models attempt to *prescribe* a better or more appropriate pattern of activities." [Cross (1994) p. 19]

The two, however, properly go together; they cannot be separated. As a result of identifying the goals, options, and rules used by designers in design, as described above, a robust methodology for design *will be established*. That is, "*one can show that a thoroughly 'scientific' and robustly 'descriptive' methodology will have normative consequences.*" [Laudan (1996) p. 133] This is because

> Once we recognize that methodological rules deal with the relationship between cognitive ends and means, we can recognize that it is an empirical question, not a matter of convention, which means promote which ends....Whether certain proposed methods do in fact promote certain ends is generally a...question about cause-effect linkages in the natural world. Methodological claims...are no less 'factual' than any claims made by natural or social scientists. [Laudan (1996) p. 18]

---

[8] Clausing calls this set of activities *partial design*: "The undergraduate engineering curriculum typically...includes one or two design courses. These concentrate on creative concepts and feasibility, the assurance of a first-order compatibility with the laws of nature. Let us call this *partial* design." [Clausing (1994) p. 5]

[9] [Dixon (1987), Finger and Dixon (1989)] are examples of this view.

### 1.2.3 Basis of axiomatic design

The foundation for this thesis work is axiomatic design theory.[10] The underlying hypothesis of axiomatic design is that there exist fundamental principles that govern good design practice.[11] It is a general theory of design which provides a time-tested, scientific basis for designers to make design decisions. It is a scientific approach to design, in which decisions are made at multiple levels of abstraction, starting at the system level and progressing in more detail until the design is completed. Axiomatic design theory can be applied recursively throughout the design hierarchy. Design problems are stated; solutions are proposed and analyzed; and decisions are made. The components that distinguish axiomatic design from other design theories are domains, hierarchies, zigzagging, and the two design axioms: independence and information.

## *1.3 Area of research: current theory and needs*

One reason that axiomatic design is a valid starting point is that it meets the criteria for a progressive research program.[12] This thesis may be considered then an extension of axiomatic design theory.

#### 1.3.1.1 Scope of decomposition and project control

The activity of focus in this thesis is *decomposition and project control*: that is, the process of defining and relating a set of sub-FRs to fulfill a parent FR-DP pair. It is an activity where the details of the object being designed emerge, in which conceptual FR-DP pairs are reduced to sub-FRs, in which sub-DPs are embodied and integrated, and that is repeated until the design is completely embodied.

Table 1-1 contrasts activities which are considered part of decomposition and those which are not. The distinction is that decomposition concerns those activities which relate multiple layers of the design hierarchy, for example, defining sub-FRs and physical integration of DPs. Other activities in design—those not part of decomposition—occur within each level of the design hierarchy, so they do not span multiple levels, for example, defining and selecting a set of DPs.

---

[10] See appendix 4 or [Suh (1990)] for a more detailed overview of the theory. They give insight about the way AD represents design tasks and its past successes.

[11] According to Suh, common features of the creative process that "permeates all fields of human endeavor" can be used to "distinguish between good and bad designs." [Suh (1990) p. 5]

[12] A progressive research program is expanding in the scope of phenomena that it explains (theoretically progressive), that its predictions correlate with empirical results (empirically progressive), and that anomalies and unclarities are being brought within the theory without being ad hoc (heuristically progressive). [Lakatos (1978), Larvor (1998)] This is discussed more in chapter 5 and especially in appendix 1.

**Table 1-1. Activities considered part of decomposition versus those that are not**

| Activities in decomposition | Activities at one level—not decomposition |
|---|---|
| • Generation of sub-FRs | • Synthesis of DPs |
| • Physical integration of DPs | • Comparison of DPs versus constraints |
| • Directing the progress of the decomposition: what to decompose next, when finished | • Analysis and selection of a design concept (choose a set of DPs using the 2 axioms) |
| • Dimensioning and configuration of DPs: how many of a particular module | • Decoupling of a coupled design |
| • Layout of DPs | • "Optimization": setting parameters of an existing, coupled design |
| • Carrying down and refining constraints | • "Tuning": setting DP parameter values of a decoupled or uncoupled design |
| • Ensuring consistency between levels: FRs, DPs, DMs, Cs | |
| • Identifying relevant CNs | |

Given the above activities that comprise decomposition—and assuming that a rational process can be abstracted that includes them—how can the results of decomposition (and the individual activities within it) be evaluated? Each activity can have its own specific criteria associated with it, but from a perspective that views the decomposition activity as a whole, the important consideration is *consistency*. By having lower-level decisions consistent with those made at higher levels, iteration in the design process is minimized, and good design decisions are made more quickly.

### 1.3.1.2 State of current theory and design practice

The aim of design theory is to improve the decision-making capability of the designers. The current state of the art is that there are certain types of questions that are asked about how decisions in design are to be made that design theory is able to answer. For example, given several alternative design concepts, the design axioms enable designers to rationally choose among them—even during early, conceptual stages of design. [Suh (1990)] There are certain other types of questions that are harder to answer. For example, how can the concept of a parent DP be detailed as a set of sub-functions? That is, as designers seek to make decisions, design theory is able to provide tools and guidelines for making some of those decisions; for others, however, current design theory is able to provide less input.

Many of the issues which have not been addressed in design theory (and specifically in axiomatic design) concern the subject of decomposition. This includes, in particular, the generation of sub-FRs, but it also includes also other decisions that the designers make that require that they have a view of the design object which includes the connections between the levels of the design hierarchy. *Decomposition* is the way that decisions and information

about the design object are organized during the design process. It is defined as the definition of a set of sub-FRs given a parent FR-DP pair, and it includes all the activities required to progress from one level of the design hierarchy to another.

The current process for decomposition is ad hoc. Axiomatic design theory provides tools and criteria to designers who are selecting between alternative designs proposed at one level of the design hierarchy;[13] however, the connections between the levels are rather tenuous.[14] The basic idea of this work is that *axiomatic design theory can be extended with descriptions of the activities that take place during design combined with decision-making tools and workflow paths to provide additional decision-making guidance to designers*. In particular, the activities articulated are primarily concerned with decomposition, an area which thus far has received little theoretical attention from a perspective that is consistent with axiomatic design.

**Example**

Before moving on with a vision of how to creation a rational basis for decomposition, consider the ad hoc nature of decomposition. This example concerns the design of an end effector to transfer parts (reticles) between different locations within a machine tool. The end effector is responsible for picking, holding, moving, and placing the reticles between 5 different locations within the machine tool. Suppose that at each location, the requirements on the design of the end effector are slightly different.

Axiomatic design describes the means to represent the different FRs that the end effector must satisfy over time. These may be represented by a series of vectors of FRs. A vector is stated for each discrete interval of time which has a unique set of requirements, or unique constraints. [Suh (1995b)]

$$\begin{Bmatrix} FR_1 \\ \vdots \\ FR_a \end{Bmatrix}_{t_1} \cdots \begin{Bmatrix} FR_1 \\ \vdots \\ FR_j \end{Bmatrix}_{t_i} \cdots \begin{Bmatrix} FR_1 \\ \vdots \\ FR_n \end{Bmatrix}_{t_z} \tag{1-1}$$

The designers who are responsible for the design of this end effector can look at its operation and describe a set of FRs for each time interval. Then appropriate sets of DPs can be selected for each, using the design axioms, and the design, hopefully, can be integrated in a manner that does not compromise the functional independence of the FRs that are to be satisfied at each time.

Now step back and consider the situation in which the designers are responsible not only for the design of the end effector, but also for the larger system in which it is to be located. Now they must consider this question: how *many sets of FRs* should this design satisfy? 5, 10, 11, 20? This is a question involving the relationship of the end effector design to the decisions

---

[13] The model proposed in [Wilson (1980)] describes the activities which are performed at one layer of the decomposition, namely, problem formulation, synthesis, and analysis.

[14] One idea that *has* been proposed concerning the relationship between the levels of the design hierarchies in axiomatic design is the definition of *leaf* and *non-leaf* DPs. (See, for example, [Suh (1998a)].) In considering the operation of the design object, it is the leaf DPs that are of primary importance; these leaves "sum together" to produce the desired system-level functionality. Therefore, the DPs at the intermediate levels must be primarily important during the design process itself. The nature of the intermediate-level DPs and the way in which they specifically provide guidance to the designers as the design progresses during decomposition has not been articulated.

that have been made at higher-levels of the design hierarchy, including, for example, the number of mechanisms, the number of end effectors within the machine, and how these are to be physically integrated.

A potential set of sub-FRs this end effector is given in table 1-2.

**Table 1-2. Decomposition of FR-DP3.3.3.1 "Handoff reticle to stage from first mechanism" using "End effector"**

| Index: 3.3.3.1 [-] | Functional Requirements (FRs) | Design Parameters (DPs) |
|---|---|---|
| P | Handoff reticle to stage from first mechanism | End effector |
| | *Description* | *Description* |
| 1 | Acquire reticle (to stage) | "Stage acquire" signal |
| 2 | Release reticle (with vacuum chuck) | Vacuum chuck geometry during <1st mechanism handoff, release> |
| 3 | Sense reticle (with vacuum) | Vacuum sensors during <1st mechanism handoff, sense> |
| 4 | Schedule 1st mechanism handoff | Command and control algorithm during <load, 1st mechanism handoff> |

In stating this set of sub-FRs, the designers need to answer questions including these:

- Does the design need to be decomposed further?
- If so, which FR-DP pair should be decomposed next?
- What are all the sub-FRs required to perform the parent FR, and how can the concept of the parent DP be detailed as a set of sub-functions?
- What constraints apply in choosing the sub-DPs?
- Can these sub-DPs be physically integrated with other DPs?

The design axioms provide criteria for selecting a set of DPs to meet a set of FRs. Once a set of DPs has been chosen at this level of the design hierarchy, the design process must progress from this point. How can a rational, systematic approach to decomposition—and physical integration, among other activities—shed light on the definition of these sub-FRs? How can these sub-FRs be determined from the parent-, or system-level design decomposition of which the end effector is a part?

Without a rational approach to decomposition, the definition of these sub-FRs is done in an ad hoc manner, and there is no guarantee that a necessary and sufficient set of FRs will be generated. Moreover, there is no guarantee that a good design will be created in a timely manner. It is argued here that *the statement of such sets of sub-FRs as these must be—and can be—done through a rational process for decomposition. This process for decomposition is the subject of this thesis.* Just like there are characteristics that distinguish good design decisions in selecting concepts, there are characteristics that distinguish good decisions in decomposition and physical integration.

## *1.4 Research goal of the thesis*

The design axioms are an effective tool for supporting designers in their design decisions at one layer of the design hierarchy, but axiomatic design theory needs to be further developed in supporting decision making which occurs across layers—going from higher to lower levels as prescribed by zigzagging during the synthesis of new designs.

Because the designers' decision making is the subject of design theory, the challenge is to create a model or a theory of the design process that is neither too broad nor too restrictive. That is, current models of the design process are either so broad that they vaguely describe the activities of any designer but do not provide assistance in real decision making, or the models are so restrictive that they cannot provide an accurate representation of actual design practice. The solution is to create a general model from which designers can structure a specific implementation of the design process. [Tate and Nordlund (1996)] This general model must be *specific* to provide useful guidance to the designers, yet it must be *flexible* to cover all instances of the design process.

For understanding and aiding decomposition, designers need a clearly defined set of activities with supporting rules and tools. The activities are not a rigid set of steps that must always be performed in a particular order. Rather, the model of decomposition consists of the possible activities with inputs, outputs, and connections between activities that have been clearly specified, and using this model, the designers can orient themselves. They can also then identify particular tools and theories to assist them in performing each of the activities. However, the model is flexible, showing the different options that the designers have in performing the set of activities: that is, what activities can be performed next? and what are the criteria for deciding which activity will be done next?

### 1.4.1 Research question

Thus, my overall research question may be stated as follows:

> *What is a process for decomposition—consisting of activities, tools, and theories—that empowers designers to make rational and consistent design decisions across multiple levels of the design hierarchy?*

### 1.4.2 Needs

The model of the decomposition process needs to describe how the activities given in table 1-1 fit together to yield good design decisions in decomposition. Each of the activities in this table needs to be detailed. The specification of the details of each activity include describing the activity, listing the types of questions the designers answer in performing the activity, and identifying clearly its inputs and outputs.

As described above, a model, or roadmap, is needed of decomposition which

- identifies the activities performed (in terms of explicitly defining their inputs and their outputs)
- provides tools and rules to assist the designers in performing the activities
- guides designers in the sequence of the design process by relating the flow of activities to one another

- places decomposition into the overall context of system design[15]

My vision for a theory to help designers is to *provide them with a design process which clearly articulates the activities that are performed* that are then supported by appropriate tools or guidelines in the same way that the design axioms provide a guideline to enable the task of choosing among alternative designs.

### 1.4.3 Scope and contributions of the thesis

While much work has been done examining individual designs at one level of the design hierarchy, in terms of selecting a set of DPs to satisfy a set of FRs by means of the design axioms, this thesis is unique in detailing decomposition (and by extension system design) and in providing tools for designers to perform these activities.

The specific challenge addressed in this work is that of decomposition, the activity through which the details of the design emerge. How can lower-level design decisions be related consistently to the decisions represented in design matrices at higher levels? While designers have done much work using axiomatic design, additional tools and guidance are needed for handling the design of larger systems. Specifically, tools are needed to guide designers in decomposition and other system design decisions (such as layout, numbers of DPs, physical integration, command and control, etc.).

The contributions of this thesis are means for enabling designers to deal with the complexity introduced in system design. This thesis provides extensions to AD in three areas:

- activities: The design process is modeled to accurately represent decomposition.
- guidelines: Criteria of good design decisions for decomposition activities are identified, and guidelines for achieving the designers' objectives are developed.
- tools: The changes of FRs and DPs over time during the operation of the design are defined and represented more clearly. Metrics of flexibility and modularity which can be incorporated into the AD measure of information content are defined.

The intent is to produce a comprehensive general theoretical basis to be used by designers during decomposition. The theoretical concepts are validated through application to cases from industry. The cases to which these ideas have been applied include the following: design of a tool-exchange mechanism which accommodates much customer variety, software control algorithms for machine control, reuse of design rationale for manufacturing cell design, and diagnosis of coupling in machines and suggestions for changes.

### 1.4.4 Structure of thesis

This thesis follows a structure parallel to that found in experimental theses. This is illustrated in table 1-3 which shows a generic thesis structure for a thesis concerned with advancing design theory and contrasts that with an experimental thesis (such as developing and modeling a new manufacturing process). A flowchart for this thesis is given in figure 1-1.

---

[15] This shows how the inputs and outputs describing the design object are refined as decomposition progresses and how the tools can change as the decomposition progresses through different levels of the design hierarchy.

**Table 1-3. Thesis outlines for experimental versus design work**

| Experimental thesis | This thesis in design theory |
|---|---|
| Introduction | Chapter 1: Introduction |
| Background | Chapter 2: Research question and theory extensions |
| Theoretical model/analysis | Chapter 3: Activities and theorems for decomposition |
| | Chapter 4: Tools for system design |
| Experimental design and setup | Chapter 5: Case study methods |
| Experimental results | Chapter 6: Case study results and discussion |
| Comparison of theory and results | |
| Conclusions | Chapter 7: Conclusions |

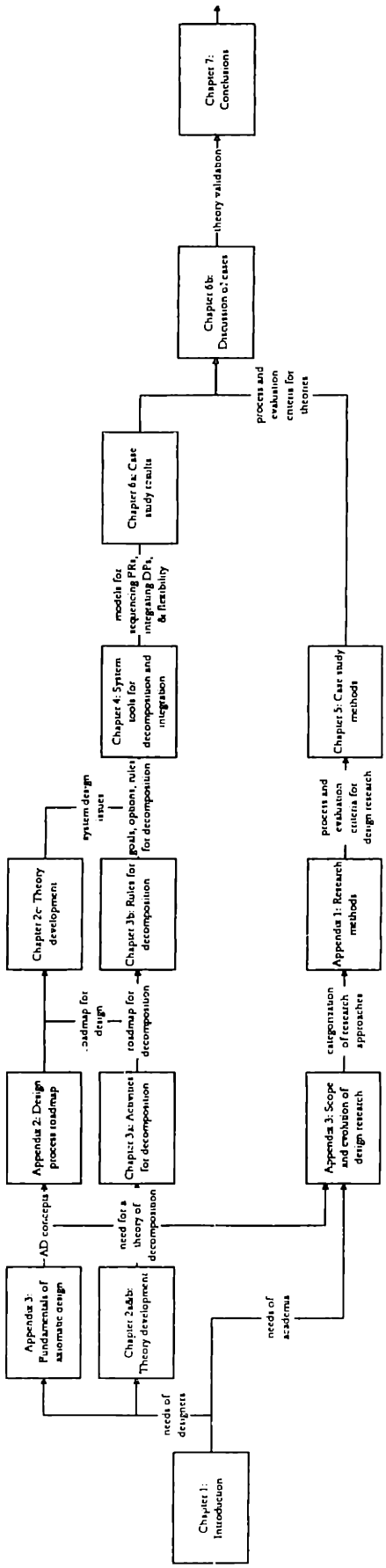


*Figure 1-1. Flow chart showing thesis structure*

# CHAPTER 2 THEORY DEVELOPMENT

## *2.1 Introduction*

This chapter motivates the theory presented in later chapters. Specifically it answers several issues that arise concerning the overall research question posed in chapter 1. That question is

> *What is a process for decomposition—consisting of activities, tools, and theories—that empowers designers to make rational and consistent design decisions across multiple levels of the design hierarchy?*

In developing an answer to this question, the characteristics of three areas need to be discussed: the design process, rules and decision-making criteria for designers, and additional field-specific tools. The discussion of these three areas motivates and provides direction for the contributions of the next chapters. In this chapter the following questions are answered:

1. What is known about the design process? Can existing models of the design process adequately capture the design decisions made during decomposition? Are they sufficiently thorough? Can they both describe the real-life design tasks of engineers and prescribe good design practice?
2. How do rules look that provide guidance to designers in terms of criteria for good decision making?
3. Does system design have characteristics that distinguish it from more simple component design? Can tools be developed to enable designers to deal with issues that occur in specifically system design? What tools are useful?

## *2.2 Vision*

As stated in chapter 1, my vision for a theory to help designers is to *provide them with a design process which clearly articulates the activities that are performed* and that are *supported by appropriate tools or theories* in *the same way that the design axioms enable designers to choose among alternative designs.* And in chapter 1, I argue that decomposition is an important area in design that is not adequately covered by current theory.

### 2.2.1 Consistency: a measure of success in decomposition

In performing decomposition, a measure of success is that of consistency. That is, do the sub-FRs, and the rest of the decomposed design, match the design decisions and the representations of the design that were made at higher levels of the design hierarchy? Good decision making in performing decomposition leads to design hierarchies that describe a design object at multiple layers of abstraction, but that *consistently describe the same design object.*

The concept of consistency is illustrated in equations 2-1 and 2-2. Equation 2-1 shows a portion of a design matrix that captures the relationships between the FRs and DPs at an arbitrary level of the design hierarchy. The elements of the design matrix indicate the effects of changes of the DPs on the FRs. A question which can be asked in evaluating the significance of the design matrix elements is this:

*Will a change in DPx.j that is consistent with fulfilling FRx.j affect the satisfaction of FRx.i? That is, does this change in DPx.j require a change in DPx.i?*[1]

$$\begin{Bmatrix} \vdots \\ FRx.i \\ FRx.j \\ \vdots \end{Bmatrix} = \begin{bmatrix} & \vdots & & \\ \cdots & Ax(i,i) & Ax(i,j) & \\ & Ax(j,i) & Ax(j,j) & \cdots \\ & & \vdots & \end{bmatrix} \begin{Bmatrix} \vdots \\ DPx.i \\ DPx.j \\ \vdots \end{Bmatrix} \tag{2-1}$$

Given a satisfactory solution to the design task at this level of the design decomposition, as evaluated by the design axioms, the designers progress to the next level of the decomposition as shown in equation 2-2. Now the question of consistency is raised.

A *consistent* decomposition is defined as one in which the lower level of the design hierarchy corresponds to the higher level. Equation 2-2 shows the decomposition of equation 2-1. In equation 2-2, *FRx.i-DPx.i* is decomposed into two sub-FRs: *FRi.1* and *FRi.2*. Likewise the *FRx.j-DPx.j* pair is decomposed into *FRx.j.1* and *FRx.j.2*. Once these sub-FRs have been formulated, sub-DPs are chosen to fulfill each of these. But are these lower-level decisions consistent with those made at the higher level?

$$\begin{Bmatrix} \vdots & & \\ FRx.i\begin{Bmatrix} FRx.i.1 \\ FRx.i.2 \end{Bmatrix} & & \\ FRx.j\begin{Bmatrix} FRx.j.1 \\ FRx.j.2 \end{Bmatrix} & & \\ \vdots & & \end{Bmatrix} = \begin{bmatrix} & \vdots & & \\ \cdots & Ax(i,i) & \begin{matrix} Ax(i1,j1) & Ax(i1,j2) \\ Ax(i2,j1) & Ax(i2,j2) \end{matrix} & \\ & \begin{matrix} Ax(j1,i1) & Ax(j1,i2) \\ Ax(j2,i1) & Ax(j2,i2) \end{matrix} & Ax(j,j) & \cdots \\ & & \vdots & \end{bmatrix} \begin{Bmatrix} \vdots & \\ DPx.i\begin{Bmatrix} DPx.i.1 \\ DPx.i.2 \end{Bmatrix} & \\ DPx.j\begin{Bmatrix} DPx.j.1 \\ DPx.j.2 \end{Bmatrix} & \\ \vdots & \end{Bmatrix} \tag{2-2}$$

The model of the design object represented at one level of the design hierarchy presents one picture, or gives one set of answers, about the design object. Either these answers are the same, that is, consistent, with the answers given at the next (or another) level of the design hierarchy, or these answers are different, that is, *inconsistent*, resulting in inaccuracies in the connections between levels of the design hierarchy.

In looking at the above equations, it can be seen that there are three fundamental ways in which inconsistencies can be manifest. These are inconsistencies in

- sub-FRs
- sub-DPs
- the design matrix elements

In addition to those shown in the above design equations, other inconsistencies can arise due to the misapplication of constraints (Cs) across levels.

These inconsistencies can be described by the following:

---

[1] We want to be able to ask questions of the model of the design to be able to propose and evaluate different design alternatives. Furthermore, these alternatives are of different types depending on the portion of the hierarchy that the designers are considering. Changing the lowest-level DPs, "leaf-DPs", corresponds to setting parameter values; changing higher-level, "non-leaf DPs", corresponds to changing details of that DP, that is, its intermediate sub-FRs.

- inconsistent FRs: Sub-FRs do not provide the functionality as described by the parent FR-DP pair. Either there is an additional, unnecessary sub-FR or the desired parent FR is not satisfied or the parent FR is satisfied in a way other than as described by the parent DP.
- inconsistent DPs: Sub-DPs do not provide sufficient capacity or have been physically integrated in a way that violates the functional independence indicated at the parent level. Alternatively, the DPs at the parent level were said to satisfy the constraints as described at that level, yet the sub-DPs do not satisfy the Cs as applied to them.
- inconsistent DM: The higher-level design matrix indicates no relationship exists between an FR-DP pair, and at least one of the corresponding sub-DM elements indicates that there is a relationship. Alternatively, as for an on-diagonal element, the higher-level design matrix indicates the presence of a relationship between an FR-DP pair, and none of the corresponding sub-DM elements indicates that there is a relationship.
- inconsistent Cs: The sub-DPs do not meet the Cs carried down from their parents.

These are the types of inconsistencies that can occur. The objective of this work is *to understand the decomposition process and provide tools so that a correct decomposition process is performed and such inconsistencies are prevented.*

### 2.2.2 Note on software implementation

It should be noted briefly that one application of a theory for decomposition is that of enabling the creation of better design software. A roadmap of design activities that includes decomposition provides a rational basis for determining the location of the designers in the design process. By clearly identifying the inputs and outputs of the activities, the software is able to track the effectiveness in performing each activity because each has a clearly defined starting and end point. Furthermore, tools and theories can be provided to the designers *in accordance with the specific activity they are performing and where they are in the decomposition.* So, the use of theories and tools will be more focused on the decisions in which they provide the most assistance. Lastly, areas for new theory or tool development will be identified. This assessment will be based on identifying both where designers need help and where applicable tools or theories do not exist.

## *2.3 Understanding the design process*[2]

The first step in understanding the role of decomposition is to examine how it fits within the design process. Then a model which details the decomposition-specific activities can be generated.

[2] Parts of this section are adapted from [Tate and Nordlund (1996)], which will also appear as [Tate and Nordlund (1998)].

### 2.3.1 Desired characteristics of the design process

This section looks at the important characteristics of the design process that must be included in a model in order to enable designers to make project plans which effectively use all available resources, methods, and tools. These characteristics must be included in the model in the sense that the model must explain, predict, or allow for explanation or prediction based on these characteristics.[3] The characteristics are

1. Decision making. The purpose of the design process is to make decisions, specifically to find a solution in terms of a design object to some design task. Thus, clearly defined decision points and decision criteria, and/or rules, must be visible in the process.
2. Performance measures. Performance of the design process is evaluated against the quantity of resources (time, cost, manpower, etc.) used to satisfy the objective (that is, solve the design problem or task). An activity is evaluated against the resources expended to produce completely its outputs.
3. Iteration. The design process includes iteration. That is, similar activities are performed at different points (historical times) in the design process applied to different portions of the design task. This should not be confused with undesirable repetition arising due to re-doing the same portion of the design multiple times.
4. Sequence of activities. Although the individual activities performed are similar throughout the design process, they can be sequenced in different ways over the deisgn hierarchy.
5. Levels of scope and levels of abstraction. The design process deals with tasks at multiple levels: levels of scope (a measure of the amount of impact the task has on the overall design) and levels of abstraction (a measure of how conceptual or how detailed the task is).
6. Information management. Data about the design object is collected, generated, used to make decisions, and stored. The information gathered varies in certainty, quantity, and relevance for current and future use.

### 2.3.2 Existing models of the design process

In this section models of the design process are grouped according to the categorization of Evbuomwan [Evbuomwan, et al. (1996) pp. 311-312]. Then the properties of these groups are compared against the characteristics described above to discuss the effectiveness of existing design process models in matching reality. In this analysis, design process models are grouped into two classifications: those which are based on activities and those based on the phases of evolution of the design object. These two classifications are shown in table 2-1.

[3] The list of characteristics of the design process was established by performing industrial case studies and by researching published papers [Andreasen (1991b), Baya and Leifer (1994), Cross (1992), Evbuomwan, et al. (1996), Fortenberry (1991)].

**Table 2-1. Some existing models of the design process [after [Evbuomwan, et al. (1996)]][4]**

| Activity-based models | Phase-based models |
|---|---|
| Archer | Asimow |
| Cross | Clausing [Clausing (1994)] |
| Harris | French |
| Jones | Hubka |
| Krick | Pahl and Beitz |
| Marples | Pugh [Pugh (1991)] |
| Wilson [Wilson (1980)] | Ullman [Ullman (1992)] |
| | VDI 2221 |
| | Watts |

#### 2.3.2.1 Activity-based models (analysis-synthesis-evaluation cycle)

One view represented in the literature is that the design process consists of repeated iterations of three activities.[5] The names used for these activities can vary, but are typically referred to as *analysis*, *synthesis*, and *evaluation*. They can be defined as the following [Jones (1962)]:

1. Analysis deals with understanding the design task and generating the requirements and specifications.
2. Synthesis deals with generating ideas and solutions by exploring the design space.
3. Evaluation deals with the appraisal of design solutions against the requirements, specifications, and "set corporate criteria". [Evbuomwan, et al. (1996)]

#### 2.3.2.2 Phase-based models

Phase-based, sequential models of the design process tend to emphasize the progression of the design in terms of the amount known about the details of its implementation—its physical embodiment. The phases can be augmented with more specific activities or steps as in the activity-based models. [Evbuomwan, et al. (1996), Pahl and Beitz (1996)] In the model of Pahl and Beitz, these phases of the design process are described by the following[6] [Pahl and Beitz (1988) pp. 40-42]:

1. Planning and clarifying the task (specification of information in a requirements list): The market, the company, and the economy are taken into account to create and select

[4] Models not given in [Evbuomwan, et al. (1996)] are shown with references.

[5] In these models, additional activities observed by Evbuomwan included "optimization, revision, data collection, documentation, communication, selection, decision making, modeling, etc." [Evbuomwan, et al. (1996) p. 312], yet the three key activities predominate.

[6] Similarly the model by Pugh describes the sequence of design activities as following a central *design core*. This core "consists of [activities which produce or identify] market (user need), product design specification, conceptual design, detail design, manufacture and sales." [Pugh (1991) p. 5]

suitable product ideas. Then, requirements and constraints are formed into a requirements list.

2. Conceptual design (specification of principle): The objective of this phase is to determine the principle solution. To do this, the essential problems are abstracted; function structures are established; suitable working principles are sought; a working structure is synthesized; and lastly, solution concepts are evaluated against technical and economic criteria.
3. Embodiment design (specification of layout): In this phase a working principle is elaborated in the form of preliminary layouts which are then evaluated and rejected and/or combined to produce a definitive layout.
4. Detail design (specification of production): In this phase all production documents are produced.

These phases are qualified with two disclaimers by their proponents. First, a clear border cannot always be drawn between these phases, and second, it is not possible to avoid backtracking. [Pahl and Beitz (1996) p. 65] The reasons why these are so are explained in the next section.

#### 2.3.2.3 Comparison with desired characteristics

The two types of models for viewing the design process are compared against the desired characteristics of a design process model outlined in section 2.3.1. When such a comparison is done, the strengths and weaknesses of each model type become apparent. See table 2-2.

**Table 2-2. Strengths and weaknesses of design process models (key: ✓=strength, ⊕=passable, ×=weakness)**

| Characteristic | Activity | Phase |
|---|---|---|
| 1. Decision making | ✓ | × |
| 2. Performance measures | ⊕ | ⊕ |
| 3. Iteration | ⊕ | × |
| 4. Sequencing of activities | × | × |
| 5. Levels of scope & abstraction | × | ✓ |
| 6. Information management | ⊕ | ✓ |

The strengths of the activity-centered models are that they acknowledge the importance, within the design process, of making decisions in order to meet needs. Furthermore, given an understanding of the products of each activity, an evaluation of the performance of each activity can be made in terms of the resources expended to complete the activity. Iteration in design is clearly indicated in some of these models (see [Cross (1994), Wilson (1980)]). The models, however, tend to emphasize repeated evaluations of multiple concepts proposed for the same problem; thus, they do not acknowledge the similar sequence of activities at multiple lower levels of the same design once the higher-level decisions have been made. Lastly, the information management consists of producing information such as lists of factors, interaction matrices, partial solutions, and combined solutions.

Phase-based models emphasize two things concerning the information produced about design objects: first, its progression from abstract to detailed and second, its increasing quantity. Understanding the design task is weighted to the front end of the process, and the solving this task can become divorced from the production of solution details. The documents that are produced tend to evolve as the design progresses; thus, since they lack a clear endpoint, it is difficult to measure the resources expended to perform each task. Furthermore, while repetition, or revisiting, of a phase is undesirable because it tends to change design details already produced, it is acknowledged to occur frequently in practice.

The discussion above explains why often neither approach can be used to trace the progression of a design in an effective manner. This result occurs because the progress of the design process does not match its description in the model. Thus these models function as ideal cases only and not as useful descriptions of what was actually done.

Looking at typical models of the design process, Bucciarelli has concluded that "[t]o anyone interested in process, these diagrams shed very little light on how design acts are actually carried out or who is responsible for each of the tasks within the various boxes. Nor is it apparent what these participants need know, what resources they must bring to their task, and, most important, how they must work with others." [Bucciarelli (1994) pp. 112-113]

**Summary**

The conclusion of this section is that a new model of the design process is needed which accurately describes the sequence of activities performed and which can be used to guide designers more effectively.

In particular a model is needed to explain decomposition that details the decisions that the designers make so that rules and guidance can be developed. The model developed is presented in chapter 3 and is used in the case studies presented in chapter 6.

## *2.4 Rules*

This section describes the development of rules to aid designers in their decision making. The objective is to develop rules to aid designers in their decisions when performing the activities identified for decomposition; however, the same argument can be made about any such rules for design, including the design axioms, as shown below.

### 2.4.1 Discussion of the relationship between goals in decomposition and the overall design process

The theorems advanced by Suh[7] concerning decomposition allow the designers to separate out the objectives of decomposition and system integration activities from their objectives and considerations at one level of the design hierarchy. This is true because functionality and independence have been taken into account in selecting the DPs according to the design

---

[7] These theorems are as follows [Suh (1999)]:

> Theorem S1: The decomposition process does not affect the overall performance of the design if the highest-level FRs and Cs are satisfied and if the information content is zero, irrespective of the specific decomposition process.
>
> Theorem S2: Two "equivalent" designs can have a substantially different cost structure, although they perform the same set of functions and they may even have the same information content.

axioms. The result is that the designers are free to satisfy other additional goals in performing decomposition activities as long as they meet the constraints imposed on the products of their decomposition—such as maintaining consistency with the higher-level design decisions.

This can be represented by the following design equation:

$$\begin{Bmatrix} \text{provide functionality} \\ \text{minimize resource use} \end{Bmatrix} = \begin{bmatrix} X & O \\ X & X \end{bmatrix} \begin{Bmatrix} \text{decisions at each level} \\ \text{decisions between levels} \end{Bmatrix} \quad (2\text{-}3)$$

This equation describes the fact that the decisions made for each design equation (at each "node" in the design hierarchy) affect the subsequent decomposition process and the decisions that are made there. If the decisions at one level change, then the decomposition will also necessarily change. The decisions in decomposition, however, are neither sufficient to ensure product functionality, nor do they adversely affect functionality, assuming they are done consistently with the decisions made at each higher level.

The reason for understanding decomposition is to help designers perform it more quickly and easily. Correctly performing the process for decomposition is not sufficient to ensure that the design satisfies the axioms (although it must be consistent with this), but it does allow the designers to concentrate more of their efforts on synthesis, analysis, etc.—that is, satisfying the axioms—rather than becoming bogged down with questions about the process elsewhere. The designers are thus enabled to focus on the aspects of the design process that have the biggest impact on the functional performance of the design: synthesizing appropriate DPs and ensuring functional independence.

In articulating the objectives of the designers, it should be noted that decomposition and system integration activities—termed *decomposition and project control*—has its overall goals including finishing it quickly, making satisfaction of the design axioms easier, etc. Likewise each of its sub-activities has its own goals such as creating a sufficient set of sub-FRs, accounting for all constraints, etc. The goals are not all at the same level of abstraction.[8] Some of the goals derive from the nature of the activity being performed, for example, the need to map DPs to physical parts; some are a means to an end, contributing to the higher-level goals of decomposition or the whole design process itself, such as keeping the number of sub-FRs to a minimum.

## 2.4.2 Approach

The objective of this thesis is to identify the activities performed by the designers during decomposition and to understand the means to achieve the desired aims of the designers as they make design choices and decisions. This approach taken in this work involves identifying for each of the activities performed by the designers, the designers' cognitive

[8] McMullin discusses different abstractions of goals in the context of scientific methods: "Some kind of ordering of means and ends is clearly needed here. Some of the values we have been talking about seem to function as goals...of the scientific enterprise itself: predictive accuracy (empirical adequacy) and explanatory power are the most obvious candidates....Other epistemic values serve as *means* to these ends; they help to identify theories more likely to predict well or to explain." [McMullin (1993)] (p. 129-130 in [Curd and Cover (1998)])

aims, the options faced by the designers, and rules (or guidelines) correlating options to cognitive aims. Statements of three types result:

- Statements of the designers' goals: These statements describe the cognitive ends that designers are trying to satisfy either for the overall design process or for a specific activity within the design process.
- Statements of options available to the designers: These statements describe the choices available to the designers, for example, identifying tools which are available for a particular activity or identifying the possible activities that can be performed next.
- Normative statements of which options lead to goals: These statements correlate the options and goals stated above. They identify which options are most likely among the available options to lead to the desired cognitive ends.

#### 2.4.2.1 Normative statements as universal laws

The use of normative statements as universal laws[9]—as is done in this work and in axiomatic design generally –may strike some people as inappropriate. As argued by Laudan, such fears are unwarranted. [Laudan (1996)] Normative statements can easily take the form of universal laws when the cognitive ends of the statements are made explicit.[10]

Justifying normative statements is no more difficult than justifying any other scientific theory. Such prescriptive statements are on the same epistemic basis as other scientific laws; that is, they can be justified using the same scientific methods as for other more obviously descriptive laws. The statements, however, should be linked to a desired cognitive end. That is, by following such a rule, what is it that the users are trying to achieve?

The two ways that these rules can be tested is through examining the historical record or through showing necessity.[11] An argument based on historical evidence answers the question: did following this rule lead to the desired cognitive ends more often than following competing rules? Alternatively, the necessity argument says that a design (or design process) that meets the cognitive aim would necessarily have a certain characteristic, or characteristics, as described by the rule. The thrust of this argument is "not to point to their efficacy in the earlier history...but to recommend them on general epistemic grounds." [McMullin (1993)] (p. 130 in [Curd and Cover (1998)])

#### 2.4.2.2 Form and justification of normative laws

The form for normative rules as universal laws is this:

---

[9] The distinction between *universal truths* and *universal laws* is given by the following: "To conceive of [a statement of the form "All F's are G"] as a universal truth is to conceive of it as a relationship between the extensions of its terms; to conceive of it as a law is to conceive of it as expressing a relationship between the properties (magnitudes, quantities, features) which these predicates express (and to which we may refer with the corresponding abstract singular term)." [Dretske (1977)] (p. 830 in [Curd and Cover (1998)]).

[10] The specific example that Laudan discusses is that of creating scientific methodology. (See [Laudan (1996) Ch. 7, pp. 125-141].)

[11] See [McMullin (1993)] which discusses the evaluation of rules for theory choice in scientific practice.

> If actions of a particular sort, *m*, have consistently promoted certain cognitive ends, *e*, in the past, and rival actions, *n*, have failed to do so, then assume that future actions following the rule "if your aim is *e*, you ought to do *m*" are more likely to promote these ends than actions based on the rule "if your aim is *e* you ought to do *n*." [Laudan (1996) p. 135]

Statements of this sort are then evaluated according to the following:

> Given any proposed methodological rule (couched in appropriate conditional declarative form), do we have–or can we find–evidence that the means proposed in the rule promotes its associated cognitive end better than its extant rivals? [Laudan (1996) p. 135]

*The same form applies to the design axioms.* In this case, the desired cognitive aim is to improve the satisfaction of requirements versus design resources (time, man-hours, iteration), and this can be tested empirically. Overall, the idea for axiomatic design is that by following the design axioms better designs will be achieved. Specifically either better designs should be created using the same amount of time (improved effectiveness), or alternatively designs of comparable quality should be achieved in less time (greater efficiency).

The above discussion must be made with one caveat. In something with as much freedom as the design process—in which decisions are made by humans and in which the number of possible design solutions cannot be numbered—the rules for decision making will need to be qualified to account for multiple contributing factors. *Ceteris paribus* clauses need to be inserted into universal laws to account for intervening cases. That is, a universal law can take a causal form such as "A causes B *ceteris paribus*" (that is, all else being equal). This accounts for situations in which B has multiple causes and another cause occurs before the effects of A transpire.[12]

**Summary**

As argued above, rules for decomposition are needed and can be structured as described. These rules are detailed in chapter 3 and tested against industrial case studies in chapter 6.

## *2.5 System design issues and tools*

The above discussion deals with decomposition in general, not being limited to a single field of design tasks, yet the particular cases to which this work has been applied (and to which AD has been applied recently) fall within the field of system design. The rules for decision making discussed above are intended to be generally applicable to all instances of decomposition; however, additional tools are needed to deal with specifically system design. These tools are motivated in this section by considering the features that distinguish system design from more simple component design.

---

[12] This is described by the following:

> Suppose some event, *e*, has two potential causes *c* and *d*, in the sense that c occurs and causes *e*, and that *d* also occurs and does not cause *e*, but would have caused *e* if *c* had not occurred. *d* is a potential alternative cause of *e*, but is pre-empted by the actual cause *c*. [Ruben (1990)] (p. 722 in [Curd and Cover (1998)])

### 2.5.1 Definition of system design

A system is defined as "a regularly interacting or interdependent group of items forming a unified whole...as...a group of devices or artificial objects or an organization forming a network esp[ecially] for distributing something or serving a common purpose".[13] [Merriam-Webster's (1996)]

### 2.5.2 Unique features of system design

System design is unique from more simple component design in several ways. As indicated by the definition above, a *system* is composed of a number of distinct elements. In terms of design, this means that an object that is designed as a system *consists of an assemblage of DPs.* Therefore when conceptualizing a system, the designers consider a DP that is decomposed into a collection of disparate elements that acting together perform the desired higher-level function. An alternative is to consider the higher-level DP as a complete unit itself; then the decomposition entails refining the single DP, and the lower-level design choices consist of the several characteristics of the one DP.

In defining a system there are clearly identifiable, or clearly defined, interfaces[14] between the DP elements that make up the system. These *interfaces* are places in the design where the output of one FR serves as the input for another function. This means that systems can in part be modeled as a series of transformations (for example, process and transport functions).

Given this description of system design, several important features, or characteristics, of this type of design can be identified:[15]

- Because a system is an assemblage of DPs, physical integration, layout, and dimensioning of these DPs are important. The system is likely composed of DPs of different types and it can have multiple copies of a single type of DP.
- The task—functional requirement—of managing these DPs and integrated resources is critical.
- Systems can be dynamic in nature. The functions being performed at one time can be a subset of the functions performed by the system.[16] Or the values of the functions required can change at different times.
- A system can be controlled through user interaction or through automation. This control enables the switching between the different functions provided by a dynamic system.[17]

---

[13] The International Council on Systems Engineering (INCOSE) defines a system as "[a]n integrated set of elements to accomplish a defined objective". The elements can "include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements". [INCOSE (1998) p. 2-4]

[14] An interface is defined as ""the place at which two independent systems meet and act on or communicate with each other". [Webster's (1988) p. 631]

[15] Not all of the above characteristics need be present in each system. A system can exhibit a subset of them and still be considered a system design.

[16] See [Suh (1995b), Suh (1997), Suh (1998a), Suh (1999)].

- The integration of DPs into a system can impose constraints upon those DPs (for example, through the design matrices), not found in situations where the DPs are present alone.[18]

Given the above description, systems can be found at many locations of a design hierarchy. Systems can be at the top or at any intermediate layer of the design hierarchy. The only locations within the hierarchy at which they can not be found is as leaves. This is because leaves are by definition complete units. Leaves can be considered as systems to *their* designers but when incorporated into a larger system must be considered as off-the-shelf, finished items.

## 2.5.3 Motivation for specific tools for system design

The process for designing systems can be distinguished from non-system design. The process for system design is this: identify FRs, determine possible DPs, *then* integrate into a system. In this case, being able to understand the relationships among the different FRs and DPs (as represented in the design matrices and hierarchies) is vital. Without this ability, the design process becomes a confusing muddle, which can ultimately lead to poor and repeated decisions. Furthermore, in system design, it is desirable that the same process be useful at all levels of the design: the same approach is followed recursively, starting at the system level and continuing until the system design is complete.[19]

### 2.5.3.1 Current tools in axiomatic design theory

Before describing new tools to aid system design, the existing tools within axiomatic design are described in this section. For both new and existing designs, the system architecture provides a valuable tool for guiding decision making. The *system architecture (SA)* is a tool for decision making and its documentation in axiomatic design. It captures the requirements, components, and their relationships. The requirements are stated in terms of functional requirements (FRs) and constraints[20] (Cs); the components, in terms of design parameters (DPs); and the relationships, in terms of design matrices (DMs) and design hierarchies. This information can be depicted in a variety of ways. These include trees of design hierarchies and design matrices, flow charts, and module-junction-structure diagrams. [Kim, et al. (1991a), Kim, et al. (1991b), Suh (1997), Suh (1998a)]

The system architecture is a model, or representation, of the design decisions that have been made for the system. As such, it is a tool for further decision making because, as described by Ross, "M is a model of A if M can be used to answer questions about A." [Ross (1985)]

---

[17] The algorithms (logic) for this control are *command and control algorithms* (CCAs) as described in [Hintersteiner and Tate (1998b)] and chapter 4.

[18] These have been termed *consequences of integrated systems* (CoIS) [Lindholm, et al. (1998)] or *consequences of configured systems* (CoCS) [Lindholm (1998)].

[19] One tool for enabling this recursive process is the definition of generic FR templates for system design based on the idea that systems share common, identifiable features (see [Hintersteiner (1999)]).

[20] Constraints have been included previously in axiomatic design, but have not been discussed as part of the system architecture.

The types of questions that the system architecture can be used to answer include the following:

1. What downstream side effects does a proposed change to the design cause (other parts of the design that can need to be changed as a consequence)? [Harutunian, et al. (1996), Nordlund (1996)]
2. In what sequence should design changes be made and/or the design be further detailed? [Tate, et al. (1998)]
3. Where does coupling exist in the design and what are options for decoupling?[21] See for example, [Lee (1999)].
4. Given two or more options for decoupling a design, which of them requires the fewer changes to the design?[22]
5. How can attribute values of DPs be dynamically allocated to meet changing required FR target values during the operation of the system?[23] [Hintersteiner and Tate (1998b)]

### 2.5.3.2 Changing FRs, DPs, and values

This section describes the nature of changes of FRs and DPs and how these changes are captured within the system architecture of the design.

Given a system architecture (SA) consisting of FRs, DPs, DMs, and Cs in a hierarchical arrangement, the items that can be changed are these:

- the Constraints (Cs) (or their values)
- the FRs
- the DPs
- the desired (or "target") FR value (or the acceptable tolerance around this value)
- the DP value

---

[21] In cases of system design, where coupling occurs because of two conflicting leaf nodes, the design fails by not satisfying the FRs at the level where the two branches converge. The designers therefore have the option to choose the branch in which to make changes including the choice to redo the whole system from the convergence point downward.

[22] The answer to this question follows from the answers to questions 1 and 3. Several options for decoupling can be identified, and their downstream effects compared. The different options should lead to the same improvement in functionality, namely satisfying the FRs that were not satisfied before; however, the means for this can be through different new DPs. Furthermore the different options (appearing at two or more locations of the design hierarchy) will require the use of different amounts of development resources (manpower, time, cost, etc.) related to the number and amount of changes required, including downstream changes.

[23] The SA is capable of showing the effects of changing FR and DP values; a single SA, however, is not capable of representing cases in which the FRs and DPs themselves change (as discussed in section 2.5.3.2).

**Table 2-3: Examples of design changes**

| Element changing | Example | Affects |
|---|---|---|
| C (or value of C) | Reduce manufacturing cost | DPs at that level and below |
| Set of FRs | New process recipe with different process steps | Parent DP |
| DP | Module with new span of spin-speeds | sub-FRs, DM, DPs |
| Value of FR | New coating thickness | DP values at that level |
| Value of DP | New spin speed | DP values at that level |

These changes are related to the DMs in the following ways:

- *Changing DP values*: In designs in which the value of an FR is set during operation, this is done through adjusting the values of its DP. For example, the amount of time a wafer is exposed to a light source in photolithography is varied in order to achieve the FR of causing a chemical reaction in different thicknesses of photoresist.
- *Changing FR values:* In many designs, the target values of the FRs change during the actual operation of a system, in addition to during the design process. For example, the user needs to be able to set the value of coating thickness during operation.
- *Changing DPs:* In some designs there are options about the DP to be used to fulfill a particular FR. This is described here as "changing the DP".
- *Changing FRs*: Changes to FRs require new DPs. In some cases a design may be dynamically adjusted to correspond to a current set of FRs. In other cases no solution can be found this way, and the design needs to be redone from that point in the decomposition forward.
- *Changing Cs (or their values)*: A change in a constraint can affect the whole design. Functions are by definition independent of one another, but there is no similar restriction on constraints. For example, the weight of a system can be considered a constraint. Therefore, if the maximum weight allowed for the system changes, all the physical elements of the design are potentially affected since the total weight is the sum of the weights of all the physical components which comprise the system.[24]

While the system architecture is sufficient for capturing changes in DP and FR values and in some cases changes to DP, it does not capture changes to FRs, changes between alternative DPs, or changes to constraints.[25]

During the design process, of course, there are many options for DPs, and the progress of the design depends of the decisions made at the different levels of the design hierarchy. Given a parent DP, if its set of sub-FRs changes, then the parent DP itself can be considered

[24] This is discussed in section 3.4.3.

[25] Work is being done to expand the applicability of the SA such as [Friedman, et al. (1998b)] and [Hintersteiner (1999)].

to change; otherwise the DP remains the same and just some of its details or attribute values change. If the DP itself changes, then a new column in the design matrix must be evaluated.[26]

When the set of sub-FRs changes, this is treated as the change of a parent DP. That is, the change of a set of sub-FRs is equivalent to the selection of an alternative parent DP. As such, for a new set of sub-FRs, a new set of sub-DPs is needed as well as a new design matrix corresponding to the two. Therefore, a change of this type is not captured by the design matrix at this sub-level or at the parent level; this requires entirely new design matrices at the parent and child levels.

#### 2.5.3.3 Need for new tools

Several decisions of the design process are difficult to capture in the current representation of the system architecture. Unfortunately, the system architecture provides only limited aid for guiding designers in stating sub-FRs and for maintaining consistency throughout the design hierarchy, that is, ensuring that lower-level design decisions do not conflict with the decisions made at higher levels. Also the SA documentation—consisting of hierarchies of FRs, DPs, and DMs—is not sufficient to capture *all* necessary information about the system. Additional information that needs to be captured includes

- the layout
- the operational states of the system (which derive from, but are not synonymous with, the operand flow) that describe when particular FRs and DPs are executed
- the configuration (for example, numbers of physical resources that DPs have been integrated into)
- the constraints (Cs)

**Summary**

To address the needs of designers in performing system design. Additional tools are required. These include tools for managing DP resource allocation and integration and for representing and evaluating the dynamics of flexible FRs. These tools are detailed in chapter 4 and applied to the cases described in chapter 6.

### *2.6 Summary*

This chapter has detailed the research question and issues addressed in this thesis. First, the state of current design practice and design theory was discussed. Next specific issues that follow from this question and which are covered in this work were identified. Then the extensions and developments to design theory that address the research issues and answer the research question were described.

In performing decomposition, a measure of success is that of consistency. That is, do the sub-FRs, and the rest of the decomposed design, match the design decisions and the representations of the design that were made at higher levels of the design hierarchy? Good decision making in performing decomposition leads to design hierarchies that describe a

[26] If the DP itself does not change, then the detailing of the design must be done in the right sequence as specified by the design matrix, but no new design matrix needs to be constructed at this level.

design object at multiple layers of abstraction, but that consistently describe the same design object.

A model is needed of the decomposition process that identifies the activities performed, provides rules and tools to assist the designers, guides designers in the sequence of the decomposition process, and places decomposition into the overall context of system design.

The goal is to understand the means by which the desired objectives of the designers are achieved. This task is accomplished by identifying for each of the activities performed by the designers, the designers' cognitive aims, the options faced by the designers, and rules correlating options to cognitive aims.

A *system* and *system design* are defined, and the characteristics which distinguish system design from component design are identified. When the designers conceptualize a system, they consider a DP that is decomposed into an assemblage of disparate elements that acting together perform the desired higher-level function. The differences between system and component design motivate the need for additional tools specifically intended for system design.

# CHAPTER 3: ACTIVITIES AND GUIDELINES FOR DECOMPOSITION

## *3.1 Introduction*

The purpose of this chapter is to discuss how decomposition fits within the framework provided by axiomatic design. This chapter discusses the activities that occur between multiple levels of the design hierarchy and therefore may be considered part of decomposition. Descriptions of the activities are given in terms of their inputs and outputs and typical questions asked by designers in performing the activities. Each description is followed by a statement of the designers' objectives in performing the activity, the options available to the designers, and guidelines stated as rules for satisfying the objectives.

### 3.1.1 Decisions that the designers make

As argued in chapter 1, the activities for *decomposition, project control, and system integration* (*decomposition* for short) concern the progression and connections between the levels of a design hierarchy. These activities may be distinguished from other activities in the design process that are not part of decomposition and that may be carried out by considering only one level of the design hierarchy.

The activities are illustrated by means of an example.

*Example: Reticle management system (RMS) [adapted from [Friedman, et al. (1998a)]]*

One branch of a decomposition of a parent FR is shown in figure 3-1 and table 3-1. The highest-level FR shown is to "manage reticles". This hierarchy is shown along one branch for 3 levels.

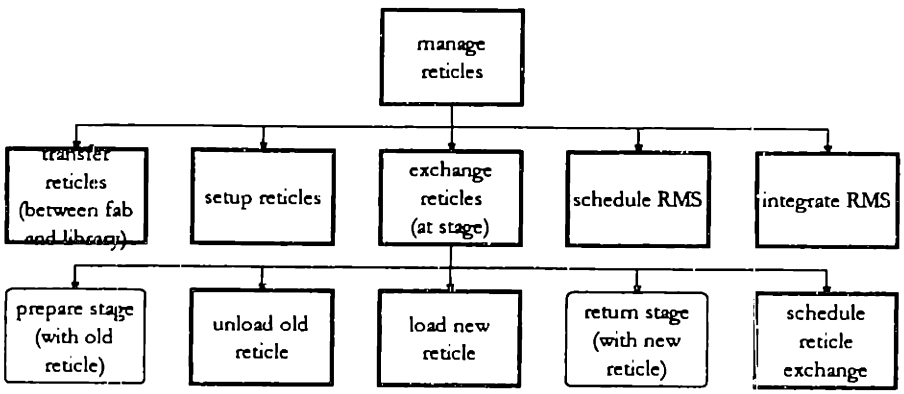


*Figure 3-1. One branch of an FR hierarchy*

In figure 3-1, the transportation FR "manage reticles" is broken down into five sub-FRs: three transportation sub-functions, one command-and-control function, and one support function. Likewise, the sub-FR "exchange reticles (at the stage)" has been broken down further into a set of transportation sub-FRs and a command-and-control sub-FR.

**Table 3-1. FR and DP to "Manage reticles" by means of a "Reticle management system (RMS) and some of their children**

| | Functional Requirements (FRs) | | Design Parameters (DPs) | |
|---|---|---|---|---|
| | Type | Description | Description |
| 3 | *Transport* | Manage reticles | Reticle management system (RMS) |
| 3.1 | *Transport* | Transfer reticles (between fab and library) | Module for reticle carriers |
| 3.2 | *Transport* | Setup reticles (at Wait, for exchange at stage) | Reticle setup scheme |
| 3.3 | *Transport* | Exchange reticles (at stage) | Reticle exchange scheme |
| 3.4 | *Control* | Schedule RMS | RMS CCA |
| 3.5 | *Support* | Integrate RMS | RMS framework |
| 3.3.1 | *Transport* | Prepare stage (with old reticle) | "Stage in position" signal |
| 3.3.2 | *Transport* | Unload old reticle | Second mechanism |
| 3.3.3 | *Transport* | Load new reticle | First mechanism |
| 3.3.4 | *Transport* | Return stage (with new reticle) | "Stage return" signal |
| 3.3.5 | *Control* | Schedule reticle exchange | Reticle exchange CCA |

The types of decisions that the designers must make that concern multiple levels of the design hierarchies are given in figure 3-2. Some of these activities occur before the designers synthesize new sub-DPs and analyze the new sub-DPs using the design axioms. Others occur after a satisfactory set of sub-DPs has been chosen by the designers.

For example, before the synthesis of DPs, the designers must identify a satisfactory set of sub-FRs. Before they can generate a set of sub-FRs, they must select the next parent FR-DP pair to detail. Furthermore, in generating sub-FRs, the designers must understand and account for the effect of the parent-level constraints, which can affect both the generation of the sub-FRs and the selections of satisfactory sub-DPs.

Once the designers have selected a set of sub-DPs, they are faced with another array of decisions. They must decide whether some of these sub-DPs should be physically integrated with other DPs. Furthermore, the designers must decide an appropriate number of instances of the physical resources of the DPs and their layout.

