DESIGN AND RECONFIGURATION OF MANUFACTURING SYSTEMS IN AGILE MANUFACTURING ENVIRONMENTS

Shamil F. Daghestani

Thesis Submitted to the Faculty of
Virginia Polytechnic Institute and State University
In partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE

in

Industrial and Systems Engineering

Dr. John P. Shewchuk, Chair
Dr. Kimberly P. Ellis
Dr. Subhash C. Sarin

December 1, 1998
Blacksburg, Virginia

Keywords: Product Life-Cycle, Design/Reconfiguration Strategy, Agile Manufacturing

DESIGN AND RECONFIGURATION OF MANUFACTURING SYSTEMS IN AGILE MANUFACTURING ENVIRONMENTS

Shamil F. Daghestani

(ABSTRACT)

Agile manufacturing has become a topic of great interest over the past several years. The entire domain of modeling and analyzing different types of agile manufacturing environments and systems, however, remain largely unexplored. The objective of this research is to provide fundamental insight into how manufacturing systems should be designed and reconfigured over time in order to cope with different agile manufacturing environments. To achieve this objective, three approaches are developed and integrated into one simulation-based model. The first approach is used to model different agile manufacturing environments. The second approach is used to define various ways in which manufacturing systems can be designed and reconfigured (i.e., design/reconfiguration strategies). The third comprises the cost and objective functions used to measure system performance when different design/reconfiguration strategies are used in different agile manufacturing environments. Based upon the assumptions adopted during this thesis, the experimental work performed suggests that despite the fact that agility incurs high costs, agile manufacturing systems are indeed necessary for certain manufacturing environments in which product life cycles are short yet demand per product type is high. Therefore, it is important in certain manufacturing environments to focus on reconfiguration in short periods of time, even at the expense of higher reconfiguration costs.

ACKNOWLEDGMENT

I dedicate this thesis to my parents and thank them for their support and patience throughout this research. I would not have been able work on my thesis without their moral and financial support to fund this research. Their helping hand would always reach out to help me during the most difficult times.

I also want to thank my thesis advisor Dr. John Shewchuk for his substantial amount of effort throughout this research, and my committee members Dr. Kimberly Ellis and Dr. Subhash Sarin for their valuable comments and feedback.

Also, many thanks to Samantha Franklin for her help and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Traditional mass and batch manufacturing environments have mainly been characterized by few product varieties, long time-to-market, long product life cycles, and product demand curves depicted by well-defined growth, maturity, and decline periods. This is because the low rate of product obsolescence causes product life cycles (PLCs) to be long and varieties to be clustered within small product families (Cordero, 1991). Long PLCs and small product varieties imply only few alternative products exist in the market. Consequently, this increases the likelihood for their demand and allows products to hold their initial selling price for relatively longer periods of time. After that, if products are successful, prices decline at relatively slow rates; and competitors can afford to enter the market later and still be able to secure their share of the profit. As a result, manufacturing facilities could afford to simply be able to produce few types of products.

As the 21st century approaches, the main focus of companies is to develop products and penetrate markets quickly, in order to acquire a high percentage of customers and dominate the market with little competition. This allows them to then charge premium prices. This approach becomes especially prevalent when a product with new technology is marketed. Millson *et al*. (1992) report that advanced technology products that come out six months late but on budget will reduce profit by 33% over a five-year product life span, whereas being on time but exceeding budget by as much as 50% reduces profit by only 4%. Therefore, the additional gains made due to early entrance into the market yields profits that by far outweigh the budgeted amount. Dominance in the early stages of a product yields a large portion of market share and can set the tone to implementing incremental advancements and staying ahead of pursuing competition (Gaimon and Singhal, 1992). Later during product life, competition starts entering into the market and prices start to decline.

Therefore, rapid changes in technology and increased customer preferences for product diversity are causing fundamental changes in mass and batch production environments. According to Cho *et al.* (1996) these changes include:

- Continually increasing product variety (more customized products).
- Continually decreasing time-to-market.
- Continually decreasing product life cycles (high rate of product obsolescence).
- Very steep growth and decline curves.

Manufacturing environments are becoming more dynamic and turbulent than ever before. Traditional manufacturing facilities, however, are not able to cope with such environments, as no single facility can be flexible enough to cope with such a large magnitude of change in products and production requirements.

As a result, new types of manufacturing facilities are needed for the emerging manufacturing environments. Such facilities must be able to be *reconfigured* over time, both quickly and easily, in order to cope with change. These are often referred to as *agile manufacturing facilities*, and the environments as *agile manufacturing environments*.

1.2 PROBLEM STATEMENT

Despite recent interest in agile manufacturing, little work has been done to quantitatively model different types of agile manufacturing environments (Boynton *et al*., 1993). In addition, little research has been conducted to investigate the design of agile manufacturing facilities. The vast amount of literature on capacity and resource planning assumes few product varieties and long life cycles extending over the entire production planning horizon (Lee and Zipkin, 1995). Traditional manufacturing facilities are not designed to be reconfigured quickly for new products. If they are to be reconfigured, very important cost tradeoffs need to be addressed.

Thus, the problem exists of determining how to design agile manufacturing facilities for different types of agile manufacturing environments. This consists of determining the strategy for establishing *how* the initial facility should be designed and

when the manufacturing facility should be reconfigured, in order to try and maximize long-term profit. Such strategies can be referred to as *design/reconfiguration* strategies. This differs from building a system while incorporating various levels of flexibility due to the fact that flexibility and agility are fundamentally different. As will be explained in Chapter 2, flexibility refers to the ability to change the system from within, whereas agility refers to the ability to change (reconfigure) the system from outside.

1.3 RESEARCH OBJECTIVE AND SIGNIFICANCE

The objective of this research is to study how manufacturing systems should be designed and reconfigured in order to cope with agile manufacturing environments. However, the entire domain of modeling and analyzing different types of agile manufacturing environments and developing design/reconfiguration strategies remains largely unexplored. Kulatilaka (1988) states that "dynamic features" have emerged in which switching between *operating modes* needs to be amply researched. These operating modes are in reference to situations including switching between the utilization of certain levels of capacity and processing capabilities, which are associated with operation and reconfiguration strategies of system capacity. This research will provide insight into the process of system design and reconfiguration in agile manufacturing environments by establishing some guidelines for agile manufacturing system design.

1.4 RESEARCH APPROACH

To study how manufacturing systems should be designed and reconfigured in agile manufacturing environments, an experimental approach is employed. Various ways in which manufacturing systems can be designed and reconfigured, i.e., design/ reconfiguration strategies, will be developed. The objective will be to specify the strategies in terms of a set of factors, each of which can be set either high or low, thus yielding a 2^k experimental design. For example, one factor could be the initial system size in terms of processing capabilities (i.e., few vs. many) – few capabilities mean more frequent system reconfigurations will be required and vice-versa. The goal is selecting the design/reconfiguration strategy (design point) to employ in order to maximize profit.

Simulation will be employed to determine the profit which results when a given agile manufacturing system (i.e., design/reconfiguration strategy) is utilized in a given agile manufacturing environment for a specified number of production periods. It is important to remember that such approach is to be undertaken for a manufacturing environment associated with a family of products in mind. One cannot, for example, assume that the best design/reconfiguration strategy for producing one type of product is also the best strategy for another type of products.

The 2^k experimental design will be implemented for three different types of agile manufacturing environments that will be presented in Section 5.1. This will allow us to see the effect to which design/reconfiguration strategies are environment-dependent. The above approach is implemented via eight steps. These steps are described as follows:

1. Develop a technique for modeling different agile manufacturing environments. Environments will differ from one another in terms of such attributes as the time between new product introduction, product life cycle length, etc.
2. Develop a technique for modeling different agile manufacturing systems. Agile manufacturing systems are considered to consist of two components.
   i. Physical processing facility.
   ii. System life-cycle strategies. These can be divided into two parts:
      - design/reconfiguration strategies.
      - operating strategies.
3. Determine how to calculate cost, revenue, and profit associated with design, operation, and reconfiguration of a given manufacturing system in a given agile manufacturing environment, for a specified period of time.
4. Integrate the models discussed in Steps 1, 2, and 3 to find the profit resulting from operation of a given agile manufacturing system, in a given agile manufacturing environment, for a given number of periods. Simulation modeling will be used for this purpose.
5. Identify three different types of agile manufacturing environments and model those environments using the technique developed in Step 1.

6. Develop a 2^k experimental design for establishing which agile manufacturing reconfiguration strategy results in the best performance (i.e., maximum total profit) for each agile manufacturing environment.
7. Perform experiments using the simulation model of Step 4.
8. Analyze results and draw conclusions.

1.5 THESIS OUTLINE

The remainder of this thesis is divided into six chapters. These chapters are described as follows:

Chapter 2 presents a literature review on traditional and agile manufacturing environments and previous work on modeling of manufacturing environments and product life cycles. In addition, a literature review on design of agile manufacturing systems is presented and discussed.

Chapter 3 presents the modeling framework: how agile manufacturing environments and agile manufacturing systems are modeled, and how cost, revenue, and profit are determined (i.e., research approach Steps 1, 2, and 3). For each, descriptions are given, assumptions are stated, and variables are defined. Furthermore, this chapter concludes with a discussion on model validation.

Chapter 4 describes the simulation model developed for determining the profit resulting from operation of a given agile manufacturing system (i.e., manufacturing system with specified design/reconfiguration strategy), in a given agile manufacturing environment, for a specified number of production periods (Step 4).

Chapter 5 presents the experimental design employed for the research. The three agile manufacturing environments are presented, the manufacturing system design variables to be used as factors are identified (and levels established), and values for non-factors are specified (Steps 5 and 6).

Chapter 6 presents and discusses the experimental results for each of the three environments investigated (Steps 7 and 8).

Chapter 7 presents concluding remarks and a cross-comparison among the three different manufacturing environments (Step 8). In addition, the contributions of the research are summarized and future research directions are suggested.

CHAPTER 2

LITERATURE REVIEW

2.1 TRADITIONAL MANUFACTURING ENVIRONMENTS

There are various books and periodicals devoted exclusively towards the subject of manufacturing environments. Those *environments* are sometimes appropriately labeled as *situations*. Yet, certain authors refer to them as manufacturing *systems*, as opposed to manufacturing *environments or situations* (Wien, 1992; Carravilla and De Sousa, 1995; Lee and Zipkin, 1995). On the other hand, others have reversed this situation; for instance, some refer to group technology (GT) system as a GT environment (Ashby and Uzsoy, 1995). This complicates matters when one attempts to study the implementation of certain types of manufacturing systems, such as flexible manufacturing systems (FMS), computer integrated manufacturing (CIM), and group technology in a specific manufacturing environment. Therefore, an attempt should be made to clarify what is commonly meant by a manufacturing environment and a system because this research attempts to model environments so that modeling results can help establish the most suitable system designs to be used.

Manufacturing environments are product-dependent. They are somewhat like a natural phenomena which evolve due to various forces at work in the marketplace. Attributes defining manufacturing environments include (Kingsman and Hendry, 1993; Brannon *et al.*, 1993; Donaldson *et al*., 1995):

1. Product characteristics. Features and functions of the product.
2. Product variety. Different types of products the company should produce.
3. Lead time. Period of time until the product enters the market.
4. Product life cycle (PLC). Length (or number of periods) and quantity a product is in demand over its life-cycle.

Such attributes are the defining factors of traditional and agile manufacturing environments. Traditional environments were identified by Bertrand and Muntslag (1993), Kingsman *et al.* (1993), and Samadhi and Hoang (1995) to be: *make-to-order* (MTO), *make-to-stock* (MTS), *assemble-to-order* (ATO), and *engineer-to-order* (ETO).

The means by which the company selects to deal with attributes of an environment and attempts to control the situation is referred to as a manufacturing system. As expected, there are a vast number of innovations and ideologies that qualify for this definition. This immense range can be in the form of equipment used, production techniques, certain methodologies or philosophies, etc. Therefore, a company can be operating under many systems at the same time.

In the literature reviewed, many authors present case studies and suggestions on which types of systems can be adapted with certain manufacturing environments. Bertrand and Muntslag (1993) acknowledge that each distinct type of manufacturing situation (environment) requires a different production control system. It was previously explained how some authors have a way of interchanging synonyms of what they mean by a system or an environment. However, it should be noted that close review of their work showed that they all had the same perspective in mind.

2.2 AGILE MANUFACTURING ENVIRONMENTS

The term *agile manufacturing* has been used by many in academia and industry as the solution to dealing with the ever-increasing turbulent and dynamic market going into the 21st century (Cho *et al.*, 1996). Unfortunately, terms associated with *agility* have been used in so many ways that they have lost much of their original distinction (Richardson, 1996). Adamides (1996) notes that Responsibility-Based Manufacturing (RBM) "falls under the umbrella of the agile manufacturing paradigm" because in mass customization environments it allows adjustments for process and product variety to be achieved dynamically and quickly and without the need for prior system reconfiguration. However, it should be clear that RBM is really a *system* or an enabling technology developed to aid in the mass customization environment.

Similarly, another enabling technology for agile manufacturing is Virtual Manufacturing (VM). According to Hitchcock (1994), VM is an integrated synthetic environment meant to enhance various levels of decision and control by improving product and process design, embellish product service and repair, reduce manufacturing risks, and support system changes. This transpires by customers' designing their products (i.e., choosing their desired features) and placing an order to the company, directly monitoring production in real-time, and receiving the final product by mail (Cho *et al*., 1996).

Most literature encountered assert that agile manufacturing refers to a dynamic manufacturing setting which allows rapid reconfiguration and is highly adaptive to quick market changes through widespread use of information technology (Graves *et al*., 1995; Xiaoyuan, 1996; Gillies *et al*., 1996; Booth, 1996). Thus, an interface between manufacturers, suppliers, and customers is required to be adaptable to continuous and unexpected changes and become successful. Agile manufacturing environments differ from traditional manufacturing environments in the fact that product life cycles are becoming shorter while the quest for higher qualities are becoming more consequential, and products are becoming increasingly diversified and global. Such product characteristics are becoming increasingly more prevalent and are being documented by many studies. For example, Sanderson and Uzumeri (1990), Cordero (1991), Hisrich and Peters (1991), Millson *et al.* (1992), Norton and Bass (1992), Stern (1992), Griffin (1993), Lawrence (1993), and Bayus (1994) report at least one of the following product characteristics:

1. Decreasing concept-to-market and, thus, the introduction time between new products.
2. Decreasing length of PLCs.
3. Increasing product variety.
4. Decreasing volumes for identical products.

To cope with such characteristics, the concept of *agile manufacturing* comes into play. This concept does not refer to a certain method of production or a specific

technology; it simply asserts the fact that many changes are occurring to production requirements.

2.3 MODELING OF MANUFACTURING ENVIRONMENTS

2.3.1 VOLUME AND VARIETY

Capacity constraints greatly affect companies' decisions regarding tradeoffs between volume and variety. Young and Greene (1986), amongst others, attempt to match different types of systems to different traditional manufacturing environments using only two decision variables: volume and variety (Figure 2.1). The figure implies that traditional manufacturing systems can only emphasize one of these items at a time. For example, transfer lines are designed for high volume but cannot be used for high variety. Therefore, an increase in volume will decrease variety, and vice-versa. However, there is no indication how the range of values for such systems are obtained, nor is it clear if those systems can be used regardless of the type of product being manufactured. Furthermore, there is no indication of the nature of product lead times for the different environments. Manufacturing lead time and delivery lead time are directly influenced by the systems being utilized simply because some systems devote more emphasis to them than others.

Although in the past traditional manufacturing environments have been granted considerable attention, they are considered by Boynton *et al*. (1993) as "old competitive strategies...of the industrial revolution." Agile manufacturing is necessary because new competitive strategies are imperative to handle today's new environments where production requirements are continuously changing.

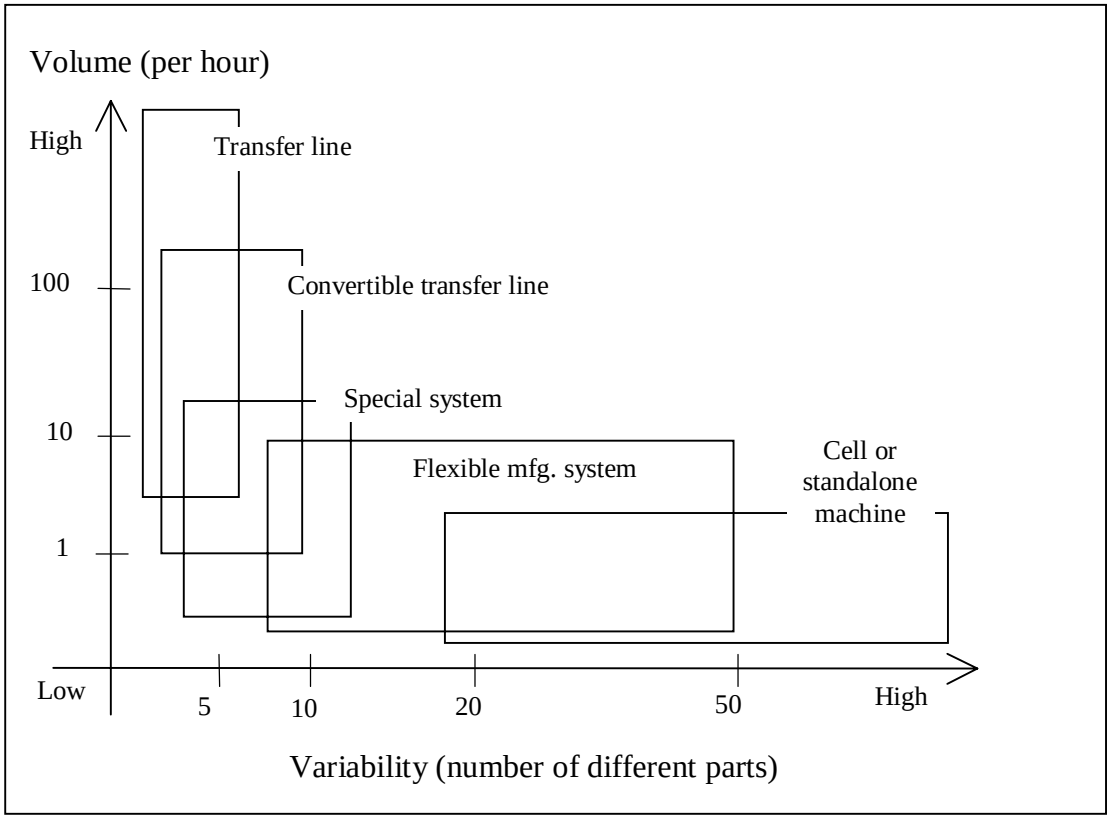


Figure 2.1. Systems used for volume versus variety. From Young and Greene (1986).