Therefore, in the example the decisions made by the designers, apart from selecting a set of sub-DPs, include among others the following typical questions:

- the next FR-DP pair to break into sub-FRs: Sub-FRs should be defined for which FR-DP pair: setup reticles, schedule RMS, unload old reticle?
- the generation of sets of sub-FRs: What is a good set of sub-FRs for the FR-DP 3.1 pair "transfer reticles" by means of "reticle carrier module"?
- the physical integration of sub-DP: Should the sub-DPs 3.3.1 and 3.3.2 "first mechanism" and "second mechanism" be physically integrated into a single resource?
- the effects of parent Cs on sub-FRs and sub-DPs: How does the constraint on exchange time impact the sub-FRs and sub-DPs for reticle exchange?

These questions arise as the designers perform the activities which comprise decomposition as presented in a roadmap in figure 3-2. The ways in which these connect with the activities at one level are shown in figure 3-3. As shown in figure 3-2, the activities in decomposition are generation of sub-FRs, identification of relevant CNs, physical integration of DPs, directing progress of the decomposition, dimensioning and configuration of DPs, layout of DPs, carrying down and refining Cs, and ensuring consistency between levels. As shown in figure 3-3, the activities at one level of the design hierarchy that must connect with project control and decomposition are design object analysis, decoupling, concept generation and selection, optimization, and tuning [Tate and Nordlund (1996), Tate and Nordlund (1998)].[1] Together these models should enable the explanation of the sequence of activities of *any* design process.

There are two motives for further understanding these decision-making activities. First, the designers should minimize the amount of ad hoc repetition of these types of decisions. Second, characteristics can be identified which distinguish good decisions from bad ones. Both of these ideas are consistent with the approach of axiomatic design.

[1] The activities shown in figure 3-3 that occur at one level of the design hierarchy are described in appendix 2.

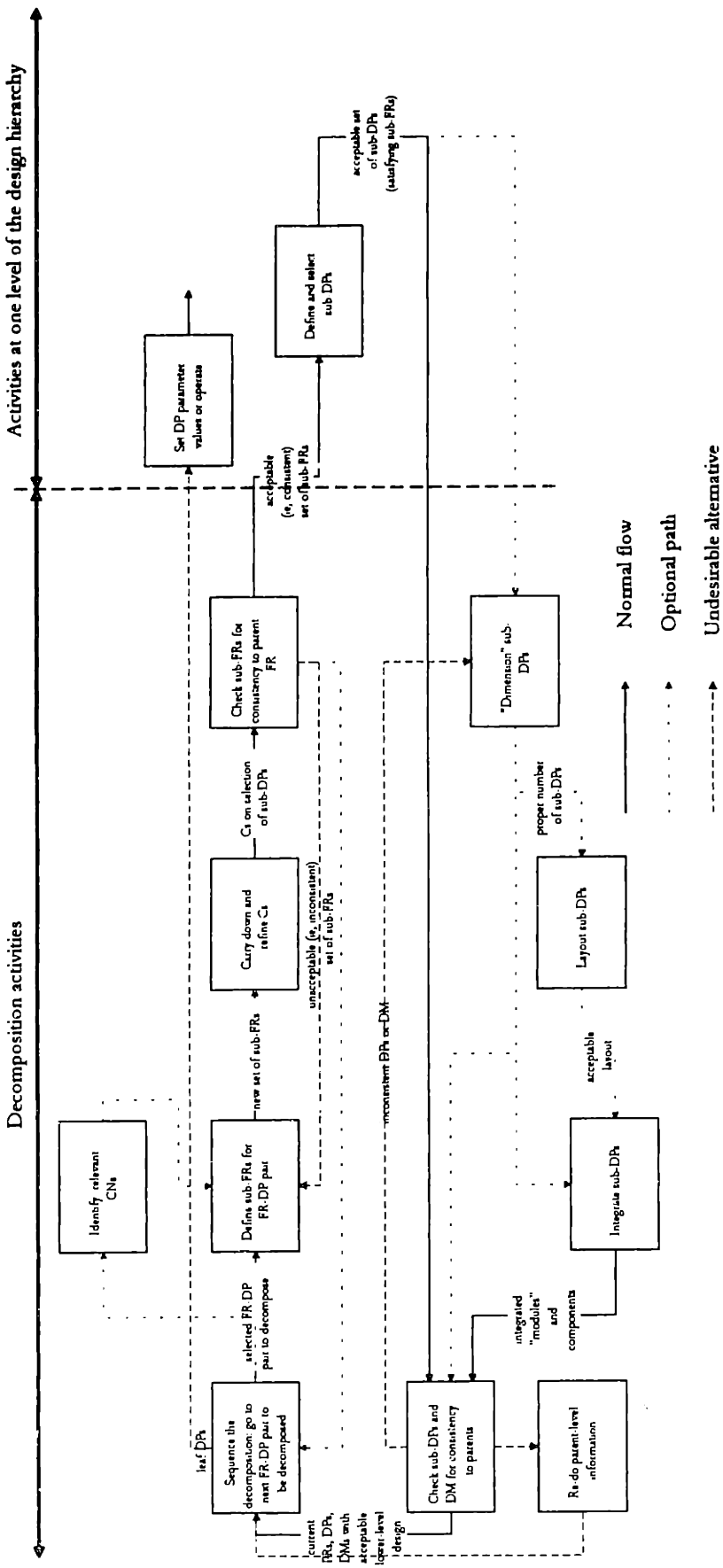


*Figure 3-2. Roadmap of activities in decomposition*

### 3.1.1.1 Minimal repetition

Because the designers want to minimize the amount of resources—in terms of time, manpower, money, etc.—needed to produce a design, they need to minimize repetition[2] of decisions. They need to minimize the number of instances in which the same decisions are made multiple times. This benefit is similarly provided by the design axioms.[3] The design axioms reduce the amount of unnecessary repetition of concept generation, or synthesis. In the absence of decision-making criteria, designers repeatedly synthesize new designs until one happens to satisfy its requirements.

Likewise, consider the decision in which a set of sub-FRs to represent an FR-DP pair is defined. If subsequent decisions—for example, either the generation sub-FRs later in the design process or the selection of sub-DPs elsewhere in the design hierarchy—requires the designers to revisit an earlier instance in which sub-FRs are defined, then this clearly would be an undesirable repetition of an already-made design decision. As such, this is a large potential timesink.

Perhaps the objection will be raised that a single designer can implicitly handle this type of concern when performing small design tasks: The individual designer is able to keep in mind all the relevant design dependencies between the parts of the design to be detailed, and the consequences of revisiting decisions is minimal because the size of the design hierarchy is small (few numbers of FRs and DPs). In the case of larger, distributed design projects, however, this issue becomes acute. Furthermore, in a design project that is automated by computer, a sequence to proceed in detailing the design necessarily needs to be determined.

### 3.1.1.2 Existence of distinguishing characteristics

This thesis builds on the scientific foundation for understanding, modeling, and guiding design that is provided by axiomatic design theory. The underlying hypothesis of axiomatic design is that there exist fundamental principles that govern good design practice.[4] The axioms distinguish good decisions from bad ones as the designers perform concept generation and analysis. Likewise, there are characteristics that distinguish good decisions from bad ones in other parts of the design process.

For example, since the design hierarchy must be complete at some point, a task of design researchers, therefore is to identify those characteristics that distinguish DPs that need to be decomposed from those that do not. This is equivalently stated as knowing whether a DP may be considered a leaf.

---

[2] *Repetition* is used to refer to re-making a decision that has already been made at one location of the design hierarchy. *Iteration* is used to refer to the performance of the same type of activity in a different location of the design hierarchy.

[3] According to Suh, "Most inventive processes are hit-or-miss activities, requiring much trial-and-error....Often people chase after worthless ideas because they do not know that their ideas have basic flaws....When [an] invention involves synthesis, there [has been] no fundamental criterion in the past [to identify good ideas]. The axioms...provide such criteria and thus can streamline the hit-or-miss process." [Suh (1990) pp. 29-30]

[4] According to Suh, common features of the "creative process [that] permeates all fields of human endeavor" can be used to "distinguish between good and bad designs." [Suh (1990) p. 5]

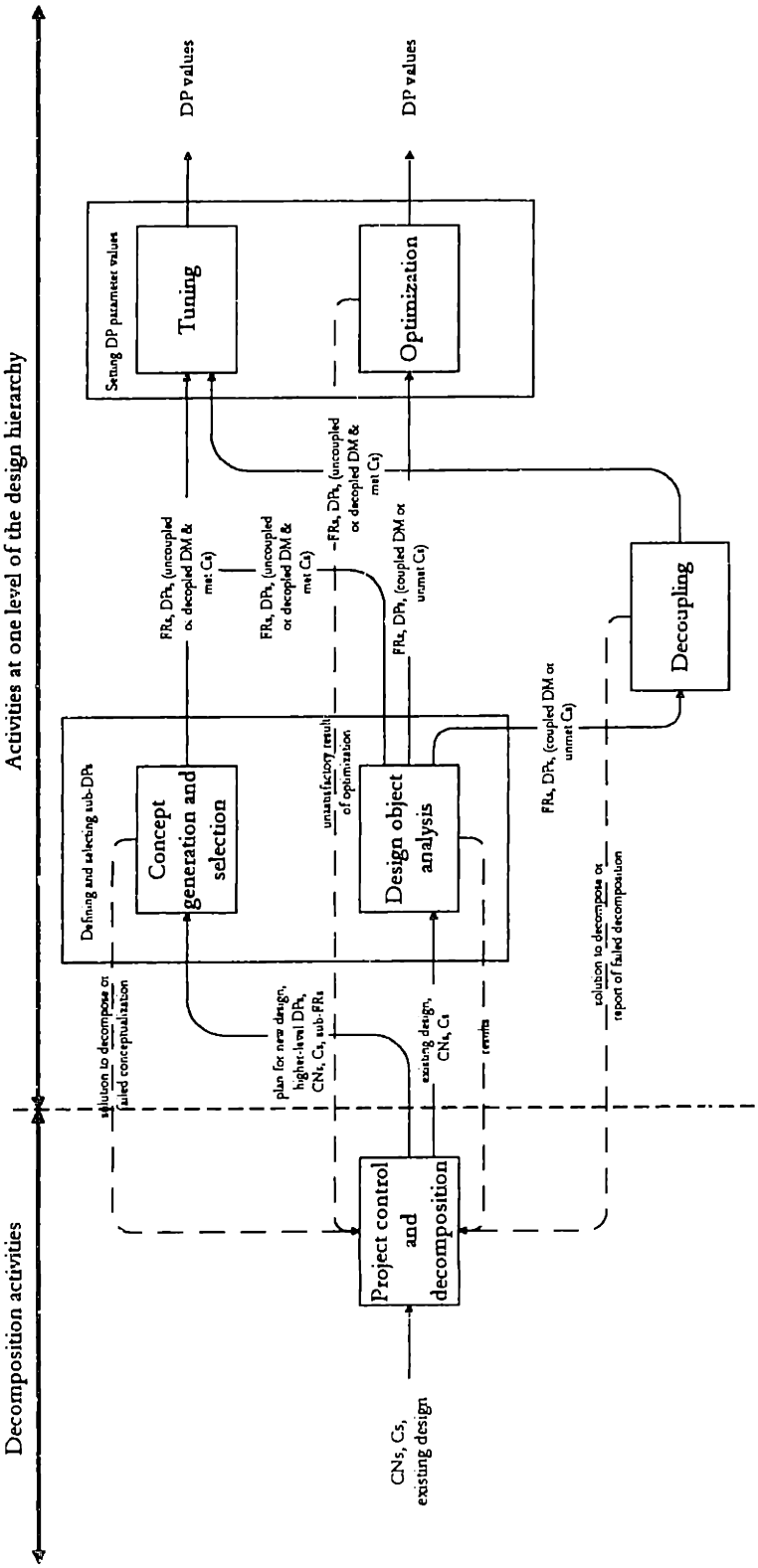


*Figure 3-3. The design process roadmap [adapted from [Tate and Nordlund (1996), Tate and Nordlund (1998)]]*

### 3.1.2 Approach to this chapter

An objective of decomposition is to enable the designers to satisfy the axioms at each level of the design hierarchy more quickly and more easily. This is done through articulating a set of activities that are performed that relate to multiple levels of the design hierarchy. Once these activities are identified, the goals for each are stated, and the corresponding options the designers have in performing the activities are given. Then, in the spirit of axiomatic design, guidelines are established which relate the options to the goals for each activity. This way the designers have a practical aid in making decisions within each of the activities.

The approach taken in this chapter is consistent with that followed in axiomatic design. I followed the approach of nomothetic research in developing a set of activities and general guidelines for decomposition. To identify rules, or criteria, for decision making in decomposition, I abstracted general principles from recent case studies in which I participated, that use axiomatic design, and that involve decomposition. This approach, in which a researcher gathers data from multiple different events to search for patterns in the data, is known as *nomothetic research*. This approach is similar to that used by Suh to generate the design axioms.[5]

The results of this nomothetic study are presented as a series of activities along with rules, or guidelines, for their satisfaction. These are illustrated by means of examples from the cases. The three contributions of this chapter are

- the model of decomposition activities (section 3.2 and figure 3-2)
- the identification of the designers' aims in performing the activities (section 3.3 and table 3-2)
- the statement of generalized guidelines for achieving the designers' aims (section 3.4 and table 3-16)

While the theory developed in this chapter—consisting of the set of activities, goals, and guidelines—is intended to be generally applicable to all designs consisting of multiple levels of design hierarchy, the guidelines are biased, if at all, towards system design. This is due to the nature of the case-study projects that were examined in this work.

Because of the number of activities considered and the number of guidelines possible for each, it is impossible for the author to give the rationale for or the thought process that generated every guideline. The author will therefore focus on a selection of activities and a selection of guidelines within each.

The activities that are treated in the greatest detail in this chapter are sequencing the decomposition (sections 3.2.1 and 3.4.1), generation of sub-FRs (sections 3.2.2 and 3.4.2),

---

[5] Suh states that he "considered several projects that he had worked on in industry and at universities, and that had turned out to be very successful, and tried to identify the common elements present in all. He then tried to generalize these common elements....Out of this exercise evolved 12 'hypothetical' axioms." [Suh (1990) pp. 20-21]

carrying down and refining Cs (sections 3.2.4 and 3.4.3), physical integration of DPs (sections 3.2.7 and 3.4.4), and ensuring consistency between levels (sections 3.2.8 and 3.4.5).[6]

Within the description of each activity, the goals of the designers are given followed by guidelines for achieving the goals[7] along with examples interspersed throughout the chapter as an aid for explanation. The examples in this chapter are taken from a number of sources. They are used to explain the concepts being presented in this chapter, *not to explore the details of the cases themselves.* Additional information on the cases from which these examples are drawn is found in section 6.3 in chapter 6.

## Terminology

Because the complete details of the examples are not given in this chapter, a few points of terminology are clarified here to aid the reader to quickly grasp the essence of the examples. First, an FR is considered as a transformation of a target object. Second, the numbering schemes for FRs, DPs, Cs, and design matrices, incorporates their relative positions within the design hierarchy. Third, systems are modeled using a standard template.

*1. FRs as transformations*

*Functional requirements* are defined as the minimum set of requirements that completely characterize the design objectives for a specific need. [Suh (1990) p. 38] Each FR specifies a target (or a span) of needed values that must be held within a tolerance. It should be emphasized that FRs are to be defined in *solution-neutral* terms.[8] These values specified by the FRs describe measurable characteristics of some object or objects. An object with a characteristic that is affected by an FR, is called (in this thesis) a "target object" of that FR. At one node of the design hierarchy, there may be more than one object whose characteristics are affected by the set of FRs, and in measuring some FRs, properties between two or more objects may be relevant.

The target of an FR is *not dependent on the choice of the DP to fulfill the FR.* For example, in the case of a function such as "transport passengers between Boston and New York City", the target object is the person, or persons, that are transported, not the DP that performs this (such as a car, a plane, etc.). The concern with this FR is a property of the passengers, namely their location, and how close this property is to the desired target value.

Some functional requirements may be described as transformations at some rate either in a continuous manner or between discrete states. Assuming it is performed correctly, the FR "transport passengers between Boston and New York City" changes the location of the passengers from their initial values (in Boston) to their final values (in New York City). Moreover, the target object affected by the FR can be a physical entity as in the case of hardware design, or it can be an abstract entity as in software design.

In the example described previously, the FR is to "manage reticles"; therefore, the target object is a reticle or reticles. The FR is stated in a solution-neutral way, and the designers

---

[6] The other activities (dimensioning of sub-DPs, layout of sub-DPs, and identification of relevant CNs) are described (in sections 3.2.5, 3.2.6, and 3.2.3 respectively) in terms of their inputs and outputs and typical questions posed and answered by designers in performing the activities.

[7] The form for the guidelines is "to achieve the designers' aim *e*, do *m*" as described in section 2.4 in chapter 2.

[8] This topic is discussed in much greater detail in section 3.2.2.

have specified only properties of the reticle in generating the FR. They have not made any decision about the DP chosen to fulfill this FR. *Design parameters* are defined as the tangible design elements which have been chosen to fulfill the FRs, that is, to perform and control the desired function, be it physical, logical, monetary, etc.

*2. The numbering scheme for FRs, DPs, Cs, and DMs*

The numbering scheme for design matrices is as follows. Consider a design matrix *[Ax]* as shown in equation 3-1; each off-diagonal term *Ax(i,j)* corresponds to the effect on *FRx.i* of changing *DPx.j*. The interpretation of the design matrices used here is that an element of the design matrix *Ax(i,j)*, which is given by $\frac{\partial FRx.i}{\partial DPx.j}$, corresponds to asking the following question: Does a change[9] in *DPx.j*–consistent with fulfilling *FRx.j*–affect *FRx.i*?[10]

$$\begin{Bmatrix} FRx.1 \\ \vdots \\ FRx.i \\ \vdots \\ FRx.n \end{Bmatrix} = \begin{bmatrix} Ax(1,1) & \cdots & Ax(1,j) & \cdots & Ax(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Ax(i,1) & \cdots & Ax(i,j) & \cdots & Ax(i,j) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Ax(n,1) & \cdots & Ax(n,j) & \cdots & Ax(n,n) \end{bmatrix} \begin{Bmatrix} DPx.1 \\ \vdots \\ DPx.j \\ \vdots \\ DPx.n \end{Bmatrix} \tag{3-1}$$

In this notation *x* is the index of the parent FR and parent DP that has been broken down into a set of sub-FRs; *i* indicates the placement of the sub-FR *FRx.i* within the vector of sub-FRs {*FRx.i*}; *j* indicates the placement of the sub-DP *DPx.j* within the corresponding vector of sub-DPs. Constraints on the sub-FRs {*FRx.i*} are numbered *Cx-k*.

*3. A template for system design*

In a general sense, the input-output transformations at the top level of a system design can be broken down into several categories: process functions, transport functions, command and control functions, and support and integration functions, as shown in equation 3-2.[11] The examples in this chapter follow this form.

$$\begin{Bmatrix} \text{Perform Process 1} \\ \text{Perform Process 2} \\ \vdots \\ \text{Perform Process } u \\ \text{Transport 1} \\ \text{Transport 2} \\ \vdots \\ \text{Transport } v \\ \text{Command System} \\ \text{Integrate System} \end{Bmatrix} = \begin{bmatrix} X & O & \cdots & O & O & O & \cdots & O & O & O \\ O & X & & O & O & O & & O & O & O \\ \vdots & & \ddots & & \vdots & & \iddots & & \vdots & \\ O & O & & X & O & O & & O & O & O \\ X & X & \cdots & ? & X & O & \cdots & O & O & O \\ ? & ? & & ? & ? & X & & O & O & O \\ \vdots & & \iddots & & \vdots & & \ddots & & \vdots & \\ ? & ? & & ? & ? & ? & & X & O & O \\ X & X & \cdots & X & X & X & \cdots & X & X & O \\ X & X & & X & X & X & & X & X & X \end{bmatrix} \begin{Bmatrix} \text{Process Module 1} \\ \text{Process Module 2} \\ \vdots \\ \text{Process Module } u \\ \text{Transport Module 1} \\ \text{Transport Module 2} \\ \vdots \\ \text{Transport Module } v \\ \text{Command and Control Algorithm} \\ \text{Structural Framework} \end{Bmatrix} \tag{3-2}$$

[9] The type of changes described here are those which do not affect the terms of the design matrix—since such changes cannot be captured in the design matrix itself. See section 2.5.3 for further discussion.

[10] Specifically if a DP may be adjusted to satisfy its corresponding FR and the other FRs at that level of the design tree are unaffected within their tolerances, then the requirement for functional independence is satisfied. See Theorem 8. [Suh (1990) pp. 121-123]

[11] This template for system design was first used in [Hintersteiner and Tate (1998b)] and is further refined in [Hintersteiner (1999)].

## *3.2 Activities in the design and decomposition process*

The model of decomposition activities, given in figure 3-2, consists of a collection of distinct activities with clear starting and end points. Each activity is a transformation from inputs to outputs. These activities can be sequenced in different ways, and as indicated, some of them are optional in any given cycle.

### 3.2.1 Sequencing the decomposition

As they detail a design object through the process of decomposition, the designers determine whether there are any FR-DP pairs that need to be decomposed further during the activity of *sequencing the decomposition.* The designers carry out decomposition until they "know what to do from there". [Suh (1998b)][12] Therefore, in design involving multiple layers of the design hierarchy, both DPs which are decomposed and DPs which are not decomposed further exist and may be termed *non-leaf* and *leaf DPs* respectively. Furthermore, when there are multiple FR-DP pairs to be decomposed, the designers select the one for which sub-FRs will be defined next during this activity. If no further decomposition is required, this activity directs the setting of the attribute values of the leaf DPs through design decisions.

Typical questions answered by the designers in performing this activity include the following:

- Does the design need to be decomposed further? Are there more FR-DP pairs to be decomposed, or are all the DPs *leaf DPs*?
- Which FR-DP pair should be decomposed next?

The input to this activity is a design object which has been detailed to some level of the design hierarchy; this includes FRs, DPs, Cs, and DMs in a hierarchical arrangement. The outputs to this activity can be several:

- a determination of whether the design needs to be decomposed further
- if the design does need to be decomposed, a selection of which FR-DP pair is to be decomposed next
- if the design does not need to be further decomposed, a description of the order in which the attribute values of the leaf nodes are to be set

### 3.2.2 Generation of sub-FRs

*Generation of sub-FRs* is the activity in which the designers develop a set of sub-FRs for one parent FR-DP pair. The design has progressed to some level of the design hierarchy; a set of parent DPs has been chosen to fulfill a set of parent FRs; and at least some of the DPs at this level are considered to be non-leaves. Therefore, the designers continue the decomposition by generating some sub-FRs.

Typical questions answered by the designers in performing this activity include the following:

- What are all the sub-FRs required to perform the parent FR?
- How can the concept of the parent DP be detailed as a set of sub-functions?

---

[12] Compare [Suh (1999)].

- Can information about the field of the design task be used in generating the sub-FRs? Does the "look" of the FRs correspond to their particular field (mechanical, software, systems, etc.)?
- Can the stated sub-FRs be combined into a set with fewer elements?
- What are different ways to define sub-FRs, and which is most appropriate for the current situation?
- Where does a command and control algorithm (CCA) fit within the hierarchies? What functions does it perform? How can it be generated?

The inputs for the generation of sub-FRs are a selected parent FR and DP to be decomposed. The output of this activity is a set of sub-FRs for the given parent FR-DP pair.

### 3.2.3 Identification of relevant CNs

*Identification of relevant CNs* is the activity in which the designers match customer needs to the current level of the design hierarchy. At the highest levels of the design hierarchy, this involves collecting and sorting the information that has been gathered from the customers. At lower levels this involves evaluating the applicability of the CNs to the current part of the design task.

Typical questions answered by the designers in performing this activity include the following:

- Is the set of customer needs complete, or should additional information be gathered, given the design decisions made thus far, at higher levels of the design hierarchy?
- Does constraint *CN-i* apply to *FRn.j*?

The input of this activity is a selected FR-DP pair to decompose and the customer needs that have been collected. The output of this activity is a set of CNs that is relevant to the selected FR-DP pair and potentially useful in defining its sub-FRs.

The organization of the customer needs (CNs) to facilitate the generation of FRs, particularly at the top-level of the design hierarchy, is an important consideration although it is beyond the scope of this work. The reader is referred to sources such as [Burchill (1993), CE (1992), LPM (1996)] for ideas about the performance of this activity and the tools available to assist it.

### 3.2.4 Carrying down and refining Cs

*Carrying down and refining constraints* is the activity in which the designers transfer a set of constraints that applies at a parent-level of the design hierarchy to the level's sub-FRs by assigning each parent level constraint to one or more sub-FRs. *Constraints* are a specification of characteristics that the design solution must posses to be acceptable to its customers and the company designing it.

Typical questions answered by the designers in performing this activity include the following:

- Does constraint *Cn-i* apply to *sub-FRn.j*?
- Have all the parent-level constraints been applied, or is there a constraint that has not been applied to a sub-FR?
- Have additional constraints been introduced at the child level that should be applied at the parent level as well?

The input of this activity is a set of sub-FRs and their parent FR and DP, along with the constraints that applied at the parent level. The output of this activity is a set of constraints applied to each of the sub-FRs as needed. The constraints that are applied at the child level can be restated, or refined, as needed to clarify or detail their application to the sub-FRs.

### 3.2.5 Dimensioning of sub-DPs

*Dimensioning of sub-DPs* is an activity which occurs after DPs have been selected. In the dimensioning activity, the throughput rate and other constraints on the FRs are used by the designers to calculate a needed number of instances of each particular DP. This activity is not necessarily performed at each level of the decomposition, but it must be performed for each DP at some point during the design process.

Typical questions answered by the designers in performing this activity include the following:

- How many of this DP are needed to fulfill the FR, given the throughput rate constraint (or takt time)?
- What throughput rate can be achieved by the selected DP?
- How many instances of this DP are needed to handle the capacity needed for the FR?
- How many transport resources are needed?
- Can a generic scheduling algorithm for assigning DPs to meet FRs automatically be developed?

The input of this activity is an FR and its selected DP. The output of this activity is a dimensioning or number of needed instances of this same DP to fulfill the FR.

### 3.2.6 Layout of DPs

*Layout of sub-DPs* is the activity in which the designers arrange a given set of sub-DPs, either physically in space or logically. This activity is like dimensioning in that it does not need to be performed at each level of the design hierarchy, yet it must be performed for all the DPs in the hierarchy at some point during the design process.

Typical questions answered by the designers in performing this activity include the following:

- How should these DPs be arranged spatially?
- How do changes in the number of process modules affect transportation or layout?

The input to this activity is a design object which has been detailed to some level of the design hierarchy; this includes FRs, DPs, and DMs in a hierarchical arrangement. Because this activity follows dimensioning, an additional input is the number instances of DPs. The outputs to this activity are a physical description of the spatial or logical arrangement of DPs. In mechanical design, this could be a set of drawings or in object-oriented software design, a grouping of attributes (variables) and operations (methods).

### 3.2.7 Physical integration of DPs

The designers perform *physical integration of DPs* to reduce the information content of the design. In this activity the designers assemble a set of separate DPs into one physical entity. This is done to reduce manufacturing cost, reduce development cost, minimize volume of the product, etc. As several DPs are integrated into one physical unit, the individual DPs do

not lose their uniqueness, their individuality, because if they did, that would compromise the independent satisfaction of their associated FRs. Rather, in physical integration two or more DPs are embodied in the same resource or set of components.

Typical questions answered by the designers in performing this activity include the following:

- What is the minimum number of components needed to perform the FRs?
- How can manufacturing costs be reduced?
- How can DPs be fit into the required volume?
- Is the same component being used to perform multiple FRs?
- Can the resources and components be scheduled to prevent functional coupling?
- What are the "interfaces" between DPs of different types?
- How can different integrations be evaluated and selected?

The input to this activity is a design object that has been detailed to some level of the design hierarchy. This design consists of sets of DPs which have been selected to fulfill FRs over the design hierarchy. The DPs can include leaves and non-leaves set during either design or operation. The output of this activity is information about the physical embodiment and integration of the design in hardware and/or software components.

### 3.2.8 Ensuring consistency between levels: FRs, DPs, DMs, Cs

*Ensuring consistency between levels* is an activity in which the designers check the layers of the design hierarchy against each other to ensure that the layers are consistent with one another. In this activity, which is shown occurring in multiple parts in figure 3-2, the FRs, DPs, DMs, and Cs must all be checked.

Typical questions answered by the designers in performing this activity include the following:

- Do these sub-FRs together produce their intended parent FR?
- Are these sub-FRs consistent with the parent DP that was selected?
- Have the constraints been refined properly?
- Are the FRs are defined at the correct hierarchical level?
- Have all the design decisions made thus far in the design process been recorded in or along with the design hierarchy (including layout, integration, etc.)?

For ensuring consistency between levels, the inputs are FRs, DPs, DMs, and Cs, at both the parent- and sub-levels. The output of this activity is an evaluation of whether or not the levels are consistent with one another.

## *3.3 Goals*

Table 3-2 summarizes the cognitive aims of the designers in performing each of the decomposition activities shown in figure 3-2.

**Table 3-2. Summary of cognitive aims for decomposition activities**

| Cognitive aims for decomposition activities |
|---|
| Sequencing the decomposition: |
| finish the design decomposition as quickly as possible (minimize use of resources: time, cost, manpower, etc.) by finishing the selection of DPs and moving on to setting their attribute values |
| distinguish between leaf and non-leaf nodes and thereby know when the design decomposition is finished |
| identify the next FR-DP pair to decompose—if there is one |
| Generating sub-FRs: |
| develop a sufficient set of sub-FRs (with respect to the parent FR) |
| describe the parent DP (consistency is especially important for analyzing an existing DP) |
| state solution-neutral sub-FRs |
| develop a necessary set of sub-FRs (with respect to the parent FR) |
| minimize the number of sub-FRs |
| Carrying down and refining Cs: |
| understand the impacts of Cs on sub-FRs |
| convert Cs from the parent level to sub-FRs as required, and distinguish Cs from FRs |
| develop a complete set of Cs for each sub-FR |
| Checking consistency of sub-FRs, sub-DPs, sub-DMs, and Cs: |
| determine if descriptions of FR-DP relationships have been correctly identified and documented in the design hierarchy (including decisions about physical integration, dimensioning, and layout) |
| determine if sub-FRs are consistent with choice of parent DP |
| Dimension sub-DPs: |
| satisfy geometric constraints |
| determine minimum number of DPs to meet the throughput rate constraints |
| Layout sub-DPs: |
| develop spatial arrangement |
| satisfy geometric constraints |
| meet throughput rate constraints |
| Integration of sub-DPs: |
| assign sub-DPs to physical (or software) resources or components |
| minimize cost |
| fit within footprint, or volume |
| minimize information content |
| Identification of relevant CNs: |

## *3.4 Guidelines for the design and decomposition process*

In this section, guidelines are presented for some of the goals associated with the following activities: sequencing the decomposition (section 3.4.1), generation of sub-FRs (section 3.4.2), carrying down and refining Cs (section 3.4.3), physical integration of DPs (section 3.4.4), and ensuring consistency between levels (section 3.4.5).

## 3.4.1 Sequencing the decomposition

For the activity of sequencing the decomposition, it is hypothesized that, to finish the design decomposition as quickly as possible (thus minimizing the use of resources: time, cost, manpower, etc.), the designers must do two things. The designers must distinguish between leaf and non-leaf nodes and thereby know when the design decomposition is finished (discussed in section 3.4.1.1), and the designers must identify the next FR-DP pair to decompose (discussed in section 3.4.1.2), if there are non-leaf nodes remaining.

### 3.4.1.1 Distinguishing between leaf and non-leaf nodes

As shown in figure 3-4, a *node* in the design hierarchy is defined as a parent FR-DP pair plus its associated children (sub-FRs and sub-DPs), constraints on the children, and the design matrix of relationships between them. If the node is a *leaf* then it does not have children; if it is a *non-leaf* then it does. Likewise, *leaf DPs* are defined as terminal DPs on the design tree, and *non-leaf DPs* are defined as DPs that are further decomposed.[13] Figure 3-4 illustrates leaf DPs, denoted by lighter colored rectangles with thin, dashed lines and non-leaf DPs, denoted by darker rectangles with thick, solid lines.

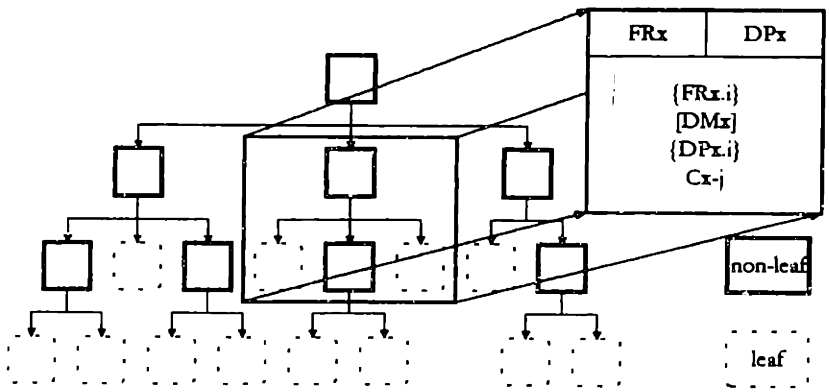


*Figure 3-4. FR or DP tree showing leaf and non-leaf nodes*

*Example: Reticle management system (RMS) [adapted from [Friedman, et al. (1998a)]]*

One branch of a decomposition of a parent FR is shown in figure 3-5 and table 3-3. This is a continuation of the example shown in table 3-1. The target object of the highest-level FR is a

[13] According to Suh, "When the decomposition process propagates down to the lower levels, a designer can reach the level where one or more FRs can be fully satisfied (or controlled) by the selected set of DPs without further decomposition. If the FRs do not have to be decomposed any further, they form terminal nodes of the hierarchical tree. The design process terminates when all the lowest branches of the FR tree form the terminal nodes. A terminal node is [called] a leaf of the FR tree." [Suh (1998a) ch. 5 p. 18] also [Kim, et al. (1991b) p. 167] and [Kim, et al. (1991a) p. 248]

reticle; that is, the FR "manage reticles" operates on a reticle. This target object appears again at lower levels. It is the target object of FRs at the next level, and it is carried down over 5 levels.[14]

**Table 3-3. FR and DP to "Manage reticles" by means of a "Reticle management system (RMS) and some of their children**

| | Functional Requirements (FRs) | | Design Parameters (DPs) |
|---|---|---|---|
| | *Type* | *Description* | *Description* |
| 3 | *Transport* | Manage reticles | Reticle management system (RMS) |
| 3.1 | *Transport* | Transfer reticles (between fab and library) | Module for reticle carriers |
| 3.2 | *Transport* | Setup reticles (at Wait, for exchange at stage) | Reticle setup scheme |
| 3.3 | *Transport* | Exchange reticles (at stage) | Reticle exchange scheme |
| 3.4 | *Control* | Schedule RMS | RMS CCA |
| 3.5 | *Integrate* | Integrate RMS | RMS framework |
| 3.3.1 | *Transport* | Prepare stage (with old reticle) | "Stage in position" signal |
| 3.3.2 | *Transport* | Unload old reticle | Second mechanism |
| 3.3.3 | *Transport* | Load new reticle | First mechanism |
| 3.3.4 | *Transport* | Return stage (with new reticle) | "Stage return" signal |
| 3.3.5 | *Control* | Schedule reticle exchange | Reticle exchange CCA |
| 3.3.3.1 | *Transport* | First mechanism handoff (reticle to stage) | End-effector |
| 3.3.3.2 | *Transport* | Move reticle (towards stage) | 6-DOF robot |
| 3.3.3.3 | *Control* | Schedule loading | Load CCA |
| 3.3.3.2.1 | *Transport* | Change end-effector position | 6-DOF arm |
| 3.3.3.2.2 | *Process* | Sense position of end effector | Joint encoders |
| 3.3.3.2.3 | *Process* | Minimize force (applied to stage) | Robot limits on $F, X, \dot{X}, \ddot{X}$ |
| 3.3.3.2.4 | *Transport* | Maintain parallelism (between reticle and stage) | Robot flexures (for compliance) |
| 3.3.3.2.5 | *Control* | Schedule move | Move CCA |

In figure 3-5 the sub-FRs in which also operate on the "reticle" are depicted by dark rectangles outlined with thick, solid lines. As these FRs dealing with the reticle are decomposed, additional target objects appear in the design hierarchy including primarily the "reticle stage" and the robot and end effector. The sub-FRs that have other target objects are shown as lighter-colored rectangles with thinner dashed lines.

[14] In total this FR has been decomposed into approximately 150 sub-FRs as described in section 6.3.3.

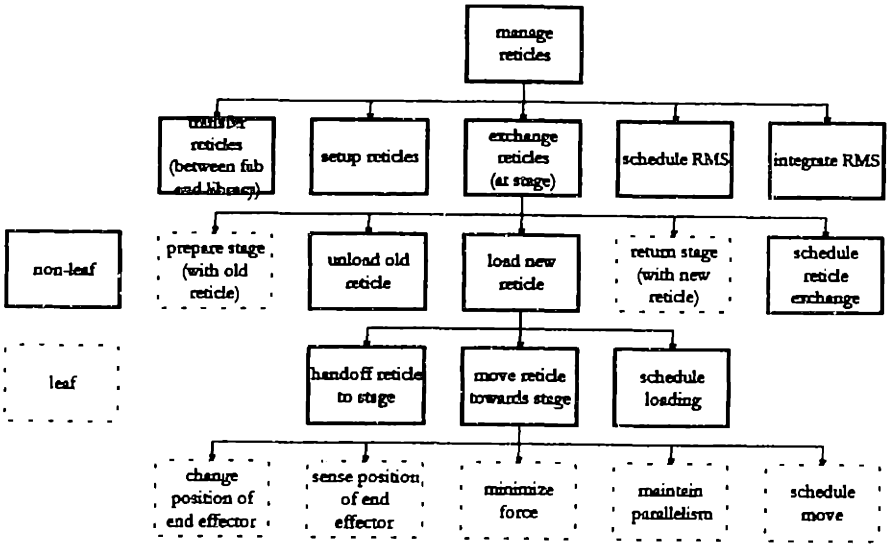


*Figure 3-5. Example of leaf and non-leaf nodes*

The FRs with reticles as target objects have been further decomposed; the others are considered to be leaves. The satisfaction of the leaves under "move reticle" have been passed on to other engineers within the project.

*Guidelines*

An FR does not need to be further decomposed if its target object is different than the target object of its parent FR. However, a DP must still be selected to satisfy the FR. At each point in the decomposition at which the target object changes between parent and child, a new target object has been introduced into the decomposition. Therefore, the designers can have the option to select a pre-existing DP which satisfies just the functionality of the child FR with this new target object. In other cases in which the designers have been performing decomposition, they have been refining and defining the functions performed on one target object, and they must continue this process until the refinement is complete.[15] Therefore, the mark of completion is the transition to other target objects. This is stated as a guideline:

*Distinguishing between leaf and non-leaf nodes*

Guideline: To determine when a node is a leaf, the designers have the option of considering a node to be a leaf—and thus not further decomposing it—when the target object of the sub-FR is different from that of its parent.

### 3.4.1.2 Identifying the next FR-DP pair to decompose

Consider in a general sense the nature of the task to identify the next FR-DP pair to decompose. One possibility is that the designers may generate any set of sub-FRs before any other set of sub-FRs. Alternatively there can be a sequence in which sub-FRs should be defined in order to reduce the amount of undesirable iteration described above. Here it is shown that the design matrices capture the necessary information for determining the sequence in which sub-FRs should be generated.

---

[15] A leaf DP for one designer may be the a starting parent for another, as for example, when a complete subsystem is bought from an outside vendor.

*Example Photolithography process module [adapted from [Hintersteiner (1998b)]]*

FR-DP1 "Transfer overlay pattern from reticle onto wafer surface" by means of "step-and-scan photolithography process" is broken down into sub-FRs and sub-DPs as given in table 3-4.

**Table 3-4. Decomposition of FR-DP1 "Transfer overlay pattern from reticle onto wafer surface" by means of "Step-and-scan photolithography process"**

| Index: 1.1-5 | | | |
|---|---|---|---|
| | Functional Requirements (FRs) | | Design Parameters (DPs) |
| P | *Process* | Transfer overlay pattern from reticle onto wafer surface | Step-and-scan photolithography process |
| | *Type* | *Description* | *Description* |
| 1 | *Process* | Project image from reticle plane to wafer plane | Projection optics (PO) assembly |
| 2 | *Process* | Align reticle image with wafer | Alignment system for "step-and-scan" |
| 3 | *Process* | Illuminate image on reticle | Laser illumination system (LIS) |
| 4 | *Control* | Schedule and coordinate process tasks | Photolithography process module CCA |
| 5 | *Support* | Integrate process components | Photolithography process module support framework |

The design matrix A1 is given in equation 3-3.

$$\begin{Bmatrix} \text{1.1 Project image} \\ \text{1.2 Align image} \\ \text{1.3 Illuminate pattern} \\ \text{1.4 Schedule and coordinate} \\ \text{1.5 Integrate} \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O \\ \mathbf{X} & X & O & O & O \\ X & X & X & O & O \\ X & X & X & X & O \\ X & X & X & X & X \end{bmatrix} \begin{Bmatrix} \text{1.1 PO Assembly} \\ \text{1.2 Alignment system} \\ \text{1.3 LI system} \\ \text{1.4 CCA} \\ \text{1.5 Support} \end{Bmatrix} \quad (3\text{-}3)$$

FR-DP1.1 "Project image from reticle plane to wafer plane" by means of "Projection optics (PO) assembly" is broken down into sub-FRs and sub-DPs as given in table 3-5.

**Table 3-5. Decomposition of FR-DP1.1 "Project image from reticle plane to wafer plane" by means of "Projection optics (PO) assembly"**

| Index: 1.1.1-7 | | | |
|---|---|---|---|
| | Functional Requirements (FRs) | | Design Parameters (DPs) |
| P | *Process* | Project image from reticle plane to wafer plane | Projection optics (PO) assembly |
| | *Type* | *Description* | *Description* |
| 1 | *Process* | Focus beam (4:1 magnification reduction at wafer plane) | Aspherical mirror assembly |
| 2 | *Process* | Parallelize beam | Lens groups |
| 3 | *Process* | Direct beam path | Fold mirror |

| | | | |
|---|---|---|---|
| 4 | *Process* | Vary numerical aperture (to adjust resolution and depth of focus) | Variable-iris diaphragm |
| 5 | *Process* | Correct aberrations in the image | Wafer lens group |
| 6 | *Control* | Control magnification and numerical aperture | Projection optics CCA |
| 7 | *Support* | Integrate PO subassembly in an isolated environment | Sealed chamber |

FR-DP1.2 "Align image on wafer" by means of "Alignment system for 'step-and-scan'" is broken down into sub-FRs and sub-DPs as given in table 3-6.

**Table 3-6. Decomposition of FR-DP1.2 "Align image on wafer" by means of "Alignment system for 'step-and-scan'"**

| Index: 1.2.1-5 | | | |
|---|---|---|---|
| | Functional Requirements (FRs) | | Design Parameters (DPs) |
| P | *Process* | Align image on wafer | Alignment system for "step-and-scan" |
| | *Type* | *Description* | *Description* |
| 1 | *Process* | Position and hold wafers precisely | Wafer stage |
| 2 | *Process* | Position and hold reticle precisely | Reticle stage |
| 3 | *Process* | Perform real-time motion adjustment of stages | AXIOM alignment system |
| 4 | *Process* | Perform calibration of machine-specific components | Automatic calibration system (Autocal) |
| 5 | *Control* | Schedule and coordinate process tasks | Stage CCA |

In this example, DP1.1 "projection optics (PO) assembly" has an impact on FR1.2 "align image on wafer" given by the matrix element A1(2,1), shown in bold in equation 3-3. Consider the types of changes in DP1.1 that are represented by the off-diagonal design matrix element A1(2,1). If DP1.1 changes such that a different depth of focus is used or a different image reduction is required, then the sub-FRs for position and calibration of the wafer and reticle stages can also change.

A change in DP1.1 that affects its sub-FRs has the potential to necessitate corresponding changes in DP1.2 "alignment system" and its sub-FRs in order to ensure that FR1.2 is still satisfied. The sub-FRs of FR1.2 are affected by the choice of DP1.1 as given by its sub-FRs, shown in table 3-5. The impact shows up in the sub-FRs of FR1.2 "align image on wafer" that are shown in table 3-6.

*Guidelines*

Two guidelines for identifying the next FR-DP pair to decompose are stated:

Guideline: To identify the next FR-DP pair to decompose, at each level, define sub-FRs in the order described by the design matrices.

Guideline: To identify the next FR-DP pair to decompose, there is no penalty in terms of time/iteration for decomposing one branch of the design hierarchy more deeply than another, provided that the order follows that given in the design matrices.

## 3.4.2 Generation of sub-FRs

Three of the designers' goals in generating a set of sub-FRs—developing a set of sub-FRs that is sufficient, describes the parent DP, and is solution-neutral—are ends in themselves.[16] They are the mark of the successful completion of this activity and are discussed in sections 3.4.2.1, 3.4.2.2, and 3.4.2.3 respectively. The other two—necessity and minimization—follow from the goals of the designers for one level of the design. If there are extra FRs—either individual FRs that do not belong (violating necessity) or the whole set of sub-FRs could have been defined more succinctly although individually none of the sub-FRs can be removed (thus violating minimization)—then the designers will have a more difficult time to synthesize and analyze an appropriate set of sub-DPs. Moreover, even if the designers are able to synthesize sub-DPs to meet *each* individual sub-FR, the challenge to produce a *set* of DPs which satisfies independence grows geometrically with the number of additional off-diagonal relationships between the FRs and DPs that must be considered.

**Discussion of completeness and consistency of sub-FRs**

Understanding completeness with respect to the generation of sub-FRs is not as simple as identifying a set of sub-FRs which produces the parent FR. This is because the set of sub-FRs must take into account the other factors which impinge on FR creation, namely the other sources for sub-FRs: the parent DP, the parent-level Cs, and the parent-level DM. Therefore, a *complete* set of sub-FRs is defined as a set of sub-FRs that is sufficient for producing the parent-level FR, that also satisfies the parent-level[17] Cs and the parent-level DM, and that describes the parent-level DP.

### 3.4.2.1 Developing a sufficient set of sub-FRs by understanding the sources of sub-FRs

What sources should the designers consider in generating their sub-FRs? How can the importance of the different potential sources be evaluated. Consider the following example.

*Example CCA [adapted from [Hintersteiner (1998b)]]*

FR-DP4 "Manage lithography tool" by means of "Lithography tool command and control algorithm (CCA)" is broken down into sub-FRs and sub-DPs as given in table 3-7.

**Table 3-7. Decomposition of FR-DP4 "Manage lithography tool" by means of "Lithography tool command and control algorithm (CCA)"**

| Index: 4.1.4 | Functional Requirements (FRs) | Design Parameters (DPs) |
|---|---|---|
| P | Manage lithography tool | Lithography tool command and control algorithm (CCA) |

[16] These goals include the conditions needed for FR consistency as described below in section 3.2.8 for the activity *maintain consistency between levels for FRs, DPs, DMs, and Cs*. Specifically FRs are consistent when they are sufficient, necessary, and descriptive. The first two describe consistency with respect to the parent FR and the last describes consistency with respect to the parent DP.

[17] The parent-level Cs are relevant for generating sub-FRs as opposed to the lower-level Cs that affect the selection of the child-level DPs but cannot be said to be *satisfied* by the lower-level FRs.

| | Description | Description |
|---|---|---|
| 1 | Send commands to process and transport modules | Tool controller logic |
| 2 | Coordinate process and transport modules according to predefined process recipe | Product file |
| 3 | Facilitate operator interaction with tool | Human interface (HI) |
| 4 | Operate machine through fab host computer | Fab computer interface (SECS/GEM) |

To see how these sub-FRs arise due to various sources as described above, consider the higher-level of the design hierarchy.

FR-DP0 "Print patterns onto photo-resist-coated wafers" by means of "Lithography tool" is broken down into sub-FRs and sub-DPs as given in table 3-8. The high-level constraints are given below in table 3-10.

**Table 3-8. Decomposition of FR-DP0 "Print patterns onto photo-resist-coated wafers" by means of "Lithography tool"**

| Index: 1-5 | | | |
|---|---|---|---|
| | Functional Requirements (FRs) | | Design Parameters (DPs) |
| P | *Process* | Print patterns onto photo-resist-coated wafers | Lithography tool |
| | *Type* | *Description* | *Description* |
| 1 | *Process* | Transfer and overly pattern from reticle onto wafer surface | Step-and-scan photolithography process module |
| 2 | *Transport* | Manage wafers (input/output) | (a) wafer handler system<br>(b) new wafer handler system |
| 3 | *Transport* | Manage reticles (input/output) | (a) reticle handler system<br>(b) reticle management system (RMS) |
| 4 | *Control* | Manage lithography tool | Lithography tool CCA |
| 5 | *Support* | Integrate tool subsystems | Tool support framework |

The corresponding design matrix A0 is given in equation 3-4.

$$\begin{Bmatrix} 1\ \text{Transfer pattern} \\ 2\ \text{Input/output wafers} \\ 3\ \text{Input/output reticles} \\ 4\ \text{Schecule and coordinate} \\ 5\ \text{Integrate} \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O \\ X & X & O & O & O \\ X & O & X & O & O \\ X & X & X & X & O \\ X & X & X & X & X \end{bmatrix} \begin{Bmatrix} \text{Photolithography process} \\ \text{(b) Wafer handler} \\ \text{(b) RMS} \\ \text{Tool CCA} \\ \text{Tool support} \end{Bmatrix} \quad (3\text{-}4)$$

By comparing the two levels of the design hierarchy, the following sources are identified:

- parent DP: FR4.1 arises because of the decision to use a software algorithm for scheduling the operation of the tool.

- Cs: Two types of interfaces are specified by the parent-level constraints (shown in table 3-10 below): human-machine interaction (C-6) and interaction with the fab facility according to a standard (C-5). These lead to two sub-FRs at this level: FR4.3 and FR 4.4

- parent DM elements: FR4.2 arises due to the need to schedule the resources within the tool that are responsible for each of the process and transport functions.

*Guidelines*

The several different sources of sub-FRs are as follows:[18]

- parent FR
- parent DP
- parent DM (system integration)
- parent (or higher-level) Cs
- CNs

An alternative view is that sub-FRs can be defined by considering only a subset of the above, listed sources. In particular, some researchers believe that it is possible to define sub-FRs by considering only the parent-level FRs.[19] However, this position directly contradicts the fundamental concept known as decomposition through *zigzagging*[20] as described in axiomatic design theory.

Guideline: To develop a sufficient set of sub-FRs, all potential sources of sub-FRs at a level should be considered. These include, parent FR, parent DP, parent-level Cs, parent-level DM (as a source of either potential Cs or sub-FRs), and the set of CNs.

Guideline: A good order to consider these sources is first to define sub-FRs based on knowledge of the parent DP. Second, define additional sub-FRs in accordance with the parent-level FRs and Cs. Finally, consider the parent DM and CNs.

Guideline: To develop a sufficient set of sub-FRs, a template for system design consists of sub-FRs of the form: process and transport, command and control, and support and integration.

These are illustrated in the previous example in table 3-7, table 3-8, and equation 3-4.

---

[18] Three of these were discussed in [Tate, et al. (1998)]: parent FR-DP pair, parent-level Cs, and parent-level DM. Also [Nordlund (1996) p. 35] discusses sub-FRs and Cs arising from CNs in general (though not mapped to a specific level in the design hierarchy) and Cs arising from the environment: "Selection of a customer domain generates an environment in which the design object must reside. The environment can generate constraints for the design process (e.g., geometrical and legal constraints)."

[19] [Schulz and Clausing (1998)] and [Sontow and Clausing (1993)] are examples of this view.

[20] Other theories that include zigzagging include [Marples (1961)], the WDK school [Aasland (1995), Andreasen (1991b), Andreasen (1998)], Olsson, and value engineering [Miles (1972)].

### 3.4.2.2 Developing a consistent set of sub-FRs by understanding the structure of the design hierarchy

Consistency of sub-FRs is an issue that depends on the nature of the parent DP since the question is whether the sub-FRs accurately represent the parent DP. Three general issues are discussed here in terms of their consistent representation within the design hierarchy:

- command and control: Controlling the operation of a design is important at many levels of the design hierarchy, particularly in cases in which the FRs and/or DPs that are active change over time.
- integration and support: Support for sub-DPs may be integrated into resources at parent levels (such as a power supply), but there will be sub-FRs at lower levels specifically tailored for the sub-DPs (such as connections to provide power to the lower-level DPs).
- views: At different levels of the design hierarchy, different views of the problem are important. For example a design that consists of FRs operating on a part being manufactured at the highest level may consist at lower levels of functions operating on elements within a machine itself.

*Example: CMP machine [adapted from [Melvin (1998)]]*

The high-level decomposition of a CMP machine is shown in table 3-9. The first two FRs are process modules required for planarizing and cleaning between layers the deposited surface on a wafer. FR3 is a system-level process module that transports wafers between the planarization and cleaning modules. FR4 is the CCA that is responsible for coordinating the interactions between the process and transport modules and for passing appropriate control parameters to each process and transport module. FR5 covers the necessary hardware for integrating the sub-systems.

**Table 3-9. System-level decomposition of the CMP machine**

| Index: 1-5 | Functional Requirements (FRs) | Design Parameters (DPs) |
|---|---|---|
| | *Description* | *Description* |
| 1 | Remove material | Planarization process<br>(a) 3-body abrasion<br>(b) 2-body abrasion |
| 2 | Clean wafer | Cleaning sub-system |
| 3 | Transport wafer | Wafer handler |
| 4 | Control tool | System-level CCA |
| 5 | Integrate tool subsystems | Tool support framework |

The relationships between the FRs and DPs for the top-level of the CMP machine are given in equation 3-5.

$$\begin{Bmatrix} \text{1 remove material} \\ \text{2 clean wafer} \\ \text{3 transport wafer} \\ \text{4 control tool} \\ \text{5 integrate sub-systems} \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O \\ X & X & O & O & O \\ X & X & X & O & O \\ X & X & X & X & O \\ X & X & X & X & X \end{bmatrix} \begin{Bmatrix} \text{planarization process} \\ \text{cleaning module} \\ \text{wafer handler} \\ \text{system-level CCA} \\ \text{support framework} \end{Bmatrix} \quad (3\text{-}5)$$

In a CMP machine capable of performing either two-body or three-body abrasion, the local CCA is in charge of selecting which material removal mode (that is, DP1a or DP1b) to activate. In addition, it is in charge of determining and sending commands to the material removal module (DP1a or DP1b) indicating the amount of material to be removed.

*Guidelines*

The following general guidelines are advanced as patterns that have been observed across multiple fields of design:

Guideline: To develop a consistent set of sub-FRs, an FR to control the time variation of a DP (sequencing, scheduling, etc.) should be at the same level as the DP that is changing.

Guideline: To develop a consistent set of sub-FRs, for sets of sub-FRs which change dynamically with time, the FR to coordinate their changes should be defined at their parent level.

Guideline: To develop a consistent set of sub-FRs, define support FRs at the parent level when support is carried to multiple sub-DPs on different branches. Then the connection, between the higher-level support DP and a particular DP requiring that support, will be a sub-FR.

Guideline: To develop a consistent set of sub-FRs, the character of sub-FRs will change from part flows to machine operations to machine adjustments as the decomposition progresses across succeeding levels that act on the same target object.

### 3.4.2.3 Developing a solution-neutral and minimal set of sub-FRs by understanding FRs as transformations

At the highest-level of the design decomposition, the target object is specified by the CNs. At lower-levels of the design, the target objects are determined by the higher-level DPs that have been chosen. In generating a set of sub-FRs, the designers have a choice abut the number of intermediate states of the parent-level target object, or target objects, to specify. The target object has one or more attributes or properties that are being transformed by the performance of the parent FR. For each attribute or property, the designers can specify an integer number of intermediate states–including zero–of the attribute into which the target object is transformed before the output of the parent FR is achieved.[21] Also, the designers are able to introduce new target objects into the design as sub-FRs are generated.

---

[21] See Ross for a description of the choice between two alternative pairs of transformations: "It is impossible for you to have a *choice* in how to break [a transformation] into *two* parts unless you realize that it *could* be broken into *three* parts." [Ross (1977) p. 26]

These concepts are illustrated in figure 3-6. In seeking to minimize the number of sub-FRs and to develop solution-neutral sub-FRs, the designers should seek to specify as few

- target object attributes to transform
- intermediate states of the target objects
- additional target objects introduced at a level

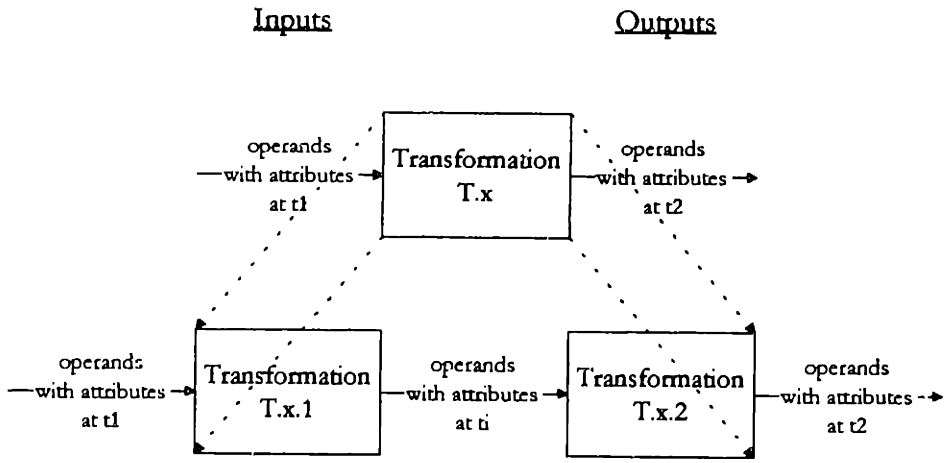


*Figure 3-6. Intermediate states of a target object in generating sub-FRs*

*Guidelines*

Guideline: To minimize the number of sub-FRs and to state the most solution-neutral FRs, specify as few attributes on the target object transformed by the parent FR as possible, and specify as few new, additional target objects as possible.

Guideline: To state the most solution-neutral FRs, articulating multiple alternative DPs serves as a check for solution neutrality.

Guideline: To minimize the number of sub-FRs, minimize the number of intermediate states of the target object that are specified.

## 3.4.3 Carrying down and refining Cs[22]

This section details the *carrying down and refining of constraints.* This is an area within axiomatic design which can benefit from a more thorough treatment than it has thus far received. This is especially true because engineers who are learning axiomatic design have found usefully distinguishing between FRs and Cs to be challenging.[23]

[22] Parts of this section on constraints are adapted from [Friedman, et al. (1998b)].

[23] See [Lindholm (1998) p. 24].

### 3.4.3.1 Documenting constraints by means of a constraints table

Table 3-10 gives generic examples of five types of constraints as applied to the top-level of a large machine tool. The refinement of these constraints to a specific sub-system is shown in table 3-11.

**Table 3-10. Generic template for listing constraints (top level)**

| Constraint Table | | | Impacts: FR | | | | |
|---|---|---|---|---|---|---|---|
| Index | Parent | Description | 1 | 2 | 3 | 4 | 5 |
| **Critical Performance Specifications** | | | | | | | |
| C-1 | Marketing | Meet throughput specifications | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-2 | Marketing | Meet process spec. (arbitrary # of instances) | ? | ? | ? | ? | ? |
| C-3 | Marketing | Meet transport spec. (arbitrary # of instances) | ? | ? | ? | ? | ? |
| **Interface Constraints** | | | | | | | |
| C-4 | Marketing | Handle customer specified target objects (arbitrary # of instances) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-5 | Marketing | Integrate tool with factory environment (host computer, air and water supply, facilities, etc.) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-6 | Marketing | Make tool "user-friendly" (ergonomics and software interfaces) | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Global Constraints** | | | | | | | |
| C-7 | Marketing | Maximize availability / reliability (minimize MTBF and minimize MTTF) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-8 | Marketing | Minimize footprint (do not exceed maximum size) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-9 | Management | Make tool serviceable (easy access for maintenance) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-10 | Marketing Management | Minimize costs (design, manufacturing, operational, maintenance, etc.) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-11 | Marketing Management | Provide ease of testability (make components compatible with standard and customer-defined tests) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-12 | Marketing | Conform to industry and safety standards | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Project Constraints** | | | | | | | |
| C-13 | Management | Integrate maximum amount of existing technology (minimize redesign of proven components, use off-the-shelf equipment whenever possible) | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Feature Constraints** | | | | | | | |
| C-14 | Marketing Management | Include specified components (arbitrary # of instances) | ? | ? | ? | ? | ? |

Table 3-11 gives examples of the application of the constraints table to the RMS design. The RMS subsystem satisfies FR-DP3 "Manage reticles" by means of a "Reticle management system (RMS)", and it is broken down into sub-FRs as has been given in table 3-3 above. The constraints from table 3-10 have been refined to apply to the RMS sub-FRs as given in table 3-11.

**Table 3-11. Constraints on decomposition of FR-DP3 "Manage reticles" using "new RMS"**

| | Parent | Constraints (C3-?) | Impacts FR3b-x | | | | |
|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 |
| | | — Critical Performance Specifications — | | | | | |
| 1 | C-1 | Exchange time (the RMS component of stage down-time ≤ 10 seconds) | | | ✓ | ✓ | |
| | | — Interface Constraints — | | | | | |
| 2 | C-4 | Accommodate multiple types of reticle carriers with minimum design effort | ✓ | | | ✓ | ✓ |
| 3 | C-5 | Transfer non-product file reticles (w/o impacting exchange time) | ✓ | | | ✓ | ✓ |
| 4 | C-5 | Accommodate multiple types of reticle stages with minimum design effort | | ✓ | ✓ | ✓ | ✓ |
| 5 | C-6 | Make tool "user-friendly" (ergonomics and software interfaces) | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | C-5 | Anticipate future FAB automation technology i.e. overhead track, AGV | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | — Global Constraints — | | | | | |
| 7 | C-7 | Meet system requirements for reliability | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | C-8 | Minimize increase to tool footprint | ✓ | ✓ | ✓ | ✓ | ✓ |
| 9 | C-12 | Protect reticles from damage or contamination | ✓ | ✓ | ✓ | ✓ | ✓ |
| 10 | C-10 | Maintain or reduce costs while increasing functionality | ✓ | ✓ | ✓ | ✓ | ✓ |
| 11 | C-12 | Conform to SEMI / industry / safety standards | ✓ | ✓ | ✓ | ✓ | ✓ |
| 12 | C-9 | Make tool serviceable (easy access for maintenance) | ✓ | ✓ | ✓ | ✓ | ✓ |
| 13 | C-11 | Provide ease of testability (make components compatible with standard and customer-defined tests) | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | — Project Constraints — | | | | | |
| 14 | C-13 | Minimize unique hardware and software components e.g. Staubli RX60CR | ✓ | ✓ | ✓ | ✓ | ✓ |
| 15 | C-14 | Production ready design w/"modular" interface available for integration in 6/99 | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | — Feature Constraints — | | | | | |
| | | N/A | | | | | |

### 3.4.3.2 Types of constraints and their impacts on the design

Table 3-10 shows a generic template for listing constraints at the highest level of the design hierarchy. In this table, the constraints are derived from the customer needs (CNs) and from considerations internal to the company. The constraints are broken down into several categories:

- critical performance specifications: constraints imposed on the attributes of the top-level target objects or on the rate at which these transforms are performed
- interface constraints: constraints imposed on the inputs and outputs that the system must accept (often at the top level, but also at lower levels if the design is a portion of an existing system)
- global object constraints: constraints with the potential to affect all DPs in the design (or some significant fraction, such as all hardware) and which are broken down in an additive way[24]
- project constraints: constraints on the development resources allowed for design or redesign, or on the decisions made across projects (standardization, etc.)
- feature constraints: constraints that apply to the choice of specific DPs within the system

Constraints can have different impacts on the design object. First, constraints can serve as filters, either allowing a DP to be chosen or necessitating its rejection. Alternatively they can serve as a source of sub-FRs, as noted above. In this latter case, each C can be directly connected to some subset of the FRs.

Some of the constraints may also arise from the CNs in such a way that they are *conditional*: they apply in case that certain DPs are chosen, but are not applicable for other selections of DPs. Furthermore some constraints are non-negotiable, while some Cs are more flexible.

The categories of constraints given above are represented in table 3-12 that shows the impact that different constraints can have on the design object. Some, like global and project constraints can potentially impact the whole design and the choice of every DP. Others, like interface and critical performance specifications, impact only a subset of the DPs.

**Table 3-12. Source and impact of constraints**

| Source \ Impact | All DPs | Few DPs |
|---|---|---|
| Direct (Customer, Marketing) | global object | critical perf. specs, interface |
| Indirect (Resource, Management) | project | feature |

Feature constraints should be avoided whenever possible since they only serve to limit the available design space. Nevertheless, they should be documented and accounted for when they are used in the design.

The guidelines stated here describe how the different constraints can impact the design.

---

[24] For a discussion of a similar idea applied to function-means trees, see [Sturges, et al. (1996) pp. 258-264].

Guideline: If the goal of the designers is to understand the impact of Cs on sub-FRs, the designers should have knowledge of *all* the constraints which apply at the parent level. At the child level, these constraints should either appear as refined constraints or be restated as sub-FRs.

Guideline: Critical performance specifications should be non-negotiable. Global and project constraints can be negotiable.

Guideline: Interface and feature constraints are conditional upon DP choices within the design object.

#### 3.4.3.3 Distinguishing Cs from FRs and conversion of Cs into sub-FRs

Differentiating Cs from FRs has proven to be one of the most difficult tasks in learning and teaching axiomatic design. The following guidelines describe the conversion of Cs into sub-FRs and the means to distinguish constraints that are converted from those that are not.

Guideline: To convert Cs from parent level to sub-FRs and distinguish Cs from FRs, *all* critical performance specifications should be refined as sub-FRs at lower levels of the design hierarchy. That is, they *will* become a sub-set of low-level sub-FRs, instead of remaining constraints.

Guideline: To convert Cs from parent level to sub-FRs and distinguish Cs from FRs, interface constraints will be refined into sub-FRs, assuming they are applicable.

Guideline: To convert Cs from parent level to sub-FRs and distinguish Cs from FRs, global object constraints will *not* be refined into sub-FRs; they will remain constraints even at the lower levels of the design hierarchy.

Guideline: To convert Cs from parent level to sub-FRs and distinguish Cs from FRs, project constraints will *not* be refined into sub-FRs; they will remain constraints even at the lower levels of the design hierarchy.

Guideline: Project constraints can be conditional upon the DP choices within the design object.

### 3.4.4 Physical integration of DPs

Integrating DPs into physical parts involves analyzing a set of DPs and deciding which DPs are integrated into which parts. The designers *have already selected the DPs.* Therefore, the question becomes how to group them. This involves grouping DPs from all over the design tree: specifically parents with children and DPs on multiple branches of the tree. At higher-levels, DPs may be considered as assemblies of parts; at lower levels, they may be part attributes, features, etc.; and the DPs at higher-levels are *not necessarily* assemblies of the DPs below them

*Example [adapted from [Do (1998)]]*

FR-DP1.1.4.1.2 "Support FR generation" by means of "Structure for FRs" is broken down into sub-FRs and sub-DPs as given in table 3-13.

**Table 3-13. Decomposition of FR-DP.1.1 "Support FR generation" by means of "Structure for FRs"**

| Index: 1.1.4.1.2.1-3 | Functional Requirements (FRs) | Design Parameters (DPs) |
|---|---|---|
| P | Support FR generation | Structure for FRs |
| | *Description* | *Description* |
| 1 | Describe the FRs | Attributes for FR |
| 2 | Display as graphics | Attributes sent to GUI |
| 3 | Manage inserted data | FR methods |

The corresponding design matrix A1.1.4.1.2 is given in equation 3-6.

$$\begin{Bmatrix} \text{Describe FR} \\ \text{Display as graphics} \\ \text{Manage inserted data} \end{Bmatrix} = \begin{bmatrix} X & O & O \\ O & X & O \\ X & X & X \end{bmatrix} \begin{Bmatrix} \text{FR attributes} \\ \text{Attributes sent to GUI} \\ \text{FR methods} \end{Bmatrix} \quad (3\text{-}6)$$

FR-DP1.1.4.1.2.3 "Manage inserted data" by means of "FR methods" is broken down into sub-FRs and sub-DPs as given in table 3-14.

**Table 3-14. Decomposition of FR-DP.1.1 "Manage inserted data" by means of "FR methods"**

| Index: 1.1.4.1.2.3.1-4 | Functional Requirements (FRs) | Design Parameters (DPs) |
|---|---|---|
| P | Manage inserted data | FR methods |
| | *Description* | *Description* |
| 1 | Store new data | "New" method |
| 2 | Support data change | "Change" method |
| 3 | Support deleting | "Delete" method |
| 4 | Support editing | "Edit" methods |

The corresponding design matrix A1.1.4.1.2.3 is given in equation 3-6.

$$\begin{Bmatrix} \text{Store new item} \\ \text{Support changes} \\ \text{Support deleting} \\ \text{Support editing} \end{Bmatrix} = \begin{bmatrix} X & O & O & X \\ O & X & O & X \\ O & O & X & X \\ O & O & O & X \end{bmatrix} \begin{Bmatrix} \text{"New" method} \\ \text{"Change" method} \\ \text{"Delete" method} \\ \text{"Edit" method} \end{Bmatrix} \quad (3\text{-}7)$$

The sub-FRs and sub-DPs associated with "supporting FR generation" are similar to those for "supporting DP generation", etc. Therefore, looking at table 3-15, since the target objects are different, the parent DPs may be integrated into one resource. In this case, the resource corresponds to an parent object class capable of handling the design elements: FRs or DPs, which provides operations such as displaying the element graphically and managing inserted data.

Table 3-15 shows when DPs may be physically integrated into common resources. A *resource* for a leaf DP is defined as the physical part or parts with which that DP is embodied. For

higher-level DPs, the resource associated with each is the collection of resources associated with its children.

**Table 3-15. Levels at which DPs may be integrated into physical resources**

| Artifact view \ Part view | Same target object | Different target object |
|---|---|---|
| Same time | 1. parent | 1. parent |
| Different time | 2. leaf | 1. parent |

The notation in table 3-15 corresponds to the following:

1. Higher-level (parent) resources can be integrated, but not leaves.
2. Resources can be integrated down to leaf level.

Given this definition of a resource, there is not in general a one-to-one correspondence then between resources and DPs. A DP *can* be embodied in a part, but a DP can also be embodied in a geometrical feature of a part, a material property, a feature between two or more parts (an "organ"), a location of a feature, the number of features, etc. Likewise for software design, a DP can be a collection of data (variables in objected-oriented modeling), an algorithm (method in objected-oriented modeling), a pointer, a function call, etc.

The following guidelines state the information presented in table 3-15.

Guideline: DPs which perform the same FRs on the same target object at different times (from the point of view of the machine), in different places in the design hierarchy may be integrated physically into the same unit.

Guideline: For FRs which are operations to be performed simultaneously (from the point of view of the machine), at sufficiently low levels of the design hierarchy, the resources embodying the DPs should consist of separate components.

## 3.4.5 Ensuring consistency between levels: FRs, DPs, DMs, Cs

The issues relating to consistency of sub-FRs and the consistency of Cs have already been discussed in the sections on generating sub-FRs (section 3.4.2) and refining Cs (section 3.4.3). The key issues are summarized as follows.

### 3.4.5.1 Consistency of sub-FRs

These goals for generating sub-FRs provide the conditions needed for FR consistency. Specifically FRs are consistent when they are sufficient, necessary, and descriptive. The first two describe consistency with respect to the parent FR and the last describes consistency with respect to the parent DP.

### 3.4.5.2 Consistency of Cs

Accurately mapping constraints from the parent level to the child level describes consistency with respect to constraints. This means that each C at the parent level applies to at least one sub-FR or is converted into a sub-FR itself and that each C at the child level is derived from a C at the parent level.

### 3.4.5.3 Consistency of DM elements

The following guidelines are advanced to assist the designers in maintaining consistency in terms of design matrix elements as the design progresses between the levels of the design hierarchy:

Guideline: The sub-DPs within one design equation will not necessarily be at the same level of abstraction as each other. Some will require more decomposition than others.

Guideline: The designers have a choice about the location of off-diagonal Xs in design matrices when the X represents an interface between two processes.

Guideline: Given the choice of where to place an off-diagonal term when considering two FRs and DPs, place it so the most technically challenging part of the design is done first. That is, the interface between two sub-systems should be defined with the sub-system that is more difficult to design.[25]

There are three options the designers have in ensuring consistency. First, they can ignore the issue of consistency altogether. Perhaps they can assume that because a single designer is working on a project that he knows everything about the design and that his understanding of the FRs and DPs and decisions that were made remain consistent throughout the design effort. As an alternative, the designers can go through and compare new decisions based on previous decisions. This can be done for all new decisions against all previous ones. Obviously, this grows exponentially, and for any realistically sized project, even if effective, would be unmanageable.[26] Third, the designers can identify decisions that are more likely to lead to inconsistencies and to check only those, and structure their decisions in a way that minimizes the risk of creating inconsistencies.

This may involve some modification in the way that decomposition is carried out. A solution put forth here is to define sub-FRs on different branches of the design hierarchy—according to the design matrices as described earlier—and to define all sub-FRs for one design matrix before selecting any of the sub-DPs. This is illustrated in equations 3-8 and 3-9. The reason this minimizes inconsistencies is that the dependencies represented by the off-diagonal terms can be adequately captured by comparing the parent-level DPs and the newly defined sub-FRs. Once the sub-FRs for *DPn.1*, *DPn.2*, and *DPn.3* have been defined in the sequence given by the parent-level design matrix *[An]*, then the sub-DPs, for each of the main-diagonal boxes may be selected in any order.

$$\begin{Bmatrix} FRn.1 \\ FRn.2 \\ FRn.3 \end{Bmatrix} = \begin{bmatrix} X & O & O \\ X & X & O \\ X & X & X \end{bmatrix} \begin{Bmatrix} DPn.1 \\ DPn.2 \\ DPn.3 \end{Bmatrix} \tag{3-8}$$

[25] Note that in this case there is a relationship between the pair of FRs and DPs. The only question is where this relationship is placed in the design matrix. The design decision is made about which DP is designed first.

[26] Researchers that have tried this approach include these: [Albano (1992), Baldwin (1994), Lee (1999)].

$$\begin{Bmatrix} FRn.1.1 \\ FRn.1.2 \\ FRn.1.3 \\ \hline FRn.2.1 \\ FRn.2.2 \\ FRn.2.3 \\ \hline FRn.3.1 \\ FRn.3.2 \end{Bmatrix} = \left[\begin{array}{c|c|c} X & O & O \\ X & O & O \\ X & O & O \\ \hline X & X & O \\ X & X & O \\ O & X & O \\ \hline X & X & X \\ O & X & X \end{array}\right] \begin{Bmatrix} DPn.1 \\ DPn.2 \\ DPn.3 \end{Bmatrix} \tag{3-9}$$

Once a DM has been determined, consistency can be checked. If the designers find an inconsistency, they have several activities in the roadmap they can progress to. For example, they can return to the selection of DPs, and they can return to dimensioning, layout, integration, etc. Alternatively they can simply update the DM and move on. This is appropriate when the corrected DM is not coupled. In this case, continuing with the design, that is, not going back would probably be the better choice.

## *3.5 Summary*

An objective of decomposition is to enable the designers to satisfy the axioms at each level of the design hierarchy more quickly and more easily. This is done through articulating a set of activities that are performed that relate to multiple levels of the design hierarchy. Once these activities are identified, the goals for each are stated, and the corresponding options the designers have in performing the activities are given. Then, in the spirit of axiomatic design, rules are established which relate the options to the goals for each activity. This way the designers have a practical aid in making decisions within each of the activities.

Therefore the designers are able to focus more of their attention, energies, and resources towards the synthesis and analysis activities, the heart of design, aided by the design axioms, instead of becoming bogged down elsewhere in the details of the design process.

The focus in this chapter is on developing a palette of activities and guidelines that can be applied in decomposition. Thus this work has a practical slant towards addressing some of the questions that arise in applying axiomatic design to real, industrial design tasks. This is apparent by looking at the questions that have been identified and answered for each of the design activities.

Specifically the following contributions are made:

- sequencing the decomposition: guidelines for identifying leaf options and guidelines for guiding the decomposition process (what factors and what order is important in generating sub-FRs)
- generating sub-FRs: guidelines for developing a complete set of sub-FRs
- constraints: tools for documenting constraints, guidelines for identifying the impacts of constraints (such as which ones become sub-FRs and which ones do not)
- physical integration: guidelines for when DPs can be integrated into one unit
- consistency: a procedure for generating sub-FRs consistently

- overall roadmap and method: understanding the relationship between activities in design and decomposition, identifying specific goals, and mapping activities to available design theories

The criteria for evaluating the activities and guidelines is whether they provide a good means for developing theory about decomposition. That is, to evaluate this work, the question needs to be asked: does this constitute a progressive research program for understanding decomposition as performed in axiomatic design?[27] This involves three sub-questions:

- theoretically progressive: Is the theory original in raising new questions and providing new explanations?
- empirically progressive: Can at least some of the explanations be empirically tested, and where they have been do they correlate with reality?
- heuristically progressive: Can the theory be refined, expanding its theoretical scope, without throwing out its basic concept, and is it reasonable to expect that the theory will be fruitful in producing *new* explanations and *new* verified results?

While the theory developed in this chapter—consisting of the set of activities and guidelines—is intended to be generally applicable to all designs consisting of multiple levels of design hierarchy, the guidelines are biased, if at all, towards system design. This is because of the nature of the project cases that were examined in this work. However, this is acceptable because this the theoretical area covered here is one that can benefit from theoretical development and because the systems view of design problems can be applied to many design problems and is commonly encountered in real-life, industrial tasks.

One application viewpoint for this work that has been kept in mind is that of design software environments, namely the use of computer software as a support tool for designers.

[27] This question will be elaborated in more detail in chapter 5 which establishes methods and criteria for evaluating the case studies related to this work.

**Table 3-16. Guidelines presented for the decomposition activities**

| | | |
|---|---|---|
| Sequence the decomposition | | |
| distinguish between leaf and non-leaf nodes | To determine when a node is a leaf, the designers have the option of considering a node to be a leaf—and thus not further decomposing it—when the target object of the sub-FR is different than that of its parent. | A1 |
| identify the next FR-DP pair to decompose | To identify the next FR-DP pair to decompose, at each level, define sub-FRs in the order described by the design matrices. | A2 |
| | To identify the next FR-DP pair to decompose, there is no penalty in terms of time/iteration for decomposing one branch of the design hierarchy more deeply than another, provided that the order follows that given in the design matrices. | A3 |
| Generate sub-FRs | | |
| develop a sufficient set | To develop a sufficient set of sub-FRs, all potential sources of sub-FRs at a level should be considered. These include, parent FR, parent DP, parent-level Cs, parent-level DM (as a source of either potential Cs or sub-FRs), and the set of CNs. | B1 |
| | A good order to consider these sources is first to define sub-FRs based on knowledge of the parent DP. Second, define additional sub-FRs in accordance with the parent-level FRs and Cs. Finally, consider the parent DM and CNs. | B2 |
| | To develop a sufficient set of sub-FRs, a template for system design consists of sub-FRs of the form: process and transport, command and control, and support and integration. | B3 |
| describe parent DP | To develop a consistent set of sub-FRs, an FR to control the time variation of a DP (sequencing, scheduling, etc.) should be at the same level as the DP that is changing. | B4 |
| | To develop a consistent set of sub-FRs, for sets of sub-FRs which change dynamically with time, the FR to coordinate their changes should be defined at their parent level. | B5 |

| | | |
|---|---|---|
| | To develop a consistent set of sub-FRs, define support FRs at the parent level when support is carried to multiple sub-DPs on different branches. Then the connection, between the higher-level support DP and a particular DP requiring that support, will be a sub-FR. | B6 |
| | To develop a consistent set of sub-FRs, the character of sub-FRs will change from part flows to machine operations to machine adjustments as the decomposition progresses across succeeding levels that act on the same target object. | B7 |
| state solution-neutral sub-FRs | To minimize the number of sub-FRs and to state the most solution-neutral FRs, specify as few attributes on the target object transformed by the parent FR as possible, and specify as few new, additional target objects as possible. | B8 |
| | To state the most solution-neutral FRs, articulating multiple alternative DPs serves as a check for solution neutrality. | B9 |
| develop a necessary set | | |
| minimize the number of sub-FRs | To minimize the number of sub-FRs, minimize the number of intermediate states of the target object that are specified. | B10 |
| Carry down and refine Cs | | |
| understand impacts of Cs on sub-FRs | If the goal of the designers is to understand the impact of Cs on sub-FRs, the designers should have knowledge of *all* the constraints which apply at the parent level. At the child level, these constraints should either appear as refined constraints or be restated as sub-FRs. | C1 |
| | Critical performance specifications should be non-negotiable. Global and project constraints can be negotiable. | C2 |
| | Interface and feature constraints are conditional upon DP choices within the design object. | C3 |
| convert Cs to sub-FRs | To convert Cs from parent level to sub-FRs and distinguish Cs from FRs, *all* critical performance specifications should be refined as sub-FRs at lower levels of the design hierarchy. That is, they *will* become a sub-set of low-level sub-FRs, instead of remaining constraints. | C4 |

| | | |
|---|---|---|
| | To convert Cs from parent level to sub-FRs and distinguish Cs from FRs, interface constraints will be refined into sub-FRs, assuming they are applicable. | C5 |
| | To convert Cs from parent level to sub-FRs and distinguish Cs from FRs, global object constraints will *not* be refined into sub-FRs; they will remain constraints even at the lower levels of the design hierarchy. | C6 |
| | To convert Cs from parent level to sub-FRs and distinguish Cs from FRs, project constraints will *not* be refined into sub-FRs; they will remain constraints even at the lower levels of the design hierarchy. | C7 |
| | Project constraints can be conditional upon the DP choices within the design object. | C8 |
| develop a complete set | | |
| Integrate sub-DPs | | |
| assign sub-DPs to components and resources | DPs which perform the same FRs on the same target object at different times (from the point of view of the machine), in different places in the design hierarchy may be integrated physically into the same unit. | D1 |
| | For FRs which are operations to be performed simultaneously (from the point of view of the machine), at sufficiently low levels of the design hierarchy, the resources embodying the DPs should consist of separate components. | D2 |
| Check consistency of sub-FRs, sub-DPs, sub-DMs, and Cs | | |
| determine if DM relationships have been identified and documented | The sub-DPs within one design equation will not necessarily be at the same level of abstraction as each other. Some will require more decomposition than others. | E1 |
| | The designers have a choice about the location of off-diagonal Xs in design matrices when the X represents an interface between two processes. | E2 |
| | Given the choice of where to place an off-diagonal term when considering two FRs and DPs, place it so the most technically challenging part of the design is done first. That is, the interface between two sub-systems should be defined with the sub-system that is more difficult to design. | E3 |

# CHAPTER 4: SYSTEM DESIGN TOOLS AND APPLICATIONS

## *4.1 Introduction*

The purpose of this chapter is to introduce additional tools that have been developed to deal with specific issues in system design. In section 2.5.1 in chapter 2, system design is defined; the features which distinguish it from other types of design are identified; and the se features motivate the need for particular tools for empowering designers to perform system design, supplementing the more generally applicable tools.

Chapter 3 is concerned with the general activities encountered in decomposition and integration; these activities occur between levels of the design hierarchy. Descriptions of the activities are given along with guidelines for the designers in performing the activities. These rules are generally applicable to any design problem in which decomposition and multiple layers of the design hierarchy are used to model the design object.

In this chapter, however, tools are presented which address situations that are encountered in specifically system design. These tools are, therefore, not as generally applicable; instead, their application is restricted to a particular class of design tasks. The tools are defined in section 4.2 by considering concepts of modularity in axiomatic design. Then, descriptions of their applications are given in section 4.3.

## *4.2 Concepts of modularity in AD*

The topic of this section is modularity which, according to Webster's, is the use of "standardized units or dimensions" as a means for providing "flexibility or variety in use" [Merriam-Webster's (1996)]. Specifically, *flexibility* is defined in AD terms as desired variety in inputs and outputs in performing functional requirements (FRs),[1] and *modularity* is one strategy for providing this desired flexibility.

The types of flexibility that are enabled by modular design include separate testing of functions, synthesis of products with custom functionality using new combinations of existing parts, and ease of product change. These benefits of modularity that are espoused by proponents[2] do not correspond to a single uniform set of product characteristics. This is shown in this section by describing systems using the axiomatic design framework.

In the discussion that follows, the designs considered are assumed to follow the independence axiom of AD. Namely, it is assumed that there is a one-to-one correspondence between FRs and DPs at each node of the design hierarchy and that each design matrix is either decoupled or uncoupled.

Three types of modularity and associated metrics are defined from an AD perspective:

- *Resource modularity* characterizes the ease of manufacturing. (section 4.2.1)

---

[1] According to Webster's *flexibility* is "characterized by a ready capability to adapt to new, different, or changing requirements". [Merriam-Webster's (1996)]

[2] A wide range of possible benefits of modularity are given in [Erixon (1998)] and [Ulrich and Tung (1991)].

- *Operational modularity* characterizes the extent to which the users have options in the operation of the system. (section 4.2.2)
- *Interfacial modularity* characterizes the amount of design effort embodied in an engineering change order (ECO). (section 4.2.3)

The first corresponds to the modularity of the DPs, the second to the operation of the FRs, and the third to the modularity described within the design matrices.

### 4.2.1 Resource modularity

The metric for resource modularity correlates the resources (for example, assemblies, physical parts, etc.) within the system with their DPs and their desired functionality. As assumed above, the design meets the independence axiom, and there is a one-to-one correspondence between FRs and DPs as required by axiomatic design. Therefore, *resource modularity* correlates *DPs with their physical embodiments.*[3] It consists of two parts: one that correlates the sub-DPs with their parent DP, and one that correlates the sub-DPs with each other.

A *resource* for a leaf DP is defined as the physical part or parts with which that DP is embodied. In the case of software, the resource would be the block of code (function, procedure, etc.) that implements that DP. The resources for *DPx.i* and *DPx.j* are represented in equations 4-1 and 4-2 where $p_1$, $p_2$, $p_3$, etc. are the physical parts associated with *DPx.i* and *DPx.j*.

$$R(DPx.i) = \{p_1, p_2\} \tag{4-1}$$

$$R(DPx.j) = \{p_2, p_3\} \tag{4-2}$$

Given this definition, therefore, there is not, in general, a one-to-one correspondence between resources and DPs. A DP *can* be embodied by a part, but a DP can also be embodied in a geometrical feature of a part, a material property, a feature between two or more parts, a location of a feature, the number of features, etc. Likewise for software design, a DP can be a collection of data (or attributes in objected-oriented modeling), an algorithm, a pointer, a function call, etc.

$$DPx.i \equiv \{a, b, c\} \tag{4-3}$$

$$DPx.j \equiv \{d, e\} \tag{4-4}$$

$$a, b \in p_1, c, d \in p_2, e \in p_3 \tag{4-5}$$

To simplify notation, a collection of parts can be termed collectively, an assembly,[4] as in equation 4-6.

$$a_1 = \{p_1, p_2, \ldots\} \tag{4-6}$$

---

[3] The resource modularity metric is the one that most closely corresponds to those proposed by other researchers. An important difference, however, is that other researchers do not make a distinction between DPs and resources.

[4] An *assembly* is defined as an "integrated set of components and/or subassemblies that comprise a defined part of a subsystem". [INCOSE (1998) p. 2-5]

The resource associated with each higher-level DP is the collection of resources associated with its children. This is shown in equation 4-7.

$$R(DPx) \equiv \bigcup_i R(DPx.i) = \{p_1, p_2, p_3, \ldots\} \tag{4-7}$$

The above discussions of resources are used to define two metrics of resource modularity.

The first measure, given in equation 4-8, looks at the overlap between resources for sibling DPs. It is a measure of the modularity of an individual DP: how separable the DP is from other DPs at its level.

$$M_{sR}(DPx.i) \equiv 1 - \frac{\left| \bigcup_j [R(DPx.i) \cap R(DPx.j)] \right|}{|R(DPx.i)|} \tag{4-8}$$

The "ideal" case in terms of modularity, the one-to-one assignment of physical entities to functions that is often referred to in the literature,[5] corresponds to the metric shown in equation 4-8 being equal to 1 for all DPs at a node. In this, there is no overlap between the assignment of parts among the different DPs so that equation 4-9 holds.

$$R(DPx.i) \cap R(DPx.j) = \{\}, \forall i \neq j \tag{4-9}$$

This metric describes how easy or how difficult it is to separate DPs (and FRs) physically from one another.

The second measure, given in equation 4-10, compares the number of instances of resources with the number of DPs at a node. This is important because a DP can be embodied in multiple instances, for example, to ensure adequate throughput.

$$M_{pR}(DPx) \equiv \frac{|R(DPx)|}{|\{DPx.i\}|} \tag{4-10}$$

### 4.2.2 Operational modularity

The metric for *operational modularity* provides a measure of the extent to which the user has freedom in the operation of the system. Figure 4-1 shows that the performance of FRs over time can be viewed from two perspectives. The first is that of a part, or data, that is passing through the system. This is the perspective that should be considered when the top-level requirements are being defined: In solution-neutral terms what are the functions that are to be performed on the target object? The second perspective is that of the system that is designed itself: What are the different FRs that are performed at any one time, and how does the system change between these?

As can be seen by looking at figure 4-1, when considered from the perspective of a part, a set of FRs can be performed in series—first one operation, then another—and when viewed from the perspective of the system, *the same set of FRs* can be performed simultaneously—for example, the system is processing multiple parts all at once.

---

[5] See, for example [Ulrich (1995) p. 422].

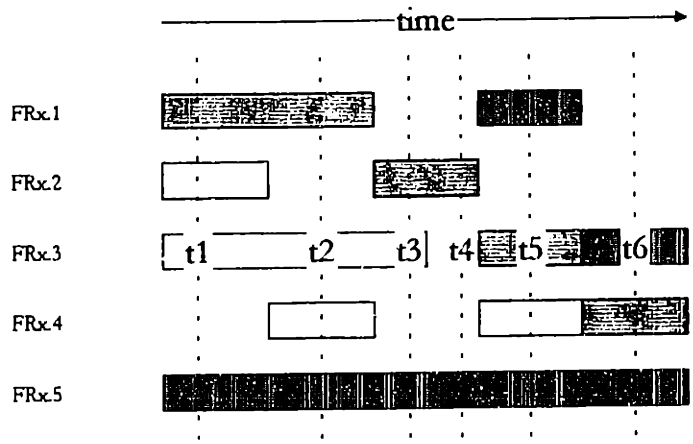


*Figure 4-1. Views of FR time variation (target objects shown as different colors, times shown by dashed lines)*

This is affected by the choice of DPs and their physical integration, as shown below. Alternative embodiments of the design solution, with different configurations and integrations of DPs, can have different values of the metric for operation modularity, given by equation 4-11. The metric compares the complete sets of FR combinations with those that are allowed based on the decomposition to some point in the design.

$$M_{sO} \equiv \frac{C_{allow}(\{FRn\})}{C_{all}(\{FRn\})} \tag{4-11}$$

This measure examines the modularity of a set of sub-FRs with respect to machine states. A machine is more modular the more it performs multiple tasks simultaneously on different target objects. That is, each target object can be processed independently. For software, the measure examines the extent to which multiple sets of data can be processed simultaneously. An example of a machine exhibiting this type of modularity is a track machine for coating and developing semiconductor wafers. It is composed of many "modules" each of which can simultaneously perform some operation on a wafer: heating, spin coating, chilling, transport, etc.

The designers are able to ask two types of questions with respect to this type of modularity:

- Is it *possible* to design the machine such that a set of functions can be performed simultaneously, for example, on multiple target objects?
- *Has* the machine been configured such that a set of functions can be performed simultaneously?

When considering the design from the point of view of the target objects which are flowing through the system, the parts or data that are being processed, the important consideration is the ways in which the different FRs can be combined to produce the desired output. In the case of the machine or system view above, the concern is with the *combinations* of FRs at different times. In considering the flow of the part, however, the concern is with different *permutations* of FRs over time. This is given in equation 4-12.

$$M_{oO} \equiv \frac{P_{allow}(\{FRn\})}{P_{all}(\{FRn\})} \tag{4-12}$$

The purpose of this measure is to examine the modularity of the set of FRs with respect to a process recipe. That is, when considered from the point of view of an target object that is being transformed (such as a part progressing through a series of manufacturing steps or data that is operated on by a series of procedures), a set of FRs is more modular if its sub-FRs can be separated into individual steps, that is, if these steps do not need to be performed simultaneously. This measure of modularity is useful in developing the software controls for a set of sub-FRs.

#### 4.2.2.1 Time Variation of FRs and DP resource allocation

This section covers tools for the resource allocation of DPs throughout the design hierarchy. That is, *time variation in FR performance and DP resource allocation* is here incorporated into the design hierarchy, thus extending the work of Suh[6] in which the FRs satisfied at any given time can be a only subset of the total set of FRs. Considerations of time variation are shown to have substantial impacts on two areas of the design:

- allocation of DP resources through command and control algorithms (CCAs) (section 4.3.1)
- physical integration of DPs (section 4.3.2)

*Time variation of FRs and DP resource allocation* is defined as keeping track of the sequence in which FRs are to be performed. This theoretical concept was introduced in [Suh (1995b)].[7] The theory presented in [Suh (1995b)] primarily concerns variations of FRs over time that occur in a way that is unknown to the designers a priori.[8] When the FRs that a design must satisfy are not known, the ability of the designers to select the best design becomes limited.

In this work, the concept of FRs changing over time is expanded to include commonly encountered systems in which designers have at least some knowledge of how the FRs vary with time. In this thesis, it is proposed to document, or model, the order of performance of sets of FRs *at each level of the design hierarchy, in all system designs.* This information is *part of the definition of the sub-FRs,* and subsequent design decisions should not conflict with these statements of the design task.

The time variation of FRs is an important consideration in many designs that needs to be accounted for in the system architecture. This section describes the variation of the sets of FRs where the FRs and the DPs to satisfy them are changed dynamically during the operation of the system.[9] This concerns cases in which the FRs themselves vary as a function of time. Cases in which just the target values of the FRs change with time is a separate topic covered in section 4.2.3.1.

---

[6] This has been termed *large-system design* [Suh (1995b)] or *large, flexible system design* [Suh (1997)].

[7] It has also been described in [Suh (1997), Suh (1998a), Suh (1999)].

[8] [Suh (1998a)] source presents 9 theorems. These are included in appendix 4, for the interested reader. Of these 9 theorems, 4 of them (theorems 2, 3, 4, and 5) deal with the distinction between systems in which the time variation of the FRs is known a priori versus systems in which it is not.

[9] This can mean that the DPs are allocated completely dynamically to meet an FR set that was unknown until that time. This is an extreme case, however, and more likely the designers have at least some knowledge of the FR variation beforehand.

*Sequential and simultaneous operation and distinction between viewpoints*

In this section, a distinction is made between FRs which must be performed simultaneously from those which are performed in a particular sequence (see figure 4-2).

For example, a process recipe for manufacturing semiconductor devices on silicon wafers can be described in terms of a series of process steps. These can be a set of FRs from the point of view of the device to be produced. Each step must be performed correctly in order for the device to be manufactured properly, and the success of a following step is likely contingent on those which proceeded it. This is illustrated with a right triangle in figure 4-2. However, within a given process step, for example, several FRs can need to be performed simultaneously in order for the step to be successfully carried out; this is shown by a diamond in figure 4-2.

In other cases, the order of FR performance does not matter. The performance of FRs in a particular sequence or not, is different than the type of information captured in the design matrix that deals with the effects of changing DPs on the satisfaction of the FRs. In general, the progress in the design decomposition is from a sequence of steps to lower-levels where FRs are performed simultaneously on the same part. This transition can occur over many levels, and the number of levels over which this transition occurs does not need to be consistent from process step to process step.

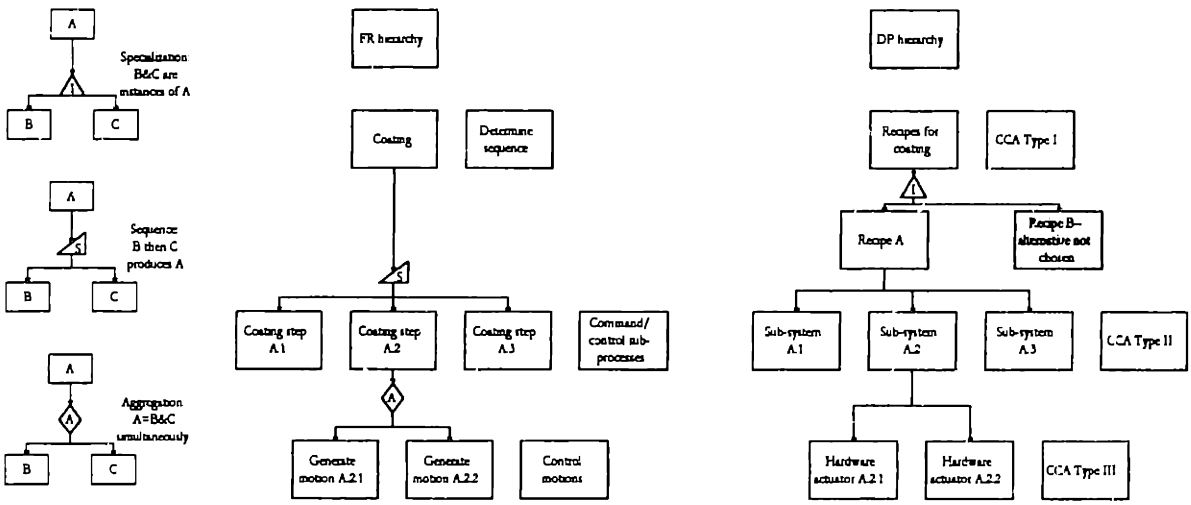


*Figure 4-2. FRs can need to be preformed sequentially or simultaneously*

*Scale of uncertainty in FR time variation*

Consider equation 4-13 in which a set of $n$ FRs is satisfied by a set of $n$ DPs. In this case, all of the FRs do not need to be satisfied simultaneously. Therefore they can be separated into groups as in expression 4-14, and a design equation can be constructed for each of them as shown in equation 4-15, so that independence can be maintained at each time. However, if the designers *do not* keep track of time considerations when defining FRs, the sets of FRs considered at each level of the design hierarchy are likely to be a mixture of choices of process recipes, process steps to be performed serially, and/or operations to be performed in parallel.

$$\begin{Bmatrix} FR_1^{t1} \\ \vdots \\ FR_n^{tj} \end{Bmatrix} = \begin{bmatrix} X & O & O \\ X & X & O \\ O & X & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ \vdots \\ DP_n \end{Bmatrix} \tag{4-13}$$

$$\begin{Bmatrix} FR_1 \\ \vdots \\ FR_a \end{Bmatrix}(t_1) \cdots \begin{Bmatrix} FR_1 \\ \vdots \\ FR_j \end{Bmatrix}(t_i) \cdots \begin{Bmatrix} FR_1 \\ \vdots \\ FR_n \end{Bmatrix}(t_z) \tag{4-14}$$

$$\begin{Bmatrix} FR_1 \\ \vdots \\ FR_j \end{Bmatrix}(t_i) = \begin{bmatrix} X & O & O \\ X & X & O \\ O & X & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ \vdots \\ DP_j \end{Bmatrix}(t_i) \tag{4-15}$$

Within the structure of the system architecture, the variation of FRs can be modeled in a variety of ways. In order of increasing uncertainty, these are the following:

1. FRs fixed for all times: This is the simplest and most basic case, in which the FR and DP hierarchies and design matrices are known, and they remain constant over the lifetime of the system.

$$\{FRn\}(t) = \mathbf{c} \tag{4-16}$$

2. Repeating sets of FRs (known sets and known times): Some systems are required to perform different sets of FRs over their life, but to do so in a cyclic manner.

$$\{FRn.i\}(t) = \mathbf{c}(n, [t] \bmod \tau) \tag{4-17}$$

3. Known combinations of FRs—lasting unknown times: This describes systems that are capable of performing in different modes of operation. These are systems that fulfill known sets of FRs, but the duration of any given active set is unknown. Since there are different modes of operation for such systems, some means of signaling the change from one mode to another must be provided, possibly by user input, or by a command and control algorithm, CCA.

$$\{FRn.i\}(t) \in \{\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_m}\} \tag{4-18}$$

4. Unknown combinations of FRs from a predefined database: This describes the case where a set of permissible FRs and DPs is known beforehand, but the specific operations required for a given target object or at a given time varies. The set that is performed is drawn from a predefined database that describes *all* the possible functions that can be performed by the system.

$$\begin{gathered} \{FRn.i\}(t) = \{FRn.1, FRn.2, \ldots, FRn.m\} \\ FRn.i \in \{FR1, FR2, \ldots FRl\} \end{gathered} \tag{4-19}$$

5. FRs unknown *a priori*: This is the most general case of FR flexibility. In this situation, the FRs are allowed to vary without limit, and the system must be designed to adapt to a highly changing environment. Theorems for this type of case have been discussed in [Suh (1995b), Suh (1997)].

Cases 3 and 4 are somewhat similar, in the sense that there are known FRs lasting for unknown times. However, the ways that these two cases are handled from a command or a design point of view are different. Given the information that is captured about the time variation of the FRs, command and control algorithms can be constructed to control the operation of the FRs at a given level of the design hierarchy.

### 4.2.3 Interfacial modularity

The third measure of modularity is the one with the most significance. It is derived by looking at the relationship between levels in the design hierarchy. One of the expressed benefits of modularity is that it empowers the designers to break the system design into pieces that can be synthesized independently.[10]

The question is this: what characteristics that are represented in a design matrix make a portion of the design easier to design independently of the decisions made elsewhere in the design? The answer to this question that gives a measure for development modularity is motivated by considering a design matrix as shown in figure 4-3.

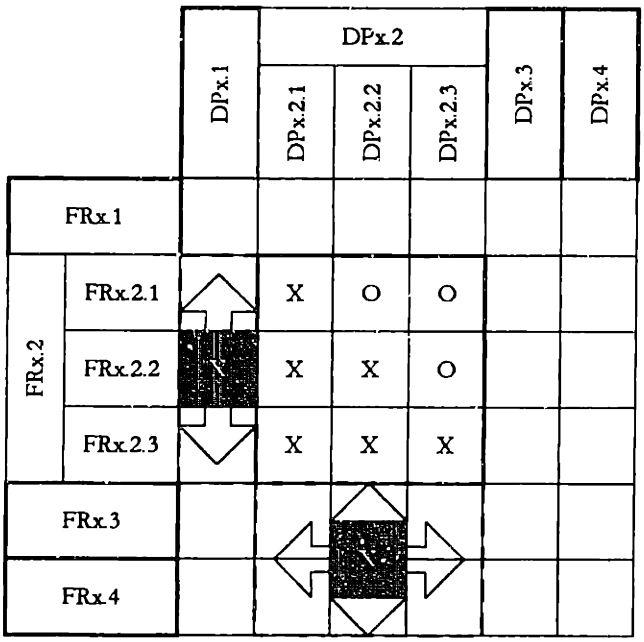


*Figure 4-3. Factors influencing interfacial modularity*

Figure 4-3 shows a decoupled (in this case, lower trianguiar) design matrix. This design is acceptable from the point of view of the independence axiom because design changes made in the correct sequence can satisfy the FRs. The design is not completely independent in the sense that the designers can make decisions about their portion of the design in isolation without impacting the work of others. This is indicated by the shaded Xs in the figure. If there were no off-diagonal terms, the design matrix would be considered uncoupled, and the design tasks *could* be done independently. Nevertheless, in the more general case, the design is decoupled, and through the off-diagonal terms, the meaning of interfaces are defined.

---

[10] See, for example [Ulrich and Tung (1991) p. 75].

The off-diagonal terms correspond to interfaces of some sort between the different DPs that make up the design at this level of the design hierarchy[11]. The relative positions and numbers of these Xs in the design matrix impacts the ease with which decisions can be made about FR-DPx.2 and the effects that these decisions will have on the rest of the design.

These considerations are captured in the next type of modularity defined, development modularity:

$$M_{uD} \equiv \frac{P}{|\{FRx.i\}|} \tag{4-20}$$

$P$ is the lowest index of the sub-FRs among $\{FRx.n.i\}$ that are affected by changes in the parent DPs $\{DPx.m\}$ where m≠n. In figure 4-3, $P$ equals 2 since *FRx.2.2* is affected by changes in *DPx.1*.

This *development* (or *interface*) *modularity* measure is useful for evaluating the amount of impact design changes have on the overall system. It provides a measure of the development time for the change of a DP.

Furthermore, it can be seen by again considering figure 4-3 that the ranges of input and output values of the transformations of the target objects are important in determining the ease with which a DP can be changed.

If the output of the previous transformation has a large span over which it can provide values, then it is easier to redesign the next DP in the series. On the other hand, if the span is smaller then the design of the second DP must be more precise to accommodate this. And the converse holds true for the DPs downstream.

This motivates the need to consider such spans in inputs and outputs in selecting DPs and why these need to be considered in the measure of information content.

#### 4.2.3.1 Flexibility

In this section the concept of *flexibility in FR spans*[12] is introduced. This deals with dynamic *spans* in the specification of FR values, how these can be accommodated in DP selection, and how the concept of information content and its calculation can be extended to deal with them.

**Definition of flexibility**

Flexibility is a measure of the variety of customer inputs and outputs which can be satisfied or provided by the design. This variety impacts the definition of sub-FRs and the choice among alternative DPs. In this thesis, it is proposed to *compare alternative designs based on the*

---

[11] Albano recognized that off-diagonal terms in the design matrices correspond to interfaces within a system. [Albano (1992) pp. 145-162] His metric of complexity termed the *interface index* is equal to the nullity of the graph that represents the design hierarchy as a vertices and edges. It is equivalent to the number of off-diagonal terms in the design matrices:

$$\text{Interface Index}, \Delta I = \textit{nullity} = \textit{edges} - \textit{vertices} + 1$$

[12] The definition of *span* used here is this: "an extent, stretch, reach, or spread between two limits" [Merriam-Webster's (1996).

*amount of flexibility that each is able to provide.* This can be done at each level of the design hierarchy, looking at each set of DPs.

*Example [from [Hintersteiner (1998b)]]*

As an example of flexibility in FR spans, consider the design of the laser-illumination system within the a photolithography tool. In this design, one of the sub-FRs is to provide laser energy within a range (FR1.3.3). This requirement is satisfied by DP1.3.3, an attenuator. The customers desire to vary the amount of laser energy that reaches the wafer in order to match the type of photoresist and the process recipe.

Within the attenuator, there are sub-FRs that expand the width of the beam (FR1.3.3.1) and then redirect a portion of the light to a beam dump (DP1.3.3.6) to dispose of the excess light (FR1.3.3.6). The amount of light redirected, which controls the intensity of the beam (FR1.3.3.3), is determined by the set position of a rotating beam splitter (DP1.3.3.3). In this particular example, the rotating beam splitter can be set to allow 25%, 50%, or 100% of the light energy, up to 15 Watts, through.

### 4.2.4 Uses and discussion

This section has described the ways in which a system design can be considered modular from several points of view:

- resource modularity: correlating DPs with their physical embodiments
- operational modularity: measuring the freedom in the operation of a system
- development modularity: evaluating the amount of impact of design changes

These metrics are introduced to show that there are several ways that designs can be "modular" from the point of view of axiomatic design. These correspond to embodiments which exhibit modularity with respect to DPs, FRs, and their relationships in DMs, respectively. When a system is modeled using axiomatic design, it is easier to explain the different characteristics of the design that promote the desired ends of modular design. Furthermore, it can be shown that the characteristics are not the same for all the different desired ends.

The different concepts of modularity in system design are clarified and related to characteristics of the design as modeled using axiomatic design. Next this understanding of the nature of system design is applied to three particular applications:

- construction of command and control algorithms (CCAs)
- physical integration of DPs into resources
- extension of information content to include flexibility in inputs and outputs

## *4.3 Applications*

The above ideas are used to explain three types of decisions made in system design: allocation of DP resources through command and control algorithms, CCAs, (section 4.3.1), physical integration of DPs (section 4.3.2), and extension of information content to include variety (section 4.3.3).

### 4.3.1 Command and control algorithms (CCAs)[13]

Given that changes in FRs and DPs are an important part of system operation, the question becomes this: how can the system adapt to meet changing requirements? The answer is given by the use of command and control algorithms (CCAs) as described in this section. *System command and control* is defined as the function of the system to adapt to changing sets of FRs and FR values. The CCAs are a means for keeping track of these changes and for dynamically allocating sets of DPs to meet the currently required sets of FRs at their current values. This section describes the functions and characteristics of the CCAs.

In a general sense, the input-output transformations[14] at the top level of the system design can be broken down into several categories: process functions, transport functions, command and control functions, and support and integration functions, as shown in equation 4-21.[15] Command and control algorithms, therefore, can be viewed as the embedded logic that controls and coordinates such input-output operations.

At a given level of the design hierarchy, once DPs have been selected to satisfy a set of process and transport steps, the selection of an appropriate CCA *at this level of the design hierarchy* can also be done. This CCA is in charge of scheduling the process and transport sub-modules that appear on this level in this branch of the decomposition, and it is related to the process and transport sub-modules by a lower-triangular decoupled design matrix as shown in equation 4-21.

$$\begin{Bmatrix} \text{Perform Process 1} \\ \text{Perform Process 2} \\ \vdots \\ \text{Perform Process } u \\ \text{Transport 1} \\ \text{Transport 2} \\ \vdots \\ \text{Transport } v \\ \text{Command System} \\ \text{Integrate System} \end{Bmatrix} = \begin{bmatrix} X & O & \cdots & O & O & O & \cdots & O & O & O \\ O & X & & O & O & O & & O & O & O \\ \vdots & & \ddots & & \vdots & & \iddots & & \vdots & \\ O & O & & X & O & O & & O & O & O \\ X & X & \cdots & ? & X & O & \cdots & O & O & O \\ ? & ? & & ? & ? & X & & O & O & O \\ \vdots & & \iddots & & \vdots & & \ddots & & \vdots & \\ ? & ? & & ? & ? & ? & & X & O & O \\ X & X & \cdots & X & X & X & \cdots & X & X & O \\ X & X & & X & X & X & & X & X & X \end{bmatrix} \begin{Bmatrix} \text{Process Module 1} \\ \text{Process Module 2} \\ \vdots \\ \text{Process Module } u \\ \text{Transport Module 1} \\ \text{Transport Module 2} \\ \vdots \\ \text{Transport Module } v \\ \text{Command and Control Algorithm} \\ \text{Structural Framework} \end{Bmatrix} \quad (4\text{-}21)$$

This is the way that the design looks at higher, system (non-component) levels. At such levels, the inputs for the CCA include information on the desired target object output (for example, the desired geometry of a part to be made) and information on the available target object inputs (for example, the set of available raw materials). The output of the CCA is a complete description of the states of the system over some period of time. This can be viewed in two ways as shown in figure 4-1:

1. information about the flow of individual target objects through the system
2. descriptions of the states of the system, in terms of the functions being performed at future times on the currently allocated target objects

---

[13] Parts of this section are adapted from [Hintersteiner and Tate (1998b)].

[14] One of the fundamental assumptions in the development of a system architecture is that the system can be modeled as a series of interacting inputs and outputs.

[15] This template for system design was first used in [Hintersteiner and Tate (1998b)] and is further detailed in [Hintersteiner (1999)].

At lower levels of the design hierarchy, the execution of a process is performed by specific pieces of hardware. The appropriate DP to satisfy the control of these hardware components is a special case of CCA, a *hardware-control algorithm (HCA)*. This algorithm is located at the same level of the decomposition as the hardware being controlled.

*Command versus control*

Command is distinct from control, from a functional point of view, in the sense that *control* determines the operating values of DP attribute for the system at a given for a given set of FRs and DPs, whereas *command* must update or switch the DPs of a system to correspond to a new set of required FRs.[16]

*Types of functions performed by command and control algorithms (CCAs)*

The functions of the CCAs are different depending on the level which is being considered within the design hierarchy. The higher-level CCAs deal with sequencing and scheduling, and the lowest-level CCAs deal with hardware control. *Mid-level CCAs can perform both sequencing and scheduling*, however, a given CCA is responsible for scheduling DPs at its level and for sequencing sub-FRs at the next, lower level.