2.3.2 PRODUCT LIFE-CYCLE (PLC)

One common nomenclature which considers product demand over time but is rarely mentioned when referring to a manufacturing environment is the product life cycle (PLC) (Figure 2.2). Bayus (1994) and Donaldson *et al.* (1995) define PLC as the measure of units sold or demanded over the product's lifetime. Others interpret PLCs as the time the product spends in the market earning profit before becoming obsolete (Cordero, 1991; Gaimon and Singhal, 1992; Millson *et al.*, 1992; Griffin, 1993). However, the first definition is more accurate because when authors represent the PLC curve graphically, the curve clearly looks like a function of both demand and time (Figure 2.2). For the purpose of this research, the first definition will be adopted and the second definition would be more appropriately called *length* of PLC.

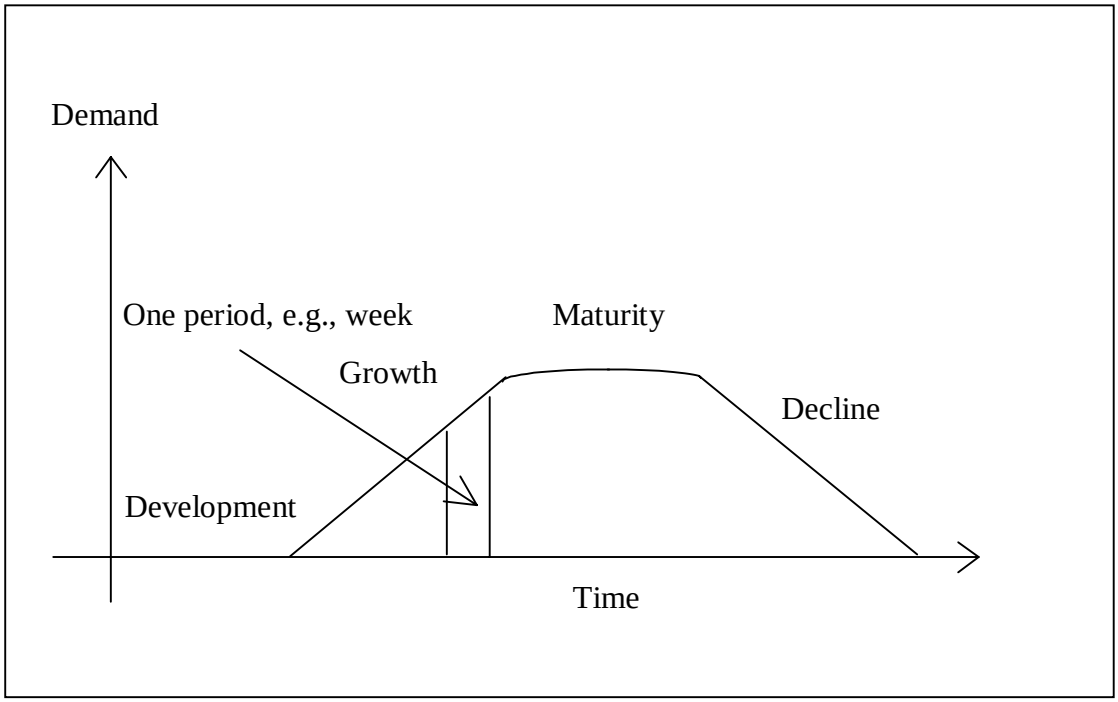


Figure 2.2. Traditional PLC curve as a function of demand and time.

Thus far, data pertaining to PLC are not only limited (Bayus, 1994), but researchers are inconsistent with their measurement standards (Griffin, 1993). For example, it is often not clear if a product is the result of complete new technology or just an incremental improvement over the previous version. Having such information is important because production volume, variety, and length of PLC depend on such factors. For example, when new technology is involved, it is best for a company to concentrate on large volumes rather than on large variety; this way they can capture as much of the market share as possible while monitoring product success. Once the product is proven successful, the company starts collecting data and getting feedback regarding customers' preferences and is able to introduce more variety. Therefore, the level of new technology involved (or incremental improvement of older technology) affects production volume and variety.

A common disposition regarding the PLC is that it is continuously decreasing. In other words, total demand over the life of a product is declining (Cordero, 1991; Gaimon and Singhal, 1992; Millson *et al*., 1992; Griffin, 1993; Donaldson *et al*., 1995). This does

not mean that the product is no longer functional; most products are functional beyond their PLC. However, newer, better, and more affordable models continually come out, which simply makes older versions look obsolete and more feasible to switch over to newer products (Cravens, 1986). Also, as product varieties are increasing, there are more choices available. Gaimon and Singhal report, for example, that from 1964 to 1976 IBM introduced only two families of mainframe computers. Then in the next four years IBM introduced 4 new computers. In the 1980s, Honda went in just 18 months from manufacturing 38 models of motorcycles to 113 models. Therefore, the likelihood of purchasing a particular model is decreasing and the PLC curve for that model does not reach high levels; but the total demand for products in the same family is increasing. In short, PLC is decreasing indeed; but that is an indication of increasing product variety and an increasingly dynamic and agile environment.

As Figure 2.2 shows, the traditional product life cycle is well-defined by growth, maturity, and decline periods (Hutchinson, 1982; Hutchinson and Sinha, 1989; Azhar and Leung, 1993). Others tend to agree, however, but they also add that growth and decline periods are getting shorter and steeper (Cordero, 1991; Brannon *et al.*, 1993; Bayus, 1994). In fact, several recent studies present PLC curves without decline periods (Figure 2.3), suggesting that instead of reducing production for older products, those products are eliminated and capacity devoted towards newer products (Brannon *et al.*, 1993; Bayus, 1994).

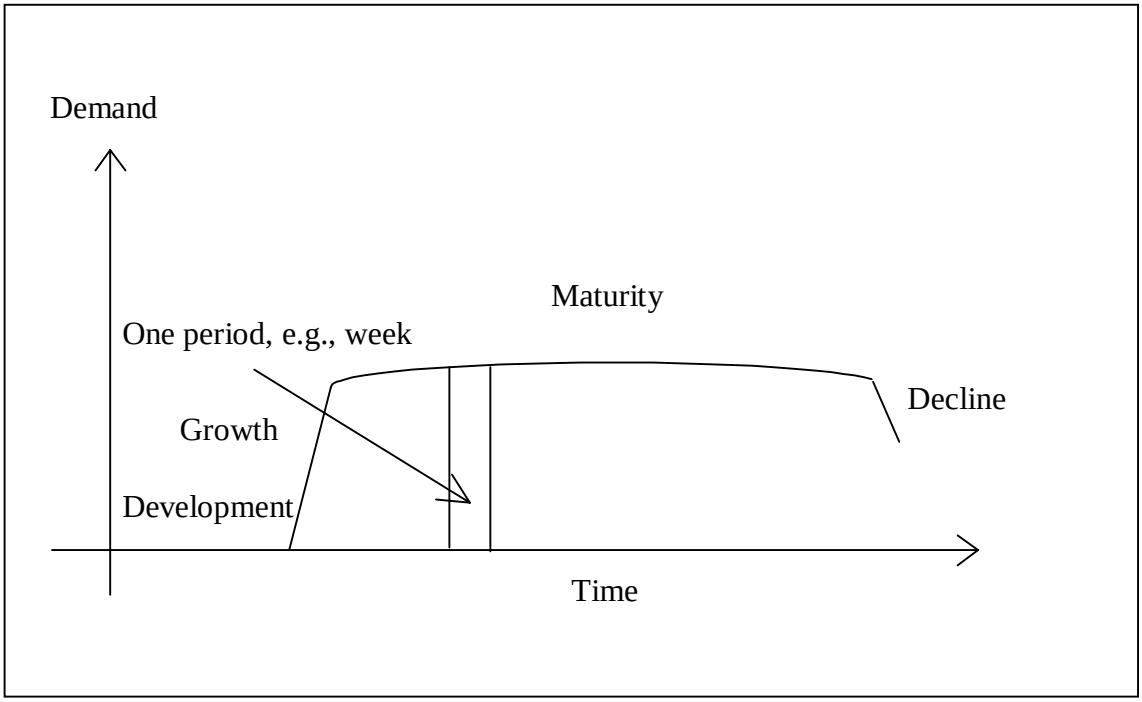


Figure 2.3. PLC curve for agile manufacturing environments.

Product life cycle (PLC) curves are used by some studies to model manufacturing environments. For example, Hutchinson and Sinha (1989) developed a model of a manufacturing environment consisting of only one product life cycle curve consisting of growth, maturity, and decline periods. Then, based upon maximizing the net present value, this model is used to choose between two alternatives: a flexible manufacturing system and a transfer line. Similarly, Gaimon and Singhal (1992) develop a model in an effort to quantify the optimal level of flexibility based upon maximizing the total profit, which takes into account the added benefits of quick changeover, as well as changeover cost associated with flexibility. They refer to *flexible manufacturing* as "the ability to quickly change over from one type of product to another at low cost". Furthermore, they state that although higher flexibility costs less to change over, the initial acquisition cost increases exponentially as the level of flexibility increases.

The framework of their modeling environment consists of several product life cycles put together to form a manufacturing environment. In order to simplify their

model, they assume that PLCs are exactly alike and do not lag or overlap. This assumption, although valid for some manufacturing environments, limits the modeling framework considerably because when PLCs do not lag or overlap, the time between new product introductions is equal to the length of PLC (See Figure 2.4); therefore, no more than one product is being produced at a time.

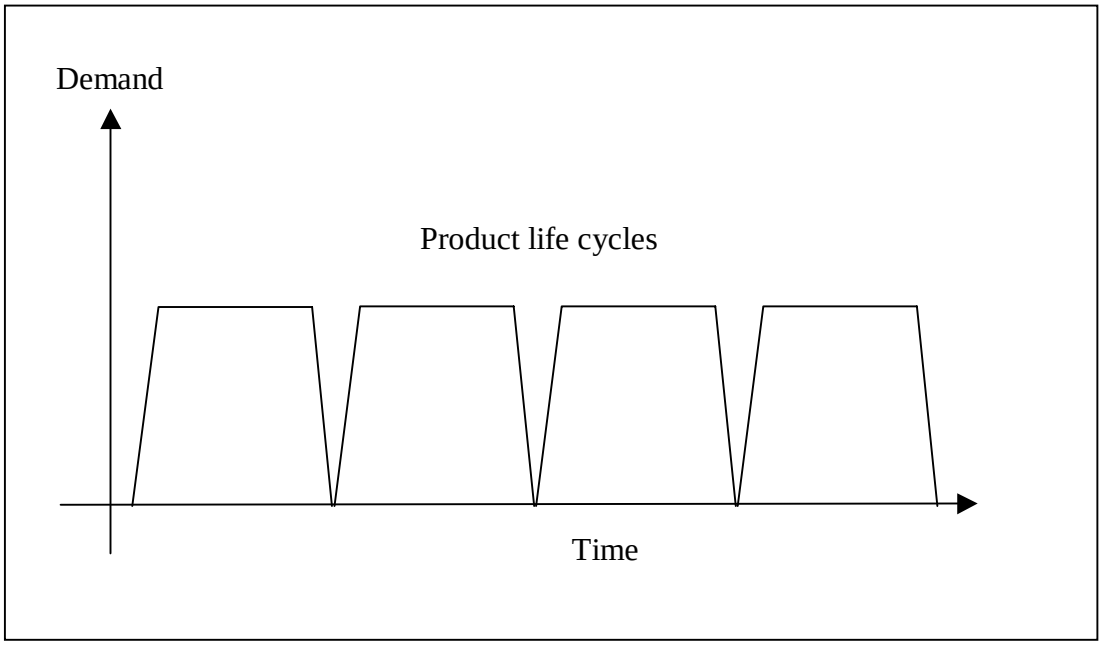


Figure 2.4. Manufacturing environment where the number of introduction periods is equal to the length of PLC.

Their argument for this assumption is that some facilities, such as automobile manufacturers, produce only one type of product at a time; therefore, PLCs do not overlap. However, such an assumption is oversimplified because the example provided regarding automobile manufacturers is just one unique case. Also, as demand for variety and custom-made products is on the rise, facilities need to be designed to handle several varieties of products at the same time. The ability to handle several varieties of products simultaneously, with overlapping PLCs, will be incorporated into this research.

2.4 DESIGN OF AGILE MANUFACTURING SYSTEMS

It is widely accepted that in order to survive in today's manufacturing environments, companies need to produce highly customized products and faster than their competitors. Unfortunately, however, most literature encountered on agile manufacturing argues *why* agile manufacturing is necessary rather than *how* to design agile manufacturing systems (Bunce and Gould, 1996; Forsythe and Ashby, 1996). Therefore, as mentioned in Chapter 1, certain influential features have emerged in which switching between operating modes needs to be researched (Kulatilaka, 1988). Those operating modes pertain to design/reconfiguration of manufacturing systems, where demands for products are continuously changing.

Therefore, in order to deal with changes in product demand, the system has to change as well (or at least re-adjust). Such changes in the system can be in the form of switching between the utilization of certain levels of capacity and processing capabilities. This includes when to switch capacity, by how much, and where to acquire it from (inside or outside the system). These need to be addressed together due to their interdependencies, and answers to such questions are likely to be different depending on the type of manufacturing environment on hand.

All too often, the term *agile manufacturing* is confused with *flexible manufacturing*; however, there is a distinction between the two. The original vision of agile manufacturing, as specified in the two-volume report *21st Century Manufacturing Strategy: An Industry-Led View* (Iacocca Institute, 1991), considered agile manufacturing as being concerned with many facets of an enterprise: processes, people, management and organizational structures, vendor relationships, business strategies, etc. It is partly because of the broad spectrum of issues addressed in this work that agility can take on so many different meanings and interpretations. In the most basic sense, however, agility refers to manufacturing utilizing resources and people which can be changed, or reconfigured, quickly and easily for coping with variability and uncertainty (Shewchuk, 1999a). Some consider agility as the ability to be flexible (able to changeover quickly and easily for producing a limited variety of products) and capable of high production

volumes at the same time (Fujii *et al.*, 1996), whereas others equate agility with mass customization, the ability to produce an almost unlimited variety of products in small quantities, even single items (Sheridan, 1993; Hormozie, 1994). So, there is no single definition of agility as it relates to product variety and volume - in fact, many different "types" of agile manufacturing systems are possible (Shewchuk, 1999b). In all cases, however, agility does have to do with the ability to change the processing facility to cope with whatever changes in production requirements do occur. As stated by Dove (Voss, 1994), "the essence of an agile corporation is the ability to reconfigure the plant facility itself." This is the adopted definition of agility for this research: the ability to reconfigure the plant facility. By reconfigure, we mean the ability to change production capabilities (what processes we can perform in the facility) and production capacity (the capacity available for performing each process). We are thus not restricting ourselves to changes within the same product line only, or high-volume production only, etc. These are characteristics of different environments within which agile manufacturing facilities must operate: a given agile manufacturing system thus may or may not be suitable for a given agile manufacturing environment (what we intend to explore in the research).

As far as flexibility goes, flexibility here refers to the ability to change a manufacturing system from within, whereas agility refers to the ability to change (reconfigure) a manufacturing system from the outside, i.e., externally (Shewchuk, 1999a). Consider an FMS: it is very flexible for what it was designed to do (changes which can be made from within the system, such as changing between those pallet and fixture types the system can accommodate, changing tools at magazines, etc.). However, FMSs are rarely agile: they are extremely rigid systems which take a lot of time and effort to change (i.e., from the outside) once they are in place and running. As we are dealing with changes to process capabilities and capacities (and incurring additional expenditure for changes), we are changing the system from the outside and hence dealing with agility.

CHAPTER 3

MODELING FRAMEWORK

3.1 MODELING OF AGILE MANUFACTURING ENVIRONMENTS

Based upon the literature review, it is obvious authors have adopted different strategies and assumptions to suite their own ways of modeling manufacturing environments. Hutchinson and Sinha, 1989 and Gaimon and Singhal, 1992, for example, depicted their manufacturing environments by one or more product life cycles which represent production requirements over time. For the purpose of modeling agile manufacturing environments, this thesis adopts certain key assumption as well, such as each product requiring a certain manufacturing capability, each product is in demand for a number of periods, and multiple products can be in the system at the same time (all other assumptions will follow soon). Given such assumptions, one simple way to model agile manufacturing environments can be through the following four variables:

1. Rate of new product introduction.
2. Length of PLC for each type of product.
3. Demand per period for each type of product.
4. Production time per unit for each type of product.

Combining these four attributes results in product life cycle curves and, as shown in Figure 3.1, combining PLC curves yield a representation of the manufacturing environment on hand. One important aspect that sets the modeling of manufacturing environments of this research apart from others is that the rate of new product introduction is a separate variable from the length of PLC. Because values of each modeling variable can vary, different levels of such variables yield different agile manufacturing environments. Accordingly, each environment will then necessitate different system design/reconfiguration strategies.

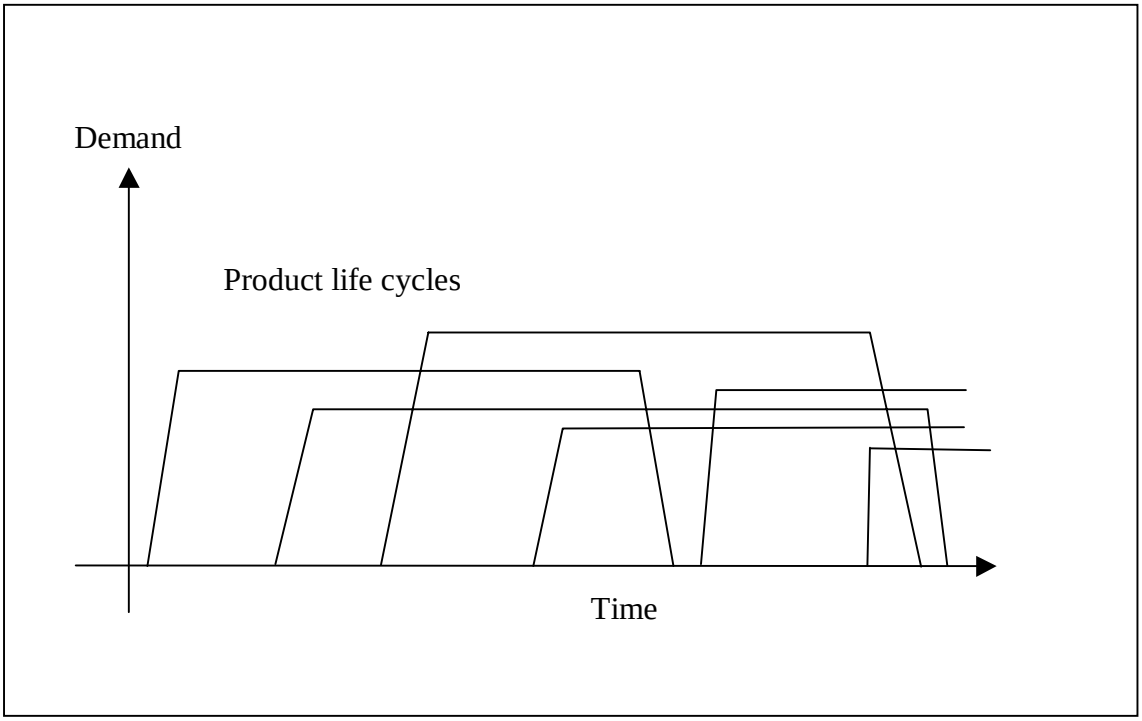


Figure 3.1. PLCs of various products combined to form a manufacturing environment.