To differentiate the CCA functions clearly, the three types of functions which can be performed by a CCA are the following: sequencing of FRs, scheduling of DPs, and control of hardware parameter values.

These three types of functions to be performed by the command and control algorithm are summarized in table 4-1 and are illustrated in figure 4-2. Here, coating step A2 is composed of two specific hardware motions (A2.1 and A2.2), and coating step A2 is one of three steps in the coating recipe (the three steps have to be performed in a particular order). At an even higher-level of control, the command and control algorithm is responsible for selecting the right process recipe that determines the sub-FRs which make up the process.

**Table 4-1: Types of functions performed by Command and Control Algorithms (CCAs)**

| Type | Input Characteristics | Output Characteristics |
|---|---|---|
| I: Selection of recipes (determine set of FRs) | target object output, rate | series of FRs to be performed ("recipe" sequence) |
| II: Scheduling of DPs (assign DPs at specific times) | series of FRs to be performed ("recipe" sequence) | schedule of DPs (assigned DPs) and FR values over time (desired output of hardware) |
| III: Hardware control (control of parameter | FR values over time (desired output of hardware) | signal to control hardware (DP |

[16] Consider the amount of uncertainty associated with the FR time variation as described in section 4.2.2.1. The first case is a system for which the FRs are known and fixed *a priori* (as shown in equation 4-16)—hence only system control is required. Command issues arise in the implementation of systems where the second through fourth cases apply (equations 4-17, 4-18, and 4-19, respectively). In these cases, there exists at least some knowledge of the FRs beforehand, but the DPs to meet these FRs must be allocated from a pre-defined database of DPs. The role of system command is to implement the switching between the DPs over the lifetime of the machine. Once a set of DPs are allocated to meet the current set of FRs, the design may be considered as in the fixed-FR case, and system control can be implemented for that particular configuration of the machine.

| values) | values over time, $\dot{\theta}(t)$) |
|---|---|

In the situation in which there are multiple sub-modules at one level of the decomposition, one command and control algorithm can be used for this whole set. All the algorithms can be integrated into one unit. Furthermore, the level of physical embodiment is not necessarily uniform at a given level of the design decomposition. For example, one FR can be decomposed into a set of sub-FRs, where some of the DPs are hardware components while others are sub-systems with *their own* sub-processes.

**Example of CCAs in the design hierarchy**

Figure 4-4 illustrates the presence of command and control algorithms at multiple levels of the design hierarchy. These include command and control at the system, sub-system, and hardware levels. In this section, a differentiation of the functions of each CCA is made, based on its position in the design hierarchy. The implementations of each are detailed in sections 4.3.1.1, 4.3.1.2, and 4.3.1.3.

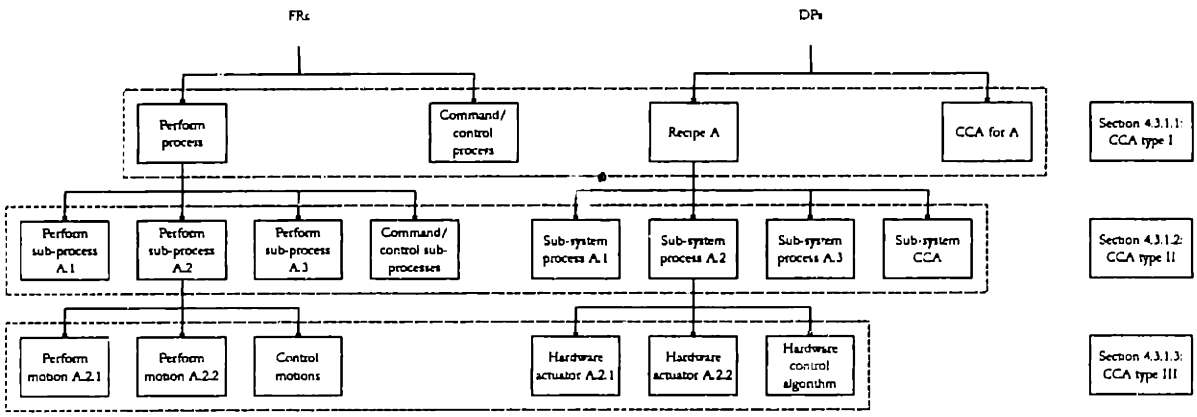


*Figure 4-4. Positions of command and control for process recipe A*

In this example, at the system level, the CCA for process A is responsible for selecting the appropriate DP that will satisfy the FR to perform process A. The choice of this DP consists of a determination of the required sub-FRs to achieve this process and the sequence[17] in which the sub-FRs should be performed.

The CCA at the middle level, responsible for the command and control of A's sub-processes, receives the FR sequence as an input. It then determines an appropriate schedule of DPs that will meet the sequence of FRs by specifically allocating DPs or DP resources at specific times. If different recipes are available, the CCA also determines the desired FRs for each, even-lower sub-process.

At the lowest level, the hardware control algorithm is responsible for translating the desired outputs of the sub-processes into specific hardware control signals for the hardware DPs.

Table 4-2 summarizes the input-output characteristics of the CCA at each level of the design hierarchy.

[17] *Sequencing* is defined as the ordering of FRs in order to produce a certain target object output. *Scheduling* is defined as the allocation of specific design parameters (DPs) over time.

**Table 4-2: I/O characteristics for CCAs at multiple levels of the design hierarchy.**

| CCA Level | Input Characteristics | Output Characteristics |
|---|---|---|
| System | desired target object output of A (*example:* desired thickness, desired rate for A) | sequence of A's sub-processes and their desired target object outputs (*example:* process recipe for planarizing wafers, desired thickness, desired rates for sub-processes of A) |
| Middle | desired target object output of sub-process of A (*example:* process recipe for planarizing wafers, desired thickness, desired rate for sub-process) | schedule for A's sub-processes, information on outputs of sub-FRs (*example:* scheduled times for sub-process sub-systems, desired rates for each sub-sub-process) |
| Hardware | desired motion characteristics of sub-processes of A (*example:* desired rates for sub-sub-process) | Control signals over time for specific hardware components (*example:* $\dot{\theta}(t)$) |

#### 4.3.1.1 FRs and DPs for system command (CCA type I: "recipe selection")

Consider a system in which the set of sub-FRs can vary with time as given by equation 4-22 in which the sub-FRs of DPx.1 are allocated dynamically based on the current CNs for the system. The sub-FRs could be, for example, a set of manufacturing operations which are performed to produce a part, be it a semiconductor device, an automotive component, etc. This includes the selection of a particular recipe from a database of sub-FR sequences as given by equation 4-22.

$$\begin{Bmatrix} \text{(FRx.1) produce flexible output} \\ \vdots \\ \text{(FRx.2) system command} \end{Bmatrix} = \begin{bmatrix} X & \cdots & O \\ \vdots & \ddots & \vdots \\ X & \cdots & X \end{bmatrix} \begin{Bmatrix} \begin{bmatrix} \text{(DPx.1a) recipe a} \\ \text{(DPx.1b) recipe b} \end{bmatrix} \\ \vdots \\ \text{(DPx.n) system CCA} \end{Bmatrix} \quad (4\text{-}22)$$

This is illustrated in figure 4-5 where there are two alternative recipes that can be followed for coating. Considered from the perspective of the parts to be processed, there are two alternative sequences of sub-FRs that can be followed given by DPx.1a and DPx.1b.

In figure 4-5, some parts are produced by following recipe A while others are produced by following recipe B. Some sub-FRs and resources are found in both recipes while others belong exclusively to one or the other.

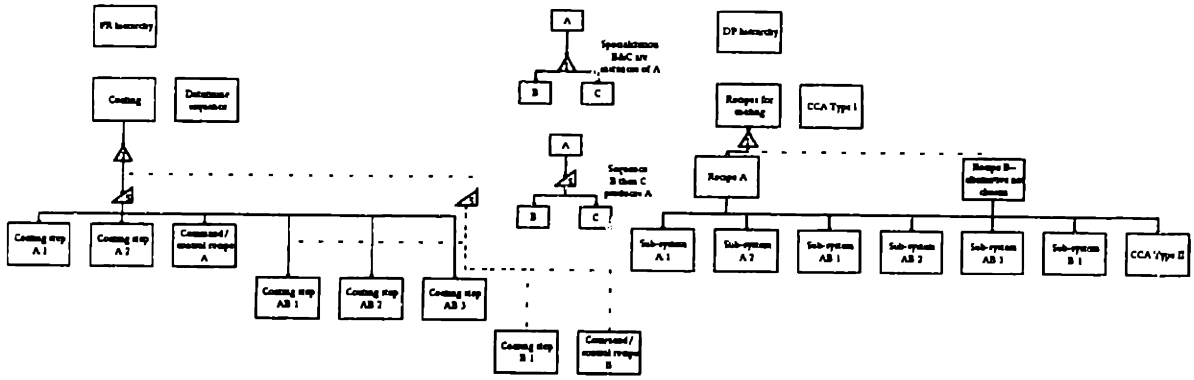

*Figure 4-5. Command and control including recipe selection*

So, at this level in the system FRx.1 can be satisfied by two alternative DPs; the choice of which depends on the particular part that is being produced. Each has its own set of sub-FRs for processing. These constitute a database of sub-FRs, and the DP resources within the system are allocated to meet these time-varying FRs. The means for selecting the correct recipe of FR sequences (that is, the choice of DPx.1a or DPx.1b) is the command and control algorithm (CCA) that is specified at this level of the design hierarchy. [Tate, et al. (1998)] The allocation of the DP resources, given the choice of the FR recipe is done by the CCA at the next, lower level of the design hierarchy as described in the next section.

### 4.3.1.2 FRs for DP scheduling algorithm (CCA type II: "scheduling")

In this section the command and control algorithms for scheduling of DPs are described.[18] They provide a general means for scheduling a set of DPs. This can be applied during the design of the system by the designers, for example in cases where the sets of sub-FRs are known a priori as in equations 4-17 and 4-18. Alternatively they can be applied during the operation of the system to match the incoming target object flow to the existing set of FRs and DPs, for example when the sets of sub-FRs are not known a priori by the designers as in equation 4-19.

In this case, a CCA fulfills the FR of scheduling a set of DPs or DP resources. This means that for the set of FRs considered at this level that at least one of them has multiple DP alternatives that can be considered.

The following functions detail the algorithm for scheduling DPs as implemented by a CCA type II. The sub-requirements for the command function given in table 4-3 can be described as follows:

- Receive sequence: In order to perform the functions for a specific part, the sequence of process operations—that is, order of sub-FRs—is given by the higher-level CCA when it assigns an appropriate DP alternative.
- Insert new part into schedule (initial): A new part is inserted into the schedule of parts around the operation which is the *bottleneck*, the operation with the longest takt time given the number of resources. This is a preliminary schedule which is checked and refined based on the other parts already scheduled.

[18] For an application of axiomatic design to scheduling, see [Lee (1999)].

- Identify critical inter-module times: Depending on the nature of the process, there can be critical process steps between which the time can be restricted, that is, the time between these steps must be less than a certain critical value. If so, these are identified.
- Balance schedule with other parts: This function ensures that the assigned process sequences do not violate any coupling in the machine that can occur when operations are being performed simultaneously. Three things must be considered in this balancing: the bottleneck in the process, the critical inter-module times, and any potential resource coupling[19]
- Finalize schedule: Once the schedule for the new part is confirmed, any needed adjustments to the database (table) are made, and appropriate commands (signals) are sent.

Table 4-3 summarizes the DP options which satisfy the CCA type II FRs.

**Table 4-3: DPs for CCA (Type II)**

| Functional Requirements (FRs) | Design Parameters (DPs) |
|---|---|
| (FR21) receive sequence | determined by higher-level CCA |
| (FR22) insert new part into schedule (initial location) | algorithm: schedule around module with least throughput |
| (FR23) identify critical inter-module times | recipe |
| (FR24) balance parts in schedule | algorithm: adjust given the critical times, coupling due to resource over-allocation, and bottleneck |
| (FR25) finalize assignment of resources | commands |

#### 4.3.1.3 FRs and DPs for control of hardware (CCA type III: "hardware control algorithms, HCAs")

This section describes briefly the design of hardware control algorithms as given in [Hintersteiner and Tate (1998b)]. In this section, an information flow model for a hardware controller that controls parameter values for specific pieces of hardware is analyzed to determine the appropriate order in which design decisions are made. Next, the components for control are examined from the perspective of the system architecture. While the sequence of design decisions must—and indeed does—remain the same, the different control components are included in different branches of the system architecture since the system architecture is intended to reflect the functional hierarchy of the system. The key results that are shown include the following:

[19] This would be a list of potential couplings in resource assignments. It could be as simple as not assigning two parts to the same resource at the same time; or it could be two resources that cannot be operated simultaneously.

- Actuators and sensors are designed or selected as a part of the process or transport module to which they belong.
- To coordinate all of the motions of a system, a hierarchical structure of control algorithms is defined.

Figure 4-6 shows a schematic diagram of the flow of information through a generic motion controller. Data are acquired from sensors, amplified and read into a computer through A/D or other sensor-specific boards (represented by SP in the diagram), and processed through one or more algorithms. The outputs of the algorithms are command signals for the actuators, which are written out through D/A, motion controller, or other actuator-specific boards, amplified as appropriate, and transmitted to the actuators.

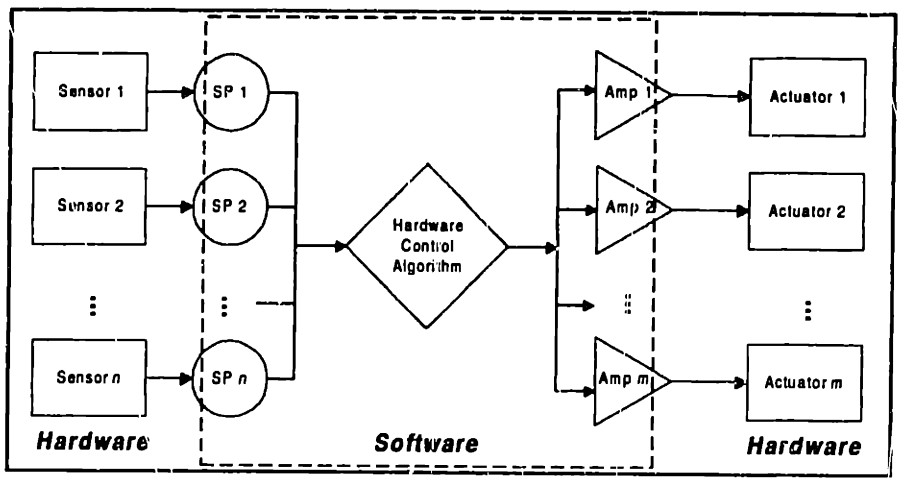


*Figure 4-6. Information flow diagram for a generic controller*

The information flow model of a controller is relatively intuitive to understand, yet it does not reflect the order in which these components are designed or selected. The actuators required for creating the individual motions and the sensors required for characterizing the results of those motions are included in the design of the process or transport module of which they are a part. Only after this hardware is specified can hardware control algorithms be devised for creating the appropriate output signals from the given input parameters. Once these algorithms are developed, decisions can be made as to the appropriate computer hardware and software required for implementing the controller, and finally the appropriate user and network interfaces to enable the controller to accept input from the user or from other systems can be selected.

The controller design matrix that shows these design dependencies is shown in equation 4-23. This can be considered a "lifting out" of command components from the system architecture. The design shows a decoupled matrix and thus specifies that design decisions are made in a non-iterative, sequential order.

$$\begin{Bmatrix} \text{Generate motion i} \\ \text{Sense parameter j} \\ \text{Control action k} \\ \text{Process input j} \\ \text{Transmit output i} \\ \text{Coordinate actions} \\ \text{Implement control} \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O & O & O \\ ? & X & O & O & O & O & O \\ X & X & X & O & O & O & O \\ ? & X & X & X & O & O & O \\ X & ? & X & O & X & O & O \\ O & O & X & O & O & X & O \\ O & O & X & X & X & X & X \end{bmatrix} \begin{Bmatrix} \text{Actuator i} \\ \text{Sensor j} \\ \text{CCA (3) k} \\ \text{Signal processor j} \\ \text{Amplifier i} \\ \text{CCA (1 \& 2)} \\ \text{Computer hardware} \end{Bmatrix} \quad (4\text{-}23)$$

The design matrix in equation 4-23 for a generic controller reflects the relationships between the controller elements in terms of the design decisions that are made. However, it does not capture the location of these elements in the system architecture. In general, several different controllers operate in parallel within a process or transport module, and coordinating communication and inputs between these different controllers must be considered. In addition, the system architecture accounts for the design of the computers and the necessary hardware interface boards required for implementing these controllers.

- Hardware: During the design of the hardware for a process or transport module, functional decomposition progresses to the point where a particular motion is preformed as a sub-FR for achieving a process or transport function. The generic DP that satisfies this FR is a "motion generator". This DP is decomposed into sub-FRs that provide motion, sense relevant parameters, and correlate input and output signals, as shown by equation 4-24.

$$\begin{Bmatrix} \text{Provide motion (1..m)} \\ \text{Transmit motion (1..m)} \\ \text{Sense parameters (1..n)} \\ \text{Correlate input \& output data} \end{Bmatrix} = \begin{bmatrix} X & O & O & O \\ X & X & O & O \\ ? & ? & X & O \\ X & X & X & X \end{bmatrix} \begin{Bmatrix} \text{Actuator (1..m)} \\ \text{Transmission (1..m)} \\ \text{Sensor (1..n)} \\ \text{Hardware control algorithm} \end{Bmatrix} \quad (4\text{-}24)$$

- Algorithms: Module command and control algorithms are required *at each level* of the design hierarchy to coordinate all of the individual hardware control algorithms, so that the module will work as one unit. Once all of the individual motion generators have been designed for a process or transport module, algorithms are defined to coordinate the motions needed to accomplish the task of that module. For example, a 3-axis robot used to transport parts between two manufacturing cells should be able to generate each axis motion independently; however, an algorithm is required to coordinate these motions so that the robot can move the part between two points. Within the framework of the system architecture, a hierarchical structure for these module control algorithms emerges, as discussed above.

- Implementation: The system architecture also includes the elements required for implementing the control system. The decomposition of the computer hardware includes a specification of the hardware and software platforms, signal interface boards, and network interfaces to other systems.

### 4.3.2 Physical integration

The earlier considerations of modularity can be used to discuss the nature of physical integration.

As described in section 4.2.1, DPs that have been physically integrated have one or more resources in common. If the distinction between DPs and resources is not understood, this can lead to confusion about the way that FRs change with time and whether or not a single DP satisfies more than one FR.

It is the resources, physical or logical entities into which DPs have been embodied, that satisfy more than one FR. Pretty clearly a robot within a track machine, or a manufacturing cell, is performing different FRs at different times. Sometimes it is picking up a wafer; sometimes it is moving a wafer; and the different locations from and to which it is moving are also continually changing.

From a functional point of view, the FRs being performed fall *in different locations* of the design hierarchy when the FRs have been defined solution neutrally at each layer of the design hierarchy. The FRs fulfilled at any given time can be *at different layers and even on widely differing branches.* There is no reason, for example, why the transportation mechanisms within each sub-system need to fulfilled by the same physical resource. In fact, depending on what parent DP is chosen for each sub-system, there may not be a required sub-FR for transportation at all!

Suppose the designers decide to use a single robot for the transport functions within each sub-system. Is it thus correct to say that *the same DP is being used to perform multiple FRs*? The answer is no.

If the designers interpret a robot moving parts between many different locations as part of different process recipes to be a single DP performing multiple FRs, then the corresponding interpretation of the independence axiom is that this design is unacceptable and a better design is to have individual transport "modules" for each part recipe. This, however, is not what the independence axiom implies. Rather, both a single-transporter design or a multiple transporter design can be acceptable from the point of view of the independence axiom.

This can be seen by examining figure 4-7. Here the design has progressed to some layer of the design hierarchy. This question is asked: can two DPs be physically, or logically, integrated? The answer depends on the nature of the FRs being performed. Are they done on the same target object? Are they done at the same time?

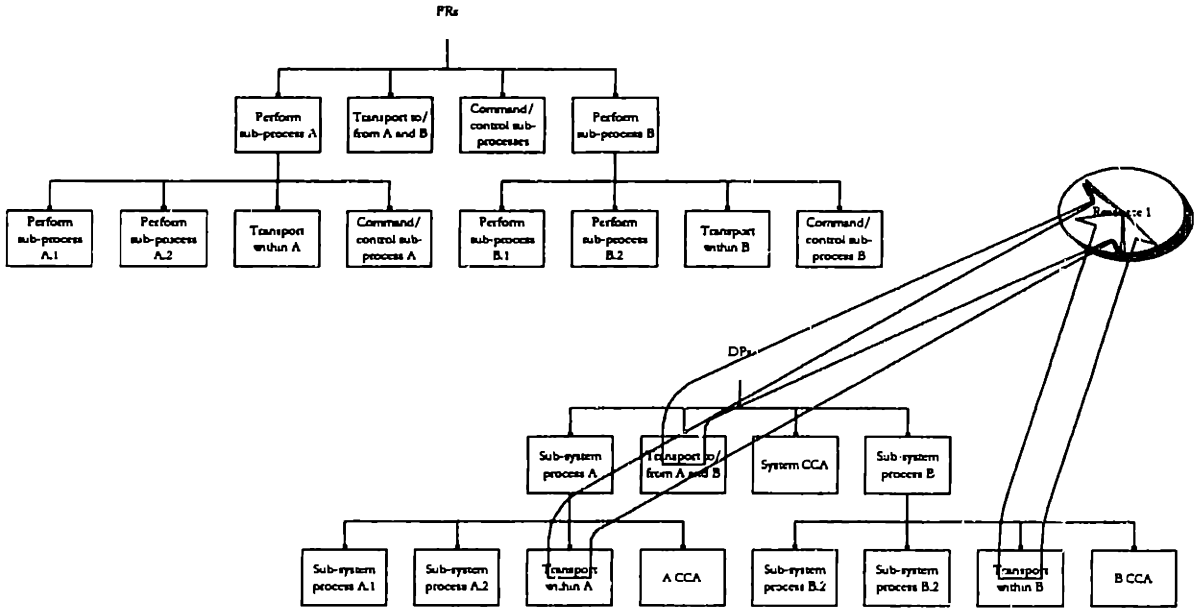


*Figure 4-7. Several DPs can be integrated into a single physical (or logical) resource*

The DPs *can* be physically integrated where they perform the same type of function on different target objects (same of different class, different instance) at the same time, or the same type of function on the same target object (same class, same instance) at different times. This is given in table 3-12 in chapter 3. They can be integrated at the parent-level in most cases, but they can be integrated down to leaves only in some special cases.

### 4.3.3 Additions to information content to account for flexibility

Flexibility is a measure of information content because the design alternatives considered are assumed to be designed to perform the same sub-FRs. That is, for a given set of functions, two different design alternatives can provide the same basic sub-FRs, but with different degrees of flexibility of the design spans.

As an example of flexibility, consider the requirement to provide a certain speed for a device, for example, the rotational speed of a drill, the speed of spinning a wafer during resist deposition, or the speed of an automobile.

Consider the definition of information content. *Information content* has been defined by Suh as the log of the inverse of the probability of success of satisfying a function.[20] [Suh (1990), Wilson (1980)]

Assume that there is a desired target value for an FR that the designers want to meet. Then an appropriate tolerance region about this target value can be specified; this region is known as the *design range,* $r_d$. Different design alternatives are able to provide the desired function within some *system range,* $r_r$. This is where the design alternative performs relative to the design range. The intersection of the system range and the design range is called the *common range,* $r_c$, shown in equation 4-25. These three areas are shown in figure 4-8. The *probability of*

[20] See appendix 4 for a more details on the fundamentals of axiomatic design.

*success*, labeled $p_t$ to indicate its basis on the tolerance of the FR, then is defined as the ratio of the common range to the system range, shown in equation 4-26, and the information content, $I$, is the natural log of this as in equation 4-27.

$$r_c = r_s \cap r_d \tag{4-25}$$

$$p_t = \frac{r_c}{r_s} \tag{4-26}$$

$$I = \log_e \frac{1}{p_t} \tag{4-27}$$

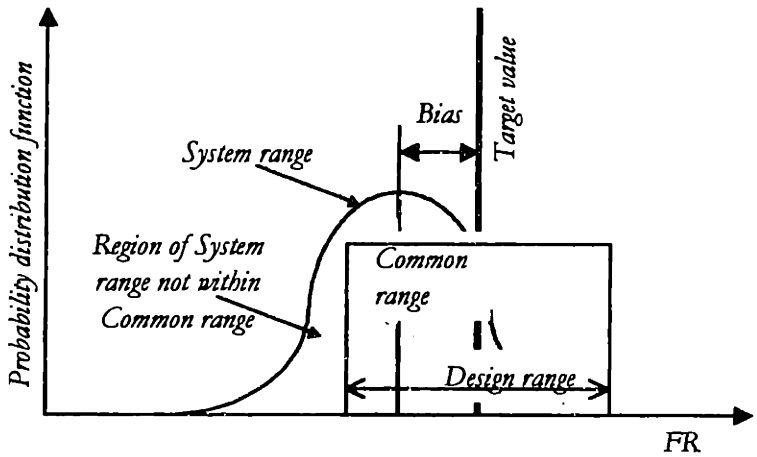


*Figure 4-8. Information content and probability of success*

Now to compare different design alternatives, the concept of flexibility needs to be incorporated into the determination of the information content. This is shown in equation 4-28.

$$I = \ln \frac{1}{p_{tolerance} \cdot p_{flexibility}} \tag{4-28}$$

Here information content is considered to consist of two components, one relating the performance of the design to the desired tolerance and the other relating the design to a desired flexibility. This indicates the performance of the design compared with an FR that is required to satisfy multiple values over its life.

Now return to the example. Designers are creating a device in which one of its FRs is to provide a desired speed, $s$. This speed must be satisfied within a certain tolerance, $t$, and there are two design alternatives that provide this. Then according to the definition of information content as in equation 4-27, both of these designs have an information content of zero and a hundred percent probability of success. So, both of these designs perfectly satisfy the information axiom.

Yet, consider the following twist: suppose the value of this FR does not remain constant as the design is used. For example, it must be designed to provide speeds over the region of (0.5s, 1.5s),[21] and suppose that the two design alternatives are not equivalent in this way. Suppose the first alternative can provide speeds within the span (0.4s, 2s), and the second alternative design can only provide speeds within the span (0.5s, s). Then clearly the first design is the better choice because it is able to satisfy the FR at all times, whereas the second design cannot meet the FR over half of its required span.

The definition of the *common span for flexibility*, $flexibility_{common}$ is shown in equation 4-29 and figure 4-9. It parallels that for tolerances. Namely, it is the overlap of the span provided by the system (system flexibility) with that required for the design (design flexibility).

$$flexibility_{common} = flexibility_{system} \cap flexibility_{design} \tag{4-29}$$

The definition of the *probability of success for flexibility*, $p_{flexibility}$ is given in equation 4-30. It is analogous to that for tolerance in equation 4-26. A notable difference is that whereas for tolerance, tighter design tolerances provided by the system are more likely to be within the design span, this is not the case for flexibility. Instead, a system which is more flexible is one which accommodates more variety in its operating spans.

$$p_{flexibility} = \frac{flexibility_{common}}{flexibility_{design}} \tag{4-30}$$

This illustrates the concept of flexibility. For some design alternatives, it is relatively easy to adjust the design to expand the span that the DP provides, particularly if the design is an uncoupled one. However, in other cases, the choice of a completely different DP is required. For example, a airplane provides more flexibility in terms of the locations that it can go than a train—not that this is the only FR for transporting packages. Likewise cellular manufacturing systems are designed to allow greater variety in the inputs and outputs of their processes than transfer lines.

Some FRs naturally need to provide a flexibility while others do not. This is part of the FR definition. Some FRs will always need to meet the same target value, while others will need to be within a tolerance, at some target value that is set dynamically, perhaps by the user, as the system operates.

[21] In this notation (x, y), x indicates the lower bound on the FR and y indicates the upper bound. So, a device that provides an FR within (x, y) can achieve a value of x or y or any value in between.

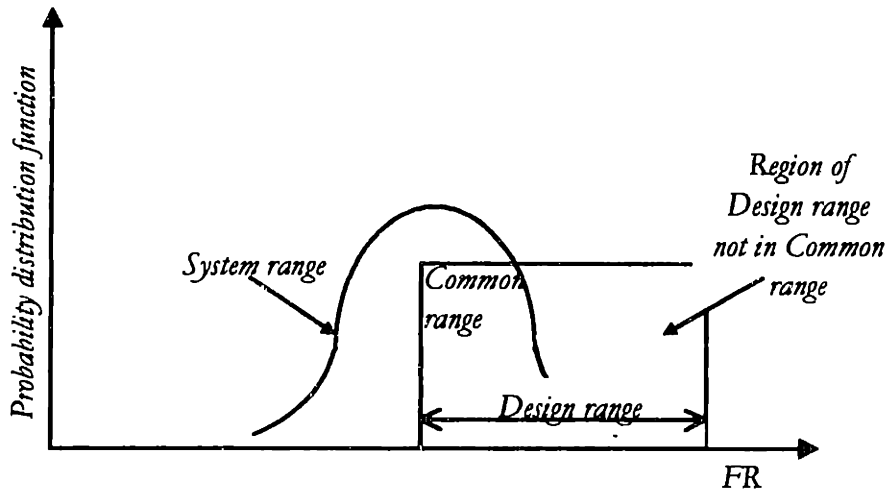


*Figure 4-9. Definition of flexibility*

The above example discussed flexibility in the output values of FRs. A system can also accommodate flexibility in the inputs it allows into the system. This is also an important concept when a set of FRs consists of a series of processes and the designers need to determine the value that will exist between two processes: the output of one FR and the input to the next process.

## *4.4 Summary*

System design is different from more simple component design in several ways. In particular, a *system* is composed of a number of distinct elements. Thus, when designers conceptualize a system, they consider a DP that is decomposed into a collection of disparate elements which acting together perform the desired higher-level function. Alternatively in *component design*, the designers consider the higher-level DP as a complete unit itself. Then the decomposition entails the refining of a single DP, and the lower-level design choices made by the designers consist of attributes or characteristics of that one DP.

In defining a system, there are clearly identifiable interfaces between the DP elements that make up the system. These *interfaces* are places in the design where the output of one FR serves as the input for another function. This means that systems can in part be modeled as a series of transformation, for example, process and transport functions.

The contributions of the chapter include detailing the following:

- explanation of the characteristics of a design which correspond to different conceptions and benefits of modularity
- software and hardware interfaces, the integration of command and control within the system architecture
- physical integration of DPs into resources that can be dynamically allocated to meet the changing FRs
- extension of information content to include variety in inputs and outputs

In this chapter, tools are presented which address situations that are encountered in specifically system design. The tools which are presented here are modeling of time variation within a system design and representation of flexibility. That is, time variation in FR performance and DP resource allocation is here incorporated into the design hierarchy, thus

extending the work of Suh (that has been termed large-system design [Suh (1995b)] or large, flexible system design [Suh (1997), Suh (1998a), Suh (1999)]) in which the FRs satisfied at any given time can be a only subset of the total set of FRs.

Considerations of time variation are shown here to have substantial impacts on two areas of the design: physical integration of DPs and allocation of DP resources through command and control algorithms (CCAs). These CCAs perform three functions: sequencing sub-FRs, scheduling DPs, and controlling hardware parameter values. This chapter demonstrates how command and control issues can be represented in a system architecture. The components (DPs) required for control are spread out over the multiple branches of the design hierarchy where active control is required. On the lowest levels, specific actuator and sensor equipment is correlated with a hardware control algorithm to coordinate low-level inputs and outputs. On higher levels of the hierarchy, command algorithms exist to schedule and coordinate all of the lower level command and control algorithms.

Additionally the concept of *flexibility in FR spans* is introduced. This deals with dynamic "spans" in the specification of FR values, how these can be accommodated in DP selection, and how the concept of information content and its calculation can be extended to deal with them.

# CHAPTER 5: CASE STUDY METHODS

## *5.1 Introduction*

This chapter places this work into the context of research in engineering design. It explains this work in light of the methods used for developing knowledge about engineering design. *Science* is seen as the application of accepted "scientific methods" to generate knowledge; the knowledge which is produced is known as *theory*.

Axiomatic design is chosen as a basis for understanding good decision making in design. This is followed by a description of the hypotheses of this work as related to the research question and the issues presented in chapter 2. These hypotheses lead to consequences, for each of the main issues (in chapters 2 through 4), which may be evaluated against the case studies (in chapter 6).

## *5.2 Axiomatic design as a "scientific" basis for this work*

In this section axiomatic design is described as a valid starting point for scientific research in design. This is followed by a description of the fundamental areas touched upon in this thesis, the underlying hypotheses for each, and the means for theory validation used.

### 5.2.1 Can axiomatic design be considered a science of design?

Axiomatic design does possess all the necessary components of a research program.[1] Moreover, axiomatic design constitutes a *progressive research program*.[2]

1. Axiomatic design makes predictions about designs and describes the successes and failures of designers in ways that no other theory of design does: Only axiomatic design combines the broad scope that describs any decision-making activity with relatively clearly articulated concepts for applying general principles to many particular cases.

---

[1] A research program or paradigm (see section A1.2.2) consists of

- ontology: an identification of the fundamental concepts or entities which make up the field of study
- aims: an articulation of the scope of the field in terms of both problems which have been solved (exemplars) and problems remaining to be solved (anomalies) which should be covered by the program, and are expected to be, but have not been yet
- methodology: guidelines for further developing the program—particularly in a manner consistent with the problems which the program has already addressed
- theories: relationships between the fundamental concepts of the field and application to the specific problems

[2] A *progressive research program* (as opposed to a *degenerating* one) meets three conditions (see section A1.5.4) [Larvor (1998) pp. 54-55]:

- Theoretically progressive condition: It must identify new and interesting predictions, that is, "undreamed of" [Larvor (1998) p. 55] by other theories. Plus, these predictions are particularly good if they are counterexamples to rival research programs. [Lakatos (1978) p. 5]
- Empirically progressive condition: Some of these new predictions must be corroborated by the experimental evidence.
- Heuristically progressive condition: As anomalies are identified, the progressive theory must accommodate and explain these anomalies in a manner consistent with the spirit of its heuristic (as opposed to in an ad hoc manner).

2. It has proven useful for answering real-world questions encountered by designers in industry.
3. Axiomatic design is not yet able to answer *every* question posed about *every* design problem. The presence of questions to be answered (or concepts to be further elaborated), however, is a hallmark of a healthy research program and provides a direction for further research.

Therefore, following the research approach (*heuristic*) of axiomatic design and addressing the problems it identifies (*anomalies* or problems to be solved) is a valid way of adding to the scientific body of knowledge of design. As such it has been chosen as the starting point for this research.

### 5.2.2 Scientific basis for this work

The fundamental knowledge areas related in this work are the design process, resources for design, and the design object. Generalizable knowledge is sought in terms of how design decisions can be made during decomposition. The outputs of decomposition are related to the process through a set of activities and guidelines (presented in chapter 3) and tools for system design (presented in chapter 4). The outputs of decomposition are presented as generalized descriptions of the design object.

In axiomatic design, it is assumed that characteristics exist "that distinguish between good and bad designs" and that these have "common elements" across creative fields. [Suh (1990) p. 5] The elements common to good design have been characterized by the two design axioms. This thesis is consistent with axiomatic design. It recognizes the importance of good design and supports the application of the design axioms. It fills in details of the process that has been sketched out in the theory so far. Moreover, it uses the same approach to theory building: identifying features or goals of good design decisions, articulating options, and abstracting general rules or guidelines.

Furthermore, the work in this thesis meets the criteria of progressiveness for research programs. The first of these is the *theoretically progressive* condition that states that research makes new and interesting predictions (or explanations[3]) that are "undreamed of" by other theories. [Larvor (1998) p. 55] This is an issue of originality and scope. As has been argued throughout, this work is unique in providing a theoretical basis for decomposition activities that is consistent with axiomatic design, and this is an area of keen interest to practicing engineering designers seeking to apply theory to their work.

The other two conditions, empirical and heuristic progressiveness, can only be evaluated in the context of the theory application. Research that is *empirically progressive* has corroborating empirical evidence, and that that is *heuristically progressive* digests new problems consistent with

---

[3] As described by Snyder, prediction and explanation both provide support for a theory:

> [W]hile the historical thesis is compatible with a personal theory of evidence, ...any purely personal theory is seriously flawed in not capturing the nature of evidence as it is used in science. In science, evidence is an impersonal concept. On impersonal concepts of evidence, ...some *e* can be evidence for *h* whether or not any person knows or believes that *e* is true. Hence, the *time* at which *e* is known is not relevant to whether *e* is evidence for *h*, as the historical thesis holds. [Snyder (1994)] (p. 475 in [Curd and Cover (1998)])

its problem-solving approach. The cases and discussion in chapter 6 argue that this work satisfies each of these conditions.

### 5.2.3 Case studies and validation

The hypotheses of this work are empirically validated through several industrial cases. *Case studies* are defined as empirical inquiries that investigate contemporary phenomena with multiple sources of evidence in which the boundary between phenomena and context is unclear. Their purpose can be description, problem solving, criticism/interpretation, or theory building. [Yin (1994) p. 4] Case studies are a common, and accepted, means for data gathering in research for design theory. (See [Dwarakanath, et al. (1996)] for examples, and see section A1.5.1 for additional discussion of case studies and their role in design research.)

In the case study projects discussed in chapter 6, engineering problems from industry are modeled as systems consisting of multiple layers of decomposition. The purpose of the case studies discussed in chapter 6 is to empirically support the work presented in chapters 2 through 4. The cases to which these ideas have been applied include the following: design of a tool-exchange mechanism which accommodates much customer variety, software control algorithms for machine control, reuse of design rationale for manufacturing cell design, and diagnosis of coupling in machines and suggestions for changes. These cases apply the theory across a wide variety of examples. Thus they provide breadth in the level of abstraction at which the theory is examined (that is, opportunities to look at the roadmap, the guidelines, and the tools). Furthermore, they provide breadth in the field of the problem (that is, system versus subsystem, hardware versus software, new design versus redesign versus troubleshooting).

In general, the hypotheses of this work are that the theory presented in chapters 2 through 4 provide an accurate basis for describing and prescribing the activities of decomposition. These can be elaborated such as the following:

1. The roadmap of decomposition shown in figure 3-2 and section 3.2 in chapter 3 comprises a complete model of decomposition which places it into the context of system design.
2. The goals of designers in performing decomposition activities are those given in table 3-2 in chapter 3.
3. The guidelines in section 3.4 and table 3-16 in chapter 3 help the designers achieve their goals in performing decomposition activities.
4. In chapter 4, modeling and documenting time variation (sequencing of FRs) provides a tool for enabling physical integration, for dimensioning of DPs, and scheduling of DP resources within a system.

Several questions will be addressed. The questions explored with the case studies include these:

- Does the roadmap of decomposition accurately represent the flow of decisions/activities as practiced in decomposition?
- Are the goals for the activities consistent with practice?
- Is the means for identifying leaves accurate?
- Is the sequence for the design decomposition accurate?

The list of hypotheses and questions are meant to be illustrative. Neither is intended to be an exhaustive list.

## *5.3 Summary*

It is the belief of the author that *research into design can be performed in a scientific way*: that scientific methods can be applied to expand our understanding of design and the principles which underlie it. The reader who is interested in a thorough treatment of scientific research methods and theory validation as applied to design is strongly advised to read appendix 1.

Axiomatic design theory possesses all the necessary components of a progressive research program: It makes unique predictions about design and the successes and failures of designers; it has proven useful for answering real-world questions encountered by designers in industry; and it continues to successfully digest new problems to be solved without resorting to ad hoc strategies in modifying the theory.

The approach followed in this work is consistent with that followed in axiomatic design theory, and it also meets the criteria of a progressive research program. The hypotheses of this work are empirically validated through several industrial cases as discussed in chapter 6.

# CHAPTER 6: CASE STUDY RESULTS AND DISCUSSION

## *6.1 Introduction*

This chapter presents the results of several case studies from industry in order to provide empirical support for the theory presented in chapters 2 through 4. The theory is evaluated according to the criteria given in section 5.2 in chapter 5.

Section 6.2 gives an overview of techniques for empirically corroborating design theory.

Section 6.3 gives descriptions of the cases to which the theory has been applied, followed by summaries of important facets of each. The cases to which these ideas have been applied and evaluated are the following:

- system analysis: photolithography machine (section 6.3.1)
- hardware-software interface: CCAs for CMP (section 6.3.2)
- design of a new sub-system: RMS (section 6.3.3)
- software design: AD software (section 6.3.4)
- system analysis: Track machine (section 6.3.5)
- software design: PDM system (section 6.3.6)
- reuse of design rationale: manufacturing cell design (section 6.3.7)

An example of the application of the theory is given in section 6.4. This discussion shows how the several parts of the theory were applied to a single case. Throughout this discussion, the similar application of the theory to other cases is highlighted.

A discussion of the results of the theory application is given in section 6.5 in three parts: academic considerations (section 6.5), industry considerations (section 6.5.2), and extensibility to software (section 6.5.3).

## *6.2 Discussion of theory evaluation*

This section presents an overview of the means for theory evaluation in design.

Table 6-1 describes the types of arguments that can be made for the theoretical concepts presented.

The concepts that are presented in this thesis fall into three categories:

- the design process model (section 6.2.1)
- guidelines for decision making (section 6.2.2)
- tools for system design (section 6.2.3)

**Table 6-1. Validation means for theoretical concepts**

| Test | | Example |
|---|---|---|
| Design process model: activities and connections | | |
| Historical | compare histories of projects | collect data on projects, compare results |
| Necessity | argue activities must connect in a particular sequence | logical necessity given inputs and outputs |
| Controlled test | have designers explain information they consider as they make decisions in design, "protocol study" | ask about goals, survey and match goals, compare results |
| Guidelines | | |
| Historical | show that a guideline matches what was done in successful designs | compare leaves, their parents, and grandparents |
| Necessity | argue from the negative: show negative results that can happen if the guideline is not followed | show how decisions on one branch affect another thus implying sequence |
| Controlled test | compare results from groups using and not using guideline, compare time, compare versus specific goals, get feedback from designers | system template |
| Tools | | |
| Historical | compare results from past cases, compare their functionality, time | successful cases versus unsuccessful ones |
| Necessity | argue why needed, argue from negative: show problems without tools (such as decisions that are not captured) | current models represent FRs with flexibility the same way, thus failing to distinguish between the designs |
| Controlled test | compare results from groups using and not using guideline | teach some engineers concept of CCAs compare with control group |

## 6.2.1 Design process: activities, connections, goals

The intended use of the model of decomposition activities is to predict or explain the decisions made by the designers. The model can be compared with reality in its descriptions of the activities, the way in which the activities connect, the inputs and outputs for each, and the goals of the designers that were identified.

A limitation, of course, is that it is difficult to perform controlled experiments on large, industrial design projects. Multiple teams of designers cannot be assigned to the same task since that this is an inefficient use of resources.

### 6.2.2 Guidelines

Each guideline for decision making that is presented in chapter 3 is associated with one of the activities that comprise decomposition. The important consideration in evaluating the guidelines is whether or not they lead to the designers' aims. Three types of arguments can be made:

- historical: Did following the guideline lead to the desired cognitive aim more often than other, rival guidelines?
- necessity: Can the consequences of not following the guideline be shown to have detrimental results, in comparison with following the guideline?
- controlled test: Using multiple sets of designers, teach some a guideline, and not others. Then compare the results.

Examples of each of these approaches are given in table 6-1 above. For a specific guideline, one or more of these strategies were identified as effective in providing empirical support.

In evaluating the guidelines, the correct identification of activities and goals must be reasonably established, and the originality or non-obviousness of the guidelines should also be established. If the guideline matches something that designers all do anyway, then merely pointing this out, provides no benefit.

In looking at the historical record or in performing controlled experiments, the results of following the guidelines are to be compared against the specific aim that they were created to promote. Furthermore, the results can be compared in terms of functionality of the design and originality of solution, time for the design task, and overall designer satisfaction in following the guidelines.

### 6.2.3 Tools

The same type of argument can be made for the tools for system design as was made for the guidelines for decision making. First, in their evaluation, the correct identification of activities and goals must be reasonably established. Furthermore, the originality or non-obviousness of the tools should be established. The results of using the tools can be evaluated against the overall effectiveness of the design task: functionality of the design and originality of solution, time for the design task, and overall designer satisfaction in following the guidelines.

#### Summary

I do not claim that this thesis answers all questions about decomposition. It provides a framework for understanding decomposition; the utility of which will prove useful or not in time. Is this framework useful for continuing to further develop an understanding of decomposition? Does it provide direction to the intellectual debate on the topic? Do the designers who use this theory find it useful?

The example in section 6.4 illustrates the usefulness of the overall theory. The roadmap is presented in the context of one specific case, and guidelines are shown to be applicable for individual activities within the roadmap. The use of the guidelines in other cases is also noted. Following this is a description of the use of the tools for system design.

### *6.3 Introductory descriptions of cases*

The theory given in chapter 2 through 4 is discussed in light of several case studies. These include cases in which the author played a substantial role and others with which the author is familiar, but was not directly involved. The cases are listed in table 6-2.

**Table 6-2. Characteristics of cases**

| Case | date | levels | nodes | participant | distributed | hardware/ software | new/existing design |
|---|---|---|---|---|---|---|---|
| System analysis: Lithography tool [Hintersteiner (1998b)] | Jun 98 - Aug. 98 | 6 | 230 | ✗ | ✓ | ✓ | e |
| Hardware-software interface: CCAs for CMP [Melvin (1998), Melvin (1999a)] | Jul. 97 - Oct. 98 | 5 | 90 | ✓ | | ✓ | n |
| Design of a new sub-system: RMS [Friedman, et al. (1998a), Friedman, et al. (1998b)] | Apr. 98 - Oct. 98 | 6 | 150 | ✓ | ✓ | ✓ | n |
| Software design: AD software [Do (1998)] | Oct. 97 - Oct. 98 | 9 | 250 | ✗* | ✗ | ✗ | n |
| System analysis: Track machine [Lee (1999)] | Oct. 97 - Dec. 98 | 6 | 100 | ✗ | ✗ | ✓ | e |
| Software design: PDM system [Lanza (1998)] | May 98 - Oct. 98 | 6 | 60 | ✗ | ✗ | ✗ | n |
| Reuse of design rationale: manufacturing cell design [Mårtensson and Tate (1998)] | Aug. 97 - Oct. 97 | 3 | 25 | ✗ | | ✗ | n |

## 6.3.1 Photolithography tool

A system architecture has been constructed for a photolithography machine (or "lithography tool"). The intention of the project is to provide a "basis for design decisions and evaluations to be made". [Hintersteiner (1998b) p. 4]

The work done captures the main subsystems of the photolithography tool including

- the photolithography process
- input/output and management of wafers
- input/output and management of reticles
- CCAs for scheduling and coordination
- support structure for subsystem integration

The system architecture shows the functionality of the major subsystems and the way their designs interrelate. Furthermore, the performance specifications are captured as high-level constraints and are carried down into more specific constraints and sub-FRs at lower-levels of the design hierarchy. The subsystems were decomposed 5 or 6 levels.

The lithography tool, being a preexisting system, was not designed using axiomatic design. Therefore the FRs, Cs, and DMs are reverse engineered from the existing DPs based on the perceived tasks the DPs perform, not necessarily those that the designers' originally intended. [Hintersteiner (1998b)]

**Results**

*Activities and guidelines*

According to Hintersteiner, the designer performing the system analysis of the photolithography tool, the theory for decomposition activities was useful in the following areas [Hintersteiner (1998a)]:

- constraint management: The theory provided guidance about the treatment of constraints. The way in which the system-level constraints were carried down to the lower-level FRs was particularly important and relevant.
- decomposition sequence: This design exhibits many instances in which the order of design decisions for one subsystem of the design must precede those of another subsystem. And knowledge of how to identify leaves and thus be able to stop decomposing was important in this analysis. Given the scope of the task–to create a system architecture for the whole machine–knowledge of when to stop kept the project manageable; otherwise it had the potential to continue indefinitely.
- levels of abstraction: It was useful to recognize that the DPs at a single layer of the design hierarchy are not all at the same level of abstraction; therefore they do not all need to be decomposed the same number of additional layers.
- defining sub-FRs: The template for system design was used to structure the sets of sub-FRs, and the multiple sources of sub-FRs were considered in developing a complete set.

The roadmap itself cannot be evaluated specifically in the context of this case because the photolithography tool is an existing design; therefore its accuracy in describing and directing the progress of the designers cannot be evaluated.

*Tools*

The concept of time variation and DP resource allocation was used in developing a hierarchy of CCAs for the system. Furthermore, the concept of flexibility was useful in defining sub-FRs more explicitly than could be done without the concept.

### 6.3.2 Hardware-software interface: CMP

Manufacture of semiconductor devices is done through the build-up and patterning of layers of metals and dielectrics to create semiconductor devices. Chemical-mechanical planarization (CMP) fulfills a growing need for planarization between process steps as line widths and depths of focus in leading-edge lithography processes decrease. In CMP, the surface

roughness of the thin-film depositions on the wafer is reduced through the combined chemical (etching) and mechanical (abrasion) process.

The CMP example is based on an on-going MIT project in machine/system design. The example as discussed here is a composite of several models for the design of the machine. [Hintersteiner and Tate (1998a), Hintersteiner and Tate (1998b), Melvin (1998), Melvin (1999a)]

**Results**

The roadmap for decomposition was able to guide the design of the CMP machine in several ways [Melvin (1999b)]:

- guidelines: The specific guidelines that the designer singled out as useful include considering all sources of sub-FRs (guidelines B1 and B2), identifying multiple alternative DPs to check solution-neutrality (guideline B9), defining support FRs (guideline B6), and carrying down and refining Cs (guidelines C1-C8).
- CCAs: The role of CCAs in enabling operational modularity is useful for the CMP tool. There are several different processes that the machine may be required to perform, and the ability to reconfigure the machine dynamically greatly increases its flexibility. The need to dynamically change the FRs and DPs of the machine depending on the particular wafer requirements may be handled in an efficient manner using CCAs. This overall machine control must be balanced with the low-level control of individual axes. The natural refinement of the CCAs as the hierarchy progresses allows the control system to develop in an integrated manner with the rest of the machine.

### 6.3.3 New sub-system: RMS[1]

The RMS case has two primary objectives: redesign of a sub-system in a semiconductor-manufacturing machine and development of company expertise and experience in using axiomatic design.[2] The customer needs for reticle management within a photolithography tool have changed since the first introduction of the sub-system several years ago, and increasing pressure to meet current—and prepare for anticipated—customer needs have prompted a project for the redesign of the reticle management system (RMS).

Reticles are quartz plates with chrome printing on one side.[3] The printing on the reticle is divided into several specific areas: the pattern to be transferred to the wafer, coarse-alignment targets, fine-alignment targets, etc. The pattern to be transferred to the wafer is reduced 4x in its transmission through projection optics onto the wafer.

---

[1] This section is adapted from [Friedman, et al. (1998a)].

[2] The confidentiality agreement with the sponsor is that the information in this chapter can be used, but that I cannot give the name of the company or its product.

[3] The current generation of photolithography machines, or "tools", uses reticles with dimension 6 x 6 x 1/4 inches (15.2 x 15.2 x 0.6 cm). New generation reticles will have dimension 9 x 9 x 3/8 inches (229 x 229 x 1.0 cm). The new, 9x9 reticles have a pattern with the same height and the same geometrical arrangement of the alignment features as the 6x6 reticles; the difference is an increase of the pattern width.

The functionality of the reticle management system includes all reticle handling and management between the introduction of the reticle into the machine and the handoff of the reticle to the reticle stage. The reticle stage performs fine alignment and synchronizes the reticle movement with the wafer as the wafer is exposed during the photolithography process. The RMS receives commands from the higher-level machine software and the fab software and performs the function to deliver the correct reticle to the reticle stage at the correct time.

The changed customer needs for reticle management include

- quick exchange time: The mechanical handoff portion of the reticle exchange is to be minimized. (target: 10s)
- accommodate variety in reticle carriers: The sub-system is to allow users to introduce reticles into the tool using one of a variety of carriers that the user can specify. This is to be accommodated with as little redesign of the machine for each carrier type as possible.
- store reticle in a "library": The users need to store significant numbers of reticles within the machine and the more, the better. (target: 18 reticles)
- quick access to reticles: Some customers need to insert and remove idle reticles while the machine is processing.

These needs are unmet by the current reticle handler design which has as a set of FRs based on an older, simpler set of customer needs (CNs).

**Results**

The following innovative results were obtained in designing and analyzing the RMS:

- development of design concept from system-level to leaves (hardware or software)
- integration of software design with hardware design, as part of the same hierarchy
- physical integration of DPs into resources for providing functions at different times including selection of appropriate hardware: end effector, mechanisms, etc.
- development of a "modular interface" that incorporates new and anticipates future CNs and the prediction of the design effort for new variants of the "modular interface"
- integration of axiomatic design on a project with multiple engineers and distributed task responsibility

The RMS case study demonstrates the usefulness of the theory presented in chapters 2 through 4. The roadmap of decomposition activities serves as a guide for the designers in performing decomposition and system integration within the context of axiomatic design. Furthermore, the goals articulated for the different decomposition activities are found to correspond to the goals of the designers in practice, and the guidelines are found to provide appropriate support to the designers in satisfying their goals. See also section 6.4.

### 6.3.4 AD Software

Because of the goals and direction of research in axiomatic design, the development of software tools based on axiomatic design is an important research topic. Design rationale and design history are both captured by combining axiomatic design with software tools for designers. *Design rationale* is defined as an information structure which can be used to answer

questions about why an object is designed the way that it is. *Design history* is a record of the decisions that have been made concerning the design, their effects on the object being developed, and the evolution of the design.

The most recent effort to develop a tool for designers to apply axiomatic design to their projects has been carried out by Do [Do and Suh (1998), Suh and Do (1998)].[4] In Do's work, two themes are stressed:

- applying axiomatic design to software design
- development of a tool for using AD in large-scale industrial design projects

#### 6.3.4.1 Application of AD to software

From one point of view, work on the axiomatic design software is interesting as an example of the application of axiomatic design theory to the field of software design. The idea of using AD to design software has been advanced elsewhere[5]; however, the axiomatic design software itself is the "first known case study of a large scale software development based on axiomatic design theory". [Suh and Do (1998) p. 3]

#### 6.3.4.2 Software for axiomatic design

An efficient and easy-to-use software program and project file for axiomatic design can facilitate application of the AD theoretical framework and data management within it. Although axiomatic design can be carried out on paper, doing this for large development efforts involving many levels of the design hierarchy becomes cumbersome.

The AD software incorporates the fundamentals of axiomatic design including: the FR, DP, and PV domains, constraints, design hierarchies, design matrices, and the system architecture. Important features of the software—in addition to such standard software functions such as file management, error handling, etc.—include functionality to deal with[6]

- tracing the effects of design changes
- reducing design time through automated design matrix rearrangement
- documentation of alternative DPs
- displaying data in a variety of graphical formats

The axiomatic design documentation scheme structures design information so that it can be further analyzed to understand how a change in one portion of the product propagates to the remainder of the product. The impact of such changes cannot readily be determined by the designer merely by looking at the design hierarchy because DPs at the same abstraction level but on different branches of the design tree may have substantially different impacts on the overall design. In recognition of this, the software automatically generates a list of functional requirements within the functional hierarchy which are affected by a change in any FR or DP in the design of the product. Once the design matrices have been entered for

---

[4] The most recent version of the software as described in [Do (1998)] is proprietary. The images shown are from earlier versions with similar functionality.

[5] See for example [Kim, et al. (1991b), Kim, et al. (1991a), Suh (1997), Suh (1998a)].

[6] An earlier version also included limited functionality for applying the Information Axiom. [Harutunian, et al. (1997)]

a hierarchy of FRs and DPs, the effects of design changes may be traced throughout, as shown in figure 6-1.

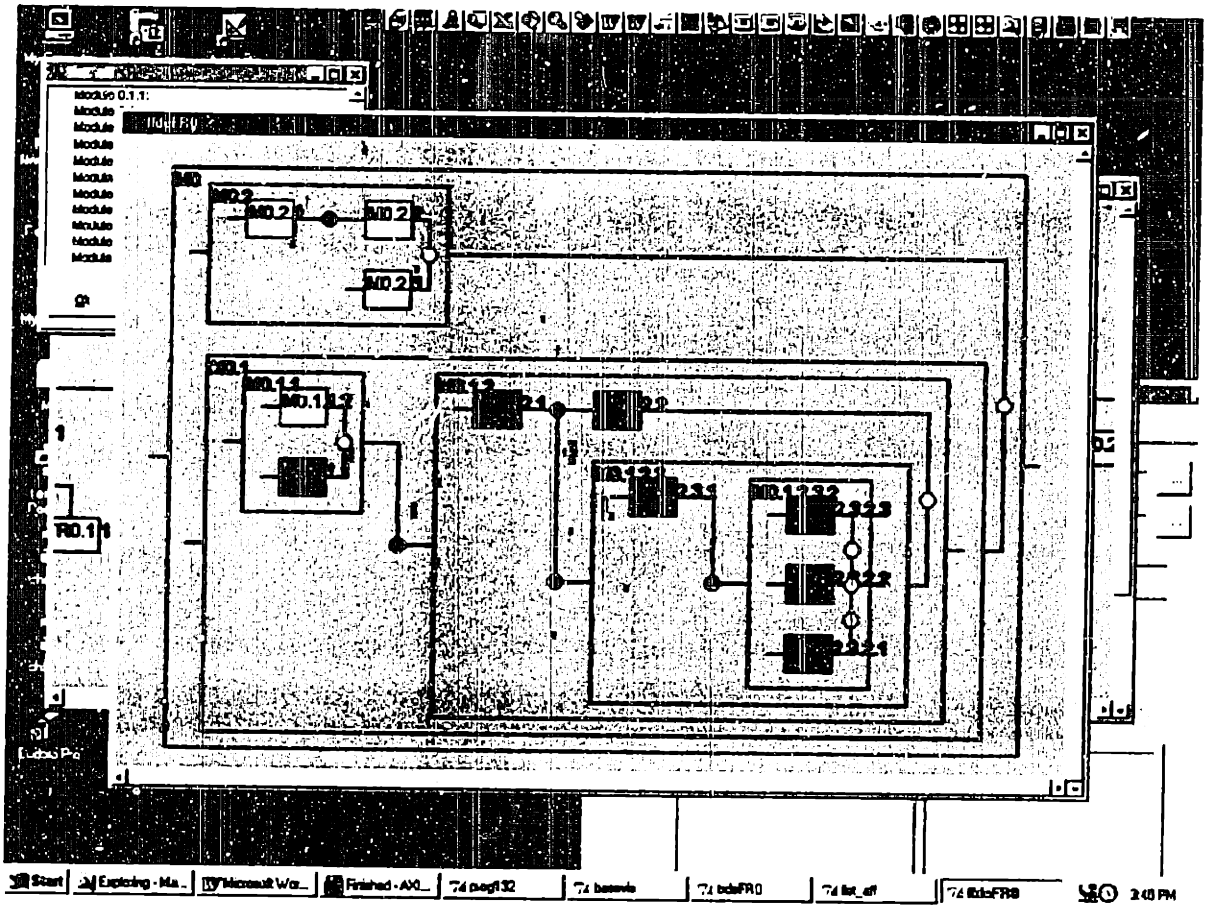

*Figure 6-1. Flow chart displaying sequence of decisions for an engineering change order (ECO) [Harutunian, et al. (1997)]*

The design matrices contain information that can be used to streamline the design process. Once the DMs are entered, the software computes the set of least coupled sequences for each matrix. The software then automates sorting through the permutations of the design matrices, shown in figure 6-2, to find the best sequence for selecting design parameters. This relieves the designer from the tedious task of manually rearranging the sequence of elements within the design matrices.

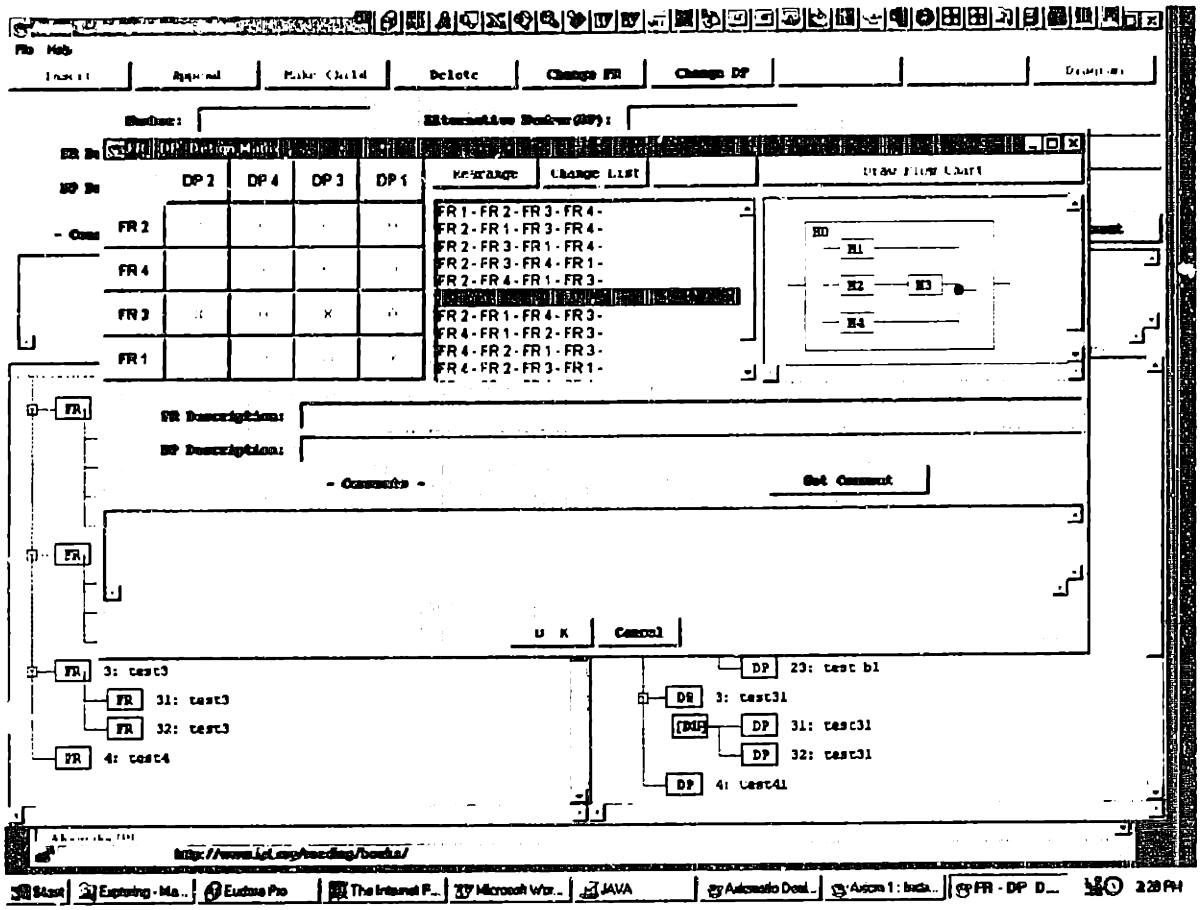


*Figure 6-2. Window enabling matrix manipulation [Do and Suh (1998)]*

The software provides a framework to enter alternative DP solutions for each function. This aids the creative process by allowing the designers to revisit proposed solutions in light of constraints on the project which change over product generations. Considering multiple design alternatives is important in systems design. A number of functional requirements are identified, and possible alternative design parameters are considered. These design parameters are then combined or integrated to make up the system design. The system which results and the nature of the design matrix will depend on the set of particular DPs that is chosen.

The design hierarchies in combination with the design matrices are used to generate a flow chart and a module-junction-structure diagram detailing the development or implementation sequence of the modules that make up the project. Each module is developed in a sequence, with respect to the other modules in the system, based on the design dependencies captured in the design matrix. The junctions between the modules are graphically displayed in the software as determined by their respective design matrices. Modules which are connected in an uncoupled design matrix may be summed to yield the desired system-level functionality and are connected by a *summation junction*; decoupled designs are connected by *control junctions*; and coupled design must have *feedback junctions* to indicate the necessary iteration.

**Results**

The roadmap for decomposition described the activities performed by the designer as the project progressed. The theory for decomposition activities was useful in the AD software case in the following areas:

- activities and goals: Articulating a set of goals for the decomposition activities was useful in guiding the designer, particularly in terms of assistance in generating sub-FRs (keeping in mind all the relevant goals). Understanding design dependencies as given in the design matrix and their implications on the sequence of decomposition was also useful.
- sequence of activities: The sequence of activities in decomposition model proved useful and is being incorporated into the latest version of the software.
- flexibility: The concept of flexibility was useful in structuring the options available to the user and providing and representing the functionality to enable the options.

## 6.3.5 Track machine

A track machine is used in conjunction with a photolithography tool in semiconductor manufacturing. The track machine performs pre- and post-exposure processing: the deposition of photoresist film on the wafer surface and the development of the exposed resist film. Both coating and developing are performed as a series of process steps. A system architecture has been constructed for two generations (200mm and 300mm) of track machines. An intention of the project is to understand the design of the previous generation machine and to apply this learning to the newer machine as it is designed. [Lee (1999)]

The work captures the main sub-systems of the machine including

- the coating process
- the developing process
- the system configuration
- CCAs for scheduling and coordination

The previous generation machine (the 200mm) since it is an existing system, was not designed using axiomatic design. Therefore, the FRs, Cs, and DMs are reverse engineered from the existing DPs. The design of the scheduling software of the newer machine is being done using axiomatic design, as part of research into the application of the system architecture to current design tasks in industry. Therefore, the objectives of the project are two-fold: 1. to facilitate the design of the next generation track machine and 2. to corroborate several hypotheses about the generation and application of system architectures.

Lee developed a set of 13 hypotheses concerning the system architecture. Of these, several were selected for detailed examination. [Lee (1999)] The hypotheses fall into 3 categories:

- accuracy of the system architecture representation
- the top-down approach to system design
- the system architecture as an analysis tool for systems

**Results**

*Activities and guidelines*

According to Lee, the theory for decomposition activities was useful in the case study in the following areas [Lee (1999)]:

- determining leaves: The guideline about leaves was useful in understanding the scope of the project.
- template: The template for systems was used to structure the FR hierarchy.
- CCAs: The concept of the command and control algorithms was useful in evaluating the functionality of the control/scheduling software.

## 6.3.6 Target-oriented product data management (PDM)

The purpose of this case is to configure a product data management (PDM) framework to support manufacturing system design among companies that are collaborating. This work is part of two German research projects: 1. Thixotec, to design and manage a virtual thixocasting manufacturing system and 2. InfoLog, which concerns field-spanning information logistics for product development. [Lanza (1998)]

## 6.3.7 Reuse of design rationale for manufacturing cell design[7]

This case concerns manufacturing cell design and the use of axiomatic design as a means to document and communicate design rationale. The cell performs operations on four families of rear axle bridges. Each family may be characterized by the diameter of its rear axle pegs. Within each family, part variants are characterized by different lengths and the fasteners with which they are equipped. In total there are 35 variants, and an example is shown in figure 6-3.

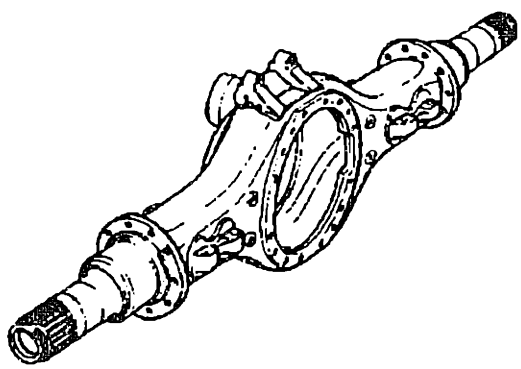

*Figure 6-3. Example of part to be transformed*

**Results**

This case from the automotive industry, looking at a cell for manufacturing rear axle bridges, shown in figure 6-4, shows the usefulness of applying axiomatic design as a means for

[7] This section is adapted from [Mårtensson and Tate (1998)].

documenting and storing design rationale in manufacturing cells. The FRs of the system are an example of the third class of time variation where fixed operations are performed on several parts, in which the number of parts produced is determined independently for each type. The necessity of fitting a cell within the current production system means that there are constraints imposed upon the cell, some of which directly come from the design parameters on a higher level of the system design. Also, as a result of this case, a number of areas were identified as candidates for further work and development: integration of similar lower-level DPs, determining the right machine capacities, and developing ways for designers to state FRs consistently.

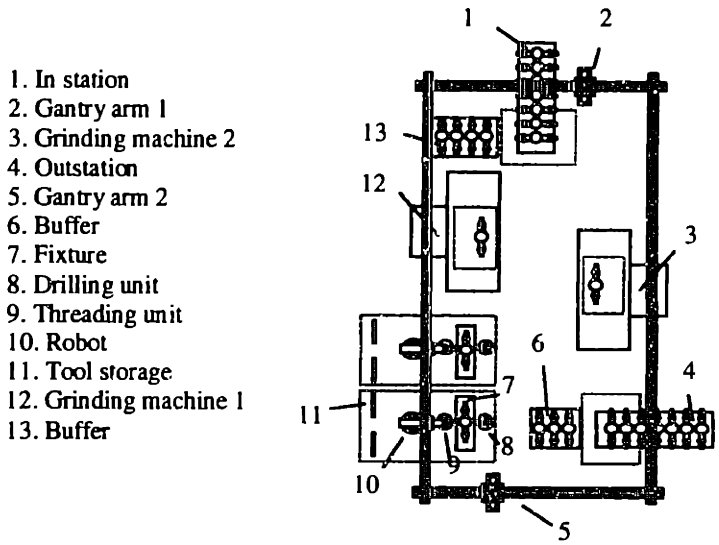


*Figure 6-4. Layout of the cell*

## *6.4 Discussion of theory application*[8]

This section shows more concretely the usefulness of the theory presented in this thesis. The RMS decomposition (and top-level of the lithography tool) is used in this section to illustrate the process of applying the roadmap, guidelines, and tools. Furthermore, as discussed in section 6.2, some of the guidelines can be corroborated through examining cases. Therefore, this discussion also includes descriptions of other cases in which each of the guidelines and tools have been applied.

### 6.4.1 Roadmap and guidelines

The roadmap for decomposition, shown in figure 6-5, described the activities performed by the designers as the design of the RMS progressed.

[8] This section is adapted from [Friedman, et al. (1998a)].

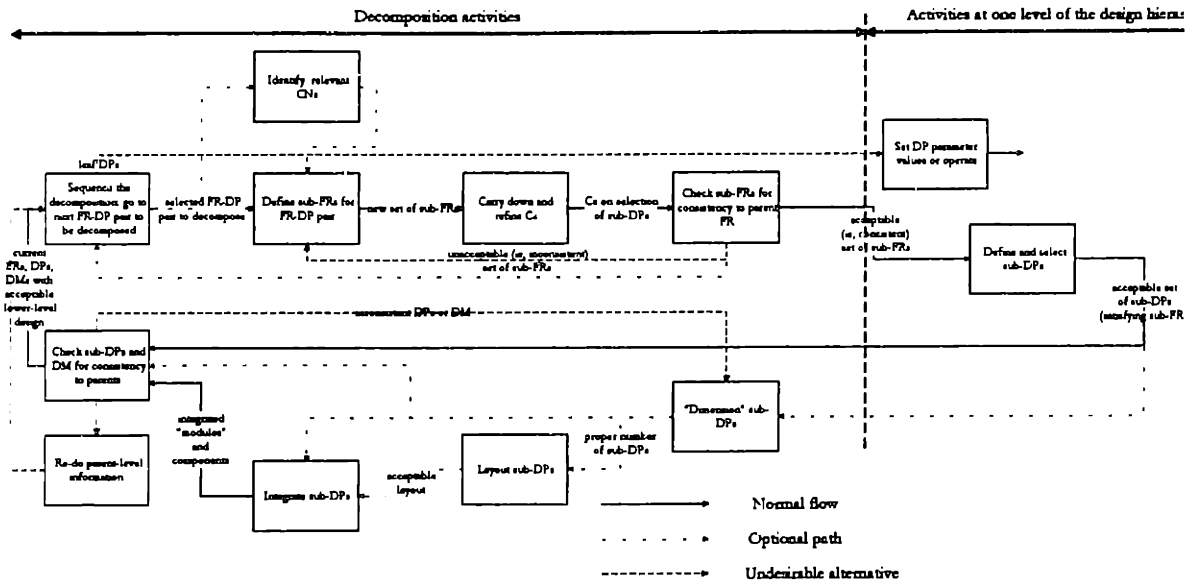


*Figure 6-5. Roadmap of activities in decomposition*

The focus in this example is on the new RMS that is being developed. Since the lithography tool itself is an existing system, an analysis of the design to the point where the RMS fits in is important to understand the impacts of the design of other subsystems of the tool on the RMS. An analysis of the lithography tool, conducted independently of the RMS design, was carried out as described in section 6.3.1 above.

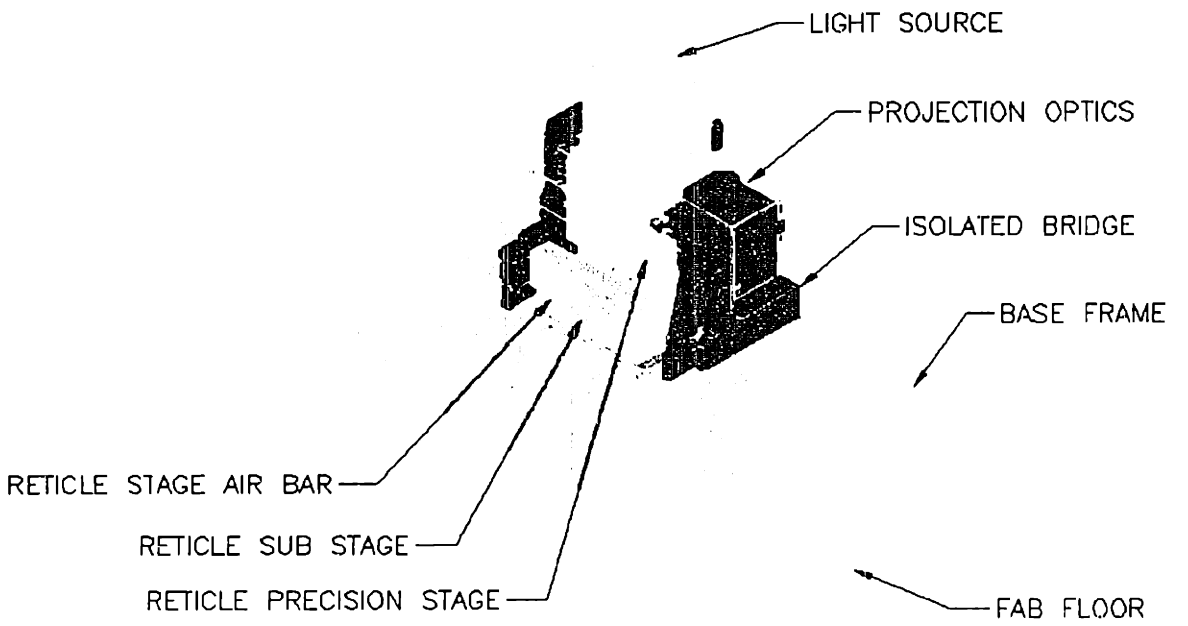


*Figure 6-6. Major components of the lithography tool*

This discussion begins with the top-level of the system architecture of the photolithography machine. The top-level FRs, DPs, DM, and Cs are given in equation 6-1 and table 6-5 below. Some of the main components of the lithography tool are illustrated in figure 6-6.

$$\begin{Bmatrix} \text{FR1 Transfer pattern} \\ \text{FR2 Manage wafers} \\ \text{FR3 Manage reticles} \\ \text{FR4 Schecule and coordinate} \\ \text{FR5 Integrate} \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O \\ X & X & O & O & O \\ X & O & X & O & O \\ X & X & X & X & O \\ X & X & X & X & X \end{bmatrix} \begin{Bmatrix} \text{Photolithography process} \\ \text{(b) Wafer handler} \\ \text{(b) RMS} \\ \text{Tool CCA} \\ \text{Tool support} \end{Bmatrix} \quad (6\text{-}1)$$

The highest-level FR defining the project is to print patterns onto photoresist-coated wafers, and the means for doing this is the lithography tool. The FRs in equation 6-1 are the result of this higher-level decision. A different choice of the parent DP (other than a lithography tool) would mean different FRs at this level. If the design of the lithography tool as a whole were being carried out, the design process could then continue with the generation of a new set of DPs for these FRs or the analysis of the existing set to see whether it is satisfactory. Over time, the constraints and FRs for the design have changed thus necessitating changes to the tool.

#### 6.4.1.1 Sequencing the decomposition

The first activity considered here is sequencing the decomposition. The starting point of this activity are the design decisions made so far as represented in the design matrix in equation 6-1 above. Guideline A1[9] states

> The designers have the option of considering a node to be a leaf—and thus not further decomposing it—when the target object of the sub-FR is different from that of its parent.

The target objects at the top-level of the lithography tool are the wafers (with photoresist coatings). At the level shown, the wafers continue to be target objects, and additional objects are introduced at this level as a result of choosing the photolithography tool as the highest-level DP (at the parent level). This is shown in figure 6-7. The FR satisfied by the photolithography tool is to create a patterned layer in order to make a semiconductor device on a silicon wafer. The DP for satisfying this is the photolithography tool itself. The FRs for the lithography tool (given in equation 6-1) that affect the wafer and its coating are in darker, shaded boxes. The FR for managing reticles does not directly affect the wafer or the coating; reticles are introduced as a result of choosing to use a photolithography tool. The designers can conceive of other means for creating patterns of light focused on the wafer surface. These, however, would not be considered photolithography tools at the top level, and their set of FRs at the next level would therefore be different.

Since this FR does not directly affect the wafer or the coating, the guideline about leaves indicates that this is a potential leaf or that it can be the start of a project, as in this case. If it is considered the start of the project, the designers need to continue decomposition until the hierarchy fully describes and determines the functions performed on the reticle. Leaves for one branch of the RMS are shown in figure 6-7.

Guideline A2 states

> At each level, define sub-FRs in the order described by the design matrices.

---

[9] The guidelines are compiled in table 3–16 in chapter 3. The numbering scheme is given there.

The nature of the design matrix in equation 6-1 indicates that the RMS can be decomposed by considering only the details of the photolithography process and the needs of reticle management.

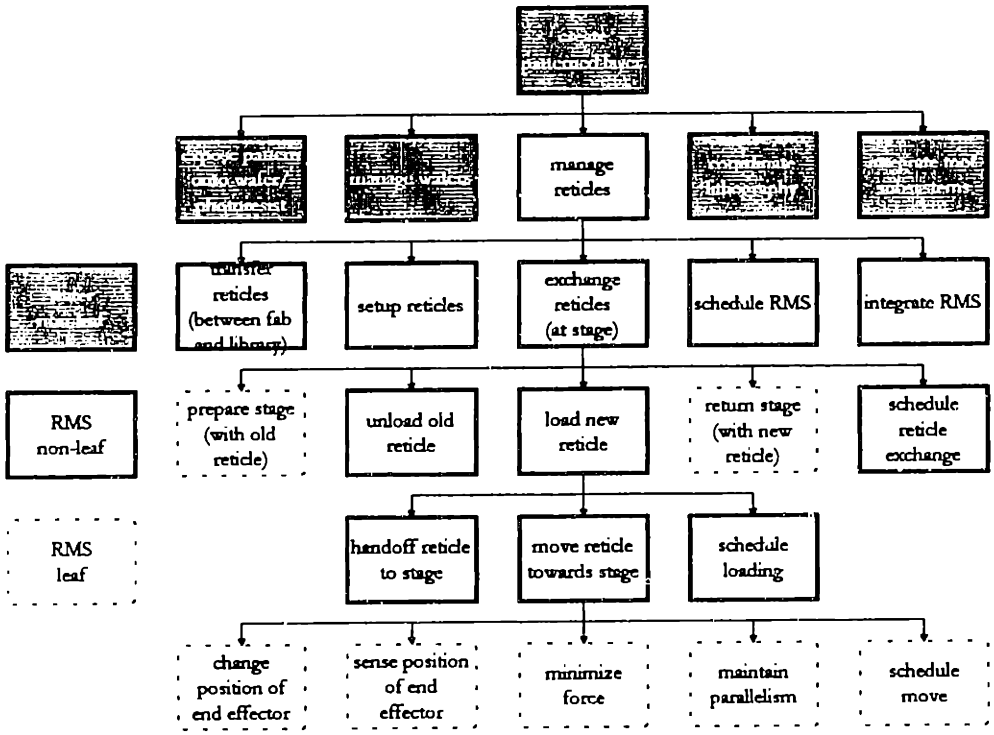


*Figure 6-7. Leaf nodes for one branch of the RMS*

*Other places where the decomposition sequence can be shown are the CMP, lithography tool, etc. Leaves can be shown with the track machine.*

### 6.4.1.2 Generating sub-FRs

The next activity considered is the generation of sub-FRs. These sub-FRs are generated for the FR "Manage reticles" satisfied by the DP "Reticle management system". The designers' goals in generating a complete set of sub-FRs are developing a set of sub-FRs which is sufficient, describes the parent DP, and is solution-neutral.

*sufficient set*

The sub-FRs generated for the RMS are shown in table 6-3 and figure 6-8. This set of sub-FRs follows the template for system design (guideline B3), and they are related to multiple sources of FRs (guideline B1).

> A template for system design consists of sub-FRs of the form: process and transport, command and control, and support and integration.
>
> All potential sources of sub-FRs at a level should be considered. These include, parent FR, parent DP, parent-level Cs, parent-level DM (as a source of either potential Cs or sub-FRs), and the set of CNs.

A good order to consider these sources is first to define sub-FRs based on knowledge of the parent DP. Second, define additional sub-FRs in accordance with the parent-level FRs and Cs. Finally, consider the parent DM and CNs.

**Table 6-3. Sub-FRs of FR-DP3 "Manage reticles" using "new RMS"**

| Index: 3b.1-5 | Functional Requirements (FRs) |
|---|---|
| P | Manipulate wafers (input/output) |
| | *Description* |
| 1 | Transfer reticles (between FAB and library) |
| 2 | Setup reticles (at WAIT, for exchange at the stage) |
| 3 | Exchange reticles (at the stage) |
| 4 | Schedule RMS |
| 5 | Integrate RMS |

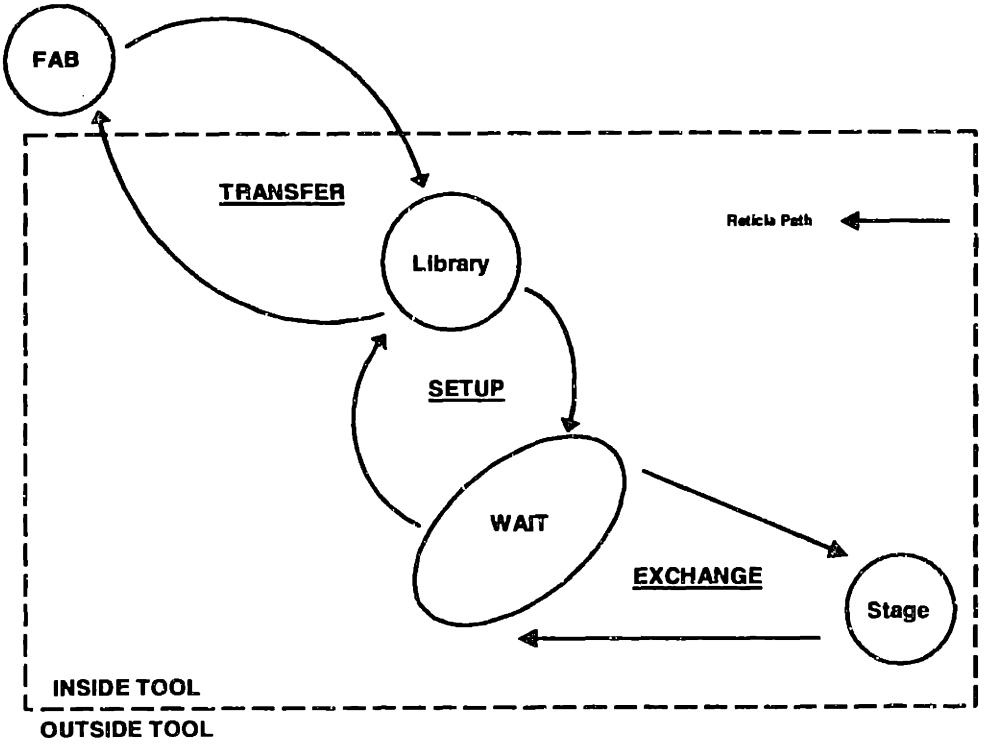


*Figure 6-8. Reticle movement through the RMS*

*The manufacturing cell serves as another example of the system template.*

*consistent set including describe parent DP*

The guidelines that concern consistently describing the parent DP deal with several elements that lack an obvious within the system architecture. The particular elements that are

considered in these guidelines are DP resources that vary with time (guideline B4), FRs that change dynamically with time (guideline B5), and "support FRs" (guideline B6).

> To develop a consistent set of sub-FRs, an FR to control the time variation of a DP resource (sequencing, scheduling, etc.) should be at the same level as the DP resource that is changing.
>
> Define support FRs at the parent level when support is carried to multiple sub-DPs on different branches. Then the connection, between the higher-level support DP and a particular DP requiring that support, will be a sub-FR.

In the example, the sub-FRs for reticle management at the top-level of the RMS are not all performed simultaneously. The guidelines describe the nature of the command and control algorithms (CCAs) that are used in the RMS. In particular, the CCA at the top-level of the RMS (FR3.4 being considered) is responsible for scheduling the DP resources at this level. In addition, the guideline about support says that some sub-FRs can be placed under the "support structure", connecting with the tool's power supplies, vacuum lines, environmental control, etc.

*The CMP example also illustrates the application of both of these ideas.*

*solution-neutral and minimum sub-FRs*

The guidelines for solution-neutral sub-FRs apply to the top-level of the RMS. Different possible DPs can be chosen (guideline B9). The number of intermediate states of the reticle are minimized (guideline B10), and few attributes on the reticle are specified per each FR (guideline B8).

> Articulating multiple alternative DPs serves as a check for solution neutrality.
>
> Minimize the number of intermediate states of the target object that are specified.
>
> Specify as few attributes on the target object transformed by the parent FR as possible, and specify as few new, additional target objects as possible.

The guidelines for generation of sub-FRs were followed. The attributes of the reticle that are specified for the FRs at this level concern the reticles' position and orientation (at the library, wait, and stage-load positions). The intermediate states of the reticle (the target object) are minimized, that is, the number of intermediate positions within the lithography tool. Setup and exchange cannot be combined with transfer because the library is located within the machine not external to it in the fab environment.

*The CMP and track machines serve as additional examples of the application of these guidelines. In the track machine, at the highest-level the attributes that are specified on the deposition (and the developing) of the photoresist are few; it is only through the selection of DPs, that other, additional process steps need to be added.*

*Furthermore, with the CMP machine, there are different ways that material can be removed, thus illustrating the creation of solution-neutral sub-FRs.*

#### 6.4.1.3 Carrying down and refining Cs

The activity of carrying down and refining constraints is important for the RMS system for two reasons. It operates within a highly constrained fab environment, and it is part of an existing system.

The top-level Cs for the lithography tool are shown in table 6-4. Their application (carrying down) to the sub-FRs that make up the RMS system is given in table 6-5.

**Table 6-4. Top-level constraints for lithography tool**

| Constraint Table | | | Impacts: FR | | | | |
|---|---|---|---|---|---|---|---|
| Index | Parent | Description | 1 | 2 | 3 | 4 | 5 |
| | | **Critical Performance Specifications** | | | | | |
| C-1 | Marketing | Meet throughput specifications | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-2 | Marketing | Meet process spec. (arbitrary # of instances) | ? | ? | ? | ? | ? |
| C-3 | Marketing | Meet transport spec. (arbitrary # of instances) | ? | ? | ? | ? | ? |
| | | **Interface Constraints** | | | | | |
| C-4 | Marketing | Handle customer specified target objects (arbitrary # of instances) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-5 | Marketing | Integrate tool with factory environment (host computer, air and water supply, facilities, etc.) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-6 | Marketing | Make tool "user-friendly" (ergonomics and software interfaces) | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | **Global Constraints** | | | | | |
| C-7 | Marketing | Maximize availability / reliability (minimize MTBF and minimize MTTF) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-8 | Marketing | Minimize footprint (do not exceed maximum size) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-9 | Management | Make tool serviceable (easy access for maintenance) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-10 | Marketing Management | Minimize costs (design, manufacturing, operational, maintenance, etc.) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-11 | Marketing Management | Provide ease of testability (make components compatible with standard and customer-defined tests) | ✓ | ✓ | ✓ | ✓ | ✓ |
| C-12 | Marketing | Conform to industry and safety standards | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | **Project Constraints** | | | | | |
| C-13 | Management | Integrate maximum amount of existing technology (minimize redesign of proven components, use off-the-shelf equipment whenever possible) | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | **Existing Constraints** | | | | | |
| C-14 | Marketing Management | Include specified components (arbitrary # of instances) | ? | ? | ? | ? | ? |

The different types of constraints are illustrated: performance specifications, interface constraints, global constraints, and project constraints.

*All* critical performance specifications should be refined as sub-FRs at lower levels of the design hierarchy.

Interface constraints will be refined into sub-FRs, assuming they are applicable.

Global object constraints will not be refined into sub-FRs; they will remain constraints even at the lower levels of the design hierarchy.

According to guideline C4 and C5, the constraints that are expected to "be refined into" into sub-FRs at lower levels of the design hierarchy are C3.1-6, the ones that are critical performance specifications and interface constraints. This plays out in the subsequent decomposition although it is not shown at this level. The sub-FRs that derive from Cs are shown in figure 6-9 as colored rectangles.

**Table 6-5. Constraints on decomposition of FR-DP3 "Manipulate reticles" using "new RMS"**

| ? | Parent | Constraints C3-? | Impacts FR3b.x 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| | | --- Critical Performance Specifications --- | | | | | |
| 1 | C-1 | Exchange time (the RMS component of stage down-time ≤ 10 seconds) | | | ✓ | ✓ | |
| | | --- Interface Constraints --- | | | | | |
| 2 | C-4 | Accommodate multiple types of reticle carriers with minimum design effort | ✓ | | | ✓ | ✓ |
| 3 | C-5 | Transfer non-product file reticles (w/o impacting exchange time) | ✓ | | | ✓ | ✓ |
| 4 | C-5 | Accommodate multiple types of reticle stages with minimum design effort | | ✓ | ✓ | ✓ | ✓ |
| 5 | C-6 | Make tool "user-friendly" (ergonomics and software interfaces) | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | C-5 | Anticipate future FAB automation technology i.e. overhead track, AGV | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | --- Global Constraints --- | | | | | |
| 7 | C-7 | Meet system requirements for reliability | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | C-8 | Minimize increase to tool footprint | ✓ | ✓ | ✓ | ✓ | ✓ |
| 9 | C-12 | Protect reticles from damage or contamination | ✓ | ✓ | ✓ | ✓ | ✓ |
| 10 | C-10 | Maintain or reduce costs while increasing functionality | ✓ | ✓ | ✓ | ✓ | ✓ |
| 11 | C-12 | Conform to SEMI / industry / safety standards | ✓ | ✓ | ✓ | ✓ | ✓ |
| 12 | C-9 | Make tool serviceable (easy access for maintenance) | ✓ | ✓ | ✓ | ✓ | ✓ |
| 13 | C-11 | Provide ease of testability (make components compatible with standard and customer-defined tests) | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | --- Project Constraints --- | | | | | |
| 14 | C-13 | Minimize unique hardware and software components e.g. Staubli RX60CR | ✓ | ✓ | ✓ | ✓ | ✓ |
| 15 | C-14 | Production ready design w/"modular" interface available for integration in 6/99 | ✓ | ✓ | ✓ | ✓ | ✓ |
| | | --- Feature Constraints --- | | | | | |
| | | N/A | | | | | |

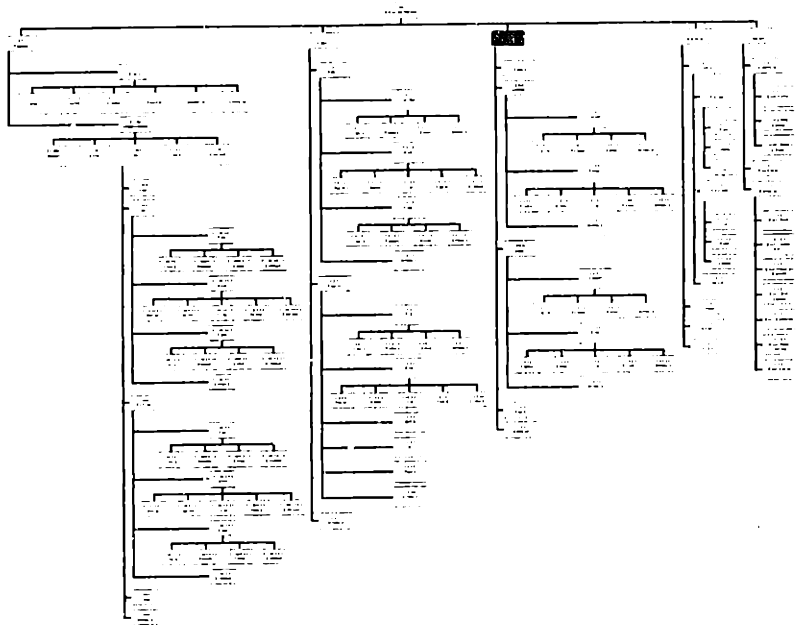

*Figure 6-9. Sub-FRs affected by constraints*

*Other cases that serve to illustrate these guidelines are the track machine and the manufacturing cell.*

### 6.4.1.4 Concept generation and selection

A set of sub-DPs shown in table 6-6 and equation 6-2 was chosen for the sub-FRs for the reticle management system. The design axioms were used in the process of generating and selecting these sub-DPs. The process then continued with the activities on the bottom row of figure 6-5. Two of these activities were not done at this time, dimensioning and layout. The one that is considered next is integration of sub-DPs.

**Table 6-6. Decomposition of FR-DP3 "Manage reticles" using "new RMS"**

| Index: 3b.1-5 | | |
|---|---|---|
| | **Functional Requirements (FRs)** | **Design Parameters (DPs)** |
| P | Manage reticles (input/output) | Reticle management system (RMS) |
| | *Description* | *Description* |
| 1 | Transfer reticles (between FAB and library) | Reticle carrier module |
| 2 | Setup reticles (at WAIT, for exchange at the stage) | Reticle setup scheme |
| 3 | Exchange reticles (at the stage) | Reticle exchange scheme |
| 4 | Schedule RMS | Command & control algorithm (CCA) during <tool, RMS> |
| 5 | Integrate RMS | RMS support framework |

The relationships between the system-level FRs and DPs for the new RMS are given in equation 6-2.

$$\begin{Bmatrix} \text{FR3.3 exchange\_reticles} \\ \text{FR3.1 transfer\_reticles} \\ \text{FR3.2 setup\_reticles} \\ \text{FR3.4 schedule\_RMS} \\ \text{FR3.5 integrate\_RMS} \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O \\ O & X & O & O & O \\ X & X & X & O & O \\ X & X & X & X & O \\ X & X & X & X & X \end{bmatrix} \begin{Bmatrix} \text{exhange\_scheme} \\ \text{carrier\_modules} \\ \text{setup\_scheme} \\ \text{command\_control} \\ \text{RMS\_framework} \end{Bmatrix} \quad (6\text{-}2)$$

### 6.4.1.5 Integrating sub-DPs

The guidelines for integration of sub-DPs are applied by considering the nature of the sub-FRs at this level. In the RMS design, the sub-FRs are not all performed at the same time, therefore according to guideline D1, the decision to integrate the DPs at this level into a single physical resources is reasonable. This means the sub-DPs on each branch are allowed to be integrated into the same resource. The details cannot be determined at this level.

> DPs which perform the same FRs on the same target object at different times (from the point of view of the machine), in different places in the design hierarchy may be integrated physically into the same unit.
>
> For FRs which are operations to be performed simultaneously (from the point of view of the machine), at sufficiently low levels of the design hierarchy, the resources embodying the DPs should consist of separate components.

Ultimately some of the sub-DPs are integrated, such as the "first mechanism". Other places in the design in which the resources for the sub-DPs cannot be integrated include the DPs for holding the reticles within the library, for example.

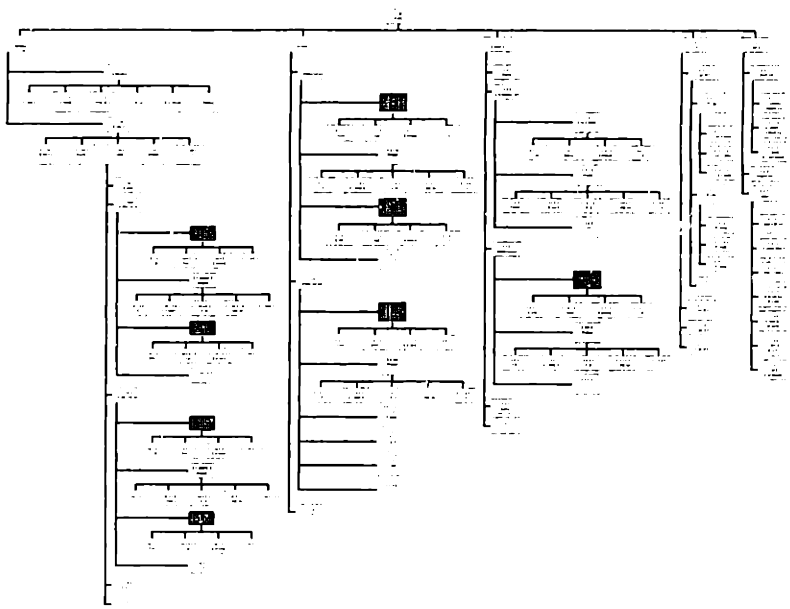

*Figure 6-10. Integration of sub-DPs into resources*

*Another example in which sub-DPs are integrated into a single physical resource is the CMP or the track machine.*

### 6.4.1.6 Check consistency of DMs, etc.

The guidelines for consistency within design matrices are applied to the RMS top-level. The sub-DPs at this level do not all require the same amount of subsequent decomposition. This guideline may seem relatively obvious, however it shows that the designers need not be concerned about whether the DPs at one node are at the same level of abstraction.

The other guidelines for this activity concern the placement of the off-diagonal terms in the design matrix (guidelines E2 and E3).

> The designers have a choice about the location of off-diagonal Xs in design matrices when the X represents an interface between two processes.
>
> Given the choice of where to place an off-diagonal term when considering two FRs and DPs, place it so the most technically challenging part of the design is done first. That is, the interface between two sub-systems should be defined with the sub-system that is more difficult to design.

The *Xs* in the design matrix for the top level of the RMS are consistent with the guidelines. They are "interfaces" between processes. The *X* between the setup and exchange is an example. Moreover, to the extent that the designers have freedom in choosing where to put the *X* between transfer and setup, it should be placed so that the design of the DP for transfer, the "modular interface" is done later.

*Other examples of the application of this guideline can be seen in the AD software.*

### 6.4.1.7 Sequencing the decomposition

After the activities in the bottom row of Figure 6-5, the designers return to the starting activity of the example, sequencing the decomposition. This time the next FR-DP pair to be broken down comes from the set of those that were just defined for the RMS. If the lithography tool itself were the focus, the sub-FRS of the RMS would not be the only candidates for decomposition, however.

None of the sub-FRs at this level are considered to be leaves (guideline A1).

> To determine when a node is a leaf, the designers have the option of considering a node to be a leaf—and thus not further decomposing it—when the target object of the sub-FR is different from that of its parent.

Furthermore, the sequence of design decisions followed was that described by the design matrix at this level (guideline A2).

> To identify the next FR-DP pair to decompose, at each level, define sub-FRs in the order described by the design matrices.

The sub-FR for reticle exchange is decomposed next. The sub-FRs were generated in the order "exchange reticles", "transfer reticles", "prepare reticles", "schedule RMS", and "integrate RMS".

## 6.4.2 Tools

*Case-study example: CCAs for CMP [adapted from [Hintersteiner and Tate (1998a), Hintersteiner and Tate (1998b), Melvin (1998), Melvin (1999a)]]*

The concepts of the command and control algorithms presented in section 4.3.1 in chapter 4 can be illustrated by examining the details of the CMP case-study example. The high-level decomposition is shown in table 6-7. The first two FRs are process modules required for planarizing and cleaning the deposited surface on a wafer between layers. FR3 is a system-level process module that transports wafers between the planarization and cleaning modules. FR4 is the CCA that is responsible for coordinating the interactions between the process and transport modules and for passing appropriate control parameters to each process and transport module. FR5 covers the necessary hardware for integrating the sub-systems.

**Table 6-7. System-level decomposition of the CMP machine**

| Index: I-5 | Functional Requirements (FRs) | | Design Parameters (DPs) |
|---|---|---|---|
| | *Type* | *Description* | *Description* |
| 1 | *process* | Remove material | Planarization process<br>(a) 3-body abrasion<br>(b) 2-body abrasion |
| 2 | *process* | Clean wafer | Cleaning sub-system |
| 3 | *transport* | Transport wafer | Wafer handler |
| 4 | *control* | Control tool | System-level CCA |
| 5 | *support* | Integrate tool subsystems | Tool support framework |

The relationships between the FRs and DPs for the top-level of the CMP machine are given in equation 6-3.

$$\begin{Bmatrix} \text{remove material} \\ \text{clean wafer} \\ \text{transport wafer} \\ \text{control tool} \\ \text{integrate sub - systems} \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O \\ X & X & O & O & O \\ X & X & X & O & O \\ X & X & X & X & O \\ X & X & X & X & X \end{bmatrix} \begin{Bmatrix} \text{planarization process} \\ \text{cleaning module} \\ \text{wafer handler} \\ \text{system - level CCA} \\ \text{support framework} \end{Bmatrix} \quad (6\text{-}3)$$

*Output of algorithms*

Figure 6-11 shows various CCAs for the CMP machine as they are distributed throughout the design hierarchy. The inputs and outputs of these different CCAs are given in table 6-8.

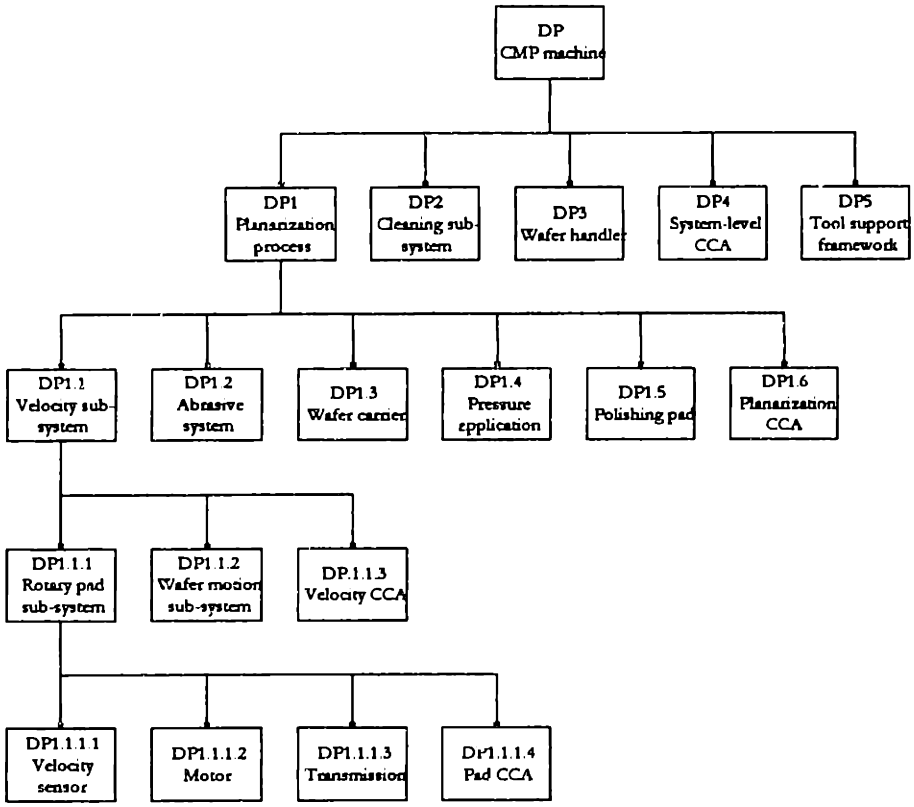


*Figure 6-11. The CCAs of the CMP machine shown among the various branches of the DP hierarchy*

**Table 6-8. Outputs of the CCAs within the CMP decomposition**

| CCA | Inputs | Outputs[10] |
|---|---|---|
| (4) system-level CCA | process recipe (sequence of FRs) | schedule of modules, MRRs, MRR for planarization[11] |
| (1a.6) planarization CCA[12] | planarization MRR | slurry flow rate(t), relative velocity(t), applied nominal force(t) |

[10] All outputs are given in terms of *desired* values, for example, desired pad velocity—except in the case of a *delta* value which is an amount calculated as the needed adjustment amount. The delta amount is based on a comparison of the desired value and measured value (from a sensor).

[11] This CCA passes its input to the CCAs at the next, lower level.

[12] CCAs are named in table 6-8 according to their parent DP, so for example, the planarization CCA is part of the planarization module, and the wafer carrier CCA is part of the wafer carrier.

| | | |
|---|---|---|
| (1a.3.6) wafer carrier CCA | planarization MRR | back-pressure(t),<br>retaining ring position(t),<br>vacuum(t)<br>wafer tilt(t)[13] |
| (1a.1.3) velocity CCA | relative velocity(t) | wafer velocity(t)<br>pad velocity(t) |
| (1a.1.1a.4) pad motion CCA | desired pad velocity(t) | delta velocity(t) |

## *6.5 Summary and conclusions*

This section summarizes academic and industry considerations.

### 6.5.1 Academic considerations

The types of cases that have been examined cover a wide breadth. They are taken from different fields, involve different amounts of detail, different numbers of designers, etc. And the parts of the theory presented have been both applicable and useful. Therefore, it is concluded that the theory presented in the preceding chapters meets the criteria for a progressive research program as described in section 5.2.1:[14]

- Theoretically progressive condition: Does the theory make new and interesting predictions or explanations?

The theory for decomposition activities makes predictions about designs and describes the successes and failures of designers in a way that augments the existing body of knowledge provided by axiomatic design: Only this theory combines the broad scope of describing decomposition activities with relatively clearly articulated concepts for applying general principles to many particular cases.

- Empirically progressive condition: Have some of these predictions and explanations been corroborated by the empirical evidence?

The theory *has* proven useful for answering real-world questions encountered by designers in industry and academia as evidenced by the cases presented in this chapter.

- Heuristically progressive condition: As anomalies are identified, are they being accommodated and explained in a manner consistent with the spirit of the heuristic of the theory, not in an ad hoc manner?

While the theory for decomposition activities is not yet able to answer *every* question about decomposition, the presence of questions to be answered or concepts to be further elaborated, is a hallmark of a healthy research program and provides a direction for further research. It *is* a reasonable expectation that *the same approach* can be used to shed light on other decisions of the designers that are not yet understood.

---

[13] The presence or absence of this CCA depends on whether there is dynamic control of the gimbal or not.

[14] See also section A1.5.4 in appendix 1.

### 6.5.2 Industry considerations

The importance and impact of this work can in part be evaluated by its applicability to real-world industrial problems. A motivation, given in chapter 1, has been to address the needs of designers working in industry in applying axiomatic design. To this end, this theory of decomposition activities along with guidelines and tools for system design, provides answers to questions that arise as designers apply axiomatic design to larger-scale, distributed design tasks.

It should be noted that companies have adopted different strategies for incorporating axiomatic design within their development processes. These have been termed the *top-down* and the *diffusive* approaches.[15] There are benefits and risks associated with each.

### 6.5.3 Extension of work to software

This work can be extended to software tools to assist designers. It can be used to orient them with respect to the task at hand. Then it can provide appropriate theories and tools as needed. This way the designers can focus on the creative activity of concept generation and analysis, aided by the design axioms.

---

[15] The terms are from [Nordlund, et al. (1996)]. Other sources which describe the adoption of axiomatic design by industry are [Fredriksson (1994), Fredriksson, et al. (1994)].

# CHAPTER 7: CONCLUSIONS

## *7.1 Conclusions*

I have intended to make the following points in this document:

*Chapter 1 (and Appendix 4)*

Companies have a variety of objectives in applying design theories to their product development processes including reducing product development time and cost, improving product functionality and reliability, and reducing product life-cycle costs. To address these needs, academia is developing the field of design theory that relates the design process, the design object, designers, specific field knowledge, and/or resources for development.

A unifying paradigm for design research is available—that used in developing axiomatic design theory—and it is used in this thesis. Important characteristics of this view are the recognition of the importance of rational decision making, an identification of the cognitive aims of designers, and the development of a robustly descriptive theory of design with obviously prescriptive consequences.

The particular facet of design theory developed in this thesis is a theory for *decomposition.* It is the way in which decisions and information about the design object are organized during the design process through the definition of a set of sub-FRs given a parent FR-DP pair and including all activities which connect multiple levels of the design hierarchy. The intention of this work is to replace the current, ad hoc process for decomposition with one more useful to designers by providing an answer to this question:

> *What is a process for decomposition—consisting of activities, tools, and theories—that empowers designers to make rational and consistent design decisions across multiple levels of the design hierarchy?*

*Chapter 2*

In performing decomposition, a measure of success is that of *consistency.* That is, do the sub-FRs, and the rest of the decomposed design, match the design decisions and the representations of the design that were made at higher levels of the design hierarchy? Good decision making in performing decomposition leads to design hierarchies that describe a design object at multiple layers of abstraction, but that consistently describe the same design object.

A model is needed of the decomposition process that identifies the activities performed, provides guidelines and tools to assist the designers, guides designers in the sequence of the decomposition process, and places decomposition into the overall context of system design.

The goal is to understand the means by which the desired objectives of the designers are achieved. This task is accomplished by identifying for each of the activities performed by the designers, the designers' cognitive aims, the options faced by the designers, and rules correlating options to cognitive aims.

A *system* and *system design* are defined, and the characteristics that distinguish system design from component design are identified. When designers conceptualize a system, they consider a DP that is decomposed into an assemblage of disparate elements that acting together

perform the desired higher-level function. The differences between system and component design motivate the need for additional tools specifically intended for system design. Two of these proposed here are command and control algorithms (CCAs) and flexibility.

*Chapter 3 (and Appendix 2)*

A palette of activities and guidelines that can be applied in decomposition is developed. This work has a practical slant towards addressing some of the questions that arise in applying axiomatic design to real, industrial design tasks. This is apparent by looking at the questions that have been identified and answered for each of the decomposition activities: definition of sub-FRs, identification of relevant CNs, physical integration of DPs, directing progress of the decomposition, dimensioning and configuration of DPs, layout of DPs, carrying down and refining Cs, and ensuring consistency between levels.

Specifically the following contributions are made:

- sequencing the decomposition: guidelines for identifying leaf options and guidelines for guiding the decomposition process (what items and order are important in defining sub-FRs)
- defining sub-FRs: guidelines for developing a complete set of sub-FRs
- constraints: tools for documenting constraints, guidelines for identifying the impacts of constraints (such as which ones become sub-FRs and which ones do not)
- physical integration: guidelines for when DPs can be integrated into one resource unit
- consistency: a procedure for defining sub-FRs so DM elements are consistent
- overall roadmap and method: understanding the relationship between activities in design and decomposition, identifying specific goals, and mapping activities to available design theories

*Chapter 4*

In this chapter, tools are presented that address situations that are encountered in specifically system design. The tools which are presented here are modeling and control of time variation within a system design and representation of flexibility.