This generic way of modeling agile manufacturing environments is significant for this research because it depicts the *time-phased* demand and *capacity requirements* needed by the manufacturing system to cope with the environment. In addition, this model allows products to be *born* regardless of the fact that other previous products may still be *alive*. This way of modeling manufacturing environments is more realistic than those previously discussed for one important reason: manufacturers do not usually wait until demand for a certain type of product diminishes completely before starting to produce another product. If that were the case, then manufacturers would have no competitive edge in the market and would be producing products with little demand and low selling prices. In addition, manufacturers have limited production capacity and do not necessarily have the means to produce all products in demand. Therefore, they adopt certain operating strategies that allow them to set their priorities and devote their capacity to producing the types of products they believe are the most important. The assumptions and variables associated with modeling of agile manufacturing environments are

presented next. Note that *algorithmic variables* here refer to non-input variables that are necessary for the simulation model to successfully execute.

Agile Manufacturing Environment Assumptions

1. Each product has a certain level of demand.
2. Each product will be in demand for a certain number of periods.
3. It takes a certain amount of processing time to manufacture each type of product.
4. It take a certain amount of time before a new product is born.
5. Product life cycle of any product may begin during any period.
6. Product life cycle of any product may end during any period.
7. Product life cycle may be in progress during any period.
8. Each product requires only one type of processing capability.

Agile Manufacturing Environment Input Variables

LB(P), UB(P) = lower and upper bounds for number of periods between new product introductions.

LB(L), UB(L) = lower and upper bounds for length of product life cycle (PLC).

LB(D), UB(D)= lower and upper bounds for demand per product during each period.

LB(T), UB(T) = lower and upper bounds for production time per product.

M_x = maximum quantity of possible processes (capabilities).

Agile Manufacturing Environment Algorithmic Variables

p_i = U(UB(P), LB(P)) number of periods between introduction of products i and i-1 generated according to a uniform distribution between lower and upper bounds.

l_i = U(UB(L), LB(L)) number of periods product i is in demand (length of PLC) generated according to a uniform distribution between lower and upper bounds.

d_i = U(UB(D), LB(D)) demand for product i in each period of its PLC. generated according to a uniform distribution between lower and upper bounds.

t_i = U(UB(T), LB(T)) production time per unit for product i generated according to a uniform distribution between lower and upper bounds.

T_v = total number of products modeled.

Dp_{ij} = demand (in units) for product i in period j.

Cp_i = $U(1, M_x)$ = process (capability) required for product i generated according to a uniform distribution between 1 and M_x.

3.2 MODELING OF AGILE MANUFACTURING SYSTEMS

3.2.1 PHYSICAL PROCESSING FACILITY

The physical processing facility refers to the physical components used to produce the required variety of products. Associated with any facility is a certain level of agility. As pointed out earlier, different studies have adopted different definitions for agility. In this thesis, however, agility refers to the effort with which a manufacturing system (capabilities and/or capacities) can be changed, where effort is a function of both time and cost. Hence, systems with higher agility can be reconfigured with less effort than less-agile systems. This will be discussed in more detail in Section 3.2.2.1.

Processing capabilities are not meant to represent individual machines. Instead, this research assumes one aggregate capacity available for each capability. One other important assumption is that the facility is 100% reliable with no breakdowns for any reason. Therefore, the only causes of idle time will be incurred due to reconfiguration time. Also, consideration is given to the fact that processing facilities have limited useful lives and cannot be considered to operate forever. Therefore, the modeling process can be considered complete when the useful life of the facility is reached. Certain other features of the processing facility include the maximum number of processing capabilities and the number of operating hours per period (for example, two shifts or 16 hours of operation per day). All assumptions and variables associated with the processing facility are presented next.

Physical Processing Facility Assumptions

1. An aggregate amount of capacity is available each period for each capability.
2. No time lost to breakdowns or inefficiencies. Processing reliability is 100%.

3. No setup time associated with any capability.
4. No post-processing time, such as removing finished parts.
5. Facility has limited useful life.
6. No learning curves (constant processing time).
7. Fixed number of operating hours per period.

Physical Processing Facility Input Variables

N = number of periods.

H_p = number of operating hours per period.

M_x = maximum quantity of possible processes (capabilities).

Physical Processing Facility Algorithmic Variables

Rc_{kj} = available capacity for k^{th} capability at the beginning of period j.

Qc_j = quantity of available capabilities at the beginning of period j.

Pa_{kj} = processing ability of the k^{th} existing capability in period j.

3.2.2 SYSTEM LIFE-CYCLE STRATEGIES

3.2.2.1 DESIGN AND RECONFIGURATION STRATEGIES

Before the manufacturing system evolves over time, it will naturally start at some initial size, which refers to processing capabilities and capacities that are present when production starts. Starting with a large number of processing capabilities will reduce the likelihood for the need to reconfigure or expand, thus allowing more types of products to be produced in early stages of their PLC. This results in increased revenue because not only more products can be produced, but also prices are usually higher for newer products. However, starting with a large number of processes results in higher initial investment costs, and this becomes more pertinent when the time value of money is taken into consideration.

Each manufacturing system has a certain level of *agility*. This refers to the effort with which capabilities can be reconfigured to produce different varieties of products.

Although more agile systems require higher initial investment costs (discussed in Section 3.3.1), once a highly agile system is acquired, reconfiguring it to produce different types of products can take place with either less time or less cost. This is supported by Kulatilaka (1988) except that instead of referring to it as agility, he expresses it as the ability to change capacity of operating modes. Interestingly enough, research that followed proved that Kulatilaka's analogy between agility and changing capacity is very accurate (Richards, 1996). Figure 3.2 shows the effect of agility on reconfiguration cost and reconfiguration time, assuming simple linear relationships. Spending an amount C_1 allows the less-agile system to be reconfigured in T_2 time units and the more-agile system in T_1 time units, $T_1 < T_2$. On the other hand, both systems can be reconfigured for the same amount of time, but the cost will be higher for the less-agile system. For example, reconfiguring in T_2 time units costs C_1 for the less agile system and C_2 (less costly) for the more agile system.

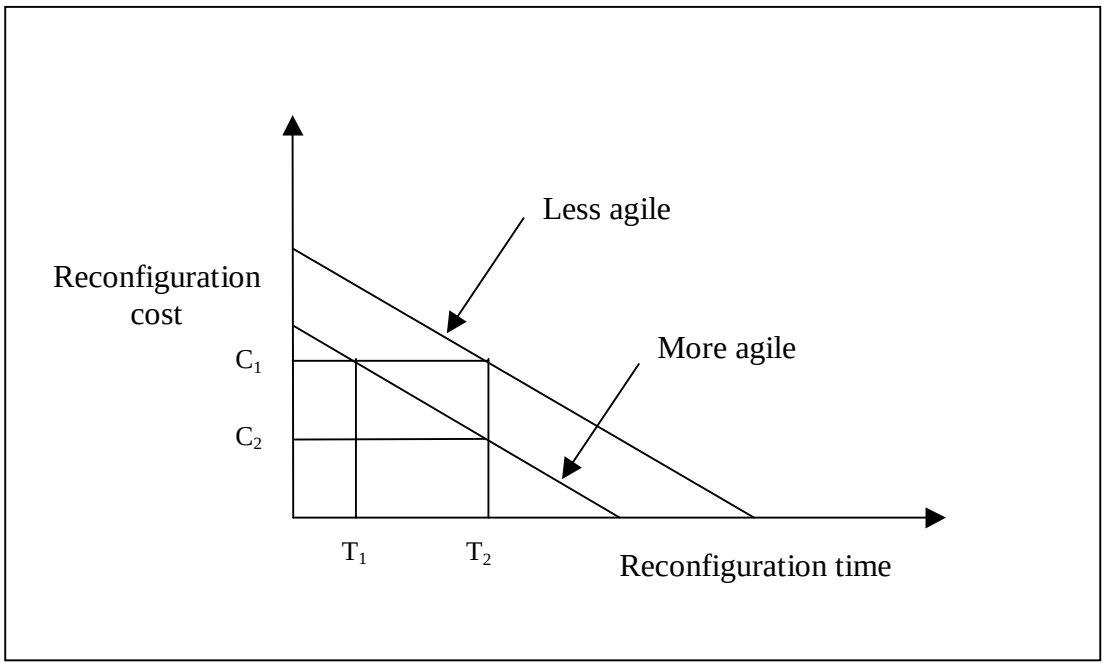


Figure 3.2. Two systems with different levels of agility.

The assumption is that given the same initial system configuration and change to be made (in order to handle the new product), the more agile system will always be able to make the change for less effort than the less-agile system. The effort required is a

function of two things, the time and cost to make the change. For a given level of effort, the change can be obtained very quickly at the expense of high cost, or very inexpensively at the expense of longer time required, or somewhere in-between. Whatever time/cost combination is used, however, the resulting effort is less with the more-agile system. In other words, the level of effort required to change the capacity is always less for the more-agile system, regardless of the change (capacity increase). Additionally, the level of effort required to change the system capabilities is always less for the more-agile system, regardless of the change (change in processes which can be performed). Also, for a given change, two systems identical in structure and functionality (same configuration) can be reconfigured in the same amount of time, but cost will always be higher for the less-agile system. Conversely, for a given change, two systems identical in structure and functionality (same configuration) can be reconfigured for the same cost, but the time required will always be higher for the less-agile system. Of course there is a price to pay for the reduced effort: the agile system is more expensive (more money spent up-front to make the system more agile).

The above characteristics of agile manufacturing are reflected in the agile manufacturing system model used in this thesis. We must keep in mind, however, that these assumptions hold true only if more-agile systems can perform every possible change with less effort. Of course, this will never be completely attained in practice. The extent to which it is attained will depend upon a variety of factors, including the nature of the products, how mature the processing technology is, how well changes in products and demand can be anticipated, etc.

Whenever there is shortage in capacity and demand cannot be fulfilled, an important decision to be considered is whether to periodically reconfigure an existing capability or to expand (purchase) capacity from outside. Expanding capacity incurs initial acquisition cost plus the additional cost and time it takes to set it up and make it ready to produce. In addition, the added capacity may only be needed for a short period of time. On the other hand, expanding capacity reduces the likelihood for the need to reconfigure other existing capabilities. Therefore, this allows the availability of other existing capabilities to produce other products that may enter into the system. On the

other hand, in order to keep the simulation model concise and not complicate matters any further, this research assumes that total system capacity is not expendable (i.e., cannot be reduced in size). Assuming the system capacity can be reduced in size will force this research to get into issues such as when to reduce system capacity, by how much, and the price of selling capacity.

It is evident from the above discussion that in order to model agile manufacturing environments many decision variables related to system design/reconfiguration can be considered. This research, however, identifies and focuses on four important such variables:

1. Initial system size (in terms of processing capabilities).
2. Level of system *agility*.
3. Level of time and cost involved in reconfiguration.
4. Level of reconfiguration vs. expansion.

These are referred to as *design/reconfiguration* variables, and comprise the decision variables for this research. Accordingly, each unique set of values for these variables represents a different *design/reconfiguration* strategy. The need to reconfigure manufacturing systems arises due to the fact that capacity required to produce a certain type of product may not always be available. In a manufacturing system, one or more of three basic cases can arise during any period in time. These three cases can be summarized as follows:

Case 1. Required capability present with enough or excess capacity at that capability to satisfy all demand for product *i* in period *j*.

Case 2. Required capability present but not enough capacity to satisfy all demand for product *i* in period *j*.

Case 3. Required capability to produce product *i* not present at all during period *j* (also implies no capacity).

In order to deal with each case, there can be one or more options from which to choose. For case 1, the obvious decision is to produce and fulfill all demand. For cases 2 and 3,

production will depend on the reconfiguration strategy being utilized. The modeling strategy presented next shows how reconfiguration takes place by considering each product type in each period.

Reconfiguration Strategy Heuristic

1) If the required capacity for any product is less than or equal to available capacity to produce that product, then satisfy all demand.
2) If the required capability is not present, then it will be created.
3) If required capacity is greater than available capacity to produce that product, then determine which type of action to take.
    a) If the capacity shortage is less than or equal to total extra capacity, then:
        i) fulfill a certain percentage of the shortage through reconfiguration, and:
        ii) acquire the remaining amount through expansion (i.e., purchasing).
    b) If the capacity shortage is greater than the total extra idle capacity, then:
        i) reconfigure as much idle capacity as possible without exceeding a certain percentage, and:
        ii) expand or purchase the remaining capacity from outside.

When the strategy is to reconfigure as much idle capacity as possible, capacity will only be purchased when no idle capacity exists. On the other hand, when the strategy is to fulfill all capacity shortage through expansion, then existing idle capacity will not be reconfigured. Naturally, there exists an option of reconfiguring a certain amount of idle capacity and purchasing the rest from outside at the same time.

Design/Reconfiguration Strategy Assumptions

1. Whenever shortage in capacity exists, capacity will be acquired through purchase or reconfiguration. *Do nothing* is not an alternative.
2. Capacity for any capability can be acquired through purchasing from outside and/or by reconfiguring idle capacity from inside the system. Therefore, capacity can be shifted among capabilities.

3. Reconfigured capacity will be obtained from the capability that has been idle for the longest time. More capacity can be acquired from the second longest idle existing capability, and so forth.
4. If no idle capabilities exist, capacity will be reconfigured from busy capabilities with extra capacity to spare, starting with capabilities with the largest amount of extra capacity.
5. The time required to set up each hour of capacity (for a certain level of agility) is the same regardless of whether the capacity was purchased or reconfigured.
6. The time required to setup each hour of capacity (for a certain level of agility) is constant (i.e., no learning curves).
7. The amount of required capacity during any period cannot be reconfigured for a different type of capability. Only excess capacity can be reconfigured.
8. Once capacity is purchased, it can be reconfigured but not sold.
9. Minimum and maximum amount of time for setting up 1 hour of capacity are known.

Design/Reconfiguration Strategy Input Variables

γ = level of initial system size (small vs. large); $\gamma \in [0, 1]$.

α = level of system agility (non-agile vs. agile); $\alpha \in [0, 1]$.

τ = level of reconfiguration time vs. cost (short and costly vs. long and less costly); $\tau \in [0, 1]$.

δ = level of reconfiguration vs. expansion preference (reconfigure vs. expand); $\delta \in [0, 1]$.

T_{min} = minimum time (hours) for setting up (reconfigured or purchased) 1 hour of capacity.

T_{max} = maximum time (hours) for setting up (reconfigured or purchased) 1 hour of capacity.

Design/Reconfiguration Strategy Algorithmic Variables

T_k = $U(T_{min}, T_{max})$ time (hrs) required for setting up 1 hour of capacity (reconfigured or purchased) for k^{th} capability when agility is minimum ($\alpha = 0$).

Cr_{kj} = required capacity for k^{th} capability at the beginning of period j.

Sc_{kj} = capacity shortage at the k^{th} capability at the beginning of period j.

Ec_{kj} = extra capacity available at the k^{th} capability at the beginning of period j.

Td_j = total extra capacity (for all capabilities) at the beginning of period j.

Pq_{ij} = production quantity for product i in period j.

Dp_{kj} = amount of purchased capacity for the k^{th} capability during period j.

Dr_{kj} = amount of reconfigured capacity *for* the k^{th} capability during period j.

Wf_{kj} = amount of reconfigured capacity *from* the k^{th} capability during period j.

Tr_k = time required to set up (reconfigured or purchased) one capacity hour for the k^{th} capability (will depend upon the level of α).

3.2.2.2 OPERATING STRATEGY

The term "operating strategy" here refers to the process of sequencing or prioritizing of products (in each period) to be considered for production. Because capacity may not always be present to produce all products, prioritizing products allows companies to produce the products that they believe are the most important. Some of the most commonly encountered operating strategies include (Bhaskaran and Pinedo, 1992):

- Products with highest selling price.
- Products with shortest processing time (SPT).
- Products with longest processing time (LPT).
- First-in-first-out (FIFO).
- Last-in-first-out (LIFO).

Such operating strategies are developed for different objective functions and would yield optimal results under specific assumptions. However, multiple experiments for this specific model have shown that the best quantitative operating strategy towards maximizing the objective function of the model is to produce products with the highest ratio of price/production time.

It should be noted that the operating strategy adopted in this model yields better results than the strategies listed above only because of the assumptions made. Other

assumptions may yield better results by utilizing one of the above operating strategies; therefore, required capability and capacity in each period (if available) are strictly devoted to the product with the highest ratio of price/production time first. The level of capacity availability and utilization will depend on the reconfiguration strategies being utilized, which were presented before.

Operating Strategy Assumptions

1. Demand will be satisfied to the extent that capacity allows.
2. Values of decision variables remain constant over the production periods of interest.
3. No backordering is allowed. Lost production due to insufficient capacity will not be compensated.
4. No inventories are held. Production will not exceed demand for any product during any period.
5. Production priority is given to products with highest ratio of selling price to production time.

Operating Strategy Input Variables

(none)

3.3 DETERMINATION OF COST, REVENUE, AND PROFIT

As was indicated before, the approach used in this research is to develop a generic modeling framework for agile manufacturing environments and design/reconfiguration strategies. This generic model is actually comprised of three models combined in one. The first model (modeling agile manufacturing environments) serves as the basis for delineating time-phased demand and capacity requirements according to the input variables. The second model goes through a set of algorithms pertaining to design, operation, and reconfiguration strategies (established via the decision variables). Finally, the third model comes in the form of cost functions used to weigh the effects of different design/reconfiguration strategies on the performance of the manufacturing system being used.

One way to measure performance is to utilize certain cost functions in which the objective function would be to maximize profit. The cost functions here refer to both revenues and costs. The revenues are generated from sales of products: it is assumed that all manufactured products are sold. In addition, due to the fact that in today's environments prices of products decline over time (Franza and Gaimon, 1997), this research assumes that the price of each product will go down at a constant rate as the PLC ages from one period to the next.

The decision variables have direct effects on both the amount of manufactured products and the costs associated with them, and hence revenue as well. For example high levels of agility incur high investment costs but, at the same time, allow capacity to be ready fairly quickly and hence more products to be produced. Because various levels of decision variables yield significant tradeoffs in revenues and costs, the model should measure the *discounted* amount of profit (i.e., take into account the time value of money).