Considerations of time variation are shown to have substantial impacts on two areas of the design: physical integration of DPs and allocation of DP resources through command and control algorithms (CCAs). These CCAs perform three functions: sequencing sub-FRs, scheduling DPs, and controlling hardware parameter values. This chapter has demonstrated how command and control issues can be represented in a system architecture. It shows that the components (DPs) required for control are spread out over the multiple branches of the design hierarchy in which active control is required. On the lowest levels, specific actuator and sensor equipment is correlated with a hardware control algorithm to coordinate low-level inputs and outputs. On higher levels of the hierarchy, command algorithms exist to schedule and coordinate all of the lower-level command and control algorithms.

The concept of *flexibility in FR spans* is introduced. This deals with dynamic "spans" in the specification of FR values, how these can be accommodated in DP selection, and how the concept of information content and its calculation can be extended to deal with them.

The contributions of the chapter include detailing the following:

- software and hardware interfaces, the integration of command and control within the system architecture
- physical integration of DPs into resources that can be dynamically allocated to meet the changing FRs
- extension of information content to include variety in inputs and outputs

*Chapter 5 (and Appendices 1 and 3)*

It is the belief of the author that *research into design can be performed in a scientific way*: that scientific methods can be applied to expand our understanding of design and the principles which underlie it. Axiomatic design theory possesses all the necessary components of a progressive research program: It makes unique predictions about design and the successes and failures of designers; it has proven useful for answering real-world questions encountered by designers in industry; and it continues to successfully digest new problems to be solved without resorting to ad hoc strategies in modifying the theory.

The approach followed in this work is consistent with that followed in axiomatic design theory and also meets the criteria of a progressive research program. The hypotheses of this work are empirically validated through several industrial cases.

*Chapter 6*

In summary, the cases presented in this chapter demonstrate the usefulness of the theory presented in chapters 2 through 4. The roadmap of decomposition activities serves as a guide for the designers in performing decomposition and system integration within the context of axiomatic design.

Furthermore, the goals articulated for the different decomposition activities are found to correspond to the goals of the designers in practice, and the guidelines are found to provide appropriate support to the designers in satisfying their goals.

The cases to which these ideas have been applied, in which the author participated, include the following: design of a tool-exchange mechanism (RMS) which accommodates much customer variety, software control algorithms for machine control, and diagnosis of coupling in machines. Other cases incorporating these ideas, but without the author's direct involvement include these: reuse of design rationale for manufacturing cell design, software designs, and system analysis.

The following innovative results were obtained in designing and analyzing the RMS:

- development of design concept from system-level to leaves (hardware or software)
- integration of software design with hardware design, as part of the same hierarchy
- physical integration of DPs into resources for providing functions at different times including selection of appropriate hardware: end effector, mechanisms, etc.
- development of a "modular interface" that incorporates new and anticipates future CNs and the prediction of the design effort for new variants of the "modular interface"
- integration of axiomatic design on a project with multiple engineers and distributed task responsibility

The roadmap for decomposition was able to guide the design of the CMP machine in several ways [Melvin (1999b)]:

- guidelines: The specific guidelines that the designer singled out as useful include considering all sources of sub-FRs (guidelines B1 and B2), identifying multiple alternative DPs to check solution-neutrality (guideline B9), defining support FRs (guideline B6), and carrying down and refining Cs (guidelines C1-C8).
- CCAs: The role of CCAs in enabling operational modularity is useful for the CMP tool. There are several different processes that the machine may be required to perform, and the ability to reconfigure the machine dynamically increases its flexibility greatly. The need to dynamically change the FRs and DPs of the machine depending on the particular wafer requirements may be handled in an efficient manner using CCAs. This overall machine control must be balanced with the low-level control of individual axes. The natural refinement of the CCAs as the hierarchy progresses allows the control system to develop in an integrated manner with the rest of the machine.

The roadmap for decomposition described the activities performed by the designer during the progress of the AD software project. The theory for decomposition activities was useful in the following areas:

- activities and goals: Articulating a set of goals for the decomposition activities was useful in guiding the designer, particularly in terms of assistance in generating sub-FRs (keeping in mind all the relevant goals). Understanding design dependencies as given in the design matrix and their implications on the sequence of decomposition was also useful.
- sequence of activities: The sequence of activities in decomposition model proved useful and is being incorporated into the latest version of the software.
- flexibility: The concept of flexibility was useful in structuring the options available to the user and providing and representing the functionality to enable the options.

According to Hintersteiner, the designer performing the system analysis of the photolithography tool, the theory for decomposition activities was useful in the following areas [Hintersteiner (1998a)]:

- constraint management: The theory provided guidance about the treatment of constraints. The way in which the system-level constraints were carried down to the lower-level FRs was particularly important and relevant.
- decomposition sequence: This design exhibits many instances in which the order of design decisions for one subsystem of the design must precede those of another subsystem. And knowledge of how to identify leaves and thus be able to stop decomposing was important in this analysis. Given the scope of the task–to create a system architecture for the whole machine–knowledge of when to stop kept the project manageable; otherwise it had the potential to continue indefinitely.
- levels of abstraction: It was useful to recognize that the DPs at a single layer of the design hierarchy are not all at the same level of abstraction; therefore they do not all need to be decomposed the same number of additional layers.

- defining sub-FRs: The template for system design was used to structure the sets of sub-FRs, and the multiple sources of sub-FRs were considered in developing a complete set.

The types of cases that are examined cover a wide breadth. They are taken from different fields, involve different amounts of detail, different numbers of designers, etc. The parts of the theory presented are both applicable and useful. Therefore, it is concluded that the theory presented in the preceding chapters meets the criteria for a progressive research program.

# REFERENCES

[1] "Discussion Sessions: A Summary Report", in *Research in Design Thinking*, N. Cross, K. Dorst and N. Roozenburg, (eds.), Delft, Netherlands: Delft University Press, pp. 213-215, 1992.

[2] Aasland K.E., "An Extensive Product Model for Design History Documentation", *Ph.D. Thesis*, Department of Machine Design and Materials Technology, Norwegian Institute of Technology (NTH), Trondheim, Norway, 1995.

[3] Albano L.D., "An Axiomatic Approach to Performance-based Design", *Ph.D. Thesis*, Department of Civil Engineering, MIT, Cambridge, MA, 1992.

[4] Altshuller G.S., *Creativity as an Exact Science*, (translated by A. Williams), New York: Gordon and Breach, 1988. ISBN 0-677-21230-5

[5] Andreasen M.M., "Design Methodology", Seminar on Design Theory and its Applications, The Norwegian Academy of Technical Sciences, Trondheim, Norway, May, 1991a.

[6] Andreasen M.M., "The Theory of Domains", Workshop on Understanding Function and Function to Form Evolution, Cambridge, UK, 1991b.

[7] Andreasen M.M., "The Role of Atrefact Theories in Design", in *Universal Design Theory*, H. Grabowski, S. Rude and G. Grein, (eds.), Aachen, Germany: Shaker Verlag, pp. 57-72, 1998. ISBN 3-8265-4265-7

[8] Antonsson E.K., "Development and Testing of Hypotheses in Engineering Design Research", *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 109, pp. 153-154, 1987.

[9] Araujo C.S., Benedetto-Neto H., Campello A.C., Serge F.M., Wright I.C., "The Utilization of Product Development Methods: A Survey of UK Industry", *Journal of Engineering Design*, Vol. 7, No. 3, pp. 265-277, 1996.

[10] Arciszewski T., "Design Theory and Methodology in Eastern Europe", ASME, *DE-Vol. 27 (DTM 90)*, Chicago, IL, pp. 209-217, 1990.

[11] Bailey M.T., "Do Physicists Use Case Studies? Thoughts on Public Administration Research", *Public Administration Review*, Vol. 52, No. 1, pp. 47-54, 1992.

[12] Baldwin D.F., "Microcellular Polymer Processing and the Design of a Continuous Sheet Processing System", *Ph.D. Thesis*, Department of Mechanical Engineering, MIT, Cambridge, MA, 1994.

[13] Baya V., Leifer L.J., "A Study of the Information Handling Behavior of Designers during Conceptual Design", ASME, *DE-Vol. 68 (DTM 94)*, pp. 153-160, 1994.

[14] Bjärnemo R., "Formaliserade Konstruktionsarbetssätt", *Licentiate Thesis*, Inst. for Machine Design, University of Lund, Lund, Sweden, 1983. LUTMDN/(TMKT-1008)/1-426/(1983)

[15] Blum B.I., *Beyond Programming*, New York: Oxford University Press, 1996. ISBN 0-19-509160-4

[16] Bucciarelli, L.L., *Designing Engineers*, Cambridge, MA: MIT Press, 1994. ISBN 0-262-02377-6

[17] Burchill G.W., "Concept Engineering : An Investigation of Time vs. Market Orientation in Product Concept Design", *Ph.D. Thesis*, Sloan School of Management, MIT, Cambridge, MA, 1993.

[18] —, "Concept Engineering, Document 7I", Cambridge, MA: Center for Quality of Management, 1992.

[19] Clausing D., *Total Quality Development*, New York: ASME Press, 1994. ISBN 0-7918-0035-0

[20] Collins R., *The Sociology of Philosophies*, Cambridge, MA: Belknap Press of Harvard University Press, 1998. ISBN 0-674-81647-1

[21] Cross N. (ed.), *Developments in Design Methodology*. Chichester: John Wiley and Sons, 1984. ISBN 0 471 10248 2

[22] Cross N., "Research in Design Thinking", in *Research in Design Thinking*, N. Cross, K. Dorst and N. Roozenburg, (eds.), Delft, Netherlands: Delft University Press, pp. 3-10, 1992.

[23] Cross N., "Science and Design Methodology: A Review", *Research in Engineering Design*, Vol. 5, pp. 63-69, 1993.

[24] Cross N., *Engineering Design Methods*, Chichester: John Wiley, 1994. ISBN 0-471-94228-6

[25] Curd M., Cover J.A., *Philosophy of Science: The Central Issues*, New York, NY: W.W. Norton and Company, 1998. ISBN 0-393-97175-9

[26] Dixon J.R., "On Research Methodology Towards a Scientific Theory of Engineering Design", *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, Vol. 1, No. 3, pp. 145-157, 1987.

[27] Do S.-H., "Design of Axiomatic Design Software", working paper, MIT, Cambridge, MA, Oct. 20, 1998.

[28] Do S.-H., Suh N.P., "Axiomatic Design Software", in *NSF-sponsored Axiomatic Design Workshop for Professors*, unpublished software, Cambridge, MA: MIT, 1998.

[29] Dretske F.I., "Laws of Nature", *Philosophy of Science*, Vol. 44, pp. 248-268, 1977. Reprinted in [Curd and Cover (1998)]

[30] Dwarakanath S., Blessing L., Wallace K., "Descriptive Studies: A Starting Point for Research in Engineering Design", in *Advances in Mechanical Engineering*, T.S. Mruthyunjaya, (ed.), New Delhi, India: Narosa Publishing House, pp. 341-361, 1996.

[31] Eder W.E., "Design Science–Meta-Science to Engineering Design", ASME, *DE-Vol. 27 (DTM 90)*, Chicago, IL, pp. 327-335, Sept. 16-19, 1990a.

[32] Eder W.E., "Engineering Design–A perspective on UK and Swiss Developments", ASME, *DE-Vol. 27 (DTM 90)*, Chicago, IL, pp. 225-234, Sept. 16-19, 1990b.

[33] Eekels J., Roozenburg N.F.M., "A Methodological Comparison of the Structures of Scientific Research and Engineering Design: Their Similarities and Differences",

*International Conference on Engineering Design (ICED 91)*, Zurich, Switzerland, pp. 58-69, Aug. 27-29, 1991.

[34] Erixon G., "Modular Function Deployment-A Method for Product Modularisation", *Docteral Thesis*, Department of Manufacturing Systems, The Royal Institute of Technology (KTH), Stockholm, Sweden, 1998. ISSN 1104-2141

[35] Evbuomwan N.F.O., Sivaloganathan S., Jebb A., "A Survey of Design Philosophies, Models, Methods and Systems", *Proceedings IMechE Part B: Journal of Engineering Manufacture*, Vol. 210, No. B4, pp. 301-320, 1996.

[36] Finger S., Dixon J.R., "A Review of Research in Mechanical Engineering Design. Part I and Part II", *Research in Engineering Design*, Vol. 1, pp. 51-67, 121-137, 1989.

[37] Föllesdal D., Wallöe L., Elster J., *Argumentationsteori, Språk och Vetenskapsfilosofi*, Stockholm, Sweden: Bokförlaget Thales, 1993.

[38] Fortenberry N.L., "Analysis and Synthesis of Structured Systematic Design Theories", *Ph.D. Thesis*, Department of Mechanical Engineering, MIT, Cambridge, MA, 1991.

[39] Fredriksson B., "Holistic Systems Engineering in Product Development", in *Saab-Scania Griffin*, vol. 1994/95, Linköping, Sweden: Saab-Scania AB, S-581 88, pp. 23-31, 1994.

[40] Fredriksson B., Killander A., Nordlund M., "An Effective Model of Transferring New Methods from Academia to Industry", Second CIRP International Workshop on Design Theory and Methodology, CIRP, Stockholm, Sweden, pp. 57-67, June 16-17, 1994.

[41] Friedman G., Hintersteiner J.D., Laganza J., Tate D., "Axiomatic Design of the Reticle Management System (RMS)", Oct., 1998a.

[42] Friedman G., Hintersteiner J.D., Tate D., Zimmerman R., "Representation of Constraints in the System Architecture", working paper, MIT, Cambridge, MA, 1998b.

[43] Fulcher A.J., Hills P., "Towards a Strategic Framework for Design Research", *Journal of Engineering Design*, Vol. 7, No. 2, pp. 183-193, 1996.

[44] Gebala D.A., Suh N.P., "An Application of Axiomatic Design", *Research in Engineering Design*, Vol. 3, pp. 149-162, 1992.

[45] Gillespie R., *Manufacturing Knowledge*, Cambridge, UK: Cambridge University Press, 1991. ISBN 0-521-45643-6

[46] Harutunian V., Nordlund M., Tate D., Suh N.P., "Decision Making and Software Tools for Product Development Based on Axiomatic Design Theory", *Annals of the CIRP*, Vol. 45/1, 1996.

[47] Harutunian V., Tate D., Suh N.P., "Axiomatic Design Software", in *Department of Mechanical Engineering*, unpublished software, Cambridge, MA: MIT, 1997.

[48] Hazelrigg G.A., "On Irrationality in Engineering Design", *Journal of Mechanical Design*, Vol. 119, pp. 194-196, 1997.

[49] Hempel C.G., *Philosophy of Natural Science*, Englewood Cliffs, NJ: Prentice-Hall, 1966.

[50] Hintersteiner J.D., personal communication, Nov., 1998a.

[51] Hintersteiner J.D., "System Architecture for the Micrascan III", Aug., 1998b.

[52] Hintersteiner J.D., "A Fractal Representation for Systems", International CIRP Design Seminar, CIRP, Enschede, the Netherlands, Mar. 24-26, 1999.

[53] Hintersteiner J.D., Tate D., "CMP System Architecture Description", working paper, MIT, Cambridge, MA, Mar. 16, 1998a.

[54] Hintersteiner J.D., Tate D., "Command and Control in Axiomatic Design Theory: Its Role and Placement in the System Architecture", Second International Conference on Engineering Design and Automation, Maui, HI, Aug. 9-12, 1998b.

[55] Hongo K., "On the Significance of the Theory of Design", *International Symposium on Design and Synthesis*, Elsevier Science Publishers B.V. (North Holland), *Design and Synthesis*, Tokyo, Japan, pp. 169-174, July 11-13, 1984, 1985.

[56] Hubka V., Eder W.E., *Engineering Design*, Zürich, Switzerland: Heurista, 1992. ISBN 3-85693-026-4

[57] Hubka V., Eder W.E., *Design Science*, London: Springer, 1996. ISBN 3-540-19997-7

[58] Hundal M.S., "Research in Design Theory and Methodology in West Germany", ASME, *DE-Vol. 27 (DTM 90)*, Chicago, IL, pp. 235-238, Sept. 16-19, 1990.

[59] INCOSE I.C.o.S.E., *Systems Engineering Handbook*, (edited by S.F.B.A. Chapter), Release 1.0, 2033 Sixth Ave., Suite 804, Seattle, WA 98121: INCOSE, 1998. http://www.incose.org

[60] Jones J.C., "A Method of Systematic Design", in *Conference on Design Methods*, J.C. Jones and D.G. Thornley, (eds.), New York: Macmillan, pp. 53-73, 1962.

[61] Karandikar H., Shupe J., "The Study of Design Theory and Methodology in Mechanical Engineering: A Commentary", ASME, *DE-Vol. 83 (DTM 95)*, Boston MA, pp. 361-370, 1995.

[62] Kim S.-J., Suh N.P., Kim S.-G., "Design of Software System Based on Axiomatic Design", *Annals of the CIRP*, Vol. 40/1, pp. 165-170, 1991a.

[63] Kim S.-J., Suh N.P., Kim S.-G., "Design of Software System Based on Axiomatic Design", *Robotics and Computer-Integrated Manufacturing*, Vol. 8, No. 4, pp. 243-255, 1991b.

[64] Kitcher P., *The Advancement of Science*, New York: Oxford University Press, 1993. ISBN 0-19-509653-3

[65] Kuhn T.S., *The Structure of Scientific Revolutions*, Chicago, IL: University of Chicago Press, 1970. ISBN 0-226-45804-0

[66] Kuhn T.S., *The Essential Tension*, Second edition, Chicago, IL: University of Chicago Press, 1977. ISBN 0-226-45806-7

[67] Lakatos I., *Philosophical Papers Volume 1: The Methodology of Scientific Research Programmes*, (edited by J. Worrall and G. Currie), Cambridge, UK: Cambridge University Press, 1978. ISBN 0-521-2803-1

[68] Lanza M., "Target-oriented Product Data Management (PDM)", working paper, University of Karlsruhe, Institute for Machine Tools and Production Science, Karlsruhe, Germany, Oct., 1998.

[69] Larvor B., *Lakatos: An Introduction*, London: Routledge, 1998. ISBN 0-415-14276-8

[70] Laudan L., "Progress or Rationality? The Prospects for Normative Naturalism", *American Philosophical Quarterly*, Vol. 24, pp. 19-33, 1987. Reprinted in [Laudan (1996), pp. 125-141]

[71] Laudan L., *Beyond Positivism and Realism*, Boulder, CO: Westview Press, 1996. ISBN 0-8133-2469-6

[72] Laudan L., Laudan R., "Dominance and the Disunity of Method: Solving the Problems of Innovation and Consensus", *Philosophy of Science*, Vol. 56, pp. 221-237, 1989. Reprinted in [Laudan (1996), pp. 231-243]

[73] Lee T.-S., "The System Architecture Concept in Axiomatic Design Theory: Hypotheses Generation and Case-Study Validation", *S.M. Thesis*, Department of Mechanical Engineering, MIT, Cambridge, MA, 1999.

[74] Lindholm D., "Towards Procedural Top-down Design: A Method based on Functional Requirements", *Licentiate Thesis*, Department of Mechanical Engineering, Linköping University, Linköping, Sweden, 1998. ISBN 91-7219-137-6

[75] Lindholm D., Tate D., Harutunian V., "Consequences of Design Decisions in Axiomatic Design Theory", Third World Conference on Integrated Design and Process Technology, Society for Design and Process Science (SDPS), Berlin, Germany, July 6-9, 1998.

[76] LPM, *The Language Processing Method, Document ML0060*, Cambridge, MA: Center for Quality of Management, 1996.

[77] Marca D.A., McGowan C.L., *IDEF0/SADT*, San Diego, CA: Eclectic Solutions, 1993. ISBN 0-9638750-0-0

[78] Marples D.L., "The Decisions of Engineering Design", *IRE (Institute of Radio Engineers) Transactions on Engineering Management*, Vol. EM-8, No. 2, pp. 55-71, 1961.

[79] Mårtensson P., Tate D., "Reuse of Design Rationale in Cellular Manufacturing Systems: Theory and Application", Second International Conference on Engineering Design and Automation, Maui, HI, Aug. 9-12, 1998.

[80] McMullin E., "Rationality and Paradigm Change in Science", in *Thomas Kuhn and the Nature of Science*, P. Horwich, (ed.), Cambridge, MA: MIT Press, pp. 55-78, 1993. Reprinted in [Curd and Cover (1998), pp. 119-138]

[81] Melvin J.W., "CMP Machine Decomposition", MIT, Cambridge, MA, Sept. 23, 1998.

[82] Melvin J.W., *S.M. Thesis*, Department of Mechanical Engineering, MIT, Cambridge, MA, 1999a.

[83] Melvin J.W., personal communication, Feb., 1999b.

[84] Merriam-Webster's, *Merriam-Webster's Collegiate Dictionary*, Tenth edition, Springfield, MA: Merriam-Webster, 1996. http://www.eb.com:180/search/search.htcl

[85] Miles L.D., *Techniques of Value Analysis and Engineering*, Second edition, New York: McGraw-Hill, 1972.

[86] Nordlund M., "Application of System Theories and AI Tools in Aircraft Design", 5th AIAA/USAF/NASA/ISSMO Multidisciplinary Optimization Symposium, Panama City Beach, FL, Sep. 7-9, 1994.

[87] Nordlund M., "An Information Framework for Engineering Design based on Axiomatic Design", *Doctoral Thesis*, Department of Manufacturing Systems, The Royal Institute of Technology (KTH), Stockholm, Sweden, 1996. ISRN KTH/TSM/R-96/11-SE

[88] Nordlund M., Tate D., "Research Methods for Design Theory", working paper, MIT, Cambridge, MA, 1996.

[89] Nordlund M., Tate D., Suh N.P., "Growth of Axiomatic Design through Industrial Practice", 3rd CIRP Workshop on Design and the Implementation of Intelligent Manufacturing Systems, Tokyo, Japan, pp. 77-84, June 19-22, 1996.

[90] Pahl G., Beitz W., *Engineering Design*, (edited by K. Wallace), New York: Springer-Verlag, 1988. ISBN 0-387-50442-7

[91] Pahl G., Beitz W., *Engineering Design*, (edited by K. Wallace), Second edition, London: Springer, 1996. ISBN 3-540-19917-9

[92] Phadke M.S., *Quality Engineering using Robust Design*, Englewood Cliffs, NJ: Prentice Hall, 1989.

[93] Pugh S., *Total Design*, Wokingham, England: Addison-Wesley, 1991. ISBN 0-201-41639-5

[94] Pugh S., *Creating Innovative Products using Total Design*, (edited by D. Clausing and R. Andrade), Reading, MA: Addison-Wesley, 1996. ISBN 0-201-63485-6

[95] Reich Y., "Annotated Bibliography on Research Methodologies", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 8, pp. 355-366, 1994.

[96] Reich Y., "The Study of Design Research Methodology", *Journal of Mechanical Design*, Vol. 117, pp. 211-214, 1995.

[97] Reitman V., Simison R.L., "Japanese Car Makers Speed Up Car Making", *Wall Street Journal*, New York, NY, p. B1 and B2, Dec. 29, 1995

[98] Ross D.T., "Structured Analysis (SA): A Language for Communicating Ideas", *IEEE Transactions on Software Engineering*, Vol. SE-3, No. 1, pp. 16-34, 1977.

[99] Ross D.T., "Applications and Extensions of SADT", *Computer*, Vol. 18, No. 4, pp. 25-34, 1985.

[100] Ross D.T., Schoman K.E., Jr., "Structured Analysis for Requirements Definition", *IEEE Transactions on Software Engineering*, Vol. SE-3, No. 1, pp. 6-15, 1977.

[101] Ruben D.-H., *Explaining Explanation*, New York, NY: Routledge, 1990.

[102] Schulz A.P., Clausing D.P., "An Embedding Process Framework for Total Technology Development", working paper, MIT Center for Innovation in Product Development, Cambridge, MA, June, 1998.

[103] Snyder L.J., "Is Evidence Historical?", in *Scientific Methods: Conceptual and Historical Problems*, P. Achinstein and L.J. Snyder, (eds.), Malabar, FL: Krieger Pbulishing Company, pp. 95-117, 1994. Reprinted in [Curd and Cover (1998)]

[104] Sohlenius G., "Presidential Address", *Annals of the CIRP*, Vol. 39/2, 1990.

[105] Sontow K., Clausing D.P., "Integration of Quality Function Deployment with Further Methods of Quality Planning", working paper LMP-93-005, MIT Laboratory for Manufacturing and Productivity, Cambridge, MA, Apr. 3, 1993.

[106] Steinberg L., "Research Methodology for AI in Design", *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, Vol. 8, pp. 283-287, 1994.

[107] Sturges R.H., Shaughnessy K.O., Kilani M.I., "Computational Model for Conceptual Design based on Extended Function Logic", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, No. 4, pp. 255-274, 1996.

[108] Suh N.P., *The Principles of Design*, New York: Oxford University Press, 1990. ISBN 0-19-504345-6

[109] Suh N.P., "Axiomatic Design of Mechanical Systems", *ASME 50th Anniversary of the Design Engineering Division: A Special Combined Issue of the Journal of Mechanical Design and the Journal of Vibration and Acoustics*, Vol. 117(B), pp. 2-10, 1995a.

[110] Suh N.P., "Design and Operation of Large Systems", *Journal of Manufacturing Systems*, Vol. 14, No. 3, pp. 203-213, 1995b.

[111] Suh N.P., "Designing-in of Quality through Axiomatic Design", *IEEE Transactions on Reliability*, Vol. 44, No. 2, pp. 256-264, 1995c.

[112] Suh N.P., "Impact of Axiomatic Design", 3rd CIRP Workshop on Design and the Implementation of Intelligent Manufacturing Systems, Tokyo, Japan, pp. 8-17, June 19-22, 1996.

[113] Suh N.P., "Design of Systems", *Annals of the CIRP*, Vol. 46, No. 1, pp. 75-80, 1997.

[114] Suh N.P., "Axiomatic Design Theory for Systems", working paper, MIT, Cambridge, MA, 1998a.

[115] Suh N.P., personal communication, 1998b.

[116] Suh N.P., *Axiomatic Design: Advances and Applications*, to be published: Oxford University Press, 1999.

[117] Suh N.P., Bell A.C., Gossard D.C., "On an Axiomatic Approach to Manufacturing and Manufacturing Systems", *Journal of Engineering for Industry*, Vol. 100, No. 2, pp. 127-130, 1978.

[118] Suh N.P., Do S.-H., "Software Development for the Axiomatic Design", working paper, MIT, Cambridge, MA, Apr,, 1998.

[119] Sushkov V.V., Mars N.J.I., Wognum P.M., "Introduction to TIPS: A Theory for Creative Design", *AI in Engineering*, Vol. 9, 1995.

[120] Tate D., Lindholm D., Harutunian V., "Dependencies in Axiomatic Design", Third World Conference on Integrated Design and Process Technology, Society for Design and Process Science (SDPS), Berlin, Germany, July 6-9, 1998.

[121] Tate D., Nordlund M., "Synergies Between American and European Approaches to Design", First World Conference on Integrated Design and Process Technology, Society for Design and Process Science (SDPS), *IDPT-Vol. 1*, Austin, TX, pp. 103-111, Dec. 7-9, 1995.

[122] Tate D., Nordlund M., "A Design Process Roadmap as a General Tool for Structuring and Supporting Design Activities", Second World Conference on Integrated Design and Process Technology, Society for Design and Process Science (SDPS), *IDPT-Vol. 3*, Austin, TX, pp. 97-104, Dec. 1-4, 1996.

[123] Tate D., Nordlund M., "A Design Process Roadmap as a General Tool for Structuring and Supporting Design Activities", *SDPS Journal of Integrated Design and Process Science*, Vol. 2, No. 3, pp. 11-19, 1998.

[124] Tomiyama T., "Engineering Design Research in Japan", ASME, *DE-Vol. 27 (DTM 90)*, Chicago, IL, pp. 219-223, Sept. 16-19, 1990.

[125] Ullman D.G., "The Status of Design Theory Research in the United States", *ICED '91*, Zurich, Aug. 27-29, 1991.

[126] Ullman D.G., *The Mechanical Design Process*, New York: Mc-Graw-Hill, 1992. ISBN 0-07-065739-4

[127] Ulrich K., "The Role of Product Architecture in the Manufacturing Firm", *Research Policy*, Vol. 24, pp. 419-440, 1995.

[128] Ulrich K., Tung K., "Fundamentals of Product Modularity", Issues in Design Manufacture/Integration, ASME, *DE-Vol. 39*, Atlanta, GA, pp. 73-80, Dec. 1-6, 1991. ISBN 0-7918-0876-9

[129] Wallace K.M., Hales C., "Engineering Design Research Areas", Engineering Design, Institution of Mechanical Engineers, *1*, Harrogate, pp. 555-562, Aug. 22-25, 1989.

[130] —, "Webster's Ninth New Collegiate Dictionary", Springfield, MA: Merriam-Webster, 1988.

[131] Weinberg S., "The Revolution that Didn't Happen", *The New York Review of Books*, Oct. 8, 1998. www.nybooks.com/nyrev/

[132] Willem R.A., "Design-Science Interactions", ASME, *DE-Vol. 27 (DTM 90)*, Chicago, IL, pp. 323-325, Sept. 16-19, 1990.

[133] Wilson D.R., "An Exploratory Study of Complexity in Axiomatic Design", *Ph.D. Thesis*, Department of Mechanical Engineering, MIT, Cambridge, MA, 1980.

[134] Yin R.K., *Case Study Research: Design and Methods*, Second edition, Thousand Oaks, CA: Sage Publications, 1994. ISBN 0-8039-5663-0

# APPENDIX 1: RESEARCH METHODS FOR DESIGN THEORY[1]

This chapter is about the development of design theory, about how research in design theory is—and should be—done. Important works in design theory since 1850 are identified. The current state of design as a pre-paradigm science is explained, in section A1.2, motivating the need for a unifying view of design. Given the state of the field, a framework for design theory is presented in section A1.3, and in section A1.4, a research process model is proposed as a scientific way of doing research in design theory. These are discussed in light of modern views in the philosophy of science and their criteria for theory evaluation and scientific progress. Research in design can be treated in a scientific manner, but this must be done from a sophisticated view of scientific practice, one with the proper criteria for evaluating progress: at the level of the research program, not the individual theory as explained in section A1.5. Lastly, the use of axiomatic design as a unifying program for practicing design research is presented in section A1.6.

## *A1.1 Introduction*

Much research in design is being pursued currently at universities around the world. While several studies that describe the subjects of research in design have been published, very few publications provide guidance about how research can be done. It is difficult to identify established research methods that have been accepted by most researchers in this field, and in fact, when they are published, research results in design are rarely accompanied by an analysis of the research method that was followed or a discussion of the validity of the method with regard to the results of the work.

In order to provide a stringent research method that can be applied in this thesis, this chapter proposes an answer to the question: what is a research framework that, if followed, will lead to scientifically valid results? From this framework, a research process is derived to apply to research in design theory. The main contribution of this chapter is a proposed research process model to achieve scientifically acceptable research. The intent here is not to make a major contribution to the philosophy of science but rather to transfer some of the thoughts from this field to the field of design, to complement these with some published ideas available in the field of design theory, and to form some relevant conclusions for the present work.

Researchers have looked at parallels between performing scientific research and following the engineering design process [Eekels and Roozenburg (1991)] or at the application of general knowledge acquired through science in doing design [Willem (1990)]. Neither of these is done here. Instead, the purpose in this chapter is distinct: It looks at *the application of scientific methods to generate knowledge about design. Science* is seen as the application of accepted "scientific" methods to generate knowledge. The knowledge which is produced is known as a *theory.*

[1] Parts of this chapter are adapted from [Nordlund and Tate (1996)] and have appeared in [Nordlund (1996)]. The differences between this chapter and those sources lies primarily in changes to "Theory validation" (section A1.5.4) and the addition of "Discussion on axiomatic design" (section A1.6).

## *A1.2 Design theory: the need for a unifying paradigm*

This section describes the ways in which research methods for design theory are an intellectually interesting topic for discussion. First, the few papers which have addressed the subject are discussed in section A1.2.1. Then the need for a unifying view (known as a *paradigm* or *research program*) is detailed in section A1.2.2.

In this section, the need for accepted research methods within the field of design theory is addressed. The main review papers for research in design in the US are the two parts by Finger and Dixon from 1989 [Finger and Dixon (1989)].[2] These papers cover a large amount of work either done in the US or translated into English. They are also significant because other researchers have continually referenced them and have used the classification of design research presented within them. The history of the field, however, may be traced back to research done in Germany in the 1850s.

Table A1-1—which is derived from material presented by Altshuller, Bjärnemo, Pahl & Beitz, Phadke, and Suh [Altshuller (1988), Bjärnemo (1983), Pahl and Beitz (1988), Phadke (1989), Suh (1990)]—lists some of the most influential work in the field of design theory.[3]

Very little European research builds on theory developed in the US and vice versa. One explanation for this is the language barrier. Finger and Dixon write that even though much work has been published in German, only a small fraction has been translated into English [Finger and Dixon (1989)], and going in the reverse direction, even less material has been translated from English into German. Research tradition, peer pressure, and pre-existing intellectual networks[4] have been additional important factors—beyond the language barrier—in isolating the different research communities.

### A1.2.1 Discussions of research methods

Only a handful of researchers have seen the need to discuss research methods for design theory. Therefore, very few papers dealing specifically with research methods in design theory are available.

---

[2] Reviews of literature in design theory are plentiful, and the field has several examples of a particular type of review in which research results are grouped according to some classification scheme—see for example, [Cross (1993), Dixon (1987), Tate and Nordlund (1995)]—or according to geographic origin—[Arciszewski (1990), Eder (1990b), Hundal (1990), Tomiyama (1990)]. None of the classifications found, however, were according to research methods.

[3] Other sources with many references for the field include the following: [Dwarakanath, et al. (1996), Evbuomwan, et al. (1996), Hongo (1985), Hundal (1990), Karandikar and Shupe (1995), Steinberg (1994)].

[4] [Collins (1998)] presents an interesting theory for the transmission of ideas ("cultural capital") across generations. Some of the main components of this theory are the existence of few rivalries within an intellectual field at any time, the importance of face-to-face contacts for identifying the current fronts of intellectual investigation and debate, and the uneven production of papers across the research community.

**Table A1-1. Some important work in design theory**

| Researcher | Publication | Year |
|---|---|---|
| F. Redtenbacher | Prinzipen der Mechanik und des Mashinenbaus | 1852 |
| F. Reuleaux and C. Moll | Konstruktionslehre für den Maschinenbau | 1854 |
| F. Reuleaux | Teoretische Kinematik: Gründzüge einer Theorie des Maschinenwesens | 1875 |
| C. Bach | Die Maschinenelemente | 1881 |
| A. Riedler | Das Machinenzeichnen | 1913 |
| A. Erkens | Beiträge zu Knostruktionserziehung | 1928 |
| F. Kesselring | Die starke Konstruktion | 1942 |
| H. Wögerbauer | Die Technik des Konstruierens | 1943 |
| F. Zwicky | The Morphological Method of Analysis and Construction | 1948 |
| G. Niemann | Maschinelemente | 1950 |
| F. Kesselring | Bewertung von Konstruktionen | 1951 |
| G.S. Altshuller | On the Psychology of Inventive Creativity | 1956 |
| R. Matousek | Konstruktionslehre des Allgemeinen Maschinenbaus | 1957 |
| M. Asimow | Introduction to Design | 1962 |
| A. Leyer | Maschinenkonstruktionslehre | 1963-71 |
| H.A. Simon | The Science of Design | 1969 |
| J.C. Jones | Design Methods | 1970 |
| W. Rodenacker | Methodishes Konstruieren | 1970 |
| G.S. Altshuller | Algorithm of Invention | 1973 |
| R. Koller | Eine Algoritmisch-physikalisch Orienterte Konstruktionsmethodik | 1973 |
| V. Hubka | Theorie der Maschinensysteme | 1973 |
| VDI 2222 Blatt 1 | Konzipieren Technisher Produkte | 1973 |
| F. Hansen | Konstruktionswissenschaft - Grundlagen und Methoden | 1974 |
| K. Roth | Aufbau und Handhabung von Konstruktionskatalogen | 1974 |
| V. Hubka | Theorie der Konstruktionsprozesse | 1976 |
| F. Olsson | Systematisk Konstruktion | 1976 |
| G. Taguchi | Jikken Keikakuho (Eng. trans. *System of Experimental Design*) | 1977-78 |
| G. Pahl and W. Beitz | Konstruktionslehre | 1977 |
| N.P. Suh, A.C. Bell and D.C. Gossard | On an Axiomatic Approach to Manufacturing and Manufacturing Systems | 1978 |
| G. Nadler | The Planning and Design Approach | 1981 |
| G. Boothroyd and P. Dewhurst | Design for Assembly - A Designer's Handbook | 1983 |
| S. Pugh | Further development of the Hypothesis of Static/Dynamic Concepts in Product Design | 1985 |
| D. G. Ullman | Mechanical Design Methodology | 1986 |
| J. Hauser and D. Clausing | The House of Quality | 1988 |

Antonsson and Dixon [Antonsson (1987), Dixon (1987)] propose ways that scientific methods can be applied to design research in order to stimulate discussion about the topic—discussion which has not materialized. Dixon discusses general scientific methods such as generation and validation of theories. He follows this with a description of three types of design theories: prescriptive, descriptive cognitive, and computational.

Antonsson discusses the validity of hypotheses and means by which to test them. A hypothesis the statement "a device to perform a certain function can be designed and fabricated" is not valid because it is not a research question; instead, the advancement of design theory requires "investigation into aids for the [design] process". [Antonsson (1987) p. 153] The example Antonsson gives, however, is inconsistent with his stated criteria: "Engineering design research *is* more than the creation of new designs. Designing a new toaster probably by itself is not EDR [engineering design research]." [Antonsson (1987) p. 154] The example concerning the design, fabrication, and testing of a robotic hand is not research into design so much as research into grasping and manipulation.

Reich [Reich (1995)] proposes the study of research methods, but he does not propose any method himself. "Since some [research methods] are better than others for different purposes, it becomes valuable to study different methodologies and their influence on research practice and results." [Reich (1995) p. 211] Although Reich realizes that there are multiple acceptable methods for performing design research, he does not recognize and consider differences in goals among design researchers. He seems to assume that "impacted design practice" is a goal of all design research. To say that it is the *only* valid goal at the level of an individual research project, however, is overly restrictive. An idiographic study (see section A1.5.1) describing the development of a particular design object,[5] say the Boeing 777, would not be considered design research according to this view.

A paper by Cross focuses on research methods to develop theories to "understand just how it is that people do design". [Cross (1992) p. 3] and gives several research methods that deal with the collection of knowledge during an actual or simulated design project. Eder describes the work of researchers following the WDK research tradition [Eder (1990a)] (see also Wallace and Hales. [Wallace and Hales (1989))] Also, several useful papers can be found dealing with research methods in general. [Reich (1994), Steinberg (1994)]

As the lack of literature dealing with research methods in design, their use, and their validity indicates, when investigating *how* scientifically valid research in design theory should be done, very few guidelines can be found in the published literature.

## A1.2.2 Design as pre-paradigm science

This section answers the question: what is the status of research in design theory? Several researchers have commented on the status of the field and noted that it is still in an early stage of development. For example, in Ullman's opinion, "there is not yet a specific theory to describe. It is still evolving...Design theory research is in the pre-theory stage. There is still a search for the basic vocabulary and building blocks of a theory." [Ullman (1991) p. 794, 800] According to Dixon, "There is much yet to be learned and formalized before we

[5] [Dwarakanath, et al. (1996)] gives an extensive list of descriptive studies.

may say that a scientific theory and foundation of principles exists for engineering design." Thus he says, the field is "in a pre-theory stage". [Dixon (1987) p. 147]

Discussion at a workshop in design theory [(1992) p. 213] concluded that the field of design theory "lacks

1. a clear agreement about the goals to pursue
2. a shared research methodology
3. a broad theoretical framework to relate the findings of isolated pieces of research to one another"

In this chapter, the approach of Kuhn is used to describe elements required for the scientific endeavor and to point out which are still needed within design research. [Kuhn (1970), Kuhn (1977)]

#### A1.2.2.1 The goal of design research

The goal of design theory can be stated in many ways. One encompassing description is that by Cross: "the study of how designers work and think, the establishment of appropriate structures for the design process, the development and application of new design methods, techniques and procedures, and reflection on the nature and extent of design knowledge and its application to design problems". [[Cross (1984)] quoted in [Cross (1993)] p. 66] Individual researchers are solving subproblems within this field, but their work contributes to the overall goal of understanding how design is, can be, and should be done.

#### A1.2.2.2 The design research community

There does exist a community of researchers in design, yet the community does not share a common framework for viewing design research. Although the goals of individual researchers fit within the overall goal of the community, the methods used vary greatly depending on background and colleagues. In Ullman's words, the design research community "is still discipline and viewpoint fragmented". [Ullman (1991) p. 800] That is, the community consists of small groups which each have their own framework for viewing—identifying, communicating, solving—research issues. (See [Tate and Nordlund (1995)] or appendix 3 for a discussion of four such groups.)

The lack of a single paradigm, however, does not preclude the existence of a community. As Dixon says, "researchers in engineering design theory...constitute a single *goal-directed research community*". [Dixon (1987) p. 146] The claim that a common, accepted paradigm is not a prerequisite for a research community is justified because, according to Kuhn, "Scientific communities can and should be isolated without prior recourse to paradigms; the latter can then be discovered by scrutinizing the behavior of a given community's members." [Kuhn (1970) p. 176]

Moreover, it is the lack of an accepted set of research methods that is the crux of the issue. It is not a disagreement about the overall goal of design theory or a lack of communication channels. The goal of design theory can be stated in such a way that it includes the work of most researchers as done above, but differences in the means by which design theorists seek to reach that goal have led to the fragmented nature of design theory research.

### A1.2.2.3 A paradigm for design theory

Paradigms for research consist of "accepted examples of actual scientific practice—examples which include law, theory, application, and instrumentation together—[that] provide models from which spring coherent traditions of scientific research." [Kuhn (1970) p. 10] A *paradigm* is a unifying view of a discipline ("the entire constellation of beliefs, values, techniques, and so on shared by the members of a given [research] community" [Kuhn (1970) p. 175]) that is brought about *exemplars* ("the concrete puzzle-solutions which, employed as models or examples, can replace explicit rules as a basis for the solution of the remaining puzzles of normal science" [Kuhn (1970) p. 175]). Thus, for example, Newton's *Principia* is a treatise which served as a unifying vision for the paradigm of Newtonian mechanics.

A *research program* can be defined as "a sequence of theories representing the development of a central idea." [Larvor (1998) p. 54] This is the term used by Lakatos, and I argue that this is fundamentally the same concept as Kuhn's *paradigm* or Laudan's *research tradition*—although none of the three makes the same argument himself.

According to Lakatos, a research program consists of a "hard core" and a "heuristic". The *hard core* is the governing idea for the research program which remains consistent throughout the life of the program and which cannot be abandoned without abandoning the research program altogether. The *heuristic* is the research programs' collection of problem-solving techniques. Similarly, for Laudan (who does not admit an unchanging hard core), a *research tradition* consists of "(1) a set of beliefs about what sorts of entities and processes make up the domain of inquiry; and (2) a set of epistemic and methodological norms about how the domain is to be investigated, how theories are to be tested, how data are to be collected, and the like." [Laudan (1996) p. 83]

Therefore from the above, a paradigm or a research program consists of

- ontology: an identification of the fundamental concepts or entities which make up the field of study
- aims: an articulation of the scope of the field in terms of both problems which have been solved (*exemplars*) and problems remaining to be solved (*anomalies*) which should be covered by the program, and are expected to be, but have not been yet
- methodology: guidelines for further developing the program—particularly in a manner consistent with the problem-solving approach that the program has been following
- theories: relationships between the fundamental concepts of the field and application to the specific problems

Kuhn describes the transition from the pre- to the post-paradigm period in the development of a scientific field. "Before [the transition] occurs, a number of schools compete for the domination of a given field. Afterward...the number of schools is greatly reduced, ordinarily to one, and a more efficient mode of scientific practice begins....[T]he transition need not...be associated with the first acquisition of a paradigm....What changes with the transition to maturity is not the presence of a paradigm but rather [the presence of a paradigm of a specific] nature....Many of the attributes of a developed science [are] consequences of the sort of paradigm that identifies challenging puzzles, supplies clues to their solution, and guarantees that the truly clever practitioner will succeed." [Kuhn (1970) p. 179]

Lack of a common paradigm within design leads to miscommunication. Design theorists hold multiple worldviews and see "different things when they look from the same point in the same direction....[T]hey see different things, and they see them in different relations one to the other. That is why a law that cannot even be demonstrated to one group of scientists may occasionally seem intuitively obvious to another". [Kuhn (1970) p. 150]

Differences between these competing paradigms can be seen by examining the literature within design theory. For example, a consistent set of terminology for the field does not exist. When one group of theorists describes the design process, this is only a fraction of the activities considered design by another group. Also, the ways in which design objects are modeled are very different as are the ways in which the models are used. (See [Tate and Nordlund (1995)] or appendix 3 for further discussion.)

The conclusion of this section is that there does exist a research community in design theory. The field of design theory, however, is in a pre-paradigm stage. There is no single accepted framework (set of examples, and set of questions and solution approaches) which define the work in this field. Rather, there exist several competing schools for design theory research. Therefore, design theory is still in a pre-paradigm stage. The remainder of this chapter articulates a view on how to move beyond this point. It describes what is needed in a paradigm for design theory, proposes a framework for viewing research in design theory, and describes a research process which contributes to theory development.

### *A1.3 Areas of fundamental knowledge for design theory*

Knowledge in design can be abstracted into fundamental areas. When knowledge is related between or within these fundamental areas, a theory of design is generated. The areas of fundamental knowledge which are covered within design theory can be abstracted as shown in Figure A1-2:

- the design process
- the design object (the product of the design process)
- designers
- specific field knowledge
- resources (such as time, money)

This abstraction agrees with fundamental areas put forth by other researchers. According to Dixon, the areas of fundamental knowledge which are required for "a descriptive cognitive theory of design" are the designer(s), the problem, the organizational environment, the design environment (including information resources and computer-based and other tools), and time. [Dixon (1987) p. 152] In the TIPS (theory of inventive problem solving) school—researchers following the work of Altshuller—the fundamental areas modeled are a process of design and products of this process [Sushkov, et al. (1995)]. In praxiologic design science, from Poland, the elements considered are designers, the design object, and the design process. [Arciszewski (1990) p. 211]

Specific design theories are then concerned with one or more of these fundamental areas. For example, axiomatic design (AD) relates the design object to decisions in the design process [Suh (1990)]. The theory of technical evolution within TIPS is an abstraction of the evolution of engineering systems, from many fields; as such, it is concerned with only design objects. Similarly other design theories provide insight about one or more of the above fundamental areas.

## *A1.4 A research process model and framework for design theory*

### A1.4.1 Research process model

In this section a model of the research process is presented, as shown in Figure A1-1. The purpose of this model is to make explicit one scientifically acceptable research process that can be applied to research in design theory.[6]

The input of the process is a research question and the output is a theory that can be tested or used for explanation and prediction. The main phases of the research process are data gathering, theory development, and theory validation. Each phase comprises one or more activities that are performed to generate knowledge. In some research projects, an individual

[6] The models in this section are presented using the SADT/IDEF0 notation. It is assumed that the reader is familiar with SADT/IDEF0 notation. For further reading on SADT/IDEF0, see for example [Marca and McGowan (1993), Ross and Schoman (1977)].

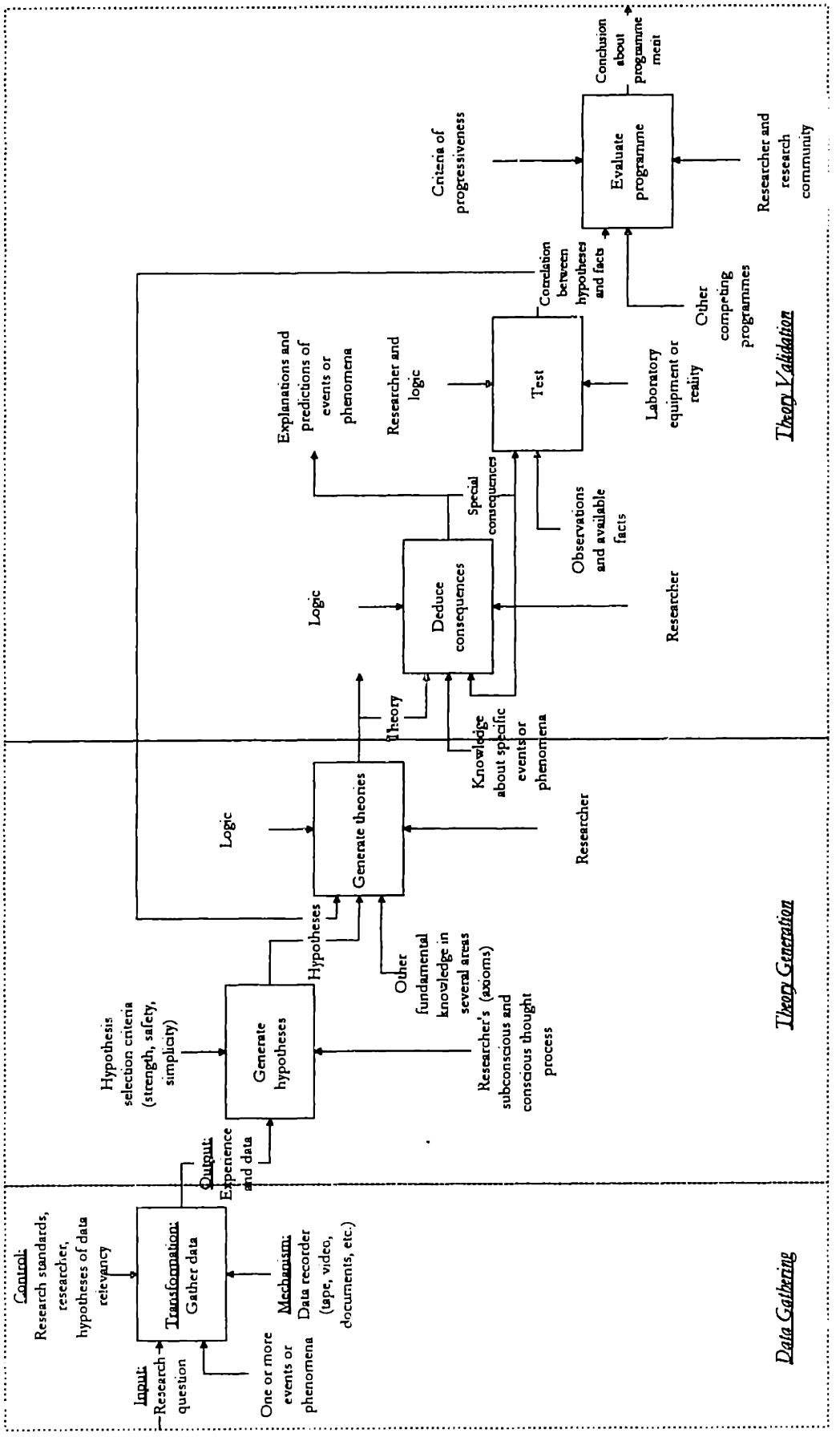


*Figure A1-1. Model of the research process*

researcher works through all phases. In other projects, the work is initiated by one researcher, and the results of this phase are passed to a different researcher who then continues the research by working in a subsequent phase.

Each phase and its activities are described in more detail in the following sections.

## A1.4.2 Framework for design theory

Figure A1-2 shows that scientific theories comprise *fundamental knowledge areas* in the form of perceptions and understandings of different entities, and the *relationships* between these fundamental areas. These perceptions and relationships are combined by the theorist to produce *special consequences* that are predictions or explanations of observations.

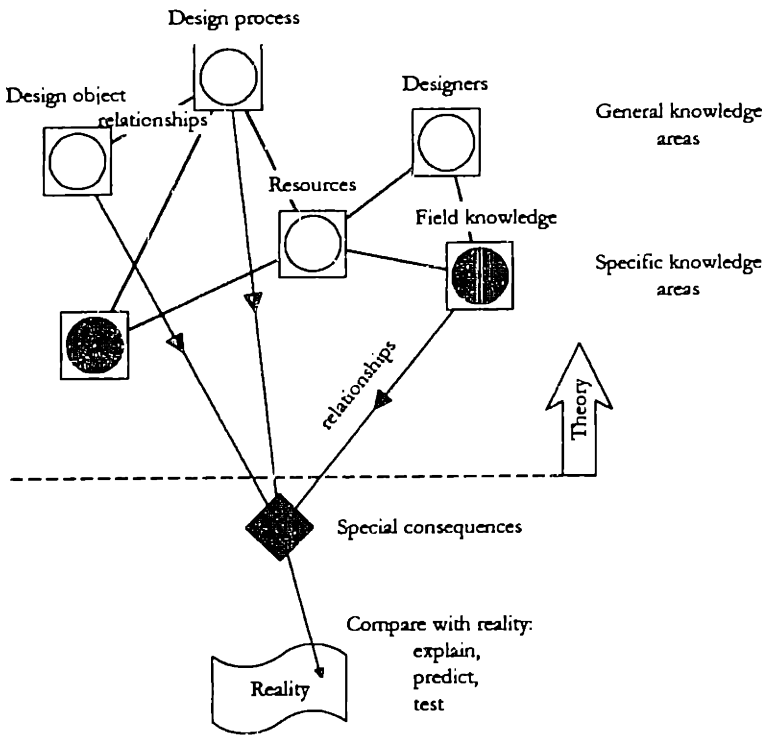


*Figure A1-2. Elements of design theory*

The fundamental knowledge areas are at a more abstract level than observations of real-world data. Such fundamental knowledge may take the form of mathematical expressions (mechanics), categorizations of phenomena or objects (biology), and other models. Such fundamental knowledge, together with the relationships between them, constitute a *theory*: "a network of statements which, in conjunction with initial conditions, lead to explanations and predictions of specific phenomena". [Laudan (1996) p. 83] A *theory* may be one of two types, depending on the way in which the fundamental knowledge areas are treated. They may be treated as either

- hypothetical
- axiomatic

If they are treated as *axioms* (or *laws*) then their validity is not questioned. The distinction is not whether they are generally accepted as true, but rather the use to which they are put. If

they are not being tested, they are being treated as axioms. The concern, in this case, is with the consequences that follow. If the fundamental knowledge areas are being tested, they are not being treated as axioms. Instead, when questioned, the fundamental knowledge areas are being treated as *hypotheses*. In this case, the consequences which result are used to corroborate or to refute the hypotheses.

So, the distinction between these two theoretical systems

- axioms (or laws), relationships, and consequences
- hypotheses, relationships, and consequences

is named by referring to the first case as an *axiomatic theory* and the second as a *hypothetico-deductive theory*. An axiomatic theory is concerned with derived knowledge; a hypothetico-deductive theory is concerned with testing the validity of hypotheses.

In either case, the fundamental knowledge areas are combined with information about particular real-world situations to derive logical consequences. The consequences themselves are not the hypotheses. The consequences are at a more detailed, less abstract, level than the hypotheses and/or axioms from which they were derived. These consequences can then be used to fulfill three purposes:

- to test
- to predict
- to explain

Hypotheses lead to consequences which are used either to test or to make predictions (which could be a test). Axioms lead to consequences which are used to explain or to make predictions (which are subsequently not tested). The use of these logically derived consequences to test or to predict is termed *applied research*. The use of the consequences to explain observed events or facts is *pure (or basic) research*. [Föllesdal, et al. (1993) p. 185]

An important consideration in understanding the expansion of knowledge is the interest of the researcher. Is the researcher more concerned with fundamental knowledge or with the application of such knowledge to a specific situation? Fundamental knowledge which results from different perceptions of phenomena or situations may be used to derive both consequences which predict and consequences which explain. In the first case of consequences which predict, the researcher's concern is with expanding the quantity or adding to the validity of the fundamental knowledge. This research adds to knowledge about the hypotheses; more is known about whether the fundamental knowledge (hypotheses, here) is valid, or not. In the second case, when consequences are used to explain, the researcher's concern is with the particular case in which observations are made and to which the theory is applied. Thus, more is known about the particular case to which the fundamentals (axioms, here) are applied. The researcher adds to an understanding of that special case.