The elements comprising the profit are the revenue (total sales), capacity purchasing cost, and reconfiguration cost. Before describing how each is determined, all assumptions and variables associated with them are presented first.

Cost Model Assumptions

1. Product selling prices decline at a constant rate over time.
2. Purchasing capacity from outside incurs initial purchasing and reconfiguration cost. Reconfiguring an existing capacity incurs reconfiguration cost only.
3. The cost of purchasing capacity will be incurred during the period in which it was bought.
4. The cost of reconfiguring capacity will be prorated over as many periods it takes to complete the reconfiguration.
5. The cost of purchasing each hour of capacity (for a certain level of agility) is constant (i.e., 100^{th} hour costs same as 1^{st} hour –no discounts or price breaks).
6. The cost of setting up each hour of capacity (for a certain level of agility) is the same regardless of whether the capacity was purchased or reconfigured).

7. The cost of setting up each hour of capacity (for a certain level of agility) is constant (i.e., no discounts or price breaks).
8. The relationship between reconfiguration time and cost is linear.

Cost Model Input Variables

R_{min} = lower bound for initial price of product.

R_{max} = upper bound for initial price of product.

P_d = percent price decline from one period to the next.

C_{min} = minimum cost for purchasing 1 hour of capacity when agility is min. ($\alpha = 0$).

C_{max} = maximum cost for purchasing 1 hour of capacity when agility is min. ($\alpha = 0$).

D_{min} = minimum rate at which cost of 1 hour of capacity increases with increasing agility.

D_{max} = maximum rate at which cost of 1 hour of capacity increases with increasing agility.

A_{min} = minimum cost of acquiring (reconfigured or purchased) 1 hour of capacity.

A_{max} = maximum cost of acquiring (reconfigured or purchased) 1 hour of capacity.

I_r = effective interest rate per period.

Cost Model Algorithmic Variables

Ip_i = $U(R_{min}, R_{max})$ initial selling price of product *i* during first period of introduction.

Sp_{ij} = selling price of product *i* during period *j*.

C_k = $U(C_{min}, C_{max})$ cost of purchasing one hour of capacity for k^{th} capability when agility is minimum ($\alpha = 0$).

D_k = $U(D_{min}, D_{max})$ rate at which cost of purchasing one hour of capacity (for k^{th} capability) increases with increasing agility.

Cc_k = cost of purchasing one hour of capacity for the k^{th} capability (any level of α).

A_k = $U(A_{min}, A_{max})$ cost of setting up one hour (reconfigured or purchased) for k^{th} capability when agility is minimum ($\alpha = 0$).

Ca_k = cost of setting up (reconfigured or purchased) one capacity hour for the k^{th} capability (will depend upon the level of α).

Pr_j = total profit in period *j*.

P_w = present worth of profit.

A_w = periodic worth of profit.

3.3.1 COST OF PURCHASING CAPACITY

The cost of purchasing capacity is affected by both the level of agility and the amount of capacity purchased. As was indicated before, the level of agility refers to the time and cost it takes to reconfigure. This translates to the level of effort in which change can be made: more agile systems require less time and/or cost to reconfigure. However, the initial investment (or purchase) cost of higher agility is higher than the cost to purchase a lesser agile system. This is illustrated in Figure 3.3.

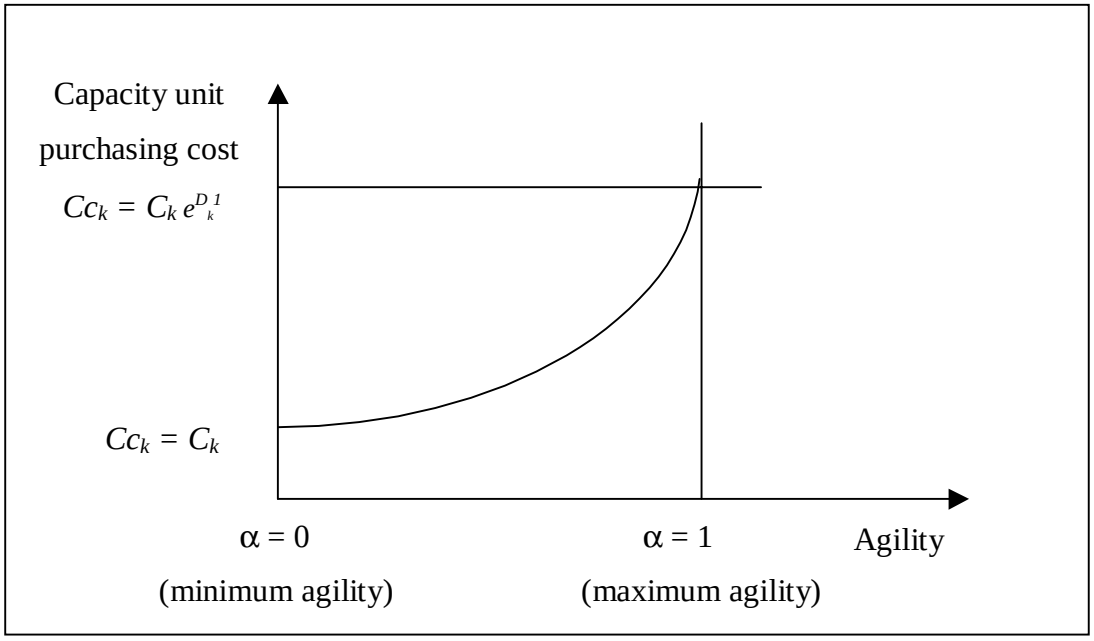


Figure 3.3. Cost to purchase capacity (per-unit basis) vs. level of agility.

From the figure, it is clear that the cost of agility increases at an increasing rate (i.e., exponentially) relative to the level of agility. This is consistent with the assumption made by Gaimon and Singhal (1992). The only difference is that they refer to it as *flexibility* although it has all the characteristics and definitions of what many others now convey as agility, such as the low cost and time of *reconfiguration*. The cost of purchasing each hour of capacity pertaining to a certain level of agility is:

$$Cc_k = C_k e^{D_k \cdot \alpha}$$

The minimum and maximum levels of agility can vary between 0 and 1 and are normalized such that the minimum level of agility will correspond to $\alpha = 0$ and maximum agility $\alpha = 1$.

3.3.2 COST OF RECONFIGURING CAPACITY

Although the level of agility measures the effort by which change can be made, there can be different ways to utilize this level of effort: quick reconfiguration while incurring a high cost, and slow reconfiguration with less cost. As Figure 3.4 shows, the level of agility sets the curve in which the system operates, and after that the time/cost of reconfiguration is established by τ. Therefore, reconfiguring in a minimum amount of time (represented by $\tau = 0$) will naturally cost the highest amount, and reconfiguring at the slowest rate will cost the least amount ($\tau = 1$). As stated previously, a linear relationship is assumed between reconfiguration cost and time.

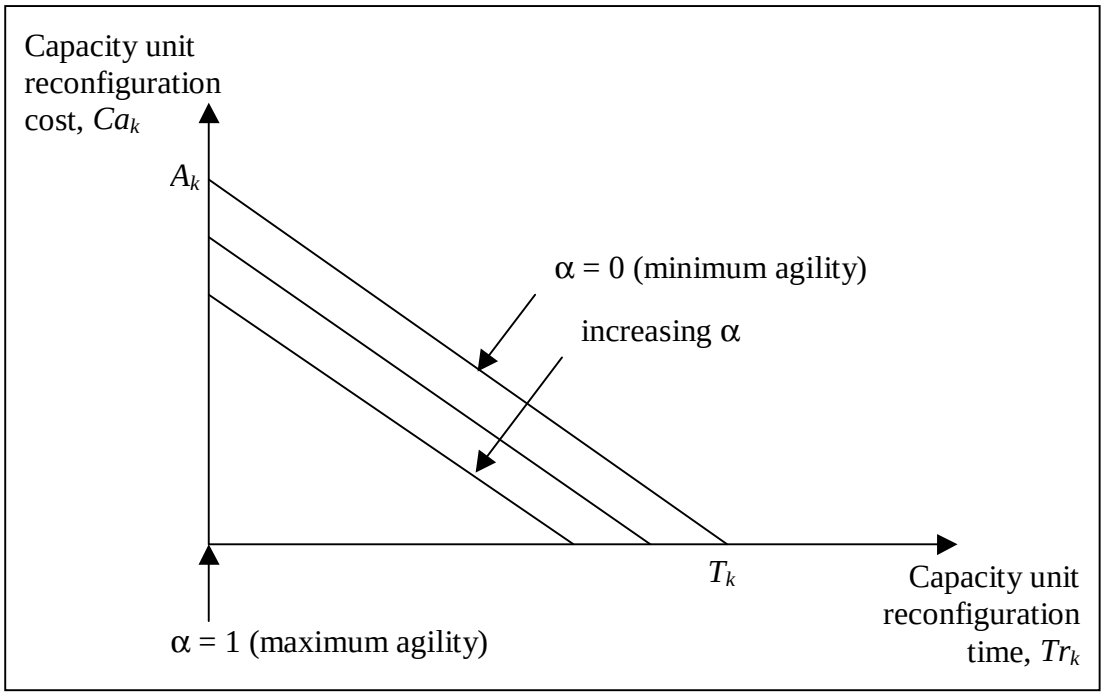


Figure 3.4. Reconfiguration cost (per-unit basis) vs. time for different levels of agility.

The above discussion illustrates that the time and cost required for reconfiguration are affected by both α and τ. The simple linear relationship between reconfiguration time and cost can be represented in the following two equations:

$$Tr_k = \tau \cdot [T_k \times (1\text{-}\alpha)]$$

$$Ca_k = A_k \cdot (1\text{-}\alpha) - (A_k / T_k) \cdot Tr_k$$

3.3.3 REVENUE AND PROFIT

All design/reconfiguration strategies have cost tradeoffs associated with them and, therefore, are expected to have an effect on revenues and costs. Although so many cost factors are involved in any manufacturing environment, this research limits its focus only on revenues, capacity purchasing, and reconfiguring cost because other costs are considered to be either fixed or unaffected by the decision variables of this model. The measure of performance takes into account the time value of money and is referred to as the *periodic worth* (A_w) of profit. Although A_w refers to the initials of *annual worth*, it will be used to imply periodic worth (or profit) because the model does not limit each period of operation to be one year. Because the profit used is merely a number representing revenues and *some* of the costs; therefore, one should not confuse it with the absolute profit that a manufacturer would make. There can be many other costs involved, but this research assumes that they will not be affected by the design/reconfiguration strategies. Hence, without discounting (or without the time-value of money), the total profit would be:

Total profit = total revenue – (capacity purchasing cost + reconfiguration cost)

$$Total\ profit\ =\ \sum_{j=1}^{N}\sum_{i=1}^{T_v} Pq_{ij} \cdot Sp_{ij} - \sum_{j=1}^{N}\sum_{k=1}^{Mx}\left(Dp_{kj} \cdot Cc_k\right) + \left(Dr_{kj} \cdot Ca_k\right)$$

Using present worth (P_w) discounting method with a constant effective interest rate I_r per period, the objective function becomes:

Maximize:

$$Total\ P_W\ profit = \sum_{j=1}^{N}\left(\frac{1}{(1+I_r)^j}\right)\sum_{i=1}^{T_v} Pq_{ij}\cdot Sp_{ij} - \sum_{j=1}^{N}\left(\frac{1}{(1+I_r)^j}\right)\sum_{k=1}^{M_x}\left(Dp_{kj}\cdot Cc_k\right)+\left(Dr_{kj}\cdot Ca_k\right)$$

Maximizing total P_w profit also corresponds to maximum periodic profit:

$$A_w\ or\ periodic\ profit\ = (total\ P_w\ profit)\cdot\left[\frac{I_r(I_r+1)^N}{(I_r+1)^N-1}\right]$$

Therefore, A_w profit can be found directly from the P_w profit. The reason A_w profit is of interest is because this allows comparison of results for cases where total numbers of periods N are unequal. In addition, P_w profit values will be very large, especially when modeling for many periods; and smaller A_w values will facilitate easier handling and analysis of results. Henceforth, unless otherwise indicated, *total profit* will always imply the A_w profit. level of agility will correspond to $\alpha = 0$ and maximum agility $\alpha = 1$.

3.4 MODEL VALIDATION

In developing and using any model, the question of validation arises. Validation is the process of ensuring that a model is valid (i.e., a good representation of reality). The extent to which a model needs to be valid depends upon its intended usage. For example, a rough-cut capacity planning model can be much lower in validity than a detailed capacity planning model: a deterministic analytical model can hence suffice for the former, whereas a detailed simulation model may be required for the latter. Another issue related to validity is tractability, the ease with which a given model can be used. In general, models which are high in validity are low in tractability and vice-versa.

To address the issue of validation, we first consider tractability. Because the scope of the problem under investigation is so large, many simplifying assumptions were necessary in order to maintain tractability. The majority of these assumptions are those typically employed to maintain tractability in manufacturing systems modeling:

- No material handling, no setups, no breakdowns, and no rework.

- No salvage value for un-needed capacity (i.e., cannot contract system).
- No backorders, no inventories.
- Single operation for each product (like single-machine scheduling).
- Constant demand over each product's life-cycle.

The remaining simplifying assumptions are not typical, however: they result specifically from the need to model reconfiguration of manufacturing systems and the associated costs. These assumptions are as follows:

- The effort associated with obtaining a certain amount of capacity is the same whether capacity is obtained from the outside (purchased) or inside (reconfigured).
- The effort required to obtain 1 hr. of capacity is constant, regardless of the total amount of capacity being obtained.
- A linear relationship exists between reconfiguration time and cost, for any level of agility.
- The cost to purchase capacity increases exponentially with the level of agility.

The result of all of the simplifying assumptions is that tractability is maintained, but the model is at a high level of abstraction and hence validity is low. This is not a problem, however, as this fits the intended use of the model - to obtain some general insights into the design and reconfiguration of agile manufacturing facilities (and not to obtain detailed design/reconfiguration procedures for a particular manufacturing facility). The low level of validity must be kept in mind, however, when attempting to apply the results to a specific manufacturing scenario. This point is noted in the presentation of results and conclusions.

Based upon the above discussion, the question of model validation for this research comes down to determining how good the latter set of simplifying assumptions are. The assumption that cost increases exponentially with agility is likely the most contentious, but it is also the most likely to be valid: cost has been found to increase exponentially with many factors (e.g., product quality) due to the law of diminishing returns. The only way to be determine with certainty how good these assumptions are, however, is to resort to exhaustive empirical research, which is beyond the scope of this

thesis. Thus, no further efforts were taken to validate these assumptions, and the fact that the results are based upon certain simplifying assumptions is consequently stressed.

CHAPTER 4

SIMULATION MODEL

The simulation model developed for this thesis integrates mathematical and logical relationships between agile manufacturing environments and design/ reconfiguration strategies. The model is coded in Turbo C^{++} and random numbers are generated using the procedure of Mars and Roberts (1983) and implemented using the code provided by Law and Kelton (1991). The main reason for using simulation modeling with random numbers is because manufacturing environments cannot be expected to have similar behavior throughout their lives. For example, demand or PLC length will differ from one product to another, even if the two belong to the same product family.

Modeling of manufacturing environments can be for any number of periods. However, simulating for 500 periods (or weeks) is adequate because it is much longer than average product life cycles in agile manufacturing environments, and many products can be simulated before the program terminates. Figure 4.1 shows how the modeling variables are affiliated with the time-phased demand of manufacturing environments. The modeling variables p_i, l_i, d_i, and t_i are the building blocks of the manufacturing environment and are generated according to a uniform distribution corresponding to the lower and upper ranges of the input variables, LB(k) and UB(k) where $k = P, L, D, T$. A uniform distribution is utilized to allow easier use of the available data for modeling purposes. Later during this research, experimental analysis will be performed using data collected from a facility that specializes in producing circuit boards. Though not all necessary types of data could be collected, the range of values of each data set that was collected seem to be roughly equally distributed between lower and upper bounds. Consequently, using a uniform distribution seems rational.

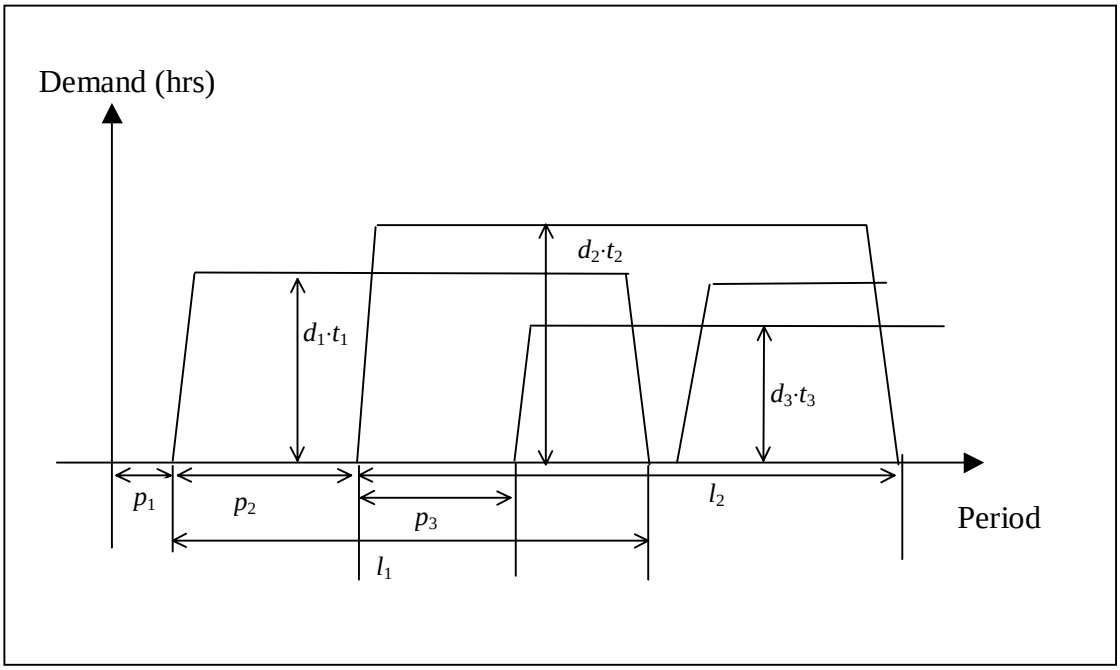


Figure 4.1. PLCs of various products, showing notations employed.

The simulation process starts during the first period of production and runs for a predetermined number of periods. Recall that for any type of manufacturing environment, during each period one or *more* of the following possibilities can occur:

1. PLC of any product(s) may begin.
2. PLC of any product(s) may end.
3. PLC of any product(s) may be in progress.

Therefore, during each period there can be any number of products going through their PLCs. Given that multiple products may exist in any period, an operating strategy should be readily available to determine the sequence (or priority) of products to be produced. This same strategy should be performed during each period because priorities may change as various products are constantly being introduced and eliminated. For example, suppose the operating strategy was to first produce products with the highest selling price and during a certain period product X had the highest priority. As product X losses its selling price from one period to the next, a new product Y with higher selling price may

be introduced. Thus, the list of priorities will have to be evaluated from one period to the next.

Because products are different, they each may require a different type of processing capability. Nonetheless, certain industries can produce many different types of products using a fixed number of versatile processing capabilities. Such is the case for a circuit board manufacturer in which assembly of any type of industrial-use circuit board can be accomplished utilizing one or more combination of 15 different processing capabilities. Later during this research, data from this circuit board manufacturer will be used to supplement the model and inferences will be drawn. However, for the purpose of this research it will be assumed that each product requires only one of 15 types of processing capability.

Whenever a particular product is considered for production, capacity to produce that product may not be sufficient. Because one key assumption for the model is that product demand will be satisfied as much as possible, the sum of the additional required capacity will be obtained either by purchasing capacity from outside the manufacturing system and/or reconfiguring an existing idle capacity within the system. This does not guarantee that all products will be produced because acquiring capacity takes time and effort, and depends on the level of agility. For example, assume each week has 40 hours of operation and it takes a total of 80 hours to obtain a certain level of capacity to produce a certain product. That additional capacity will not be ready for production until after two weeks. Because it takes time to compensate for shortage in capacity, not all demand may be satisfied; therefore, available capacity will be devoted to producing the most important products first, which is determined by the operating strategy.

At the beginning of the modeling process, one area of focus should be on the initial system size. One decision variable, γ, addresses the initial number of processing capabilities. However, initial capacity for each capability should also be considered. Various simulation experiments during this research indicate that one important mathematical relationship of the modeling variables is observed to be the following:

$$A_c = (L \div P) \times D \times T$$

where,

A_c = average capacity required per period.

P = average number of introductions periods.

L = average length (in periods) of product life cycle (PLC).

D = average demand (in units) per product during each period.

T = average production time per product.

Based upon analysis of multiple simulation runs, this equation holds after a number of periods equal to L from the start of the simulation. Consequently, total capacity required reaches a steady state (or at least fluctuates around A_c) after an average of L periods. This equation comes in handy because the model assumes that initial system capacity will be equal to the average demand per period. Therefore, the total starting capacity will be equal to A_c and the capacity for each individual capability will be equal to $A_c \div$ initial number of capabilities. Thus, each capability will start with the same amount of capacity as any other. After simulation starts, capacities of capabilities will continuously expand and/or shift around, depending on the manufacturing environment (modeling variables) and design/reconfiguration strategies (decision variables). An algorithm showing how capacities expand and shift around is presented next.

4.1 SIMULATION ALGORITHM

1.0 Set simulation modeling variables (e.g., I_r, N).

2.0 Set ranges for manufacturing environment variables (e.g., UB(P), LB(P)).

3.0 Set cost model variables (e.g., I_p, C_{min}, C_{max}).

4.0 Select low or high values for system decision variables γ, α, τ, and $\delta \in [0, 1]$.

5.0 Implement initial system. Start with $\gamma \cdot M_x$ unique capabilities; capacity of each capability = $A_c \div \gamma \cdot M_x$.

6.0 Set period $j = 1$.

7.0 Generate product requirements for period j.

8.0 Rank products according to their priority. For the product with highest priority during that period $i = 1$, second highest $i = 2$, and so forth.

9.0 Start with product $i = 1$.

10.0 For product i (considered according to rank order), determine which one of the following cases is in effect:

Case 1. Required capability present with enough capacity on that capability to satisfy all demand for product i in period j.

Case 2. Required capability present but not enough capacity to satisfy all demand for product i in period j.

Case 3. Required capability not present at all during period j (also implies no capacity).

11.0 Do one of the following:

a) If required capacity (Cr_{kj}) for product i is less than or equal to available capacity to produce that product (Rc_{kj}), then satisfy all demand by producing Pq_{ij} units. Go to Step 12.0.

b) If required capability is not present, then it will be created.

c) If required capacity is greater than available capacity to produce that product, then determine which type of action to take.

- If capacity shortage (Sc_{kj}) ≤ total extra capacity (Td_j), then:
  i. reconfigure an amount $Dr_{kj} = \delta \times Sc_{kj}$, and:
  ii. expand by purchasing an amount $Dp_{kj} = (1\text{-}\delta) \times Sc_{kj}$, and:
  iii. go to Step 12.0.
- If capacity shortage (Sc_{kj}) > total extra capacity (Td_j), then:
  i. reconfigure an amount $Dr_{kj} = \delta \times Td_j$, and:
  ii. expand by purchasing an amount $Dp_{kj} = Sc_{kj} - (\delta \times Td_j)$, and:
  iii. go to Step 12.0.

12.0 Update all algorithmic variables.

13.0 Calculate all applicable revenues and costs.

14.0 Increase $i = i + 1$. If products exist during period j that are unaccounted for, go to 10.0. Otherwise go to 15.0.

15.0 Increase $j = j + 1$. If j is greater than N, then go to 16.0. Otherwise go to 7.0.

16.0 Calculate the total A_w profit for all periods and stop.

4.2 ABOUT THE PROGRAM

To facilitate modeling of manufacturing environments and conduct 2^k factorial design experiments, a computer program in Turbo C^{++} is developed and presented in the Appendix. The necessity for this computer program stems first from the fact that this is a simulation-based model where many random numbers will have to be generated; and second, because of the tremendous amount of computations required. The reason this program code is included in this thesis is because there are many other algorithms involved in the model which are not presented in Chapter 4 (although the modeling assumptions necessitating those algorithms are clearly stated); such as searching and reserving the necessary capacities at the beginning of each period. It would be nearly impossible to sufficiently demonstrate how the program executes exactly without providing the code. In addition, anyone wishing to further this research or develop similar models will find it much more helpful to have the code readily available.

The model incorporates all the procedures of the 2^k factorial design experiment and constructs 90% confidence intervals for the *t-distribution* with 9 degrees of freedom (presented during next chapter). Also, the presence of random number-generator seeds in the code allows reproduction of the exact same results for each of the three manufacturing environments obtained during this research. Therefore, once the user inputs the same values used in this research and allows the simulation program to execute, the same results will be generated. However, the reader should keep in mind that this program does not analyze the results of the simulation model. It only generates the necessary data for one to be able to analyze and draw conclusions.

When developing the code for the model, the main concern was on making it as short and efficient as possible. There are two reasons for this:

1. So that less time will be spent running the model and waiting for results. Each replication of 500 periods took about 0.8 seconds; and with thousands of replications many hours accumulate.
2. To make it easier for someone to look at the code and understand the algorithms and the basic structure of the model.

Therefore, the code was modified several times to make it more compact; and no particular effort was put forth to create a high-level user interface model. Finally, it should be noted that the program was debugged and compiled and is fully functional on a C^{++} compiler.

CHAPTER 5

EXPERIMENTAL DESIGN

For studying the effects of several variables on a system, 2^k factorial design experiments are very useful. One way to measure the effect of a certain factor (or a decision variable in this case) on the response variable (total profit) is to fix all other variables at certain levels and replicate the simulation model several times. This can be done by incrementally varying the variable of interest and observing how the profit responds. Similarly, the entire process can be repeated with another variable of interest (by fixing all other variables) until all variables are contemplated. Unfortunately, this method has limited validity and lacks proficiency because it requires extensive simulation replications and assumes no interactions exist among factors (Law and Kelton, 1991).

On the other hand, using simulation in 2^k factorial design experiment is more potent for this research because it allows experimentation with alternative configurations of decision variables (Box and Draper, 1987). As previously stated, the objective of this research is to show how design/reconfiguration strategies (which are dictated by system decision variables) should be set (i.e., low vs. high) to maximize profit. Using a 2^k factorial design, each decision variable will have two levels which will be considered either low or high represented by "–" and "+", respectively. This approach is used in correspondence with the 2^k experimental procedure described by Box *et al*. (1978) in which each factor can have just two levels, and each level can be represented by one attribute (for example "–" for blue and "+" for red) or a range of values denoted by either "–" or "+". Deciding on the range of values that correspond to either "–" or "+" depends on the factor of interest and can vary from one type of experiment to another. For example, Law and Kelton (1991) presented a simulation model in which one of the factors represents machining times (in minutes) and the values representing "–" are generated according to a uniform distribution $U(0.65, 0.70)$, and the "+" values generated

according to a uniform distribution $U(0.585, 0.630)$. Notice that higher values are represented by "–" and vice-versa. However, it is generally more convenient to denote low values by "–" and high values by "+" (Box *et al.*, 1978).

Therefore, given that the four decision variables (γ, α, τ, and δ) in this research are normalized between 0 and 1 (i.e., the range of possible values for each factor can be anywhere between 0 and 1), low values corresponding to "–" will be randomly generated from a uniform distribution $U[0, 0.5)$ and high values corresponding to "+" will be from $U[0.5, 1]$. With two levels for each of k factors, the total number of design point combinations is equal to 2^k different design points. Therefore, this research deals with $2^4 = 16$ design points (or design/reconfiguration strategies). Table 5.1 shows a typical 2^k factorial design matrix and each row corresponds to a certain design point. This matrix facilitates the process of computing main effects and interactions.

Design Point	Factor 1	Factor 2	Factor 3	Response
1	–	–	–	R_1
2	+	–	–	R_2
3	–	+	–	R_3
4	+	+	–	R_4
5	–	–	+	R_5
6	+	–	+	R_6
7	–	+	+	R_7
8	+	+	+	R_8

Table 5.1. Design matrix for a 2^3 factorial design. From Law and Kelton (1991).

The *main* effect of each factor f, denoted by e_f, is interpreted as the average change in response resulting from moving factor f from a low "–" level to the high level "+". In other words, e_f (the effect of decision variable f) is the difference between the average profit when f is at a low level and the average profit when it is at a high level. Therefore, with the aid of Table 5.1, e_f can be computed as follows:

Step 1 Substitute the "–" sign by –1 and the "+" sign by +1 in the design matrix.

Step 2 In the column for factor f, multiply the –1 or +1 of each row by the response on the same row. Hence, some responses will change signs and some will not.

Step 3 Sum up all the new responses.

Step 4 Divide that sum by 2^{k-1}. The result will be the value of e_f.

To demonstrate how effects of main factors in Table 5.1 are computed, e_1, e_2, *and* e_3 are calculated as follows:

$$e_1 = \frac{-R_1 + R_2 - R_3 + R_4 - R_5 + R_6 - R_7 + R_8}{2^{3-1}}$$

and

$$e_2 = \frac{-R_1 - R_2 + R_3 + R_4 - R_5 - R_6 + R_7 + R_8}{4}$$

also

$$e_3 = \frac{-R_1 - R_2 - R_3 - R_4 + R_5 + R_6 + R_7 + R_8}{4}$$

It is noted from the above procedure that e_f is the average of all responses (due to a change in factor f) being taken over all possible design point combinations. This way of interpreting results provides limited insight because it assumes that the effect of one factor does not depend on any other, a case rarely factual. When one or more factors are indeed dependent on each other, they are said to be *interactive*. The *interaction* effect of any combination of factors can be computed simply by following the steps outlined before with one exception: in Step 2 all the –1s and +1s on each row (for the factors being considered) will be multiplied by the response value on the same row. For example, the level of interaction between f_1 and f_2 in Table 5.1 can be computed as follows:

$$e_{12} = \frac{R_1 - R_2 - R_3 + R_4 + R_5 - R_6 - R_7 + R_8}{4}$$

This *two-factor* (or *two-way*) interaction effect is interpreted as the difference *between* the following two cases:

1. Average response when factors f_1 and f_2 are both at the same level ("–" or "+").
2. Average response when factors f_1 and f_2 are at opposite levels (one at "–" and the other one at "+").

Three-way (or more) interactions are just as easy to compute but become more difficult to interpret. The level of interaction between f_1, f_2, and f_3 in Figure 5.4 can be computed as follows:

$$e_{123} = \frac{-R_1 + R_2 + R_3 - R_4 + R_5 - R_6 - R_7 + R_8}{4}$$

This measures "half the difference between the average two-factor interaction effect between factors 1 and 2 when factor 3 is at its "+" level, and the average two-factor interaction effect between factors 1 and 2 when factor 3 is at its "–" level" (Law and Kelton, 1991). All interaction effects are symmetrical and, thus, $e_{12} = e_{21}$, $e_{123} = e_{132} = e_{231}$ and so forth. Also, the total number of main effects and interactions is equal to:

$$2^k - 1$$

With k equal to the total number of factors (or 4 decision variables in this research), the total number of main effects and interactions will be $2^4 - 1 = 15$ (which includes the main effects e_1, e_2, e_3, and e_4).

To determine if the main effects and interactions are consistent with practical conclusions rather than illustrated by random fluctuations, confidence intervals must be constructed by replicating the entire process several times. Therefore, for each type of manufacturing environment 10 random samples will be generated, and each sample will be used for each of 10 replications of the 16 design/reconfiguration strategies. In other words, the result will be 100 replications for each design point and type of environment. With 16 different design points the total number of replications is equal to 1600 for each environment. The total number of simulation runs is thus $3 \times [10 \times (16 \times 10)] = 4800$.

5.1 AGILE MANUFACTURING ENVIRONMENTS

To analyze the model and be able to draw reliable inferences, this research attempts to experiment with some collected data. Acquiring data to model a manufacturing environment is relatively easy. However, collecting data pertaining to a manufacturing system is much more difficult to do, and certain assumptions will have to

be made. This research uses limited data from one of the facilities of a circuit board manufacturer; and at the same time assumes certain values for variables for which data were not available. Hence, it should be noted that data pertaining to one particular manufacturer should not be used as the basis for another manufacturer even if both specialize in producing the same type of product. The circuit board manufacturer from which the data were collected specializes in production of very large and expensive industrial-use circuit boards. To be able to withstand high voltages and vibrations, each board can have up to 5,000 pins (about twice the size of regular pins), and most pins on those circuit boards are plated in gold. Hence, those are not the types of circuit boards typically found in television sets and radios, which may be the specialty of other circuit board manufacturers. Naturally, the two manufacturers face completely different environments.

To help provide insight on how design/reconfiguration strategies should be set according to different agile manufacturing environments, experiments will be conducted for three different types of environments. The first environment pertaining to the circuit board manufacturer was briefly discussed and the following data were collected:

LB(P), UB(P) = 3, 6 weeks
LB(L), UB(L) = 8, 15 weeks
LB(D), UB(D) = 950, 1400 units per product per week
LB(T), UB(T) = 2.3, 3.5 hours per product

Figure 5.1 (a) shows a pictorial representation of product life cycles corresponding to this manufacturing environment. The attributes of each PLC (i.e., p_i, l_i, d_i, t_i) were generated according to the above uniform distributions. In addition, with those lower and upper ranges, the means according to the above uniform distributions are as follows:

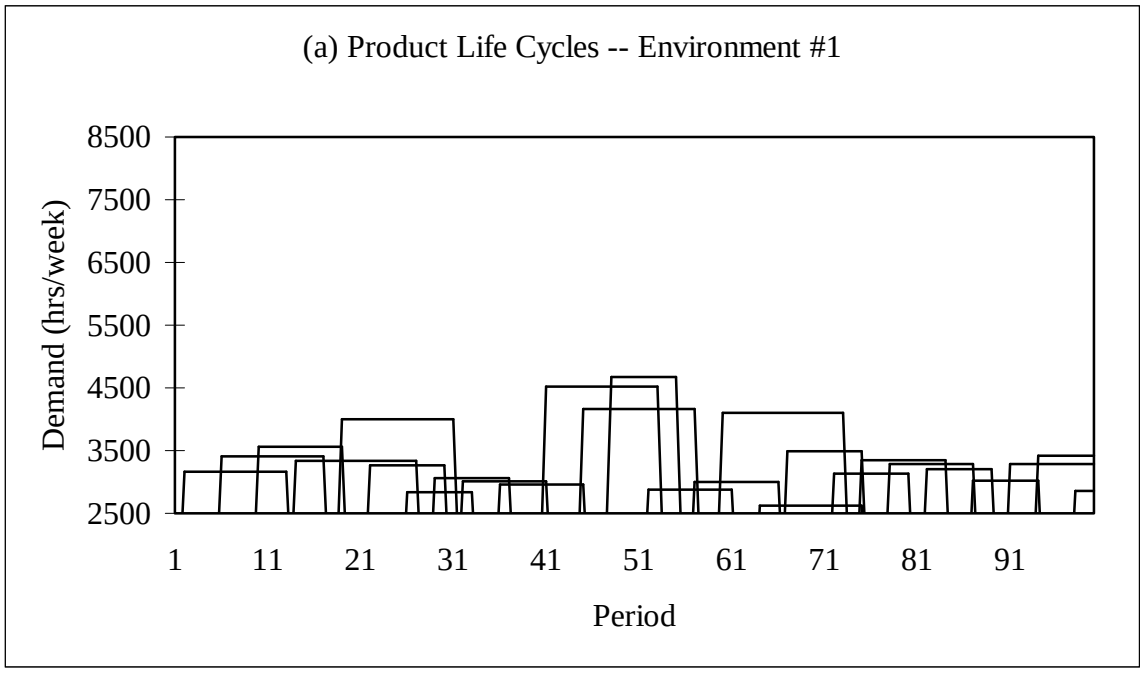
P = 4.5 weeks
L = 11.5 weeks
D = 1175 units per product per week
T = 2.9 hours per product

Using the capacity equation presented in Chapter 4, those values yield an average capacity/period value equal to:

$$A_c = (L \div P) \times D \times T$$

$$A_c = (11.5 \div 4.5) \times 1175 \times 2.9 \cong 8700 \text{ hrs/week}$$

This is close (only 3% lower) to the average capacity required after steady state (i.e., after 12 periods) obtained through simulation modeling (Figure 5.1 (b)), which yielded 8970 hrs/week. This is the average amount of capacity for this particular environment in which different processing capabilities will have to deal with. However, it is obvious from Figure 5.1 (b) that actual demand often either exceeds or falls below the average demand. Because the level of required capacity for each capability could be different in each period, the notion of constantly changing capacities of capabilities now seems conceivable as there are various products being introduced that may have different processing requirements than those that are present or those just eliminated. Although the simulation model runs for 500 periods, Figure 5.1 shows only the first 100 periods.