In design theory, the role of a design researcher is to develop new design theories and to verify these. In the following sections, a process for such activities is presented. It is not intended that the algorithm below be interpreted as the only valid means to perform design research, yet it should be a sufficient one.

The above approach may be interpreted as overlapping too much with the positivist approach to science. Therefore a discussion of philosophy of science is needed. In particular

the approach described here is distinguished from the positivist approach by the means for theory evaluation (see section A1.5.4.2).

## A1.4.3 Approaches to philosophy of science

Philosophy of science answers questions about ontology, epistemology, and methodology of science. That is, what are the important concepts which make up science? How is knowledge generated by science? What is good scientific practice?

Important developments in this field over the past fifty years are discussed here; this provides background for understanding the development of design as a scientific discipline.

The following views of science are discussed here:

- Logical positivism/logical empiricism
- Falsificationism (Popper)
- Scientific revolutions (Kuhn)
- Methodology of scientific research programs (Lakatos)
- Normative naturalism (Laudan)

The positivist view of science was well-accepted in the early part of the twentieth century. Its emphases were on separating theory generation from theory validation and for developing means to validate theories. The characteristics of positivism are dependence on empirical observation, defined vocabulary, explanations of phenomena deduced from laws and initial conditions, and a belief in theory reduction (that the laws of specialized disciplines can be explained by deduction from more basic laws, or axioms). Scientific inquiry in the positivist view is seen to comprise four steps: recording of observed facts, analysis and classification of facts, generalization of the facts by induction to form laws (or axioms), and further testing of the laws. This view of science focused on using rules of induction to prove general scientific laws. To account for the fact that a series of observation could not, strictly speaking, *prove* the theories, statistical means were proposed for assigning probabilities to the likelihood of different theories' validities. The reader is referred to works such as [Hempel (1966)] for a description of the positivist position.

Problems with the positivist view were recognized by Popper who attempted to solve them by describing science as a process which grows through critical experiments and falsification. In this view, scientists spend their time devising critical experiments and respond rationally by rejecting theories as soon as they produce false results. The key intellectual quest for supporters of this view is to devise means for demarcating science and non-science. As applied, this is equivalent to demarcating conditions under which a theory could be said to be falsified and articulating experiments to test this.

The problem with the falsification view of science, of course, is that it is unsupported by the history of science itself. It does not accurately describe the day-to-day activities of scientists, nor does it describe the means by which new theories become accepted and replace old theories. These faults were identified and rectified by Kuhn who proposed that science advances by means of successive scientific "revolutions". According to Kuhn, scientists adopt a particular paradigm of their field which serves for establishing the important concepts, for describing their relationships, for teaching, for identifying important problems, and for directing research activities. [Kuhn (1970), Kuhn (1977)]

To accurately account for the historical development of science (as done by Kuhn) while maintaining a desire for demarcating good science from bad (as desired by Popper), Lakatos developed a theory of scientific research programs. This theory is notable for its sophisticated view of theory validation and choice of a research program. [Lakatos (1978), Larvor (1998)]

Further clarifications of scientific practice have been articulated by Laudan. [Laudan (1996)]

### *A1.5 Research activities*

This section discusses research activities which contribute to the growth of knowledge within a scientific field. The useful output of theories are special consequences which serve to explain or predict different phenomena. These consequences and their usefulness then serve as a metric for measuring and evaluating the successes of theories and the research programs of which they are a part. The use of theories and derivations to explain and predict events is generally agreed upon by philosophers of science; the means by which such consequences support theories or lead to scientific growth is, however, a subject of intense debate and sharply diverging views.

## A1.5.1 Data gathering

Data gathering is normally the first phase of a research project. It begins after a research question has been proposed, and its outputs are data about one or several events or phenomena.

The researcher can either gather data to provide a detailed description of one event or phenomenon (this is called *idiographic research*), or the researcher can gather data from multiple different events in order to search for patterns in this data (this is called *nomothetic research*). In figure a1-1, it is observed that the researcher acts as the control for these data gathering activities; therefore the data that is gathered will be biased by the researcher's implicit or explicit hypotheses about which data are relevant and which are not.

#### A1.5.1.1 Empirical studies and case studies

Case studies are found in literature of design theory, yet their role in the research activity and their scientific validity are often not discussed. *Case studies* are empirical inquiries that investigate contemporary phenomena with multiple sources of evidence where the boundary between phenomena and context is unclear. They can be either practical or scholarly, or preferably, both. Their purpose can be description, problem solving, criticism/interpretation, or theory building. [Yin (1994) p 4]

Because the data gathering activities lay the foundation for the research work, it is important to ensure that the results of this activity are scientifically rigorous. Scientifically rigorous case-study research should produce results that are [Bailey (1992) p 51]

- Generalizable: Good case studies identify those features that are uniform and generalizable across organizations or events. At the same time, case studies will also identify those features that appear to be relatively unique.
- Transferable: The findings or research solutions can be applied (some times with minor modifications) in other similar organizations,

- Replicable: It should be possible to reproduce the method of experimentation and analysis. Given the same conditions and using the same methods, the same or similar results should be obtained in other studies or with other organizations. However, exact replication in design is most likely impossible since conditions can never be the same twice. Changes in context; time; market and economic conditions; or the possibility of Hawthorne effects (see [Gillespie (1991)]) make research projects involving human beings singular events.

To facilitate replication, Yin [Yin (1994) p 70] prescribes the use of a case study protocol. Such protocols contain the case study method as well as the procedures and general rules that should be followed. A protocol should include the following sections [quoted from [Bailey (1992) p 51]:

- "Overview of the case-study project (project objectives and auspices, case-study issues and relevant readings about the topic being investigated);
- field procedures (credentials and access to the case-study sites, general sources of information and procedural reminders);
- case-study questions (the specific questions that the case-study investigator must keep in mind in collecting data, table shells for specific arrays of data and the potential sources of information for answering each question); and
- guide for the case-study report (outline, format for the narrative, and specification of any bibliographical information and other documentation)."

The greatest pitfall in case study research is researcher bias. Researchers must avoid having preconceived notions to prevent bias and to allow the discovery process to occur. In order to avoid bias, it is recommended that researchers continually discuss their research design and interpretations with colleagues. The need to avoid such bias thus precludes "a study conducted retrospectively or concurrently on an organization by one of its employees" from being acceptable research. [Bailey (1992) p. 52]

## A1.5.2 Theory development

Theory development is made up by two different activities: hypothesis generation and theory generation. A *theory* is defined as a set of fundamental concepts (axioms or laws) and the relationships between them. *Hypotheses* are those fundamental concepts and relationships which are being tested. Hypothesis generation is discussed in section A1.5.2.1, and theory generation is discussed in section A1.5.2.2.

### A1.5.2.1 Generating hypotheses

The hypothesis generation activity uses the data from the data gathering phase as its input, and delivers hypotheses as its output. Hypotheses are always proposed as more or less brilliant guesses.

When generating hypotheses, there can be a large number of hypotheses that can be used to explain a finite number of observations. From all these potential hypotheses, a small number should be selected for further investigation.

The criteria for a good hypotheses are quantity and variety of corroborating evidence (evidence the hypothesis was constructed to explain), strength of new predictions (those the hypothesis was not constructed to explain), deductive support from other well-supported hypotheses or theories, and simplicity (relative to other competing hypotheses). [Hempel (1966) Ch. 4 pp. 33-46]

Hypotheses are stated as general laws for deductive-nomological explanation and prediction. Specifically they are "statements of universal form"; that is, they assert a uniform connection between different phenomena. They take the form of a statement that when certain conditions are met, a certain outcome occurs, for example: "In all cases when conditions of kind F are realized, conditions of kind G are realized as well". Also a law can support both subjunctive and counterfactual conditionals (things that might occur, but haven't and things that might have happened, but didn't). [Hempel (1966) pp. 54-57]

### A1.5.2.2 Generating theories

The characteristics of a theory are that it clarifies the concepts which make up the theory, and more importantly, it clarifies the relationships between the concepts. [Föllesdal, et al. (1993) p 75] According to Popper, a theory "should proceed from some *simple, new, and powerful, unifying idea* about some connection or relation (such as gravitational attraction) between hitherto unconnected things (such as planets and apples) or facts (such as inertial and gravitational mass) or new 'theoretical entities' (such as fields and particles)" [Popper in [Blum (1996) p. 241]

A theory is created by grouping different concepts and then using logic to deduce principles. Thus, the inputs to this activity are fundamental knowledge in several different areas, and hypotheses. [Föllesdal, et al. (1993) p 76] This is shown in figure a1-1, where it can be seen that during theory generation the researcher acts as the mechanism for generating a theory while being guided by logic.

#### *A1.5.2.2.1 Normative statements as universal laws*

Normative statements as universal laws may strike some people as inappropriate. As argued by Laudan, such discomfort is unwarranted. Normative statements may easily take the form of universal laws when the cognitive ends of the statement are made explicit.

Laudan argues that justifying such statements is no more difficult than justifying any other scientific theory. Such prescriptive statements are on the same epistemic basis as other scientific laws; that is, they can be justified using the same scientific methods as other more obviously descriptive laws. However, Laudan does stress that such statements should be linked to a desired cognitive end. That is, what is it that the users are trying to achieve by following such a rule?

The specific example that Laudan discusses is that of creating scientific methodology. (See [Laudan (1996) Ch. 7, pp. 125-141].) These are prescriptive statements that researchers should follow in choosing hypotheses and theories. The statement of a rule, could be that to find a theory with a broader scope, researchers should not adopt ad hoc hypotheses. Then this type of rule can be tested by examining the historical record and seeing if this is actually the case. Did following this rule lead to the desired cognitive ends more often than following competing such rules?

The form that Laudan proposes is this:

> If actions of a particular sort, *m*, have consistently promoted certain cognitive ends, *e*, in the past, and rival actions, *n*, have failed to do so, then assume that future actions following the rule "if your aim is *e*, you ought to do *m*" are more likely to promote these ends than actions based on the rule "if your aim is *e* you ought to do *n*." [Laudan (1996) p. 135]

and this is evaluated according to the following:

> Given any proposed methodological rule (couched in appropriate conditional declarative form), do we have–or can we find–evidence that the means proposed in the rule promotes its associated cognitive end better than its extant rivals? [Laudan (1996) p. 135]

## A1.5.3 Use of consequences derived from theory

A theory, once generated, can be used to deduce special consequences that are used for one or more of three purposes: to support or refute the underlying hypotheses during the theory validation phase or to explain or predict events or phenomena. These uses are explained in this section.

### A1.5.3.1 Deducing consequences

Based on the theory and knowledge researchers have about a specific event or phenomena (used as inputs), they can deduce special consequences based on the theory that apply to the event or phenomena under consideration. These consequences are used to [Föllesdal, et al. (1993) p 59]

- explain something that has been observed (section A1.5.3.2)
- predict something that has not yet been observed (section A1.5.3.3)
- test the theory and its hypotheses (section A1.5.3.4 and A1.5.4)

The special consequences also serve as feedback for the theory generation activity and guide modification to the theory, if needed.

In the situation where the principles which underlie the deduction of consequences are assumed to be certain, this type of deductive system is referred to as a *axiomatic system.* In this case, the interest of the researchers is not in verifying the truth of the underlying principles (thus referred to as *axioms*); rather the researchers are concerned with the implications of the axioms with regard to the specific event or phenomenon. [Föllesdal, et al. (1993) p 77] If, on the other hand, the researchers are deducing consequences in order to test the validity of the underlying hypotheses, this approach to theory validation is known as the *hypothetico-deductive method.*

Researchers use laws (or hypotheses) for explanation and prediction by specifically deducing an explanation of an event E from a set of general laws ($L_1$, $L_2$, ...$L_n$) and initial conditions ($C_1$, $C_2$, ... $C_m$):

$L_1, L_2, \ldots L_n$
$C_1, C_2, \ldots C_m$
Therefore E

The only difference in this explanation between explanation and prediction is the time at which the event occurs.

### A1.5.3.2 Explaining

The *explanation* answers the question "why?" The phenomenon to be explained (explanandum) can be an *event*, for example, why did the Chernobyl nuclear accident occur? Alternatively a more *permanent state of affairs* can be explained, for example, why is the Boeing 777 wing shaped the way that it is?

There are three categories of explanations commonly used in science:

- Causal explanations are used in physics, where, for example, they contain references to various forces that are causing some phenomenon. The ways in which these forces act can, in turn, be explained by atomic physics. A causal explanation of an event X, consists of identifying an earlier event Y and a set of initial conditions Z, such that there is a causal law saying that under the conditions Z, Y will always be followed by X. [Föllesdal, et al. (1993) p. 199]

- The functionalistic explanation is the opposite of the causal explanation. A phenomenon is explained by what it causes rather than what has caused it. Biology is characterized by explanations of this kind: An organism's behavior and structure are explained by the organism's function, for example, favorable effects on the organism's ability to reproduce. To provide a functionalistic explanation of a property X in the organisms of a population is to show that X has better consequences for an individual organism's ability to reproduce, than all close alternatives to X: That is, X is a local maxima with respect to reproductive ability. [Föllesdal, et al. (1993) p. 208]

- Intention explanations presume that there is a person or a "being" that has the intention one is referring to when explaining the phenomenon. These explanations are important when explaining actions or choices, for example, to explain decisions made by organizations, corporations or nations. An intention explanation of an action consists of showing that the action—according to the acting agent—was the best means to realize some desired ends. Furthermore, it must be shown that this correspondence between action, desires, and ideas was not random, but rather that the action was executed *because* it corresponded to the acting agent's ideas and desires. [Föllesdal, et al. (1993) p 219]

### A1.5.3.3 Predicting

Predictions have the same logical form as explanations. The difference is that the facts that are expressed as the logical conclusion have not yet been observed.

### A1.5.3.4 Testing

In order to test a theory, researchers first logically deduce a prediction and then observe whether an event happens that fulfills the prediction. In order for a prediction to be useful in hypothesis testing, it must be empirical. Such a prediction is then called an *empirical consequence.* [Föllesdal, et al. (1993) p 62]

Strictly speaking the validity of consequences derived from a theory do not prove the validity of the theory. Logically, it is impossible to conclude, based upon a true (successful) prediction, that the theory from which the prediction was derived is also true. This faulty reasoning is known as *affirming the consequent*:

If H is true, then E
E is true (as shown by the empirical evidence)
Therefore, H is true

Given that such conclusions are not justified, how is it that science progresses? And how can empirical evidence be used logically to guide researchers in theory selection? These are the issues and questions that have plagued philosophers of science for the last fifty years or more. An approach for dealing with them is discussed in the next section.

## A1.5.4 Theory validation

The purpose of this phase is to validate the theory by testing consequences that regard some specific situation and that have been derived from the theory. There are three activities during this phase:

- deducing consequences (as described above)
- testing to support or refute the theory, and its underlying hypotheses (as described in section A1.5.4.1)
- evaluating the research program of which the theory is a part (as described in section A1.5.4.2)

### A1.5.4.1 Testing the theory

The last phase in the research process is the validation of the theory. As its next activity, after deriving predictions, is testing to correlate predictions with empirical results; this is the activity of *testing the theory*. It has as its inputs the special consequences from the previous consequence-deduction activity empirical observations or facts, and the output of the activity is a correlation between the predicted consequences and the observed facts. As can be seen in figure a1-1, the researcher and logic act as the controls for the hypothesis testing activity. Empirical studies and case studies are often also used during this activity to gather observations (see section A1.5.1.1).

### A1.5.4.2 Evaluating research programs, not theories

The next activity, following the correlation of predictions with empirical results is to evaluate the overall performance of the research program of which the theory is a part.

The way in which a research program advances is through developing and refining theories. What largely remains consistent throughout the life of the program are the key concepts, the scope of the field, and the history of past solved problems. However, the details of the theories will change as more and more anomalies are brought within the scope of the program and as the researchers' understanding of the field grows and matures. Therefore, according to Lakatos, this "shifts the problem of how to appraise *theories* to the problem of how to appraise *[a] series of theories*. Not an isolated theory, but only a series of theories can be said to be scientific or unscientific: to apply the term 'scientific' to one single theory is a category mistake." [Lakatos (1978) p. 34]

The judgement of whether a research program is suitably scientific, or not, is done in terms of several factors as discussed next. Specifically, two things are needed:

- criteria for evaluating whether a research program is progressing or degenerating (section A1.5.4.2.1)
- alternative research programs to consider (section A1.5.4.2.2)

*A1.5.4.2.1 Criteria for progressive and degenerating programs*

The criteria for choosing a research program is related to—but not synonymous with—the existence of anomalies or counterexamples. That is, anomalies are defined as "recalcitrant instances, not [as] refutations" [Lakatos (1978) p. 4] Specifically, anomalies are identified with the expectation that they will be "solved" by the research program. The issue is whether the process of solving these anomalies is done in a manner consistent with the programs' heuristic—that is, its program-specific set of problem-solving techniques. [Larvor (1998) p. 55] (As an example, for Newtonian mechanics, its heuristic consists of its mathematics: differential calculus, differential and integral equations, etc. [Larvor (1998) p. 53]) Given that all theories have anomalies (and thus according to Popper's definition would be considered to be *falsified*), their quality may be judged according to the following criteria. A "progressive" research program meets these conditions [Larvor (1998) pp. 54-55]:

- Theoretically progressive condition: It must make new and interesting predictions, that is, "undreamed of" [Larvor (1998) p. 55] by other theories. And these predictions are particularly good if they are counterexamples to rival research programs. [Lakatos (1978) p. 5]
- Empirically progressive condition: Some of these predictions must be corroborated by the experimental evidence.
- Heuristically progressive condition: Furthermore, when anomalies are identified, the progressive theory must be accommodating and explaining these anomalies in a manner consistent with the spirit of its heuristic (as opposed to in an ad hoc manner).

Degenerating programs are ones in which the new theory lags behind the facts.

*A1.5.4.2.2 Need for a better program to switch*

Another thing which must be considered when evaluating research programs is the existence of other, competing research programs. That is, no research program is abandoned without a better research program to switch to. As Lakatos describes,

> "Contrary to naïve falsificationism, *no experiment, experimental report, observation statement or well-corroborated low-level falsifying hypothesis alone can lead to falsification.* There is no *falsification before the emergence of a better theory*....But, of course, if falsification depends on the emergence of better theories, on the invention of theories which anticipate new facts, then falsification is not simply a relation between a theory and the empirical basis, but a multiple relation between competing theories, the original 'empirical basis', and the empirical basis resulting from the competition....The very term 'counterevidence' has to be abandoned in the sense that no experimental result must be interpreted directly as 'counterevidence'. If we still want to retain the time-honored term,

> we have to redefine it like this: 'counterevidence to $T_1$' is a corroborating instance to $T_2$ which is either inconsistent with or independent of $T_1$ (with the *proviso* that $T_2$ is a theory which satisfactorily explains the empirical success of $T_1$). This shows that '*crucial counterevidence*'–or '*crucial experiments*' can only be recognized as such among the scores of anomalies [to theory $T_1$] only *with hindsight*, only in light of some superceding theory." [Lakatos (1978) p. 35-36]

### A1.5.4.3 Switching to a new research program

Once a research program is degenerating and programs exist that rival it, how is the decision made to switch from one program to another? This question can be posed from the point of view of the individual researcher, and it can be posed from the point of view of the research community as a whole.

The criteria for evaluating programs discussed above provides some insight into the nature of the choices which are made. Specifically if a program is progressing, it is a perfectly reasonable decision to stick with a research program, and if a program is not progressing, it is advisable to seek another program. The decision to abandon a research program, however, is not made in a vacuum. It may be clear to researchers that the program to which they belong is not longer progressing, yet their ability to switch to a new program is inherently limited by the presence of other competing, attractive programs or by their ability to create such.

The issues in switching to new programs are discussed in this section. These involve the following:

- revolutions as the mechanism for change (section A1.5.4.3.1)
- incommensurability (section A1.5.4.3.2)
- gains and losses in the transition (section A1.5.4.3.3)
- the number and availability of competing programs (section A1.5.4.3.4)

#### *A1.5.4.3.1 Revolution as the mechanism for change*

The switch from one research program to another is what Kuhn refers to as a *scientific revolution.* This term is used because of the radical change in world view that can accompany the switch from one research program to another. It is not a sudden shift of the entire research community; rather it is a gradual shifting in the numbers of the individual researchers who practice one research program versus another.

In the situation where two programs are competing and one is progressing while the other is degenerating, the choice between programs is straightforward. However, Laudan discusses the choice researchers make between competing research programs when both are considered progressive. (See [Laudan and Laudan (1989)].) In this case, conflict arises between the groups of researchers in terms of their program choice when the groups do not share the same criteria for evaluation. That is, the different researchers place differing importance on the conditions listed above (for example, theoretical versus empirical results).

Some scientists will initiate the switch to a new program. Their motivation may be the greater scope of explanatory power of the new program, (that is, its application to a wider variety of problems) or perhaps its internal consistency or simplicity. Another attractive

criterion is the explanation of novel facts which the old program is unable to explain, and these can be the facts that the new program is constructed to explain. Lastly is the prediction of novel facts which were not used in the generation of the program is also attractive.

As more work is done to flesh out a new program, the amount of evidence which supports or refutes the program will grow, and as the evidence grows, increasing numbers of researchers will become convinced of the program's merits—or lack thereof. Therefore, over time, one program is likely to become dominant.

#### *A1.5.4.3.2 Incommensurability*

According to Kuhn, different paradigms exhibit *incommensurability*.[7] This quality prohibits researchers who practice the different programs from having conversations that are completely meaningful. The scientists are unable to communicate because—although they may use the same words—the meanings of the words are different. Furthermore, these scientists are unable to judge the validity of each other's work because these differences in concepts result in the fact that there is no uniform standard for what counts as evidence.

Scientists using different paradigms "inevitably see differently certain of the experimental or observational situations to which both have recourse. Since the vocabularies in which they discuss such situations consist, however, predominantly of the same terms, they must be attaching some of those terms to nature differently, and their communication is inevitably only partial. As a result, the superiority of one theory to another is something that cannot be proved in the debate. Instead,...each party must try, by persuasion, to convert the other." [Kuhn (1970) p. 198]

One recourse to this problem is translation between the two views. As Kuhn describes, "Briefly put, what the participants in a communication breakdown can do is recognize each other as members of different language communities an then become translators....Each [participant] will have learned to translate the other's theory and its consequences into his own language and simultaneously to describe in his language the world to which that theory applies. This is what the historian of science regularly does (or should) when dealing with out-of-date scientific theories." [Kuhn (1970) p. 202]

There may not be complete communication between researchers within the two different programs. However, once a rival program has been translated into one's own, a researcher should be able to make judgements about its merits in problem solving: the ones already solved and its promise in new problems to be solved. Then the researcher can decide whether to stay with the old program or to switch to the new.

#### *A1.5.4.3.3 Some loss of explanatory ability in transition to new paradigm*

The shift from one research program to another is not a uniformly progressive change. On the one hand, it many address *only some* of the anomalies that the old program was unable to solve, particularly at the beginning of the new program. So, some of the anomalies of the old program will remain as anomalies at the start of the new one. [Lakatos (1978) p. 36n]

---

[7] For a counter argument, see [Weinberg (1998)].

In addition to the above lack of progress, a new research program may not even explain *all* of the successes of its predecessors. "When a new candidate for a paradigm is first proposed, it has seldom solved more than a few of the problems that confront it, and most of those solutions are still far from perfect....Of course, it handles some problems better...But the older paradigm can presumably be articulated to meet these challenges as it has met others before [at least according to its defenders]....In addition, the defenders of traditional theory and procedure can almost always point to problems that the its new rival has not solved but that for their view are no problem at all." [Kuhn (1970) pp. 156-157] Even Lakatos recognizes that "*one must treat budding programs leniently.*" [Lakatos (1978) p. 92]

Laudan shows that such non-cumulativity can be due to the fact that some problems cannot even be formulated in competing theories, let alone solved. The proper way of viewing scientific progress then is progress towards a cognitive aim—be it, number of problems solved, maximizing probability of beliefs, etc. [Laudan (1996) Ch. 6, pp. 113-122]

*A1.5.4.3.4 One versus many paradigms in "normal science"*

A key idea of Kuhn's is the existence of *one* unifying paradigm for a field at a time. This is true, he argues, for mature sciences, and he sees this as a positive characteristic of science, one that allows a field to progress. It is, in effect, a *defining* characteristic of a science: a field which has one unifying paradigm may be termed *scientific*. [Kuhn (1970) pp. 160-164] In addition to mature sciences, Kuhn describes a stage in the history of any discipline in which that field may be said to be in a *pre-paradigm* phase or a *pre-scientific* phase of its development.

The progress which Kuhn describes as happening within a mature science is the solving of new "puzzles"; that is, science is fundamentally a "puzzle-solving" activity. Scientists draw upon their education, experience, etc. (within a particular paradigm) in order to solve new previously-unsolved puzzles, and the fundamentals of the field are no longer questioned. "Before it [the transition from the pre- to the post-paradigm period in the development of a scientific field] occurs, a number of schools compete for the domination of a given field. Afterward, in the wake of some notable scientific achievement, the number of schools is greatly reduced, ordinarily to one, and a more efficient mode of scientific practice begins. The latter is generally esoteric and oriented to puzzle-solving, as the work of a group can be only when its members take the foundation of their field for granted." [Kuhn (1970) p. 178]

One a research community has adopted a common paradigm, at least the fundamentals are no longer *constantly* questioned, and to a large extent the focus of the field's practitioners is on the solving of the puzzles articulated by the now-accepted paradigm, thus leading effectiveness and efficiency to the field's progress.

"When, in the development of a natural science, an individual or group first produces a synthesis able to attract most of the next generation's practitioners, the older schools gradually disappear. In part their disappearance is caused by their members' conversion to the new paradigm. But there are always some men who cling to one or another of the older views, and they are simply read out of the profession, which thereafter ignores their work." [Kuhn (1970) pp. 18-19]

### A1.5.5 Application to history of science

One advantage of Lakatos' and Kuhn's approach to the philosophy of science is that they justify their work based on the actual historical progress of science. Lakatos goes so far as to argue for his criteria of progressiveness as a means for evaluating his own work. Also Laudan acknowledges the importance of evaluating methodological criteria against the verdicts of history. [Laudan (1987)] That is, does a rule for theory choice lead to useful results, validated by history? In contrast, the results from applying Popper's falsification criteria to *his* work leads to the conclusion that since some scientists stuck with their theories despite empirical results which could be considered to have falsified them, then *Popper's work has been falsified* and should be rejected! [Larvor (1998) p. 50]

### A1.5.6 Summary

Therefore, the key idea of validation is that scientists do not look at individual critical experiments, but rather evaluate the overall success of a program (series of theories) relative to its competitors. No theory is ever going to be "proven" inductively from empirical evidence, so in program evaluation, the questions becomes these: Which program does the evidence support more? Which program holds more promise for continued good results (correlation with reality) and for improved knowledge (new problems solved)?

## *A1.6 Discussion on axiomatic design*

This section brings together the above ideas about design theory and research methods and uses them to describe axiomatic design as a unifying research program for design theory. This section discusses how axiomatic design fits the criteria for a unifying program and evaluates it against the criteria for being progressive or degenerating.

### A1.6.1 Axiomatic design: a science of design?

The components of a research program (as described in section A1.2.2.3) are these:

- ontology: an identification of the fundamental concepts or entities which make up the field of study
- aims: an articulation of the scope of the field in terms of both problems which have been solved (exemplars) and problems remaining to be solved (anomalies) which should be covered by the program, and are expected to be, but have not been yet
- theories: relationships between the fundamental concepts of the field and application to specific problems
- methodology: guidelines for further developing the program—particularly in a manner consistent with the problem-solving approach that the program has been following

The underlying hypothesis of axiomatic design is that there exist fundamental principles that underlie good design practice. This is combined with a view of design practice as a broadly applicable process including creativity and decision-making: a set of activities in which designers develop and/or select the means to satisfy objectives, subject to constraints.

Of course, this idea is stated very broadly, but to flesh it out, Suh and his colleagues have articulated specific concepts and examples to show, to teach, and to more fully understand

how these ideas can be applied to specific situations.[8] Here, only enough of the theory will be presented to explain how axiomatic design may be viewed as a research program for understanding design.

## Fundamental concepts

The fundamental concepts of axiomatic design are domains (including CNs, FRs, DPs etc.), hierarchies, zigzagging, independence, and information content. Each of these concepts is a hypothesis or set of hypotheses within one of the knowledge areas of design theory.

*Relationship to fundamental knowledge areas and hypotheses*

The above concepts of axiomatic design fall within the scope of the field of design as given in sections A1.2.2.1 and A1.3. Specifically the statements about domains and hierarchies are hypotheses about the design object; the statements about zigzagging and information deal with the design process and the design object; and the statement about independence is a theory which relates the two areas with the resource of time. The concepts are listed in table a1-2 along with their hypotheses and associated knowledge areas.

**Table A1-2. Concepts and hypotheses in axiomatic design**

| Concept | Knowledge Areas | Hypotheses[9] |
|---|---|---|
| Domains | Design object | Design problems can be modeled as a mapping between domains.<br>Requirements are to be articulated in a solution-neutral environment, that is, in a distinct domain from the physical (or even intangible) solution. |
| Hierarchies | Design object | Design consists of a hierarchy of design tasks at multiple levels of abstraction. |
| Zigzagging | Design object and process | Decisions made at higher levels of the design hierarchy constrain and formulate design problems at lower-levels. |
| Information content | Design object and process (more object) | Information content is a means for measuring the "quality" of a design.<br>The quality of the design must be evaluated against a clearly stated set of requirements (including tolerances, etc.) |
| Independence | Design object and process and resources | There is a positive correlation between design decisions that maintain independence among functional requirements and both improved design quality and reduced use of resources (such as time)<br>Independence can be modeled in terms of design matrices. |

[8] An overview of axiomatic design is given in appendix 4 in which are discussed the basic concepts of the theory, its development, and recent advances.

[9] This is not meant to be an exhaustive list of hypotheses associated with each concept, rather, a representative sample.

**Theories**

The theories of axiomatic design relate the above concept and knowledge areas. The areas which are connected by theory in axiomatic design are the design process and the design object. Axiomatic design consists of the concepts plus the relationships between them. There is a correspondence between the goodness of designs and the decisions made by designers: Specifically, if designers follow independence in making their design decisions, their designs will have a higher probability of meeting the designs' requirements.

Two universal laws of design are the design axioms. These are the Independence Axiom and the Information Axiom. They are usually stated as follows:

*The Independence Axiom:*
Maintain the independence of the functional requirements.

*The Information Axiom:*
Minimize the information content of the design.

Can these properly be considered as scientific theories? The answer is yes.

*Methodological statements as universal laws*

An argument was made in section A1.5.2.2.1 that universal laws can have the form of normative statements. The same argument can be applied to the design axioms. In axiomatic design, the desired cognitive aim is to improve the satisfaction of requirements versus time (and other resources), and this can be tested empirically.

Overall, the idea for axiomatic design is that by following the design axioms better designs will be achieved. These better designs may be interpreted in terms of probability of satisfying requirements and should also include a notion of time. Specifically either better designs should be created given the same amount of time (improved effectiveness), or alternatively designs of comparable quality should be achieved in less time (greater efficiency).

**Aims and methodology**

In this section the components which make up axiomatic design have been discussed, its fundamental concepts and its theories. Moreover, the scope of axiomatic design is fully as broad as that of design theory in general because axiomatic design is generally applicable to all design problems—as evidenced by the breadth of the case studies to which it has been applied. Finally axiomatic design provides a mechanism for researchers or engineers to addresses new problems through modeling the problems in terms of domains, hierarchies, and design matrices, and then to examine the independence and information content of the design.

Therefore, yes, axiomatic design does possess all the necessary components of a research program, yet, is axiomatic design a *progressive* research program? The next section addresses the growth of the field. Is axiomatic design capable of answering more and more questions? Is it bringing anomalies (or problems to be solved) within its scope? Is it doing this in a manner consistent with its heuristic? The answers to these questions are vital in proper theory validation and program choice.

## A1.6.2 Axiomatic design as a *progressive* research program

There are three conditions for a research program to be progressive (see section A1.5.4.2.1):

- Theoretically progressive condition: Does axiomatic design theory lead to new, "undreamed of" predictions—particularly in contrast with other theories?
- Empirically progressive condition: Does axiomatic design theory have experimental corroboration of some of these predictions?
- Heuristically progressive condition: Is axiomatic design explaining and accounting for anomalies in a way that is consistent with the spirit of its heuristic?

The answers to each of these questions is yes as described briefly in the next sections, A1.6.2.1, A1.6.2.2, and A1.6.2.3 respectively. Furthermore, it should be mentioned that there are no competing, better research programs for design theory which explain the successes of axiomatic design and make better predictions.

### A1.6.2.1 Theoretically progressive condition: new predictions

Axiomatic design makes predictions about designs and describes the successes and failures of designers in ways that no other theory of design does. Other research may describe what it is that designers do (as in idiographic, descriptive research). Other research programs may provide prescriptive advice about narrow design problems (for example, specific field-related rules). Only axiomatic design, however, combines the broad scope of describing any decision-making activity with relatively clearly articulated concepts for applying principles to many particular cases thus enabling use of the theory and its empirical validation.

Axiomatic design as created explained the successes of a small number of designs—principally from the field of manufacturing process design. (Granted the axioms were intended to be universally applicable laws, but the number of concrete examples Suh drew upon in creating the axioms was very small, less than a dozen.) Subsequently axiomatic design has been successful widely beyond the scope of the few design successes it was inductively created to explain. Therefore, yes, axiomatic design leads to some interesting predictions beyond the scope of the problems it was developed to explain.

### A1.6.2.2 Empirically progressive condition: corroborating empirical results

Axiomatic design has proved useful in many real-life, industrial cases. The number and variety of its applications are too numerous and broad to discuss in detail here. (See for example [Nordlund (1996), Nordlund, et al. (1996), Suh (1996), Suh (1999)] for descriptions of recent successes in using axiomatic design.) Axiomatic design has been applied to many fields—including products, software, systems, business planning, etc.—beyond the initial one of manufacturing processes.

### A1.6.2.3 Problem-solving approach and digesting of new problems

Axiomatic design poses interesting research questions which remain to be solved.

The heuristic for axiomatic design consists of its analysis of designs in terms of design hierarchies and matrices. These concepts have been partially systematized in the form of system architectures which capture the relevant information and have been assisted by software tools for their capture.

Recent advances in axiomatic design are the incorporation of time-varying FRs, robustness, command and control, CAD tools, application to business planning, incorporation of other

tools (for example, TIPS), and ECOs. No one would have thought to look at some of these issues without first understanding axiomatic design and identifying design questions the researcher or designer needs to answer. Some areas of work to be done include cost, system design, and diagnostics.

### A1.6.3 Summary

The argument made here is that, axiomatic design does possess all the necessary components of a research program. Moreover, axiomatic design constitutes a *progressive research program*:

1. Axiomatic design makes predictions about designs and describes the successes and failures of designers in ways that no other theory of design does: Only axiomatic design combines the broad scope of describing any decision-making activity with relatively clearly articulated concepts for applying general principles to many particular cases.
2. It has proven useful for answering real-world questions encountered by designers in industry.
3. Axiomatic design, however, is not yet able to answer *every* question posed about *every* design problem, yet the presence of questions to be answered (or concepts to be further elaborated), is a hallmark of a healthy research program and provides a direction for further research.

Therefore, following the research approach (*heuristic*) of axiomatic design and addressing the problems it identifies (*anomalies* or problems to be solved) is a valid way of adding to the scientific body of knowledge of design. As such it has been chosen as the starting point for this research.

## *A1.7 Conclusions*

In conclusion, this chapter has sought to make several points.

First, the need for a unifying view of design is established. Second, the ideas of philosophy of science are applied to the field of design to elaborate the activities performed by researchers who seek to understand design. It is the belief of the author that *research into design can be performed in a scientific way*: that scientific methods can be applied to expand our understanding of design and the principles which underlie it. The criteria for judging the results of scientific research—theories and research programs—are articulated and applied to research in the field of design theory.

Third, the above ideas are brought together as axiomatic design is presented as fulfilling the need for a unifying vision of design. Axiomatic design theory possesses all the necessary components of a progressive research program: It makes unique predictions about design and the successes and failures of designers; it has proven useful for answering real-world questions encountered by designers in industry; and it continues to successfully digest new problems to be solved without resorting to ad hoc strategies in modifying the theory.

# APPENDIX 2: DESIGN PROCESS ROADMAP[1]

## *A2.1 Introduction*

An effective product development process, supported by scientifically validated design theories and tools, is becoming an increasingly useful asset in industry for reducing lead-times and costs as well as for improving quality. However, engineers and managers in industry have not been able to integrate theory and practice into their product development processes because they lack a means by which to match their unique problem situations and activities with available design theories and methods—including, for example, Axiomatic Design [Suh (1990), Suh (1995a)], Pahl and Beitz's method [Pahl and Beitz (1988), Pahl and Beitz (1996)], Quality Function Deployment (QFD) [Clausing (1994)], Robust Engineering (Taguchi methods) [Phadke (1989)], Structured Analysis and Design Technique (SADT) [Ross (1977), Ross (1985)], Theory of Inventive Problem Solving (TIPS/TRIZ) [Altshuller (1988)], Total Design [Pugh (1991), Pugh (1996)], VDI 2221, and the WDK school (Hubka's theory) [Andreasen (1991b), Hubka and Eder (1992), Hubka and Eder (1996)].

As stated by Cross [Cross (1992) p. 9], "[W]e lack a successful, simplifying paradigm of design thinking. Those simplifying paradigms which have been attempted in the past—such as viewing design simply as problem-solving, or information-processing, or decision-making, or pattern-recognition—have failed to capture the full complexity of design thinking."

Ross defines a *model* as "M is a model of A if M can be used to answer questions about A" [Ross (1985)]. This chapter presents a model of design process activities for the use of those interested in implementing product development processes, supported by theories and tools, and for those interested in explaining the events and outcomes of these processes. The purpose of this model is to provide answers to questions including

- What is a design process model which is flexible enough to include all instances of the design process and which is specific enough to enable designers and managers to integrate available design theories and tools into their practice?
- How can an observed phenomenon in a design process be explained?
- How can design theories and tools be used to better support the activities of the design process?
- What is the context of a current research effort?

To answer these questions specific design activities which comprise all design processes were identified so that practitioners can piece them together in an appropriate sequence. This yields a flexible tool for structuring unique design processes for design situations encountered in industry, yet the tool is sufficiently specific to allow mapping of theories and methods to activities. It can also be used in academia to identify new research questions and to place research in a context useful for industry.

[1] This chapter is adapted from [Tate and Nordlund (1996)], which will also appear as [Tate and Nordlund (1998)].

## *A2.2 Model of the design process*

The design process model presented in this section is an abstraction and generalization of several design processes that were studied in European and US industries. [Nordlund (1996)] In the model, terminology from axiomatic design [Suh (1990), Suh (1995a)] has been used in order to identify consistently the information developed during and transferred between the activities.

### A2.2.1 Properties of the design process model

The design process model consists of a collection of distinct activities with clear starting and end points. Each activity is the transformation of inputs to outputs. These activities can be sequenced in many ways. The activities are to be fit together into a project-specific design process. Each project will have its unique sequence, depending on its status, scope, and goals.

Figure A2-1 shows all activities and the possible links between them in a design process roadmap. This model should enable the explanation of the decisions regarding the sequence of activities of any design process. The design process model consists of a set of activities: project control and decomposition, concept generation and analysis, design object analysis, decoupling, optimization, and tuning.

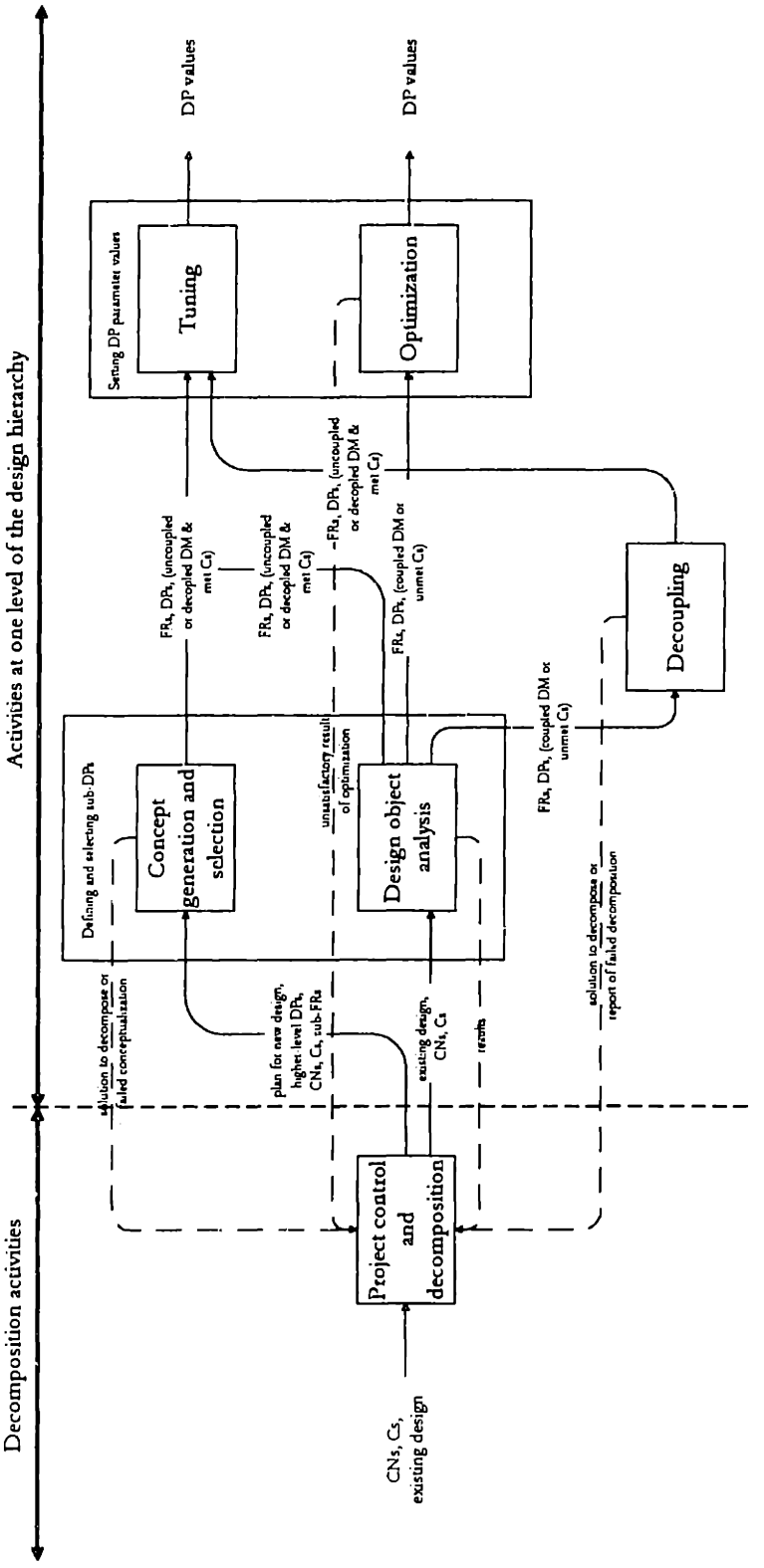


*Figure A2-1. The design process roadmap*

The start of a design process (i.e., project) is at the left side of figure a2-1. Here, customer needs and constraints imposed by the customer and the environment are the input. At the end of the design process, at the right of figure a2-1, a solution is specified in terms of product and implementation (i.e., manufacturing) details.

## A2.2.2 Descriptions of the generic activities and their problem situations

This section contains a detailed description of each activity in the design process model in figure a2-1. Each description explains the activity's purpose, inputs, outputs, and typical questions encountered in the activity.

### A2.2.2.1 Project control and decomposition

Project control and decomposition is the activity in which the scope of and controls on the design project are established. In a design project, this activity will be revisited multiple times. This activity will be performed when it is necessary either to plan for work at a more detailed level of the design or to plan activities in response to an inability to solve a previous problem.

During project control, possible courses of actions are evaluated and decided upon. The scope of the design project and other project management issues such as a budget, milestones, etc. are decided. Typical questions of this activity include the following:

- What resources are available to solve the problem?
- Will a project be a new design or an adaptation of an existing design?
- How can the project be decomposed into subproblems?

The inputs of project control and decomposition can vary. At the beginning of a clean slate project, the inputs are customer needs and constraints. At the start of a re-design, or evolutionary design project, the inputs will include customer needs, constraints, and additionally a representation of the existing design object. When this activity occurs in an ongoing design project, the inputs will be descriptions of the design object at the current level of abstraction and a description of problems which have occurred, if any.

Outputs of this activity are project goals, constraints, and instructions on for conducting the design project through performing a sequence of activities.

### A2.2.2.2 Design object analysis

Design object analysis, the analysis of an existing solution, presupposes an existing design object about which there are questions regarding functionality or feasibility. The analysis may follow a specific approach, for example, axiomatic design [Suh (1990)], functional analysis (value engineering) [Miles (1972)], or TRIZ [Altshuller (1988)]. It is often a central activity in feasibility studies where the results of this activity can be fed back to the project control activity to allow a detailed planning and control of the design process. Design object analysis can also be conducted to identify areas for improvement in existing designs. Typical questions of this activity include the following:

- Are off-the-shelf, or other existing, solutions acceptable for this project, or is a new solution needed?
- Does the design meet its constraints?

- What are the DPs and FRs of the design, and are there couplings in the design?
- What improvements can be made to the design?
- How can the design object be changed to reduce cost? reduce the number of parts? reduce assembly time?

In design object analysis, there are two inputs: an understanding of the customer needs, and an existing design to analyze. The output of this process is a description of the design in terms of its functions, constraints, physical implementation, and functional independence (or dependence).

### A2.2.2.3 Concept generation and selection

Concept generation and selection (conceptualization) follows project control and decomposition. The objectives of the conceptualization activity are

- to develop concepts that satisfy the specifications derived in project control and decomposition
- to decide which of these concepts to implement

Hence, this activity is supported by concept generating (synthesis) tools and by concept selection (analysis) tools. Typical questions addressed during this activity include the following:

- Given a set of functions, what are possible solutions?
- What are different ways can this problem be solved?
- Will this concept work?
- Which concept should be chosen, and why?
- How can conceptual solutions be physically integrated while achieving their functions?

Concept generation tools include brainstorming, databases, morphological tables, etc., while analysis tools include the design axioms [Suh (1990)], group decision making, Pugh concept selection [Pugh (1991), Pugh (1996)], and other design rules. Suh illustrates this activity as a feedback control loop (see figure a2-2) and emphasizes the complementary nature of synthesis and analysis: "If we cannot analyze a design solution, then we cannot rapidly generate the 'best' design since we cannot distinguish a good design from a bad design." [Suh (1990)]

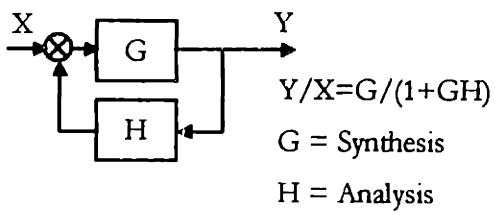


*Figure A2-2 Feedback control loop depicting synthesis and analysis during conceptualization, from [Suh (1990)]*

Inputs to this activity are a set of functional requirements (FRs) and constraints. Outputs are a set of FRs, constraints, design parameters (DPs), and design matrices (DMs) which show the functional dependencies within the design object.

### A2.2.2.4 Decoupling

Decoupling (conflict resolution) is one possible follow-up activity after design object analysis in the case where the result of the analysis shows that the existing design object does not satisfy Suh's independence axiom (see [Suh (1990)]). (The other alternative in this case is the optimization activity, described below.) The desired output of the decoupling activity is an uncoupled or decoupled design and is sought by applying problem-solving strategies for conceptual design (for example, Altshuller's principles [Altshuller (1988)], su-field analysis [Altshuller (1988)], Suh's theorems [Suh (1990)], etc.).

Often, decoupling is performed when the designer must improve the design's performance, but the designer does not have the freedom to make major changes to the design. Typical questions that this activity is intended to answer include these:

- What modifications to the design object would ensure functionality?
- How can the conceptual solutions be implemented better to meet the constraints?

The input to this activity is a description: either of a design or concept that is coupled or of a design object that does not satisfy its constraints. In the first case, the design object does not perform its functions satisfactorily, in the second case, the design object performs its intended functions, but the physical solution is unacceptable for another reason (e.g., it is too big, or too heavy, or it does not meet some legal requirement, etc.).

The preferred output of this activity is a description of a design object, or concept, that is functionally uncoupled and which meets all its constraints. With this result, the designer decomposes the design further into sub-projects (in project control and decomposition), or the designer progresses to implementation.

In cases which produce an unsatisfactory (coupled) design, a report of the activity's failure is then the input for a repetition of the project control.

### A2.2.2.5 Optimization

The objective of the optimization activity is to get the best possible performance out of a design which does not satisfy Suh's independence axiom (see [Suh (1990)]). (The tuning activity concerns determining parameter values for designs which do satisfy the independence axiom and is described below.)

The optimization activity becomes necessary when an existing design has been found to have inherent problems, but no changes to the concept may be made other than small changes to some DP values. In this situation, the optimization activity entails setting the design parameters to values that, as closely as possible, provide the desired functionality. Typical questions answered during this activity include the following:

- How should the values of these parameters be set to provide the best possible functionality?
- What is the best functionality that can be achieved with this design?

The input to this activity is an unacceptable design (not independent or not meeting its constraints) described in terms of functional requirements, constraints, design parameters, and design matrices. Outputs of this activity are functional requirements, constraints, design matrices, and design parameters that are set at their optimal values given that they are a part of a design which does not satisfy the independence axiom.

#### A2.2.2.6 Tuning

The tuning activity is the desired final activity for any design project. The key difference between tuning and optimization is that when tuning is performed, the analyses during the design process have shown that all functional requirements can be achieved because the design satisfies Suh's independence axiom. A typical question for this activity is

- How should the values of these parameters be set to provide optimal functionality?

Inputs to tuning are functional requirements, constraints, design parameters, and uncoupled or decoupled design matrices. Outputs for tuning are functional requirements, constraints, design matrices, and design parameters that are tuned to provide optimal functionality for this design object.

### A2.2.3 Connecting the activities

The design process roadmap presented in this paper consists of a set of activities as described in the previous sections. These activities are project control and decomposition, design object analysis, decoupling, concept generation and analysis, optimization, and tuning.

A design process (project) begins with the project control activity at the left side of figure a2-1. Here, customer needs, constraints imposed by the customer or the environment, and any existing solutions are the inputs. At the end of the design process, the final activity is either optimization or the tuning activity in which a solution is specified in terms of a design object and its implementation (drawings, models, and manufacturing) details. If no solution—at all—can be found, a decision is made during an instance of the project control activity to terminate the project.

The specific path which is followed between the starting and the end points is the responsibility of the design team. The preferred outcome of the project is to reach the end of the tuning activity—and thus be done with the whole project—as quickly as possible.

Each time an activity is completed, a decision is made about the progress of the design. By understanding the desired outcomes of the various activities as described in the model (FRs, DPs, uncoupled/decoupled DM, etc.) and by comparing these with the actual outcomes, decisions can be made as needed and can be based firmly on concepts of independence and minimum information (maximum probability of success) [Suh (1990)].

Often the choice to perform one activity over another if performed successfully will allow the design team to reach its goal in less time with better results. For example, the choice to decouple a design, as opposed to performing a optimization, will—if successfully done—allow a design to be implemented which satisfies the independence axiom. Optimization, on the other hand, is a much more involved and iterative process which is not guaranteed to yield a desirable result.

### *A2.3 Summary*

Existing design process models have shortcomings in that they either cannot be adjusted to reflect an actual design process or do not provide sufficient specificity to support the practitioners in this design process. Thus, a new tool for structuring and explaining the product development process has been introduced. It consists of a roadmap of specific design activities that can be assembled to describe any unique design process.

# APPENDIX 3: SCOPE AND EVOLUTION OF DESIGN RESEARCH[1]

## *A3.1 Introduction*

In recent years, researchers and practitioners worldwide have recognized the importance of structured, scientifically-based, and industrially-tested theories and methods for product (and process) design and development. Recent research has sought similar goals: reduced development time, reduced product costs, and increased value delivered to customers. However, American and European research in engineering design and product development have evolved differently and are distinct in their scope of application. Consequently, little integration and cross-learning have been done. In this chapter a categorization of design research approaches[2] based on evolution and scope is given. This categorization is used to explain the reasons for lack of integration of design research.

## *A3.2 Scope and evolution of design research*

Four prominent programs of design research are covered:

- Suh's axiomatic design [Suh (1990)]
- total quality development (TQD) as elaborated by Clausing and others (including the house of quality, quality function deployment (QFD), Pugh concept selection, and quality engineering) [Clausing (1994)]
- Altshuller's theory of inventive problem solving (TIPS) [Altshuller (1988)]
- the Workshop Design Konstruktion (WDK) school—the work of Hubka and Eder [Hubka and Eder (1992)], and Andreasen [Andreasen (1991a), Andreasen (1991b)]

Each program approaches engineering design from a different perspective. In discussing the evolution (development) and scope (extent of treatment, activity, or purpose) of these design theories and methods in this chapter, it is assumed that the reader is familiar with the basic concepts and tools of each of these design theories and methods. The objectives of the discussion are

- to explain the reasons for lack of integration of design research in academia
- to illustrate the difference between the creation of theories and methods and the selection of such theories and methods for use in industry

### A3.2.1 Evolution of research methods

This section addresses how researchers in the field of engineering design have answered the question: how is knowledge and practice in the field of engineering design advanced? That is, what general approach to studying, systematizing, and transferring engineering design knowledge is to be followed?

By comparing different research programs and examining the literature on engineering design, one realizes that design researchers have evolved their theories in different ways.

---

[1] This chapter is adapted from [Tate and Nordlund (1995)].

[2] *Approach* is defined as the broad approach researchers take to develop knowledge for their research programs.

These differences can be explained by the different ends the research programs are seeking to achieve as well as the approach the researchers have taken to achieve these ends.

In this section categories for classifying the evolution of design research are presented. Several prominent research programs will be mapped onto this categorization in section A3.2.3.

Systematic research in engineering design began in Germany in the 1850s. Material presented by Altshuller, Bjärnemo, Pahl & Beitz, Phadke, and Suh [Altshuller (1988), Bjärnemo (1983), Pahl and Beitz (1988), Phadke (1989), Suh (1990)], lists some of the most influential work in the engineering design field.[3]

Very little recent European research builds on material developed in the US and vice versa. One explanation for this is the language barrier. Finger and Dixon write that "even though a large body of research has been published in German, only a small fraction of this (e.g., Hubka and Pahl & Beitz work) has been translated into English". [Finger and Dixon (1989)] Going in the reverse direction, even less material has been translated from English into German. Research culture and peer pressure have been other, unmentioned important factors in isolating the research communities.

When examining how knowledge has been developed by the researchers in this field, two general approaches were found, these are described briefly in sections A3.2.1.1 and A3.2.1.2 below.

### A3.2.1.1 Generation of new knowledge

*Theories and methods developed from some fundamental principles*

This is analogous to developing a new design from scratch. Based on the researcher's understanding of the problems with current design practice, a general statement of objectives is created. This approach then leads to the development of prescriptions to meet these stated objectives. The prescriptions take the form of theories or general principles. The principles, as defined, cover a broad range of design problems. Lastly, a process is created to apply these principles to the specific situations encountered by designers working in their particular fields (e.g., mechanical design)

*Theories and methods developed based on the study of good design practice*

This is analogous to redesign of something which exists: We like the results we're getting; we'd just like to do things in fewer steps, cheaper, etc. Existing techniques for creating product X are studied. Prescriptive principles are developed to aid in the design of the next X. Finally these principles are applied to the design of new X.

### A3.2.1.2 Selection/use of existing knowledge

*Objectives then selection of tools*

This is analogous to selection from among existing designs. The objectives for the particular problem at hand are stated, and design tools which exist are evaluated against a set of

[3] See table A1-1 in appendix 1

selection criteria. Finally, the design tools that best satisfy the objectives are integrated to form a design process.