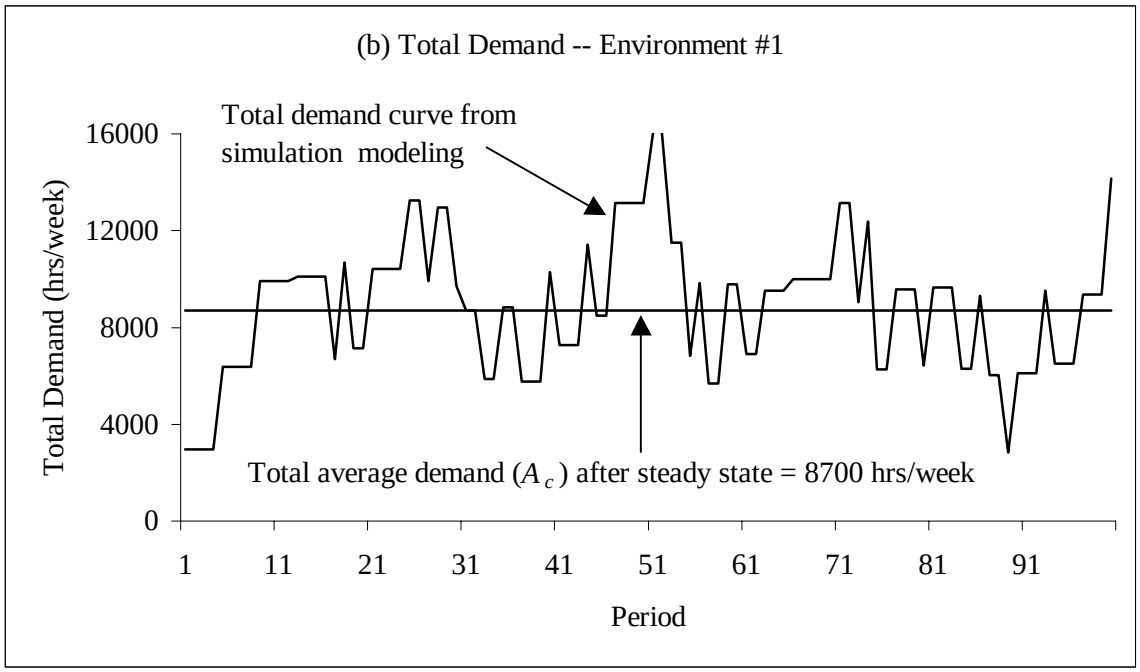


Figure 5.1. (a) product life cycles and (b) total demand, environment #1.

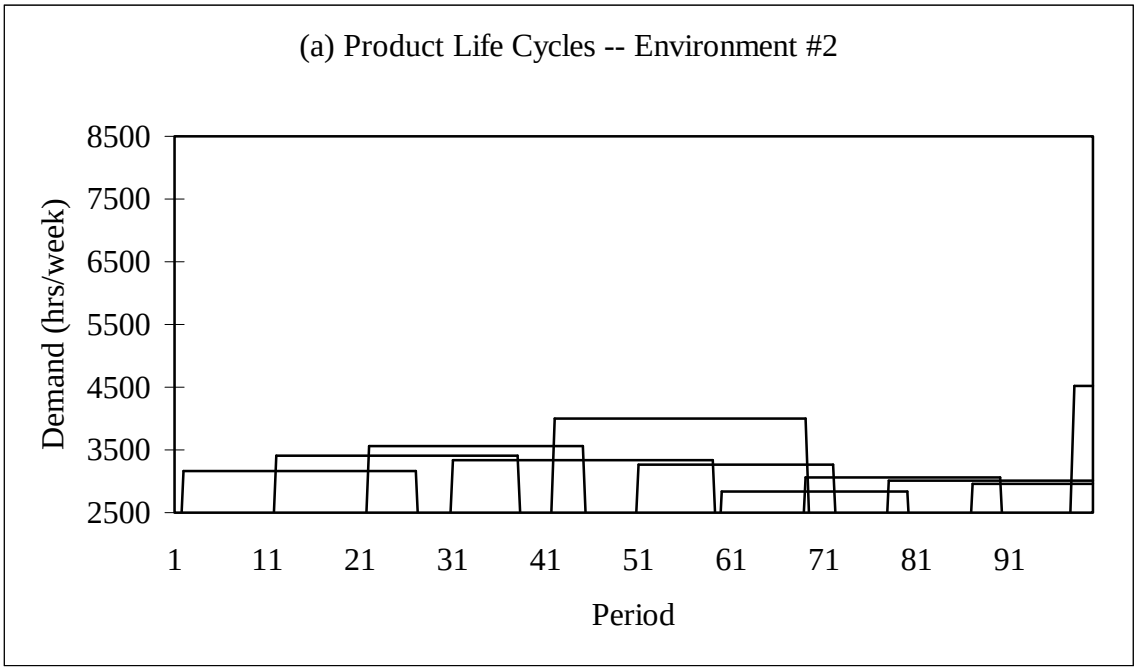
To explore basic changes in environments and determine how they are affected by different design/reconfiguration strategies, two other manufacturing environments will be

contrived, without loss of generality, by holding their required capacities at similar levels. This can be done by using upper and lower ranges in which they yield an average capacity equal to 8700 hrs/week. Table 5.2 presents the values of the original manufacturing environment #1 and new environments #2 and #3.

	Environment		
Input Variable	**#1**	**#2**	**#3**
LB(*P*), UB(*P*) weeks	3, 6	8, 12	16, 24
LB(*L*), UB(*L*) weeks	8, 15	21, 31	21, 30
LB(*D*), UB(*D*) units/product/week	950, 1400	950, 1400	1900, 2800
LB(*T*), UB(*T*) hours/product	2.3, 3.5	2.3, 3.5	2.3, 3.5
Average Capacity Required hours/week	≅ 8700	≅ 8700	≅ 8700

Table 5.2. Input values for 3 different manufacturing environments.

Although the average capacity required per period is similar for all three environments, there are rather basic differences among them. The next two pages present figures showing product life cycles and demand capacities for environments #2 and #3. The product life cycles in Figure 5.1 (a) (environment #1) are short and close to each other. Environment #2 (Figure 5.2 (a)) has longer product life cycles that are also farther than those in environment #1, but with similar heights (demand levels). Finally, environment #3 (Figure 5.3 (a)) shows product life cycles similar in length to those in environment #2; however, they are spread farther apart and are higher. As for demand, Figures 5.1 (b), 5.2 (b), and 5.3 (b) show that demand fluctuations and volatility are strongest in environment #1 and develop some kind of a trend that becomes more obvious when going to environment #2 and then to environment #3.



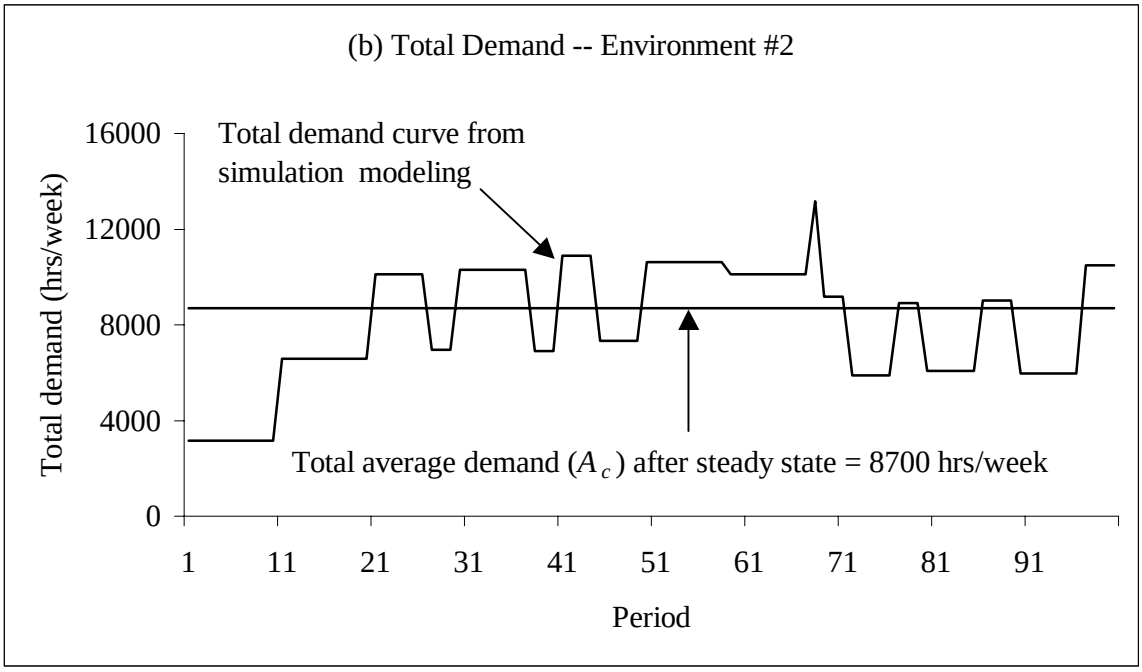
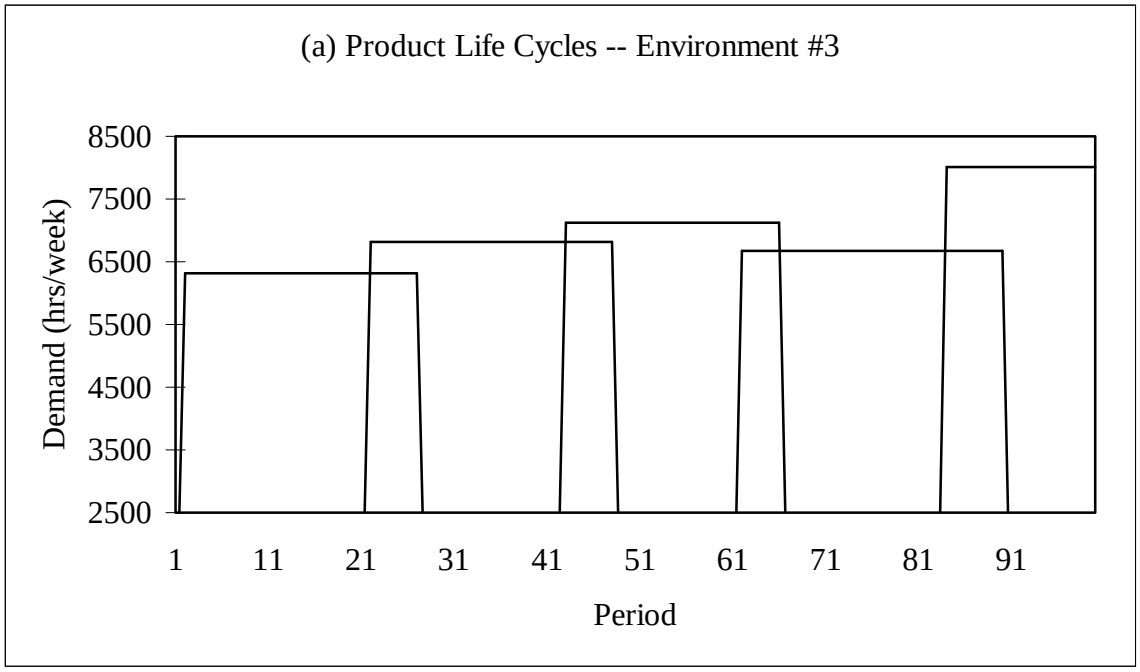


Figure 5.2. (a) product life cycles and (b) total demand, environment #2.



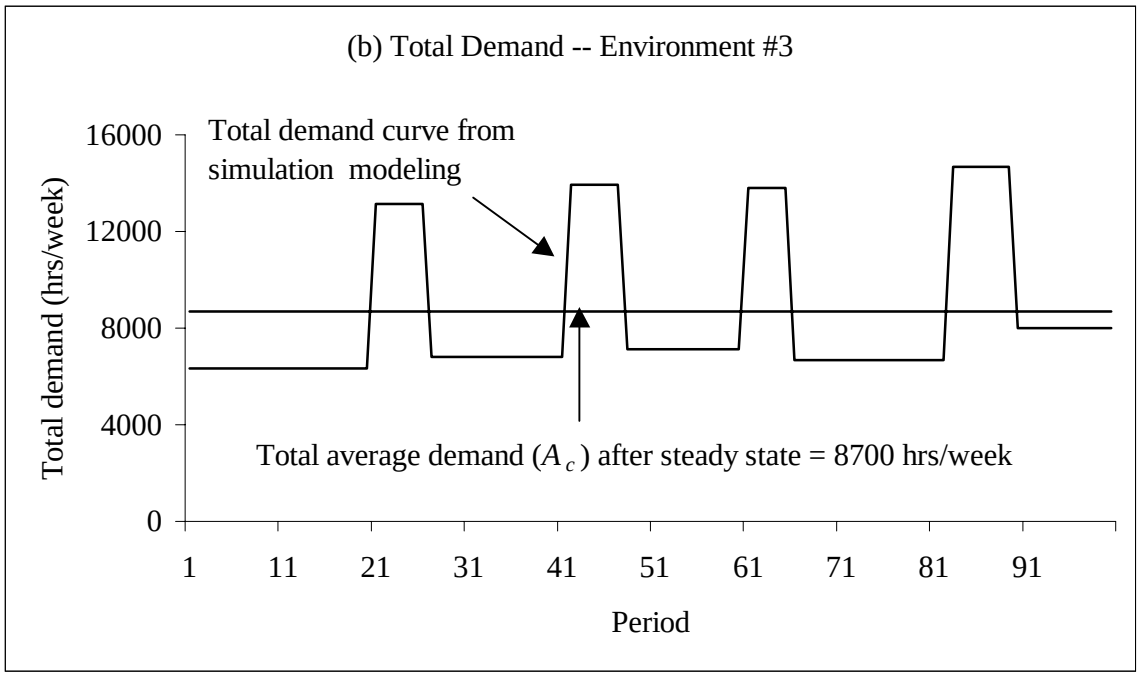


Figure 5.3. (a) product life cycles and (b) total demand, environment #3.

5.2 AGILE MANUFACTURING SYSTEMS

5.2.1 DESIGN FACTORS

As was previously described in Section 3.2.2.1, the four design factors used in this model (also referred to as decision variables) are as follows:

γ = level of initial system size (small vs. large); $\gamma \in [0, 1]$

α = level of system agility (non-agile vs. agile); $\alpha \in [0, 1]$

τ = level of reconfiguration time vs. cost (short and costly vs. long and less costly); $\tau \in [0, 1]$

δ = level of reconfiguration vs. expansion preference (reconfigure vs. expand); $\delta \in [0, 1]$

To facilitate the modeling process, all design factors are normalized on the same scale between 0 and 1. To demonstrate how values for such factors are interpreted, assume they hold the following values:

$$\gamma = 0.6 \quad \alpha = 0.7 \quad \tau = 0.4 \quad \delta = 0.3$$

and $M_x = 15$. The above values will yield a certain amount of profit and can be interpreted as follows:

1. Start the initial system with $0.6 \times 15 = 9$ unique capabilities.
2. The system's level of agility is = 0.7 (70% of maximum).
3. Reconfiguration takes place with a little more emphasis on quick (and costly) reconfiguration rather than slower and less costly reconfiguration.
4. 30% of shortage in capacity in each period is purchased from outside and 70% is reconfigured from inside.

Although this way of interpreting factors' values provides some insight on how the systems should be set, it does not provide any basis on how those factors affect the system. For example, it could be that the initial system status has no effect on the profit and that any value of γ has no significance on the profit in one way or another. On the other hand, the value of γ might have substantial significance on the profit. As was

mentioned before, the significance of all factors, as well as interactions among all factors, can be established with the aid of 2^k factorial design experiments. This way, it will be determined whether or not factors have an effect on the profit and, if so, how.

To develop an understanding of decision variables' effects on the manufacturing environment, an additional response variable, total average cost, will be measured and briefly compared to the relative values of the profit (Table 5.3). The main reason behind this is because what is referred to as *profit* in this model is really the difference between revenues and *some* of the costs involved. While it is expected that sales are the only source of revenue for a company, there are other costs, such as overhead and fixed costs, which have an impact on the profit but not on design/reconfiguration strategies. Data for such costs are difficult to acquire partly due to companies' confidentiality policies; and one has to assume that the actual costs will, by far, exceed the amount from the simulation model. Therefore, because not all costs are accounted for, profit will tend to be high, and the *magnitude* of interaction effects will appear lower than they actually are.

Design Point	γ	α	τ	δ	**Average Profit Response**	**Average Cost Response**
1	–	–	–	–	$R_{1,profit}$	$R_{1,cost}$
2	+	–	–	–	$R_{2,profit}$	$R_{2,cost}$
3	–	+	–	–	$R_{3,profit}$	$R_{3,cost}$
4	+	+	–	–	$R_{4,profit}$	$R_{4,cost}$
5	–	–	+	–	$R_{5,profit}$	$R_{5,cost}$
6	+	–	+	–	$R_{6,profit}$	$R_{6,cost}$
7	–	+	+	–	$R_{7,profit}$	$R_{7,cost}$
8	+	+	+	–	$R_{8,profit}$	$R_{8,cost}$
9	–	–	–	+	$R_{9,profit}$	$R_{9,cost}$
10	+	–	–	+	$R_{10,profit}$	$R_{10,cost}$
11	–	+	–	+	$R_{11,profit}$	$R_{11,cost}$
12	+	+	–	+	$R_{12,profit}$	$R_{12,cost}$
13	–	–	+	+	$R_{13,profit}$	$R_{13,cost}$
14	+	–	+	+	$R_{14,profit}$	$R_{14,cost}$
15	–	+	+	+	$R_{15,profit}$	$R_{15,cost}$
16	+	+	+	+	$R_{16,profit}$	$R_{16,cost}$

Table 5.3. 2^4 factorial design matrix employed for experiments.

The response values in Table 5.3 will be averages of 10 replications of each design point for one random sample of manufacturing environment. Therefore, they yield only one set of main effects and interactions. In other words, one value for each of $e_{\gamma,profit}$, $e_{\gamma,cost}$, $e_{\alpha,profit}$, $e_{\alpha,cost}$, …, $e_{\gamma\alpha\tau\delta,profit}$, and $e_{\gamma\alpha\tau\delta,cost}$. However, the same process is repeated 9 more times (by generating 9 more matrices) with a different random sample of manufacturing environment each time. Hence, the resultant will be a matrix with 10 different values for each of $e_{\gamma,profit}$, $e_{\gamma,cost}$, $e_{\alpha,profit}$, $e_{\alpha,cost}$,…, $e_{\gamma\alpha\tau\delta,profit}$, and $e_{\gamma\alpha\tau\delta,cost}$.

From these, 90% confidence intervals can be constructed by first computing sample means and variances such as:

$$\bar{e}_{\gamma,profit}(10), \qquad \mathrm{Var}[\bar{e}_{\gamma,profit}(10)]$$

Therefore, 90% confidence intervals using *t-distribution* will be:

$$E[e_{\gamma,profit}] = \bar{e}_{\gamma,profit}(10) \pm t_{9,0.95}\sqrt{\frac{\mathrm{Var}[\bar{e}_{\gamma,profit}(10)]}{10}}$$

The above is conducted for all three manufacturing environments. Results will be presented in Chapter 6.

5.2.2 VALUES FOR NON-FACTORS

The values of variables for modeling manufacturing environments were previously presented and discussed. Before proceeding any further, all non-factor values must be specified – these are presented in Table 5.4. As was indicated before, not all the necessary data could be collected and some had to be presumed based upon reasonable deduction. In the case of the circuit board manufacturer, the most difficult data to obtain are associated with the cost functions and all other data were much easier to collect. For example, H_p (number of operation hours/week) is obviously equal to 80 hrs/week because the facility operates 8 hrs/shift, 2 shifts/day, and 5 days/week. Thus, $2 \times 8 \times 5 = 80$ hrs/week. In addition, the maximum possible number of processes (M_x) is indeed equal to 15 for that particular manufacturer, and records indicate that the starting selling price

for the majority of the circuit boards ranges between $1680 and $2115. Other values such as I_r (interest rate per period) are $0.3\% \times 52 = 15.6\%$ APR, which corresponds with what is normally exhibited in most minimum attractive rate of return (MARR) investments.

Variable	Description	Values
N	Number of simulation periods (weeks)	500
I_r	Effective interest rate per period	0.3%
M_x	Maximum number of all possible processes	15
T_{min}	Minimum time (hours) for acquiring (reconfigured or purchased) 1 hour of capacity	0.15
T_{max}	Maximum time (hours) for acquiring (reconfigured or purchased) 1 hour of capacity	0.3
H_p	Number of operation hours per week	80
R_{min}	Minimum initial selling price	$1680
R_{max}	Maximum initial selling price	$2115
P_d	Percent price decline from one period to the next	5%
C_{min}	Minimum cost for purchasing 1 hour of capacity when agility is minimum.	$200
C_{max}	Maximum cost for purchasing 1 hour of capacity when agility is minimum.	$400
D_{min}	Minimum rate at which cost of 1 hour of capacity increases with increasing agility	0.5
D_{max}	Maximum rate at which cost of 1 hour of capacity increases with increasing agility	1.0
A_{min}	Minimum cost of acquiring (reconfigured or purchased) 1 hour of capacity	$50
A_{max}	Maximum cost of acquiring (reconfigured or purchased) 1 hour of capacity	$100

Table 5.4. Input values for non-factors of the simulation model.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 RESULTS OF THE 2^k FACTORIAL DESIGN EXPERIMENTS

To briefly demonstrate how the simulation results are computed (using the values of variables in Table 5.4), Table 6.1 shows the results of 10 replications of each design point for a random sample of manufacturing environment #1. From these 10 replications, the average profit and cost are calculated and then one value for each of the main effects and interactions on the profit and cost are also computed.

Design Point	γ	α	τ	δ	Average Profit Response $\times 10^6$ ($)	Average Cost Response $\times 10^6$ ($)
1	–	–	–	–	3.73	0.23
2	+	–	–	–	4.16	0.29
3	–	+	–	–	4.18	0.31
4	+	+	–	–	4.40	0.25
5	–	–	+	–	3.87	0.38
6	+	–	+	–	3.26	0.42
7	–	+	+	–	4.27	0.34
8	+	+	+	–	4.52	0.34
9	–	–	–	+	3.58	0.47
10	+	–	–	+	4.04	0.25
11	–	+	–	+	4.43	0.17
12	+	+	–	+	3.43	0.19
13	–	–	+	+	2.45	0.59
14	+	–	+	+	2.73	0.57
15	–	+	+	+	3.37	0.54
16	+	+	+	+	3.74	0.31

Table 6.1. Results of the 2^4 experimental design for a random sample of manufacturing environment #1.

For instance, the main effect of γ on profit and cost is as follows:

$$e_{\gamma,\,profit} = \frac{-3.73 + 4.16 - 4.18 + 4.40 - 3.87 + ... + 3.74}{8} = \$0.049 \text{ million}$$

$$e_{\gamma,\cos t} = \frac{-0.23 + 0.29 - 0.31 + 0.25 - 0.38 + ... + 0.31}{8} = -\$0.053 \text{ million}$$

Solving in a similar manner for the other main effects and interactions yields a total of $2 \times 15 = 30$ data points. Moreover, executing the entire process 10 times (with different samples of random environments) will produce a set of 10 data points for each main effect and interaction (i.e., 10 points for each of $e_{\gamma,profit}$, $e_{\gamma,cost}$, $e_{\alpha,profit}$, $e_{\alpha,cost}$, …, $e_{\gamma\alpha\tau\delta,profit}$, $e_{\gamma\alpha\tau\delta,cost}$). From each of these, sample means and variances can be computed to construct 90% confidence intervals with *t-distribution*. For example:

$$\bar{e}_{\gamma,\,profit}(10) = \$0.034 \text{ million}, \qquad \text{Var}[\bar{e}_{\gamma,\,profit}(10)] = \$0.008025 \text{ million}$$

$$E[e_{\gamma,\,profit}] = \bar{e}_{\gamma,\,profit}(10) \pm t_{9,0.95}\sqrt{\frac{\text{Var}[\bar{e}_{\gamma,\,profit}(10)]}{10}}$$

$$E[e_{\gamma,\,profit}] = 0.034 \pm 1.83\sqrt{\frac{0.008025}{10}} = 0.034 \pm 0.052 = \$[-0.018,\ 0.086] \text{ million}$$

This experiment is repeated for all three manufacturing environments. The next two sections present all confidence intervals in tabular and graphical formats for all main effects and interactions on profit and cost. To facilitate the presentation, Table 6.2 summarizes each of the three manufacturing environments as well as the values for all input variables.

Modeling Variables	Environments #1	#2	#3
LB(P), UB(P) weeks	3, 6	8, 12	16, 24
LB(L), UB(L) weeks	8, 15	21, 31	21, 30
LB(D), UB(D) units per product/week	950, 1400	950, 1400	1900, 2800
LB(T), UB(T) hours/product	2.3, 3.5	2.3, 3.5	2.3, 3.5
Average Capacity Required hour/week	≅ 8700	≅ 8700	≅ 8700

Variable	*Description*	*Values*
N	Number of simulation periods (weeks)	500
I_r	Effective interest rate per period	0.3%
γ	Level of initial system size (small vs. large); $\gamma \in [0, 1]$	– / +
α	Level of system agility (non-agile vs. agile); $\alpha \in [0, 1]$	– / +
τ	Level of implementation/reconfiguration time; $\tau \in [0, 1]$	– / +
δ	Level of reconfiguration/expansion preference; $\delta \in [0, 1]$	– / +
M_x	Maximum number of all possible processes	15
T_{min}	Minimum time (hours) for acquiring (reconfigured or purchased) 1 hour of capacity	0.15
T_{max}	Maximum time (hours) for acquiring (reconfigured or purchased) 1 hour of capacity	0.3
H_p	Number of operation hours per week	80
R_{min}	Minimum initial selling price	$1680
R_{max}	Maximum initial selling price	$2115
P_d	Percent price decline from one period to the next	5%
C_{min}	Minimum cost for purchasing 1 hour of capacity when agility is minimum.	$200
C_{max}	Maximum cost for purchasing 1 hour of capacity when agility is minimum.	$400
D_{min}	Minimum rate at which cost of 1 hour of capacity increases with increasing agility	0.5
D_{max}	Maximum rate at which cost of 1 hour of capacity increases with increasing agility	1.0
A_{min}	Minimum cost of acquiring (reconfigured or purchased) 1 hour of capacity	$50
A_{max}	Maximum cost of acquiring (reconfigured or purchased) 1 hour of capacity	$100

Table 6.2. Summary of environments and inputs for experiments.

6.1.1 RESULTS IN TABULAR FORMAT

The lower and upper limits for all main effects and interactions are computed using the simulation model and are presented in Tables 6.3 – 6.5. The first four rows of each table are for the main effects. Those intervals mean that there is a 90% chance that the difference in profit will be within lower and upper ranges by interchanging values of factors (decision variables). For example, referring to the first row of Table 6.3 (a) and assuming, just for now, that γ has no higher-order interaction with any other factors, then changing γ from low to high yields an expected *increase* in profit by the amount of $34,295 per week. Also, there is a 90% chance that the change in profit can be within the lower and upper bounds of –17,550 and $86,139 per week, respectively. Furthermore, because the lower and upper bounds have different signs (i.e., the confidence interval crosses 0), it is interpreted that γ has no statistical significance on the profit of environment #1. In other words, whether the initial system size is small or large has no effect on the profit.

The effects of other main factors and interactions can be interpreted in a similar fashion, and any confidence interval that cross zero will be considered to be statistically insignificant. However, when interpreting results of factorial designs, caution should be rendered not to rush into conclusions simply by looking at the individual effects of main factors. For example, it cannot be concluded that γ has no effect on the profit simply because the confidence interval of main effect of γ crosses zero. It may be that γ has no statistical significance on the profit when other factors are ignored, yet may exhibit statistical significance through a two-way or higher order interaction. Similarly, main effects of certain factors may exhibit significance in a certain direction (such as increasing profit when set high "+"), but opposite effect on the response through two-way (or higher order) interaction.

Thus, if interactions are present, it cannot be interpreted that the main effects by themselves effect the response values. To interpret the results correctly whenever there are multiple factors, all effects (main, two-way interactions, etc.) should be considered to find the highest-order effects which are significant.

		(a) Profit		
	Effect	**Lower Bound**	**Mean**	**Upper Bound**
1	γ	-17550	34295	86139
2	α	375856	472632	569408
3	τ	-609118	-520283	-431449
4	δ	-456750	-375654	-294558
5	γα	-143402	-72530	-1658
6	γτ	-103246	-12798	77650
7	γδ	-59738	5702	71142
8	ατ	276784	341243	405701
9	αδ	67221	152899	238577
10	τδ	-235355	-158794	-82233
11	γατ	-97131	24183	145497
12	γαδ	-99435	-41915	15604
13	γτδ	21753	142903	264053
14	ατδ	82222	120949	159676
15	γατδ	-37654	61249	160153
		(b) Cost		
	Effect	**Lower Bound**	**Mean**	**Upper Bound**
1	γ	-26867	-6386	14094
2	α	-122472	-105190	-87907
3	τ	149835	178199	206562
4	δ	2236	39993	77750
5	γα	-34118	-4320	25477
6	γτ	-24812	-6588	11636
7	γδ	-35964	-9011	17943
8	ατ	-98020	-77770	-57519
9	αδ	-76914	-50360	-23805
10	τδ	27115	63379	99644
11	γατ	-54617	-12690	29238
12	γαδ	-20716	-2257	16203
13	γτδ	-37686	-15298	7090
14	ατδ	-14421	1105	16631
15	γατδ	-37069	-5167	26735

Table 6.3. 90% confidence intervals for the main effects and interactions on (a) profit and (b) cost, environment #1.

		(a) Profit		
	Effect	**Lower Bound**	**Mean**	**Upper Bound**
1	γ	-59685	-26768	6149
2	α	215539	249245	282952
3	τ	-332776	-295558	-258339
4	δ	-214678	-152177	-89676
5	γα	-24994	18599	62192
6	γτ	-90061	-28907	32248
7	γδ	-34086	18170	70426
8	ατ	135146	198241	261336
9	αδ	-24292	36427	97146
10	τδ	-134738	-105673	-76609
11	γατ	-69839	-22547	24745
12	γαδ	-71036	-21832	27372
13	γτδ	-90404	-51725	-13047
14	ατδ	12831	52814	92797
15	γατδ	-55541	-11891	31760
		(b) Cost		
	Effect	**Lower Bound**	**Mean**	**Upper Bound**
1	γ	-16392	-3052	10288
2	α	-65574	-48307	-31040
3	τ	105766	120382	134998
4	δ	6719	31866	57013
5	γα	-18493	-2477	13540
6	γτ	-13256	5728	24711
7	γδ	-7683	6331	20346
8	ατ	-42006	-28056	-14107
9	αδ	-50413	-27404	-4395
10	τδ	22807	37490	52172
11	γατ	-24158	-8459	7240
12	γαδ	-24377	-5236	13905
13	γτδ	56	14973	29890
14	ατδ	677	12832	24988
15	γατδ	-26343	-7795	10753

Table 6.4. 90% confidence intervals for the main effects and interactions on (a) profit and (b) cost, environment #2.

		(a) Profit		
	Effect	**Lower Bound**	**Mean**	**Upper Bound**
1	γ	-172739	-80041	12656
2	α	324330	390031	455733
3	τ	-510658	-438304	-365950
4	δ	23026	96355	169684
5	γα	-54499	37496	129492
6	γτ	-45383	4829	55041
7	γδ	-60058	-13099	33860
8	ατ	196611	244251	291892
9	αδ	-163358	-85114	-6869
10	τδ	-109405	-36719	35967
11	γατ	-140728	-71052	-1377
12	γαδ	-64893	-10596	43701
13	γτδ	-134926	-56494	21938
14	ατδ	-88236	-21182	45872
15	γατδ	-64238	602	65441
		(b) Cost		
	Effect	**Lower Bound**	**Mean**	**Upper Bound**
1	γ	-26876	-11148	4580
2	α	-85077	-67464	-49851
3	τ	102981	113837	124692
4	δ	-150800	-144793	-138787
5	γα	-5820	9593	25006
6	γτ	-11981	-1830	8321
7	γδ	-19626	-2370	14887
8	ατ	-59414	-44408	-29402
9	αδ	13788	28761	43734
10	τδ	-59620	-49446	-39272
11	γατ	-5764	7435	20634
12	γαδ	-13185	2525	18234
13	γτδ	-19849	-4509	10831
14	ατδ	-59	16749	33556
15	γατδ	-28268	-5295	17679

Table 6.5. 90% confidence intervals for the main effects and interactions on (a) profit and (b) cost, environment #3.

6.1.2 RESULTS IN GRAPHICAL FORMAT

To aid in identifying statistically significant effects and interactions, as well as the magnitude of such effects, graphical presentation of results is helpful. Accordingly, the results are presented in graphical format in Figures 6.1 – 6.3. For each effect, the mean response and 90% confidence interval are shown. Once again, any effect for which the confidence interval does not cross zero is considered to be statistically significant. The farther away the interval is from zero, the stronger the magnitude of the effect or interaction.

Statistical *insignificance* between two or more factors does not necessarily mean that they do not affect the profit; it only means that setting one factor at a certain level does not instigate the way another factor should be set to influence the profit. In other words, the factors that do not show any interaction effect are independent of each other, but yet each may have an effect on the profit. Once interaction effects are computed, the highest-order interaction effects are selected and the procedure presented by Box *et al*. (1978) can be used to determine how the factors should be set to maximize profit. This will be presented shortly.

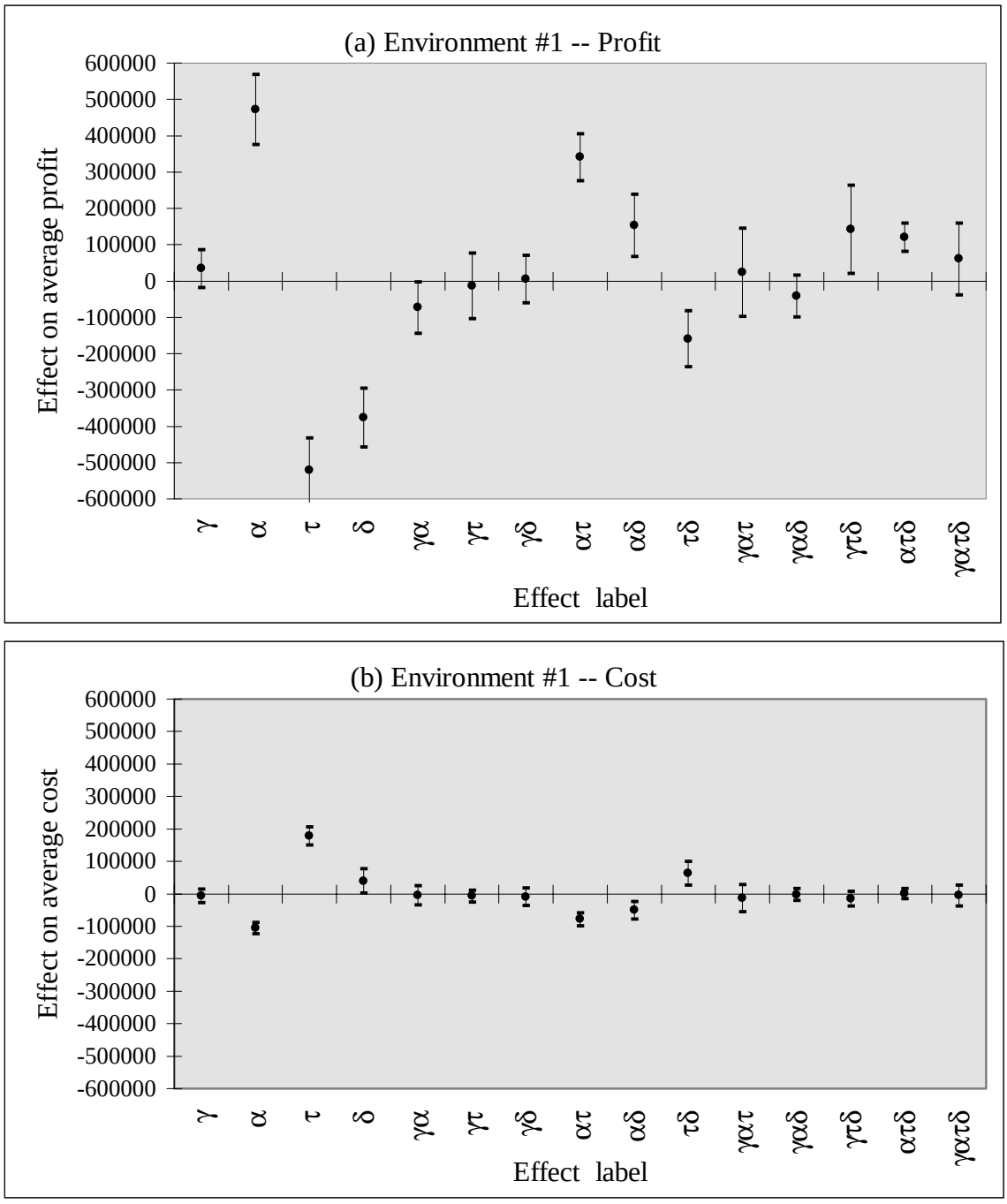


Figure 6.1. Level of significance for the main effects and interactions on (a) profit and (b) cost, environment #1.