## A3.2.2 Scope[4] and motivation

In this section, several specific research programs are examined and the objectives which the researchers have stated are matched with the methods that they have followed. In particular the research programs which are examined are those put forth by Altshuller [Altshuller (1988)], Andreasen [Andreasen (1991a), Andreasen (1991b)], Clausing [Clausing (1994)], Hubka and Eder [Hubka and Eder (1992)], and Suh [Suh (1990)]. These programs were chosen because they represent a cross section of research in design today—and are available in English. Additionally by examining these, insight is gained into differences between European theories which have drawn upon the early work of German researchers and those which arose independently of German research over the last half century.

Two questions are answered in this section with respect to each of the programs examined:

1. What is the motivation for each program?
2. What is the scope of each program?

The first question deals with the overall goal for the research: why is research in design important? The second deals with how large a problem is addressed: what should be covered by the program, and what are the limits of its applicability?

*Altshuller*

Altshuller recognized the need for a scientific approach to invention after listening to scientists and inventors speaking of "[design as] sudden enlightenment, the impossibility of controlling the creative process, but also [the impossibility of] understanding what it is and how it comes about." These discussions prompted the following questions: "Why should everything but creativity be open to scrutiny? What kind of process can this be which unlike all others is not subject to control?" The consequences of creativity being an uncontrolled process was clear to Altshuller in that "[m]any inventions have come too late [and that i]nventors make frequent mistakes, dreaming up [unrealistic solutions]." [Altshuller (1988) pp. ix-x]

The motivating objective for Altshuller is to make creativity become a controlled process. Creativity may be taken to be the activity of generating new designs. It does not include the selection from among existing designs, rather it is concerned with the statement of problems, an analysis which identifies a key area of conflict, and the application of solution guidelines to the specific situation at hand.

Altshuller's program is intended to enhance the engineer's thinking during innovative work thereby contributing to the overall design process. According to Altshuller, the scope of his method of creative or innovative thinking is general, although it is mainly aimed at engineers. "The principle of controlled thinking in the solution of inventive problems (the principles and not concrete formulae and rules) can be transposed to the organization of creative thinking in any sphere of human activity." [Altshuller (1988) p. xi] Applying this method in

[4] The definition of *scope* used here is the following: "extent of treatment, activity, or influence". [Webster's (1988) p. 1053]

the context of a design project should provide benefits in the form of a reduced number of iterations and better solutions (based on Altshuller's definition of what is a good product). Altshuller describes how to go about solving a technical conflict in his algorithm for solving inventive problems.

*Clausing*

The driving force behind Clausing's work has been a very broad one—to improve industrial performance. Our interpretation is that Clausing is trying to use product design research to improve the overall product development process. He incorporates the work of other researchers into his framework, which he calls "total quality development".

The scope of design as viewed by Clausing is very broad: "*Total quality development* is the modern way of developing new products that will be competitive in the global economy. It combines the best engineering, the best management, the best strategy, and especially, the best teamwork. The resulting improvements are greatly reduced development time, a reduction in all costs, higher quality, and increased product variety. Combined, these improvements greatly increase customer satisfaction". [Clausing (1994) p. 3]

*Suh*

Suh's primary motivation for developing axiomatic design is education; he wants designers to learn how to make good design decisions. Suh's goal is to establish an "academic [discipline] for design and manufacturing". [Suh (1990) pp. 21-22] The reason is found in the following: "[i]n order to obtain better performance, both engineering and management structures require fundamental, correct principles and [methods] to guide *decision making in design;* otherwise, the ad hoc nature of design can not be improved". [Suh (1990) p. 5] To be effective "the student must be taught to see the big picture and [be taught] the ability to conceptualize a solution, as well as how to optimize an existing product or process". [Suh (1990) p. 22]

Suh's view of the scope of design may be summarized by the following: "Design, as the epitome of the goal of engineering, facilitates the creation of new products, processes, software, systems, and organizations through which engineering contributes to society by satisfying its needs and aspirations". [Suh (1990) p. 5] This is a more restricted view than Clausing, but encompasses more than Andreasen or Hubka. In contrast to Clausing, Suh does not describe how to connect design activities to the company's general activities; Suh's theories and methods are focused on decision making in the design process.

In his book [Suh (1990)] Suh considers designs primarily in three fields: manufacturing process design, product design, and organizational design. Although the bulk of his personal experience in applying axiomatic design is limited to these three areas, he recognizes the potential for its application in other fields. Industrial use and acceptance of axiomatic design has been growing in a variety of fields. Recent applications of the theory have included product design, manufacturing process design, the design of software configuration control systems, organizational design, and corporate planning.[5]

[5] See [Nordlund (1994), Nordlund (1996), Nordlund, et al. (1996)] for a description of these applications.

*The WDK School*

Hubka and Eder are working to enable systematic work in designing, and "independent auditing" in order to improve the efficiency of the designer. [Hubka and Eder (1992) p. iii] The motivation for this is given as follows: "to make [design] more efficient by scientifically reducing or eliminating waste of labor, time, or materials". [Hubka and Eder (1992) p. 45]

Andreasen observes that European schools of design in general have the utilization of the theories and methods in practice as the declared aim of their research. Andreasen considers the aim of design methodology to be "to structure design procedures and to model them, and also to give support to each step through models and methods with the aim of increasing efficiency and of making the area easy to learn and transparent". [Andreasen (1991b) p. 1]

Our interpretation is that although some literature on this school suggests that different groups perform their tasks in an integrated manner, that these groups are separate entities: product planning, engineering design, etc. Furthermore, these groups are not using the same fundamental process (that is, the design process) for performing their activities. The design process is restricted to the certain stages of the product development process: after specification of needs, but before manufacturing.

Looking at the beginnings of the design process, Hubka and Eder state that design can be considered broad enough to include defining needs and product planning as well as the narrow view of designing. [Hubka and Eder (1992) p. 49] Clearly though, they do not feel that this is *necessarily* within the scope of design. An engineering design team begins its design task when it receives a set of requirements (either from a customer or another sponsor). "This document is the start of the design sequence, the engineering design team accepts the assignment of the problem." [Hubka and Eder (1992) p. 74] The end point of the design process is a description of a technical system, specifically a "full and complete description of an optimal product (i.e. a technical system) is considered the aim of an engineering design process, its *output*". [Hubka and Eder (1992) p. 46]

#### A3.2.2.1 Discussion

European schools of design tend to separate out a portion of product development activity and address this primarily. There is no specific word in English for this activity, but Clausing has a term for it: "The undergraduate engineering curriculum typically...includes one or two design courses. These concentrate on creative concepts and feasibility, the assurance of a first-order compatibility with the laws of nature. Let us call this *partial* design." [Clausing (1994) p. 5] This term is unsatisfactory, however, because it unfairly misleads. The activity is not partial in the sense that it is incomplete; rather it only covers a small, well-defined fraction of design. The Germanic word *konstruktion* is proposed for this narrow, detailed activity.

### A3.2.3 Evolution

In this section, the categorization among the evolution approaches described in sections A3.2.1.1 and A3.2.1.2 of the theories and methods covered is described in more detail.

### A3.2.3.1 Generation of new knowledge

*Theories and methods developed from some fundamental principles*

*Altshuller*

Altshuller identified "a need for new methods for managing the creative process capable of radically reducing the number of 'empty' trials." [Altshuller (1988) p.3] during a trial-and-error approach. Also needed was a new organization of the creative process that would permit the effective application of new methods. All this required a scientifically based theory for the solution of inventive tasks that is capable of being implemented in practice. [Altshuller (1988) p. 3]

In order to develop such a theory three requirements were established. If these requirements were satisfied, it would be possible to guarantee a solution to any technical problem. The requirements were 1) "information about the whole of physics," 2) "tables linking the type of problem to the respective physical effects," and 3) "control of psychological factors that inhibit the thinking of the inventor." [Altshuller (1988) p. 35]

Altshuller begun work on an algorithm[6] for the solution of inventive problems [Altshuller (1988) p. 36] in 1946. He studied the experience of inventive creativity from a fundamental point of view and brought out the characteristic features of good solutions (that is what distinguished them from bad solutions). As a result of these studies, Altshuller discovered that "the solution of inventive problems turned out to be good if it overcame the technical contradiction[7] contained in the problem presented, and bad if the technical contradiction was not revealed and eliminated." [Altshuller (1988) p. 40]

*Suh*

Suh started the development of his program by asking: "Given a set of functional requirements for a given product, are there generally applicable axioms which yield correct decisions in each step of manufacturing (i.e., starting from the design stage to the final assembly and inspection stages) so as to devise an optimal manufacturing system?" [Suh, et al. (1978)]

A heuristic approach was used to develop the axioms. This approach involved positing an initial set of axioms that were subject to trial and evaluation in manufacturing case studies. This evaluation would then be used in order to expand, redefine, and refine the original set of axioms, until the process converged on a comprehensive set of axioms. [Suh (1990)] Based on such a set of axioms, many specific methods for analysis and problem solving could be developed. [Suh (1990) p. 171] Out of this exercise evolved twelve hypothetical axioms which later have been reduced into two and a set of corollaries and theorems. [Suh (1990) p. 20]

Suh had started his search for design axioms by observing that that there are good design solutions and unacceptable design solutions. Because these can be distinguished, this indicated that there exist features or attributes that distinguish these. The first axiom defines

---

[6] Altshuller defines an *algorithm* as any sufficiently clear program of action. [Altshuller (1988) p. 36]

[7] A technical contradiction exists "if [when using] certain methods [to improve] one part (or one parameter) of a technical system, it is inadmissible for an other part (or other parameter) to deteriorate in the process". [Altshuller (1988) p. 28]

an acceptable design as one where design parameters and the functional requirements are related in such a way that a specific design parameter can be adjusted to satisfy its corresponding requirement without affecting other functional requirements. The second axiom states that the best design of several proposed is the one that has the lowest information content (highest probability of success). [Suh (1990) p 47-8]

*Theories and methods developed based on the study of good design practice*

*The WDK School*

In contrast with the other programs discussed here, this school, as presented by Hubka and Eder, has its primary focus to develop descriptive models—of both technical systems and the design process. [Hubka and Eder (1992) pp. 71-102] When such descriptive theories are established, "it would be desirable if the [prescriptive] statements (of advice and compulsion) could be derived from the descriptive [theories]". [Hubka and Eder (1992) p. 116]

Based on this general procedural model of the design process, a procedural plan for a specific situation can be "derived and adapted from the ideal model". [Hubka and Eder (1992) p. 59]

Andreasen, has further evolved Hubka and Eder's work, based on his belief that designers are, in general, unable to describe large parts of their work, as "it takes place in unnamed patterns of ideas, rapid experimental patterns of association, and partly sub-consciously." Therefore, Andreasen determined that "the task of design research must be to create the conceptual framework and the patterns of thought." In order to support design of mechanical systems, Andreasen concludes that the design theory must be based on a theory of the design process and a theory of mechanical systems. [Andreasen (1991b) pp. 1-2] He also believes that if "we are to make progress in design science, we have to create a theoretical apparatus so that we can discuss design and attempt to derive laws, models and methods". [Andreasen (1991b) p. 10]

### A3.2.3.2 Selection/use of existing knowledge

*Clausing*

As was described in section A3.2.2, the motivation behind this approach is pragmatic; if a technique works (that is, improves the design process or design object), it is more important to put it into use than to understand exactly why it works. Thus this school consists almost entirely of methods, not theories.

In developing his approach to design, Clausing has been using two primary sources: personal experience from industry and benchmarking the best practices around the world then integrating the best components he has found into a holistic approach to design. [Clausing (1994) p. xix]

This approach of developing a design method is different in that it is totally goal oriented - improve industrial performance. Clausing doesn't make any claims to be scientific in his approach, but implicitly claims that it works better than any other approach (that he is aware of) in an industrial setting. Clausing's contributions are mainly: 1) analyzing pragmatically the different design methods and placing them in the context of the total development process in a corporation, and 2) integrating the best design theories and methods with management and strategy to form a cohesive approach to design. By its evolutionary nature, this program

will continually change and improve as Clausing continues to search for new components that complements or improves his approach.

#### A3.2.3.3 Discussion

Both Altshuller and Suh's established a set of principles or axioms from which a variety of methods or algorithms to solve specific problems can be developed. Both also attempt to define what is a "good" solution or design, *and* interestingly, they arrive at virtually the same definition independently of one another!

Based on their principles or axioms, Altshuller and Suh have developed different but complementary approaches to arrive at a good design. Altshuller developed a system of methods to separate contradictory properties through clever synthesis and integration of parameters, while Suh developed a metric and analysis rule that warns the designer if he or she is creating a bad design.

One of the complementary properties of these two methods is that while Suh's analysis method points out when interdependencies are harmful and can easily visualize interdependencies between *several* variables, Altshuller's method lacks this property. However, once the conflicting interdependencies are identified, Altshuller provides a set of tools to resolve it—something Suh's method lacks. It has been proposed to use these methods in a complementary fashion, making use of their respective strengths, to further enhance these principled approaches to design. [Nordlund (1994)]

The approach that Hubka and Eder followed to develop WDK school is based on observation and systematizing what designers already do, complemented with a theory for modeling technical systems. This, appears to yield an after-the-fact approach to design, that is, it will provide a scientific description of what the designers do—but no statement on whether this is the right thing to do. The models of the technical system will be used by the designer to describe how the technical system will function; however these models will not provide any fundamental reason why it will or will not work.

Furthermore, the WDK school of Hubka and Eder does not appear to define what constitutes a good design—something that is central to both Altshuller and Suh. Instead, Hubka and Eder use the ISO 9000 definition of quality, "the totality of those properties and characteristics of a product or an activity that relate to its suitability to fulfill the stated requirements". [Hubka and Eder (1992) p. 21] The quality is evaluated against a set of criteria, and a composite quality number (representing e.g., technical and economical value) is calculated. Hubka and Eder recognize that this method has problems, but implicitly defend it in that all methods have their disadvantages. In comparison to Suh and Altshuller it appears as a weakness of this school to lack a clear definition of what constitutes a good design.

Clausing provides an important technology transfer link from the academic research community to the community of users in industry. Clausing uses an unique approach in this research field in applying a concept selection method, combining useful features from several different methods to create a holistic method, then using benchmarking to ensure that the new method indeed is superior.

Clausing's way of evolving a design approach requires a number of properties: a broad network of contacts that can provide information on new developments; more focus on

pragmatic value than scientific value; and an open mind which is not committed to any individual component of the approach, but rather is willing to replace components with new ones that are better. Perhaps Clausing is the only researcher in this field who has found a way to satisfy the following challenge by Ullman: "If only we could use a sound design [method] to approach the problem of designing a theory....." [Ullman (1991) p. 801] Even if Clausing's approach by some would be called unscientific, it is nevertheless effective.

This differences in approaching the development of knowledge in this field is captured in the following statement where Sohlenius expands on a thoughts presented by Von Karman: "The engineer creates what has never been, the scientist analyses what is, and the engineering scientist analyses what is, imagines what should be, creates what has never been, analyses the results of the creation." [Sohlenius (1990)]

### A3.2.4 Process for selection of design tools by industry

In this section, the difference in perspective concerning design methods held by design researchers and by practitioners is discussed. Furthermore, a process is proposed by which practitioners can identify methods which meet their needs. Such a process implies that design researchers need to provide information more accessibly to industry to facilitate the selection of appropriate methods.

The perspective of researchers in academia must necessarily be different from that of practitioners in industry. In particular academia asks the question: how can knowledge be generated, or collected, to expand the body of work which constitutes engineering design? That is, design theories and methods are the product of design research, and researchers seek to *create, refine, and expand* these by adopting a means to evolve their work. In contrast, in industry design theories and methods are a means to achieve the end of product development. Thus, rather than create a new knowledgebase of engineering design, the strategy is to *select* from among existing design research those tools which will meet the objectives of the firms designers.

The procedure to be followed in selecting methods for use by industry is as follows: understand the design process used currently, formulate specific objectives (improvements) to be met in performing the design process, identify design tools/methods (products of design research) which could potentially meet these objectives, analyze the situation to identify conflicts (coupling) and synergies in the proposed process, and choose the best proposal.

This is the same process which is followed when design is not performed from a blank slate, but rather selection between existing design objects is performed. Here the important consideration is not the synthesis of new methods, but rather the selection of methods which realize specific objectives that have been formulated and which are compatible with each other.

## *A3.3 Summary*

In this chapter, some current design theories are classified according to their method of evolution:

- Generation of new knowledge
- Selection/use of existing knowledge

No single theory or method today covers all aspects of the product development activities. This chapter demonstrates, however, that the way to evaluate and select design research for use in a corporate setting is to identify the objectives which need to be satisfied for the company and then to select from among theories which have been developed in academia.

In order to make this possible two things are necessary: 1) the company articulates its objectives for product development and 2) the design theories and methods are presented in such a way that the company can match its needs with the respective capabilities of each theory and method. This requires advanced, systematized understanding of the design process within the company as well as a more informative label attached to the design theories and methods provided by academia.

# APPENDIX 4: FUNDAMENTALS OF AXIOMATIC DESIGN

## *A4.1 Introduction*

This chapter provides a basic introduction to the concepts of axiomatic design, and it places the work of this thesis into to the context of the larger body of axiomatic design research. This chapter is organized in 3 sections. The first, section A4.2, defines the important terms and concepts found in axiomatic design. The second, section A4.3, describes the diverse ways in which the theory has been applied and extended. Lastly, section A4.4 relates the work of this thesis to the existing, fundamental concepts of axiomatic design.

## *A4.2 Basic concepts of axiomatic design*[1]

At its most basic, axiomatic design is composed of 5 concepts. These concepts are domains, hierarchies, zigzagging, and the two design axioms.

The starting point for axiomatic design, as stated by Suh, is that "there exists a fundamental set of principles that determines good design practice". [Suh (1990) p. 18] These principles, the axioms, are to be used in a design process that consists of at least three activities: problem formulation, concept generation, and concept evaluation and selection. This section focuses on information about the design object and its generation within the practice of axiomatic design.

**The design process and design objects**

The *design process* is the set of activities whereby designers develop and/or select the means to achieve a set of objectives, subject to constraints. The design process may entail the creation of a new solution, the selection of an existing solution, or a combination of the two. A series of activities are performed by which the customers' perception of a design task is transformed into an output—the *design object*, which is any satisfactory solution to this task. The transformation occurs by means of designers working with design tools/methods, with their knowledge of discipline-specific information, and with a set of available resources.

### A4.2.1 Domains and mapping

During the design process, the task which is being addressed can be divided into four *domains*. The number of domains remains constant at four for all design tasks, but the nature of the design elements in each domain changes depending on the field of the problem.[2]

The four domains are generalized as the *customer domain*, the *functional domain*, the *physical domain*, and the *process domain*. Associated with each domain are the *design elements* it contains. In the order listed, the elements within each domain are *customer needs* (CNs), *functional requirements* (FRs), *design parameters* (DPs), and *process variables* (PVs). Furthermore in addition to these elements that are each specifically associated with a particular domain, constraints on the design task can also exist. *Constraints* are a specification of the characteristics that the

[1] Parts of this section are adapted from [Harutunian, et al. (1996)]. For a more thorough explanation of axiomatic design see, of course, [Suh (1990)].

[2] Gebala and Suh [Gebala and Suh (1992)] list examples of the breakdown of tasks into the 4 domains.

design solution must possess to be acceptable to its customers and to the company designing it.

*Functional requirements* are defined as the minimum set of requirements that completely characterize the design objectives for a specific need. [Suh (1990) p. 38] These FRs must be specified in a "solution-neutral environment" in terms of the functions to be achieved, *not in terms of particular solutions. Design parameters* are defined as the set of elements of the design object that have been chosen to satisfy the FRs.

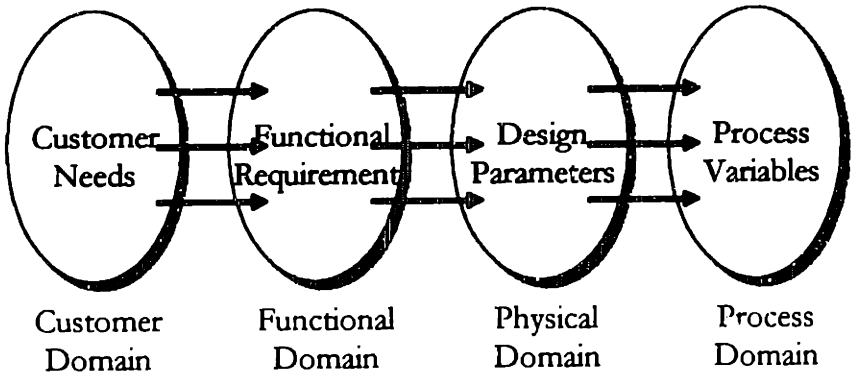


*Figure A4-1. Design domains*

The four domains are shown in figure a4-1. AD provides guidelines consisting of axioms, theorems, and corollaries. They specify the relationships that should exist between the FRs and the DPs of a design. These guidelines answer the question—will a set of DPs satisfy the FRs in an acceptable manner? These relationships also hold between DPs and PVs. The relationships between CNs and FRs, however, are more loosely structured.

### A4.2.1.1 Hierarchies

The design process progresses from a system level to levels of more detail. It progresses from systems to subsystems to assemblies to parts to part features. This may be represented in terms of a design hierarchy. The decisions about the design object are structured in three of the domains in a hierarchical manner, and hierarchies exist for any design object in each of the domains: functional, physical, and process. Thus an FR hierarchy, a DP hierarchy, and a PV hierarchy exist for a design object. The information in the customer domain is an exception and cannot be structured as rigorously.[3]

Domains, mapping, and hierarchies provide a structure for information about the design decisions that have been made. The framing of design tasks in this way enables the identification of regularities in design decisions.

### A4.2.1.2 Zigzagging

The decisions that are made at higher levels of the design hierarchies affect the statement of the design tasks at lower levels. The designers go through a process in which they zigzag between domains—functional, physical, and process—in decomposing the design problem.

---

[3] Strictly speaking customer needs *can* be structured *in a hierarchy*, however, they *do not necessarily need to be.* If they are so structured into a tree, it does not correspond to the other trees: FR, DP, or PV.

At a given level of the design hierarchy, a set of functional requirements exists. Before these FRs can be decomposed, the corresponding design parameters must be selected. Once a functional requirement can be satisfied by a corresponding design parameter, that FR can be decomposed into a set of sub-requirements, and the process is repeated. Zigzagging between the functional and the physical domains is illustrated in figure a4-2.

For example, in designing a transportation system for a city, if the top-level solution is the use of personal automobiles, then the further decomposition of the design task will be very different from the situation in which a mass-transit system is selected.

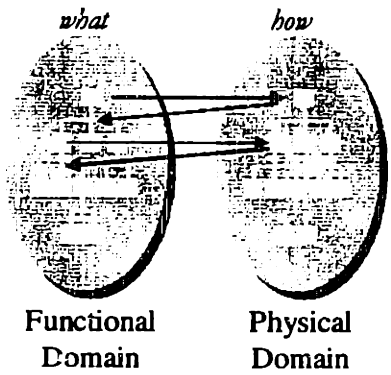


*Figure A4-2. Decomposition by zigzagging*

Designer should realize what choices they make, their options should be identified, and a good solution selected. The criterion for evaluation should be—is this option the most likely one to provide a satisfactory result? The design axioms, combined with the designers' knowledge, are a way to answer this question at early—even conceptual—stages of the design process. The designers follow the zigzag approach, checking the correctness of the design at each level, until they have decomposed the problem to a point where the solutions to the remaining sub-problems is known.

### A4.2.1.3 The design axioms: Independence and information

As described, the designers follow a design process in which decisions are made about a design object starting with high-level, system decisions and progressing to levels of increasing detail. In following this process—at each level of detail—the steps through which the designers progress can be described as in the Wilson model (see section 3.1): problem formulation, synthesis, and analysis. [Wilson (1980) The design axioms provide a criterion for analysis, particularly during conceptual design. The two design axioms are stated as follows [Suh (1990)][4]:

- *The Independence Axiom (First Axiom):*
  Maintain the independence of functional requirements.
- *The Information Axiom (Second Axiom):*
  Minimize the information content [of the design].

[4] The original reference on axiomatic design lists seven design axioms [Suh, et al. (1978)]. These were quickly reduced to the current two.

Once a set of FRs has been formulated and possible sets of DP alternatives have been synthesized, the two design axioms are applied to evaluate the proposed designs. The designers apply the Independence Axiom by using a design matrix. [Suh (1990)]

*A4.2.1.3.1 Evaluating the design matrix*

The design matrix (DM) shows the relationships between the FRs and DPs at one level of the design hierarchy. The design matrix–[A]–results from a design equation of the form shown:

$$\begin{Bmatrix} FRx.1 \\ \vdots \\ FRx.n \end{Bmatrix} = [Ax] \begin{Bmatrix} DPx.1 \\ \vdots \\ DPx.n \end{Bmatrix} \tag{A4-1}$$

The elements of the design matrix are determined from the set of equations [Suh (1990) p. 122]:

$$\begin{aligned} \Delta FRx.1 &= \frac{\partial FRx.1}{\partial DPx.1}\Delta DPx.1 + \cdots + \frac{\partial FRx.1}{\partial DPx.n}\Delta DPx.n \\ &\vdots \\ \Delta FRx.n &= \frac{\partial FRx.n}{\partial DPx.1}\Delta DPx.1 + \cdots + \frac{\partial FRx.n}{\partial DPx.n}\Delta DPx.n \end{aligned} \tag{A4-2}$$

where each

$$Ax(i,j) = \frac{\partial FRx.i}{\partial DPx.j} \tag{A4-3}$$

In a design matrix an X represents a strong effect by a DP on an FR, and an O indicates a weak effect, relative to the tolerance associated with the FR. There are three possibilities for the nature of the design matrix. It can be a matrix populated both above and below the diagonal, a triangular matrix, or a diagonal matrix. These are illustrated in design equations A4-4a, A4-4b, and A4-4c, respectively. A triangular matrix, as in equation A4-4b, is known as a *decoupled design.* A diagonal matrix, as in equation A4-4a, is an *uncoupled design.* Any other matrix, as in equation A4-4c, is known as a *coupled design.*

$$\begin{Bmatrix} FRx.1 \\ FRx.2 \\ \vdots \\ FRx.n \end{Bmatrix} = \begin{bmatrix} X & O & \cdots & O \\ O & X & & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & X \end{bmatrix} \begin{Bmatrix} DPx.1 \\ DPx.2 \\ \vdots \\ DPx.n \end{Bmatrix} \tag{A4-4a}$$

$$\begin{Bmatrix} FRx.1 \\ FRx.2 \\ \vdots \\ FRx.n \end{Bmatrix} = \begin{bmatrix} X & O & \cdots & O \\ O & X & & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & X \end{bmatrix} \begin{Bmatrix} DPx.1 \\ DPx.2 \\ \vdots \\ DPx.n \end{Bmatrix} \tag{A4-4b}$$

$$\begin{Bmatrix} FRx.1 \\ FRx.2 \\ \vdots \\ FRx.n \end{Bmatrix} = \begin{bmatrix} X & O & \cdots & O \\ O & X & & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & X \end{bmatrix} \begin{Bmatrix} DPx.1 \\ DPx.2 \\ \vdots \\ DPx.n \end{Bmatrix} \tag{A4-4c}$$

The strength of off-diagonal terms in the design matrix is defined mathematically by Theorem 8. Given an FRx.i, the change in its value due to adjustments in other DPx.js must not exceed the designer-specified tolerance δFRx.i. [Suh (1990) p. 122] That is,

$$\delta FRx.i \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| \frac{\partial FRx.i}{\partial DPx.j} \Delta DPx.j \right| \tag{4-5}$$

In an uncoupled design, the FRs can be satisfied independently by means of the corresponding DPs. In a decoupled design, the FRs can be satisfied if the DPs are varied in the correct sequence. A coupled design has no guaranteed point where the FRs can be satisfied. A similar matrix captures the relationships between DPs and PVs.

*A4.2.1.3.2 Calculating the information content*

*Information content* has been defined by Suh as the log of the inverse of the probability of success of satisfying a function. [Suh (1990), Wilson (1980)]

$$I = \log_e \frac{1}{p_i} \tag{4-6}$$

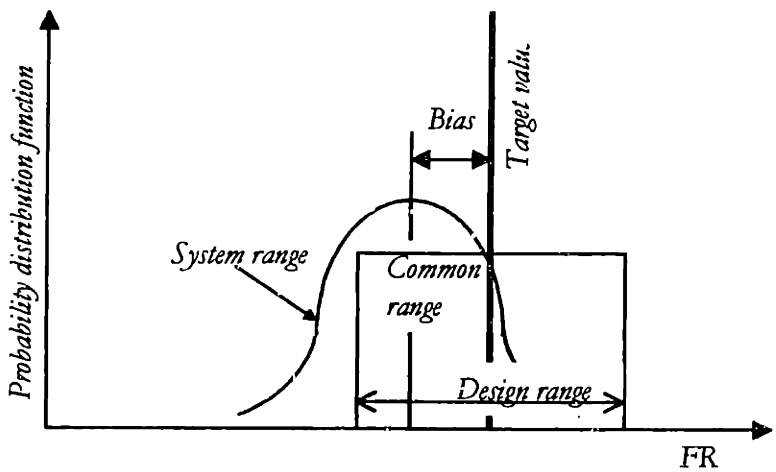


*Figure A4-3. Information content and probability of success*

In generating an FR, the designers define a desired target value for the FR. They also specify an appropriate tolerance region about this target value; this region is known as the *design range*, $r_d$. Each available design alternative is able to provide the desired FR within its system range. This *system range*, $r_s$, is the region in which the design alternative performs relative to the design range. The intersection of the system range and the design range is called the *common range*, $r_c$, shown in equation 4-7.

$$r_c = r_s \cap r_d \tag{4-7}$$

These three areas are shown in figure a4-3. The *probability of success*, labeled $p_s$ to indicate its basis on the tolerance of the FR, therefore, is defined as the ratio of the common range to the system range, shown in equation 4-8, and the information content, $I$, is the natural log of this as given in equation 4-6.

$$p_s = \frac{r_c}{r_s} \tag{4-8}$$

For uncoupled designs the FRs may be considered independent variables. Therefore the total information content for a set of $n$ FRs in an uncoupled design is equal to the sum of the sum of information contents for each of the $n$ FRs:

$$I_{total} = \sum_{i=1}^{n} I_i = \sum_{i=1}^{n} \log_e \frac{1}{p_i} \tag{4-9}$$

## A4.2.2 Use of AD concepts

Axiomatic design provides a framework for describing design objects. This framework is consistent for all types of design tasks and at all levels of detail. Therefore, different designers can quickly understand the relationships between the intended functions of an object and the means by which the functions are achieved.

Furthermore, axiomatic design theory encompasses a design process that has several benefits for the creation of designs. First, the design axioms provide an analysis tool that is unique among design theories. The design axioms provide a means for evaluating the quality of proposed designs so that design decisions may be made on a *rational* basis, supported by easily understood analytical results. Second, the design process used guides designers to consider alternatives at all levels of detail and makes choices between these alternatives explicit.

### *A4.3 Advanced topics in AD*

Most research in axiomatic design has been in its application. Because axiomatic design is generally applicable to all types of design tasks, rapid growth in the numbers of areas in which it has been applied is only natural. AD can be applied in the same way to problems in many fields: product design, manufacturing process design, software design, system design, business planning, and organizational design. Suh's second AD book is structured around several application areas and gives numerous examples in which axiomatic design has been applied. [Suh (1999)

### *A4.4 Relationship of this work to AD*

The three types of statements that are made in axiomatic design are axioms, theorems, and corollaries.[5] Within axiomatic design, an *axiom* is a statement that is "believed to be universal in [its] application" but cannot be proven. [Suh (1990) p. 15] *Theorems* and *corollaries* are

[5] Suh presents 8 corollaries to the axioms and 16 theorems in [Suh (1990)]; additional theorems are given in [Suh (1995b)], [Suh (1995c)], and [Suh (1999)].

propositions that follow from axioms or other propositions that have been proven. Both are valid if their "referent axioms and deductive steps are valid". [Suh (1990) p. 47] Furthermore, a theorem is "equivalent to a law or principle", and corollaries "can be applied to actual situations more readily than the [design] axioms" and thus "may be...called design rules". [Suh (1990) pp. 47, 52]

These definitions are consistent with common usage. According to Webster's an *axiom* is "a maxim widely accepted on its intrinsic merit" or "an established rule or principle or a self-evident truth"; a *theorem* is "a formula, proposition, or statement in mathematics or logic deduced...from other formulas or propositions"; and a *corollary* is "a proposition inferred immediately from a proved proposition with little or no additional proof". [Merriam-Webster's (1996)]

According to the Oxford English Dictionary an *axiom* is a "proposition that commends itself to general acceptance; a well-established or universally-conceded principle; a maxim, rule, law"; a *theorem* is a "universal or general proposition or statement, not self-evident (thus distinguished from an axiom), but demonstrable by argument"; and a *corollary* is a "proposition appended to another which has been demonstrated, and following immediately from it without new proof". [OED (1989)]

The distinction between a theorem and a corollary is that a theorem is less self-evident and must be derived from a set of starting premises. A corollary, on the other hand, naturally follows from a theorem that has been derived.

The three areas of contribution of this thesis are the roadmap and activities, the guidelines, and the tools for system design.

The existence of the activities in the roadmap of the design process given in appendix 2 may be considered a corollary to the design axioms and the body of knowledge within axiomatic design.

This can be seen by considering the options available to the designers as they follow a decision-making strategy based on the axioms. For example, when the designers analyze an existing design and find that it is coupled, they have two options available: Either the designers can decouple the design through developing and/or selecting new design parameters to satisfy the functional requirements, or the designers can leave the design as it is and seek through optimization to find an operating point were the design is less sensitive to the coupling. (Obviously the latter choice is undesirable and in most cases unacceptable.) By examining the options available to the designers in their decision making, the activities identified in the roadmap of the design process, shown in figure 3-3 and A2-1, follow from the framework of axiomatic design theory. Likewise the roadmap for decomposition activities is also a corollary in the same way.

The guidelines for decomposition activities may be considered theorems. They do not follow immediately from the axioms, but may be derived by considering the axioms, the options available to the designers, and the decisions that will lead to the best designs using the least resources. For example, in the activity of generating a set of sub-FRs, the question is how the designers can do this in such a way that leads to designs that satisfy the independence axiom and minimize the information content of the design. The starting premises are the basic concepts of axiomatic design described above in section A4.2. Following the sequence of the design matrix in generating these sub-FRs will lead to less iterative design processes.

Consider a design matrix as shown in equation 4-10 below; each off-diagonal term *Ax(i,j)* corresponds to the effect on *FRx.i* of changing *DPx.j*. Specifically if a DP may be adjusted to satisfy its corresponding FR and the other FRs at that level of the design hierarchy are unaffected, then the requirement for functional independence is satisfied as per theorem 8.

The interpretation of the design matrices used here is that an element of the design matrix *Ax(i,j)*, which is given by $\frac{\partial FRx.i}{\partial DPx.j}$, corresponds to asking the following question: Does a change in *DPx.j*–consistent with fulfilling *FRx.j*–affect *FRx.i*?

$$\begin{Bmatrix} FRx.1 \\ \vdots \\ FRx.i \\ \vdots \\ FRx.n \end{Bmatrix} = \begin{bmatrix} Ax(1,1) & \cdots & Ax(1,j) & \cdots & Ax(n,1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Ax(i,1) & \cdots & Ax(i,j) & \cdots & Ax(i,j) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ Ax(n,1) & \cdots & Ax(n,j) & \cdots & Ax(n,n) \end{bmatrix} \begin{Bmatrix} DPx.1 \\ \vdots \\ DPx.j \\ \vdots \\ DPx.n \end{Bmatrix} \quad (4\text{-}10)$$

If *DPx.j* changes, for example due to a change in the technology used in the product or due to a change in the customer needs concerning the value of *FRx.j*, then the following argument applies: Because of the nature of the design, *DPx.i* has been chosen to fulfill *FRx.i*; therefore, if the desired value of *FRx.i* has not changed, then the value of *DPx.i* will need to be changed to counteract the effect that changing *DPx.j* has on *FRx.i*.

Equation 4-11 illustrates a design situation in which a change made to *DPx.1* requires no change in either *DPx.2* or *DPx.3*. However, if a change is made to *DPx.2* (for example, because the target value of *FRx.2* changes), then a change *will be* required in *DPx.3*, assuming that the target value of *FRx.3* does not change. This shows that, as described by Suh, there is a preferred sequence to changing the DPs. [Suh (1990)] These considerations apply to both changes in *DPx.j* itself and to changes in its attribute values. This includes changes made by the designers during conceptual design.

$$\begin{Bmatrix} FRx.1 \\ FRx.2 \\ FRx.3 \end{Bmatrix} = \begin{bmatrix} X & O & O \\ O & X & O \\ O & X & X \end{bmatrix} \begin{Bmatrix} DPx.1 \\ DPx.2 \\ DPx.3 \end{Bmatrix} \quad (4\text{-}11$$

Decomposition within one node of the design hierarchy *does* have a sequence that is dependent on the nature of the design matrix at that level. All changes to DPs necessarily invoke the type of questions considered above—even such minor changes as changes in the values of DP attributes. Therefore it follows that the decisions made in generating sub-FRs correspond to "changes" in the sense described above. Consider the generation of a set of sub-FRs for *DPx.2*. In this activity, the parent *DPx.2* and the design matrix stay the same; however, additional attributes of the DP are defined and for leaves their values are set. The off-diagonal matrix element *Ax(3,2)* indicates that there is a relationship between the decisions the designers make about *DPx.2* and the satisfaction of *FRx.3*. Changes to *DPx.2* including the generation of a set of new sub-FRs have the potential to affect FRx.3. Therefore, *FR-DPx.3* should be decomposed second to compensate for these potential effects.

### *A4.5 Summary*

This chapter provides a basic introduction to the concepts of axiomatic design. Axiomatic design provides a framework for describing design objects. This framework is consistent for all types of design tasks and at all levels of detail. Axiomatic design theory encompasses a design process that has several benefits for the creation of designs. First, the design axioms provide an analysis tool that is unique among design theories. The design axioms provide a means for evaluating the quality of proposed designs so that design decisions may be made on a *rational* basis, supported by easily understood analytical results. Second, the design process used guides designers to consider alternatives at all levels of detail and makes choices between these alternatives explicit.

# APPENDIX 5: FUNCTION STRUCTURES OF PAHL AND BEITZ

## *A5.1 Introduction*

The work of Pahl and Beitz is considered a well-known and thorough approach to design. A part of their approach is the development of function structures. Because these function structures are ostensibly similar to FRs in axiomatic design, this section explores the relationship between the work in this thesis and the work of Pahl and Beitz. How do the ideas compare? Were the same guidelines developed from different starting assumptions? If not, are the approaches complementary or contradictory?

### *Aim of Pahl and Beitz's work*

The goal of the work of Pahl and Beitz is to provide a comprehensive *design methodology*, "a concrete course of action for the design of technical systems that...includes...plans of action to link working steps and design phases...; strategies, rules and principles to achieve general and specific goals...; and methods to solve individual design problems or partial tasks." [Pahl and Beitz (1996) p. 10] Their approach is to "combine various methods in a coherent and practicable way". [Pahl and Beitz (1996) p. 25]

### *Scope of function structures*

The overall design process presented by Pahl and Beitz is analyzed elsewhere (see section 2.3.2); here a portion of their theory is discussed. Created during the conceptual design phase, a *function structure* is a "meaningful and compatible combination of sub-functions into an overall function..., which may be varied to satisfy* the overall function." [Pahl and Beitz (1996) p. 31]

The functions comprising the function structure may be classified according to two types: main and auxiliary functions. The functions are connected to describe three types of flows through a system: energy, material, and signals. An schematic is shown in figure 5-1.

* Note that the word *satisfy* is used differently here than in axiomatic design. In axiomatic design, an FR is *satisfied* by its DP, not by its sub-FRs.

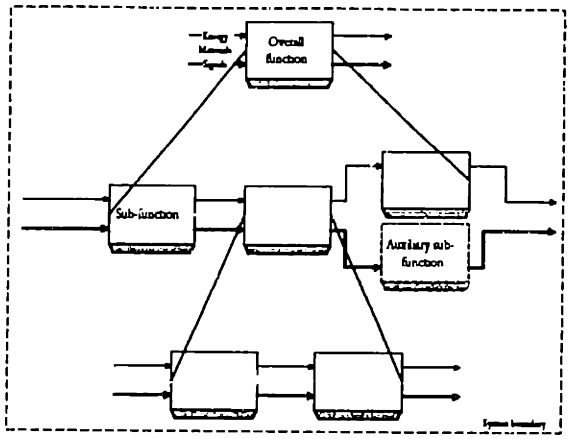


*Figure 5-1. Function structures showing operational flows of materials, energy, and signals*

## *A5.2 Compare and contrast Pahl and Beitz versus AD*

Several observations can be made about the relationship between function structures within the work of Pahl and Beitz and the overall approach of axiomatic design.

1. Pahl and Beitz agree with axiomatic design about the nature and definition of functions. They are usually stated as verb-noun pairs.
2. Pahl and Beitz recognize the need for solution neutrality: "At this stage [of defining functions] there is no need to stipulate what solution will satisfy this kind of function. The function thus becomes an abstract formulation of the task, independent of any particular solution." [Pahl and Beitz (1996) p. 31] This is seen as a desirable, but an unobtainable ideal, however, because "function structures are seldom completely free of physical or formal presuppositions, which means that the number of possible solutions is inevitably restricted to some extent". [Pahl and Beitz (1996) p. 160]

In these two ways the theory of Pahl and Beitz is consistent with axiomatic design. There are also important ways in which it differs.

1. Pahl and Beitz lack a clearly delineated concept of domains. For example, the closest that their theory comes to DPs is the identification of "working principles" and "working structures". Furthermore, since there is no concept of domains, there is also no information produced that corresponds to the relationships contained within design matrices in axiomatic design.
2. Also, the concepts of hierarchies and zigzagging do not exist in the theory. An example is given of developing a function structure for the overall function to "measure and indicate quantities of liquid". Once a set of sub-functions is developed, Pahl and Beitz note, "The initial search for a solution will focus on this sub-function [to 'receive signal'] and the solution to this will largely decide to what extent individual sub-functions can be changed round or omitted." [Pahl and Beitz (1996) p. 161] The functions to be changed or omitted are identified at the same time as at the function to "receive signal". Clearly this is different than zigzagging since the selection of a DP at this level is said to affect the existence of other FRs *at this same level.*

3. Functions are classified as main or auxiliary functions. "*Main functions* are those sub-functions that serve the overall function directly, [and] *auxiliary functions* are those that contribute to it indirectly." [Pahl and Beitz (1996) p. 32] This is distinct from axiomatic design in which all sub-FRs are important to providing their parent FR and thus must all be satisfied by their sub-DPs, with no weighting factors.

4. Pahl and Beitz emphasize the operational flows and the ability to identify minor changes to a design; axiomatic design is more concerned with good and possibly radically creative design decisions. For example, Pahl and Beitz describe the development of variants, "The relationship between sub-functions and overall function is very often governed by certain constraints, inasmuch as some sub-functions have to be satisfied [performed] before others. On the other hand, it is usually possible to link sub-functions in various ways and hence create variants. In all such cases, the links must be compatible." [Pahl and Beitz (1996) p. 31]

### *A5.3 Comparison of specific guidelines*

Pahl and Beitz advocate several specific guidelines, which may be adapted for use within axiomatic design. Their ideas are presented following a different structure of design phases, rather than decisions made, but several guidelines may be separated out of the text.

For the purposes of this discussion, the functions in function structures are assumed to correspond to functional requirements in axiomatic design terms. Significant differences exist, of course, in the way that a function structure is developed as compared with the development of an FR tree. This is due especially to the lack of explicit consideration of domains and hierarchies as noted above.

The guidelines of Pahl and Beitz are therefore evaluated in terms of their usefulness in generating sub-FRs.

**Guidelines in Pahl and Beitz**

Pahl and Beitz present the following guidelines for the development of function structures.

1. "From a rough structure, or from a structure obtained by the analysis of known systems, it is possible to derive further *variants* and hence to optimize the solution." [Pahl and Beitz (1996) p. 160]

The emphasis in this guideline is on producing variants in the way functions are carried out. The idea that functions may be carried out in different sequences of operation is consistent with axiomatic design, and the methods that Pahl and Beitz describe are useful for the designers to consider thoroughly the different alternatives. In axiomatic design, however, creative synthesis of DPs, more than simple rearrangement of FR operation, is considered necessary to create an optimal design.

2. "It is useful to distinguish between main and auxiliary functions." [Pahl and Beitz (1996) p. 32]

As noted already, there is no distinction in axiomatic design between main and auxiliary functions, therefore this guideline is inconsistent. Furthermore, the use and definition of auxiliary functions is unclear. For example, in "some problems...*auxiliary flows* have a *crucial* bearing on the design." [Pahl and Beitz (1996) p. 154] The confusion between these functions probably arises due to the lack of a hierarchy established through zigzagging.

3. "[U]sing generally valid functions...at a higher level of abstraction...leaves open all possible solutions and makes a systematic approach easier." [Pahl and Beitz (1996) p. 34]

This guideline is not consistent with axiomatic design because an objective in axiomatic design is to generate FRs at a proper level of abstraction, consistent either with the customer needs and/or with the decisions that have been make for the higher-level FRs and DPs. Furthermore, Pahl and Beitz are unclear in their advise about the merits of using generally valid functions or task specific functions, that is, when to use one or the other. For example, they state that the use of "generally valid sub-functions...may be helpful...because they lead to the discovery of task-specific sub-functions". [Pahl and Beitz (1996) p. 151]

4. "The optimum method of breaking down an overall function—that is, the number of sub-function levels and also the number of sub-functions per level—is determined by the relative novelty of the problem and also by the method used to search for a solution." [Pahl and Beitz (1996) p. 150]

This guideline states that a function structure is dependent on novelty of problem. Designers who have much experience with a specific design task are therefore expected to develop a necessarily different function structure than those who are working on a project that is new to them. This conflicts with the representation of systems within axiomatic design. Different designers should be able to create and use similar if not identical descriptions of the same design.† This is especially important in cases in which a design already exists and the designers need to understand the impact of potential design changes. While different designers having different experiences may make analyzing and/or designing a system more or less easy, nevertheless, axiomatic design should promote consistency in understanding the design task.

5. "If existing assemblies can be assigned directly as complex sub-functions, the substitution of the function structure can be discontinued at a fairly high level of complexity. In the case of new assemblies or those requiring further development, however, the division into sub-functions of decreasing complexity must be continued until the search for a solution seems promising." [Pahl and Beitz (1996) p. 151]

This provides a guideline for when to quit developing the function structure. The application of this guideline is unclear because the authors do not state a criterion for the designers to know when an existing design may be assigned directly to a sub-function.

6. "It is...possible to begin with sub-functions whose inputs and outputs cross the assumed system boundary. From these we can determine the inputs and outputs for neighbouring functions, in other words, work from the system boundary inward." [Pahl and Beitz (1996) p. 156]

This guideline specifies that functions are to be defined inward from the boundary of the system. This could be useful (and consistent with this work) in generating a set of sub-FRs. In generating a particular set, consider the inputs and outputs of the parent FR-DP pair and work inward in conceptualizing the sub-FRs.

---

† Currently this may be considered an imperfectly realized ideal toward which the system template serves as an aid. (See section 3.1.2.)

This guideline is not consistent with this work, however, if it is interpreted to be an instruction for sequencing the decomposition. There is no reason to expect that the design matrix at some level of the design hierarchy is necessarily such that decisions about the designs of the FR-DP pairs at the operational extremes of the system (that is, those that are next to the system boundaries) should be made first. For example, in a track machine, wafer motion occurs both first and last in the operational flow of the wafer through the system; however, the design matrix for such a system indicates that the processes which make up photoresist coating and developing should be detailed first, before decisions are made about the transportation subsystem.

7. There exits a particular sub-function that is identified as being "[a]n important sub-function that must be satisfied first, and on which the working principle of which the others depend". [Pahl and Beitz (1996) p. 161]

This guideline may be considered consistent with this work. It describes the proper role of hierarchies and zigzagging which should be incorporated into the development of the function structures of Pahl and Beitz.

8. Pahl and Beitz propose several sets of functions that are commonly found together in specific applications.

This idea is consistent with that of templates, such as the system template and command and control algorithms. As an example, the controller for a technical system needs to perform the sub-functions "operate", "indicate", "detect", and "activate", interacting with both an operator and a piece of hardware to be controlled. [Pahl and Beitz (1996) p. 157]

### *A5.4 Summary*

In summary, the guidelines presented in this thesis have a broader scope than those presented by Pahl and Beitz for the construction of function structures. The differences between the directions of Pahl and Beitz and the guidelines presented in this thesis can largely be traced to differences in modeling the design. This work starts with the concepts of axiomatic design. In contrast, the lack of recognition of these concepts in Pahl and Beitz—especially domains and zigzagging—leads to instructions that are not especially useful or complementary to this work.

## Acknowledgments

This thesis owes a large intellectual debt to the colleagues I have had the privilege of working with. I was introduced to axiomatic design by Chul Bum Park. He told me about the subject when I first arrived at MIT as I interviewed with the Microcellular Plastics Research Group about an available RA position. I accepted that position and completed my Master's studies under the direction of Professor Nam P. Suh.

Although I changed my research area, I continued working with Professor Suh for my Ph.D. This has proven to be an extremely valuable learning experience for me as well as one with numerous enjoyable qualities. I would be hard-pressed to exaggerate his intuitive grasp of physics, his creativity, his speed in problem solving, and the strength of his intellectual convictions. In him these qualities combine with a desire to interact with his students and a passion for big-picture research topics. I am grateful to him for the opportunity to work on interesting research topics and for the freedom he allowed me to follow my own path.

In addition to Professor Suh, I could not ask for a committee composed of a more distinguished set of design researchers. Each has contributed to my work in his unique way. Professor David Cochran was always available to listen to the work I was up to and to bounce ideas off of. Dr. Don Clausing gave invaluable, constructive comments on the last drafts of my thesis. Professor Steven Eppinger kept my committee meetings on track and forced them to be productive.

Another person who largely impacted the shape of this research is my good friend Mats Nordlund. This thesis would have an altogether different look, I'm sure, if I had not gotten to know Mats as I was starting my Ph.D. The process of establishing a research direction in axiomatic design and persevering to realize my thesis would have been more challenging, if not altogether impossible, without the views of design and design research I picked up from him.

I've been around MIT so long that I've worked with so many good people. Dan Frey, Vigain Harutunian, Shaun Abrahamson, Brian Lim, and Tae-jung Kim each worked for a time in the Axiomatic Design Group. Shaun was always able to bring an interesting perspective to discussions of AD research topics and was often available with an encouraging word for my ideas. I'm glad he found a place in which he was happy to finish his Master's with David Wallace.

The current members of the research group deserve much credit for their support: Jason Hintersteiner, Tae-Sik Lee, Yun Kang, and Jason Melvin. I appreciate the camaraderie within the group, the many drafts of my thesis read, suggestions made, etc. I wish you all the best of luck as you continue to work in axiomatic design.

Beyond MIT students, the group at various times was privileged to have an international flavor: Ronald Popma, Hiroshi Igata, Kazuhiko Umezawa, Dan Lindholm, Sudhakar Reddy, Sung-Hee Do, Cord Becker, Fritz Seemann, Marco Lanza, and Woo-Seok Choi. Two other students I should also thank are Armin Schulz and Pär Mårtensson. Many have been friends as well as co-workers.

My time as a Ph.D. student was supported through generous funds from Draper Laboratory and Silicon Valley Group. At Draper I would like to single out Peter Sebelius for his

assistance; at Thermco, Bob Parvin, Mike Nogues, Waldo Rogers, Scott Thomas, and Jason Zhou; and at SVGL, Dan Cote, Larry Oh, Glenn Friedman, and Joe Laganza.

I should also thank the many staff at MIT who been friendly and helpful to me as I was a grad student. This includes Patsy Sampson, Marion Gross, Pat McCosco, and Pam McCarthy. I should single out Leslie Regan who has been a source of grace and has been understanding and accommodating beyond-the-call-of-duty.

I also owe a debt to the communities at MIT of which I have been a part. The first was Ashdown that was a welcoming escape from the lab, in addition to its initially appealing qualities as cheap and convenient housing. I lived on the 4th floor all my years at MIT and got to know many friends in the kitchen. I suppose that, at some point, the recurring fight to save Ashdown from the fate of becoming an undergrad dorm will be lost. When that happens, MIT will have suffered an irreparable loss.

The other community of which I became a part is the MIT Graduate Christian Fellowship. To the extent that I understand the subject now, it was in GCF as well as at Park Street Church that I learned how being a Christian—my previously vague commitment to following Christ—was supposed to impact my life. To name a few of the friends I want to thank: Grace Foster, Shane and Jee-Hoon Krska, Araz Inguilizian, Cedric Logan, Tom Lee, Steve and Lisa Reid, Kristie and Christoph Mertz, Pauline Bennett, Scott Socolofsky, Steve Mascaro, Debbie Lightly, Andrew Crabtree, Inn Yuk, Alana Wiens, Mike Crockett, and Yann Schrodi. I am indebted to too many people for the creation and completion of this work, and for maintaining some semblance of my sanity, to adequately thank them all.

I am grateful to God for bringing me to MIT and Boston and for seeing me through the process of getting my degree. While it's hard to accept sometimes, he has a plan for me and knows what the future holds even—or especially—when I do not. In the words of David, "All the paths of the Lord are faithful love towards those who honor the covenant demands." (Psalm 25:10)

My parents and sister have also been supportive in my struggles: academic, romantic, or whatever they happen to be at any given time. Thank you for being there to listen and for your words of encouragement.

## Biographical information

> Charleston has a landscape that encourages intimacy and partisanship. I have heard that an early inoculation to the sights and smells of the [South] Carolina lowcountry is an almost irreversible antidote to the charms of other landscapes, other alien geographies. You can be moved profoundly by other vistas, by other oceans, by soaring mountain ranges, but you cannot be seduced. You can even forsake the lowcountry, renounce it for other climates, but you can never completely escape the sensuous semitropical pull of Charleston and her marshes. It is...one of those cities that never lets go, that insinuates its precedence by the insistent delicacy of its beauty.—Pat Conroy, *The Lords of Discipline*, p. 16

Derrick Tate has studied prominent approaches to design in America and Europe and applied these ideas to design tasks in many fields: robotics, software, product design, corporate strategy, and manufacturing processes. He has collaborated with and taught engineers from several industries: semiconductor manufacturing, financial services, automotive, and R&D. His research interests include the design process, the integration of design theories into industrial practice, and the use of decomposition in system design.

Finally, we arrived. The Product Ideation and Design Facility was a large wedge-shaped room stuffed floor to ceiling with instrument panels throbbing with electric life, glittering with lights and luminous screens and flashing dials. Arthur sat on a bench near Carl, who was hunched over what appeared to be some sort of computerized drafting board–a wide flat screen crawling with moving diagrams and charts.

...

"How the hell did he figure out how to work the equipment?"

"Oh it's not so hard as you might expect." Arthur rose, walked over, and peered over Carl's shoulder. "In this plant, in its day, engineers were looked upon as artists. They really didn't need to know much about engineering. Here, machine intelligences supply all the data, all the formulae, all the know-how. They do all the dirty work. The only thing that organic brains can supply is creativity. That's what Carl's doing. He's telling the machines what he wants and what he wants it to do, and the machines are helping him design it. And if the design is judged a worthy work of art, they just might build a prototype model."

"That's really something," I said. And it was, it was.

...

After lunch the plant foreman spoke to us. "We have begun production of the prototype. Would you like to observe?" He sounded a lot like the design chief, and I suspected that the latter was merely a subsystem of the former.

Would [we] like to observe, yes.

We all boarded another robocart and swung out into the plant.

The place had come to life. We rode for an hour through the throbbing heart of technological wizardry. What had been hulks of dead machinery now flashed and sparked, whirred and hummed, chimed and beeped and thrummed and sang, while pink and violet electrical discharges leaped between giant coils, translucent tubing glowed and pulsed, luminescent motes swam inside huge transparent spheres, and veils of energy fluttered in the air overhead like auroral displays.

"Goddamn Frankenstein movie," was Carl's reaction.

At last we came to a large quiet empty chamber. We got off and waited. Before long, the far wall retracted, and two robots hauled the prototype out onto the showroom floor. — DeChancie, *Paradox Alley*, pp. 125-127.