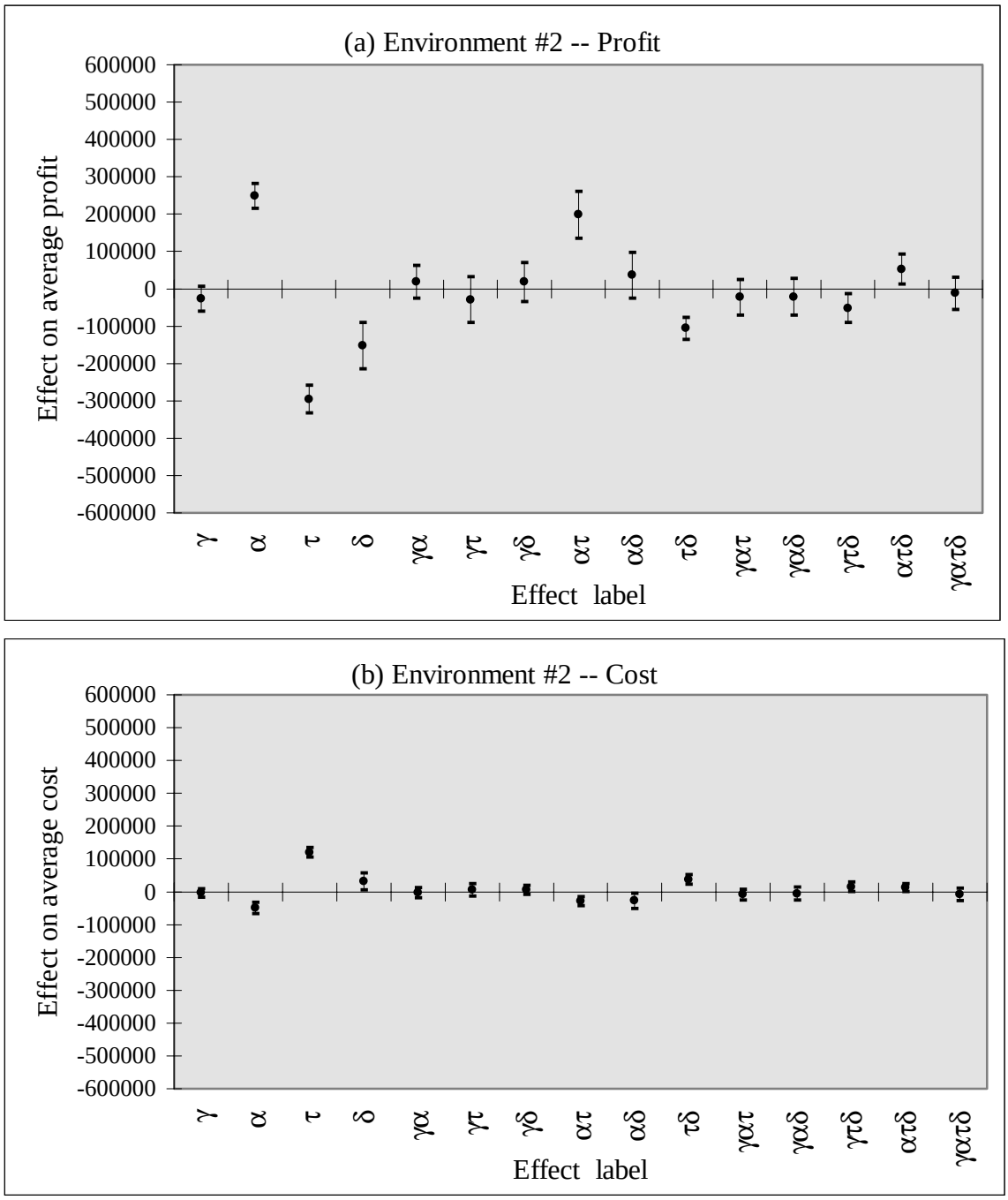


Figure 6.2. Level of significance for the main effects and interactions on (a) profit and (b) cost, environment #2.

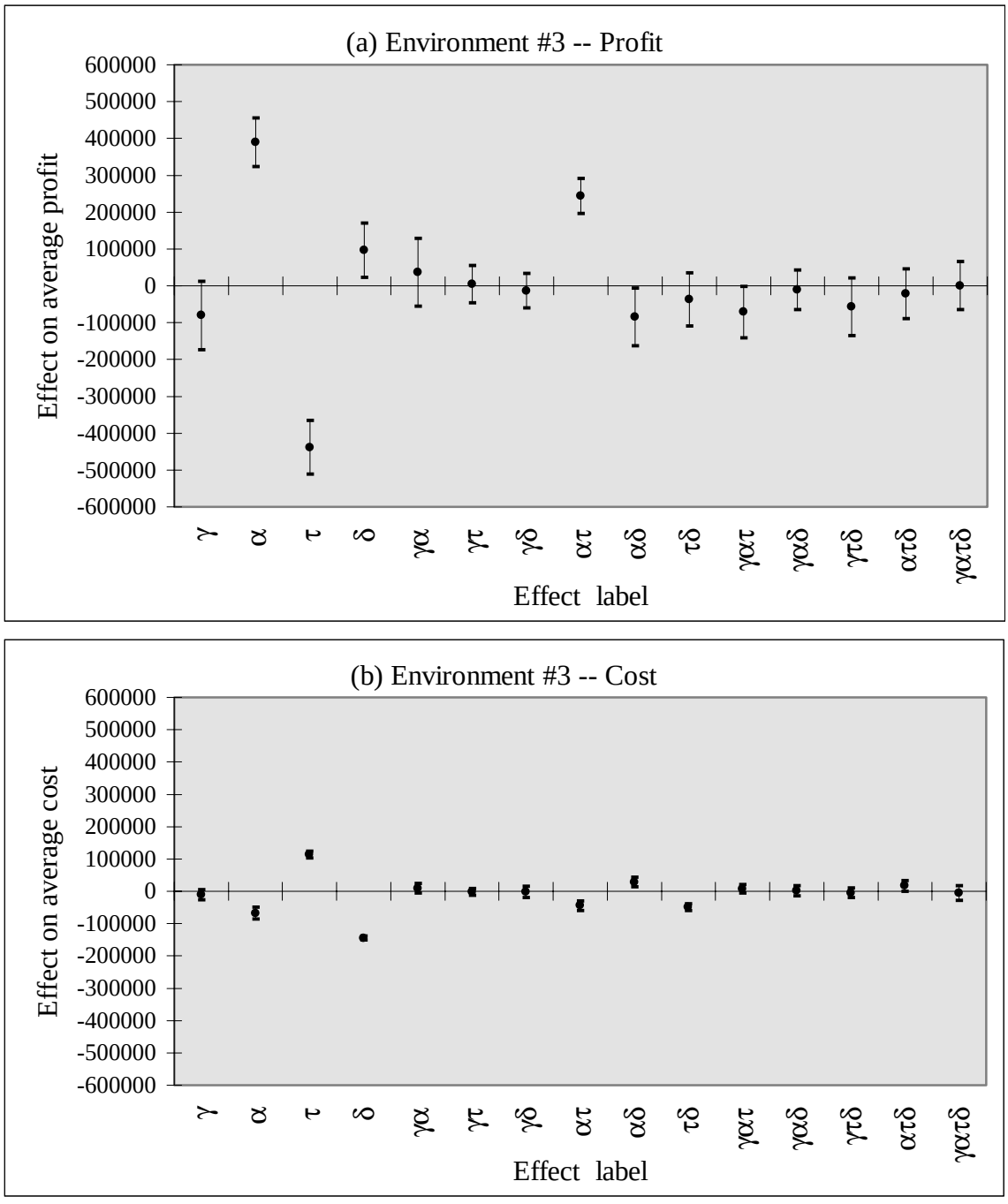


Figure 6.3. Level of significance for the main effects and interactions on (a) profit and (b) cost, environment #3.

6.2 DISCUSSION OF THE RESULTS

A substantial amount of discussion and analyses can be inferred from results of factorial design experiments in this research. However, although there are a total of 15 main effects and interactions for each manufacturing environment, not all of them yield statistical significance or have any effect on the profit. Based upon the 90% confidence intervals in Tables 6.3 – 6.5 and Figures 6.1 – 6.3, the statistically significant effects of the three manufacturing environments are presented in Table 6.6. The highest-order effects encompassing all factors are identified by enclosed frames, and only they can be used to draw conclusions regarding interaction effects. Note that the enclosed frames for each environment account for all decision factors (γ, α, τ, and δ).

	Effects														
Env.	γ	α	τ	δ	$\gamma\alpha$	$\gamma\tau$	$\gamma\delta$	$\alpha\tau$	$\alpha\delta$	$\tau\delta$	$\gamma\alpha\tau$	$\gamma\alpha\delta$	$\gamma\tau\delta$	$\alpha\tau\delta$	$\gamma\alpha\tau\delta$
#1	-	+ve	-ve	-ve	-	-	-	+ve	+ve	-ve	-	-	+ve	+ve	-
#2	-	+ve	-ve	-ve	-	-	-	+ve	-	-ve	-	-	-ve	+ve	-
#3	-	+ve	-ve	+ve	-	-	-	+ve	-ve	-	-ve	-	-	-	-

Table 6.6. Main effects and interactions with statistical significance on profit.

To determine how the variables should be set (low or high) to maximize profit, a method presented by Box *et al*. (1978) is used with the help of the average response (profit) for each design point. Table 6.7 presents the average responses where, as can be recalled, for each design point 10 random samples of a manufacturing environment were generated and each random sample was replicated 10 times. Therefore, with 16 different design points and three different types of manufacturing environments, the total number of replications is equal to 4800, which is the result of multiplying 16 design points $\times$ 3 manufacturing environments $\times$ 10 random samples per design point per manufacturing environment $\times$ 10 replications per random sample (with different random number-generator seeds). Note that each average value presented in environment #1 in Table 6.7 is actually the grand average of 10 different replications, each of which is similar to Table 6.1.

Design Point	γ	α	τ	δ	Env. #1		Env. #2		Env. #3	
					Profit	Cost	Profit	Cost	Profit	Cost
1	–	–	–	–	3.98	0.27	2.87	0.16	2.75	0.26
2	+	–	–	–	4.15	0.26	2.76	0.16	2.51	0.25
3	–	+	–	–	4.07	0.29	2.89	0.18	2.84	0.23
4	+	+	–	–	4.26	0.29	2.88	0.18	2.83	0.21
5	–	–	+	–	3.41	0.45	2.51	0.29	1.95	0.49
6	+	–	+	–	3.35	0.47	2.46	0.27	1.97	0.47
7	–	+	+	–	4.02	0.33	2.83	0.23	2.72	0.32
8	+	+	+	–	3.83	0.33	2.83	0.22	2.69	0.33
9	–	–	–	+	3.76	0.30	2.74	0.22	2.91	0.16
10	+	–	–	+	3.87	0.29	2.79	0.19	2.76	0.14
11	–	+	–	+	4.13	0.21	2.78	0.13	2.88	0.14
12	+	+	–	+	3.84	0.22	2.86	0.14	2.94	0.14
13	–	–	+	+	2.48	0.62	2.12	0.36	2.18	0.26
14	+	–	+	+	2.68	0.61	2.11	0.39	2.08	0.23
15	–	+	+	+	3.59	0.41	2.71	0.29	2.75	0.18
16	+	+	+	+	3.73	0.35	2.61	0.27	2.56	0.17

Table 6.7. Average profits and costs $\times 10^6$ ($) at different design points for three different manufacturing environments.

6.2.1 MANUFACTURING ENVIRONMENT #1

LB(P), UB(P) weeks	3, 6 weeks
LB(L), UB(L) weeks	8, 15 weeks
LB(D), UB(D) units	950, 1400 units/product/week
LB(T), UB(T) hrs	2.3, 3.5 hours/product
Average capacity required, hours/week	≅ 8700

According to environment #1 in Table 6.6, interaction effects are exhibited among factors α, τ, δ, and γ, τ, δ. Therefore, the first interaction effect implies that in order to increase profit, setting any one of α, τ, or δ depends on the way the other two are set. Similar interpretation can be made for the γτδ interaction. From this, it is clear that all factors have interaction effects among each other except between α and γ. In order to

determine how the factors should be set to maximize profit, the average profits presented in Table 6.7 are displayed geometrically in Figure 6.4, where the numbers in bold case are averages of the two options at each point (i.e., setting the forth variable that does not exhibit interaction effect low or high).

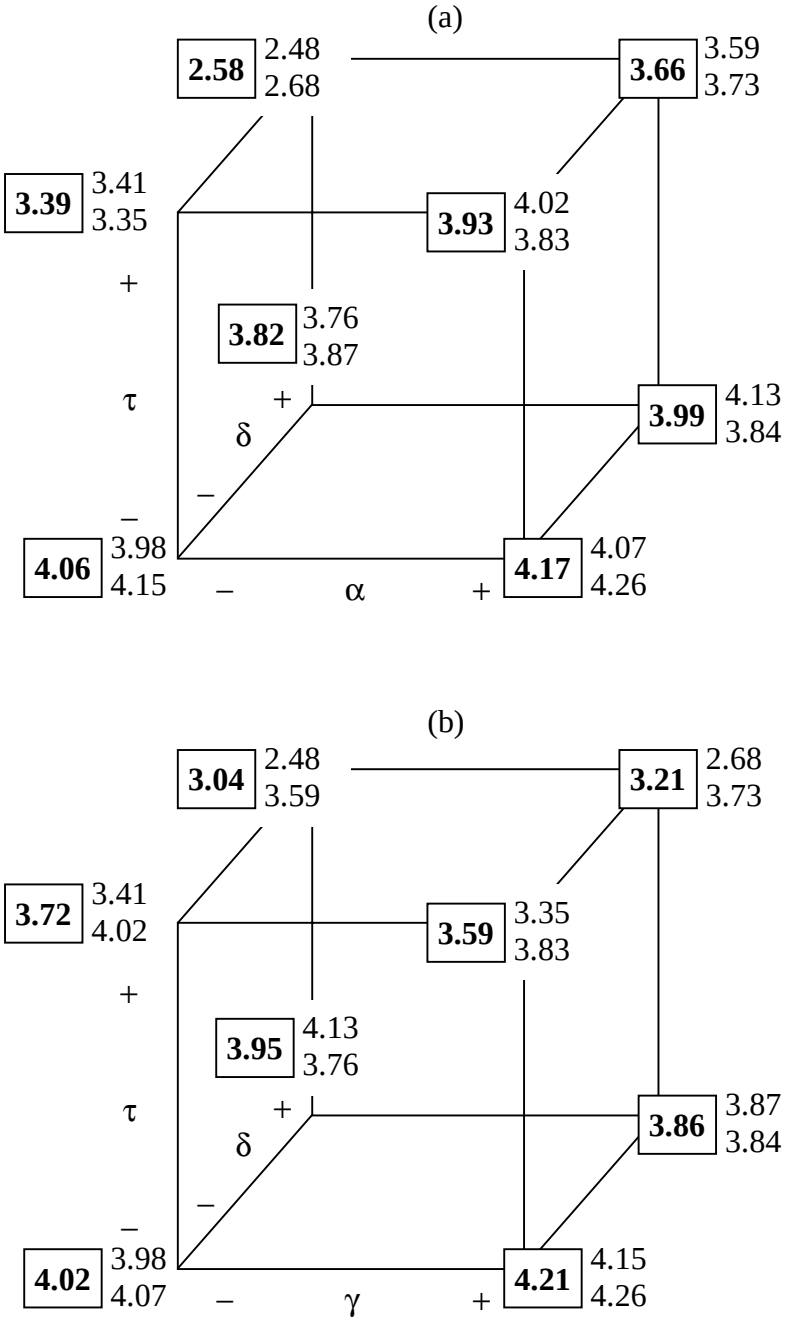


Figure 6.4. Interaction effects among (a) ατδ and (b) γτδ, environment #1.

Figure 6.4 (a) suggests that profit is highest when α is high (+) and both τ and δ are low (-). This is because average profit yield is \$4.17M and is higher than for any other average profit. At this point, no inferences can be made regarding γ. On the other hand, Figure 6.4 (b) suggests that profit is highest when γ is high and both τ and δ are low, which yield an average profit of 4.21M. Therefore, it is concluded that in order to maximize profit both γ and α should be set high and both τ and δ set low. Furthermore, note that while there is no interaction effect between γ and α, they both have an effect on profit. In short, profit can be maximized in environment #1 by doing the following:

1. Start with a large initial system size (γ = high).
2. Acquire a highly agile system (α = high).
3. Reconfigure quickly, at the expense of higher cost (τ = low).
4. Reconfigure more capacity from within the system rather than purchasing from outside (δ = low).

Once all factors are appropriately set to maximize profit, if for any reason one of the factors has to be set at an opposite level, then settings of all other factors will have to be re-evaluated because some may need to be reset in the opposite direction. For example, if δ is to be kept high (regardless of its effect on profit), then Figure 6.4 (a) suggests that keeping α high and τ low still have a positive influence on profit. However, according to Figure 6.4 (b) while τ should remain low, γ should now be set low too. Therefore, forcing the level of δ from low to high will cause γ to be set differently to increase profit.

Although both types of interaction effects are statistically significant, their magnitudes do not necessarily indicate *practical* significance. The average magnitude of the $\gamma\tau\delta$ interaction on profit is around \$143K and for the $\alpha\tau\delta$ interaction it is around \$121K (Table 6.3 (a)); which means that the $\gamma\tau\delta$ interaction effect is stronger and can have more influence on the profit than the other one. In addition, the 90% confidence interval for the $\gamma\tau\delta$ interaction is larger than for the $\alpha\tau\delta$ interaction and, therefore, the $\alpha\tau\delta$ interaction appears to have more consistency on the profit than the other $\gamma\tau\delta$

interaction. However, whether or not the magnitude is practically significant is largely dependent on each individual's own perception.

6.2.2 MANUFACTURING ENVIRONMENT #2

LB(*P*), UB(*P*) weeks	8, 12 weeks
LB(*L*), UB(*L*) weeks	21, 31 weeks
LB(*D*), UB(*D*) units	950, 1400 units/product/week
LB(*T*), UB(*T*) hrs	2.3, 3.5 hours/product
Average capacity required, hours/week	≅ 8700

Confidence intervals for environment #2 indicate that the highest-order interactions with statistical significance are for factors γ, τ, δ and α, τ, δ (Table 6.6), which is similar to environment #1. Using the same procedure as before, the averages corresponding to each interaction effect are displayed geometrically in Figure 6.5. According to Figure 6.5 (a), profit is highest when α is high and both τ and δ are low, with average profit yield at \$2.89M. Also, Figure 6.5 (b) suggests that profit is highest when all γ, τ, and δ are low, which yield an average profit of \$2.88M. Therefore, it is concluded that in order to maximize profit α should be set high and all three γ, τ, and δ set at low levels. Once again, there is no interaction effect between α and γ, but in the presence of the two other factors (τ and δ) they both have an effect on the profit. To summarize, profit can be maximized in environment #2 by doing the following:

1. Start with a small initial system size (γ = low).
2. Acquire a highly agile system (α = high).
3. Reconfigure quickly, at the expense of higher cost (τ = low).
4. Reconfigure more capacity from within the system rather than purchasing from outside (δ = low).

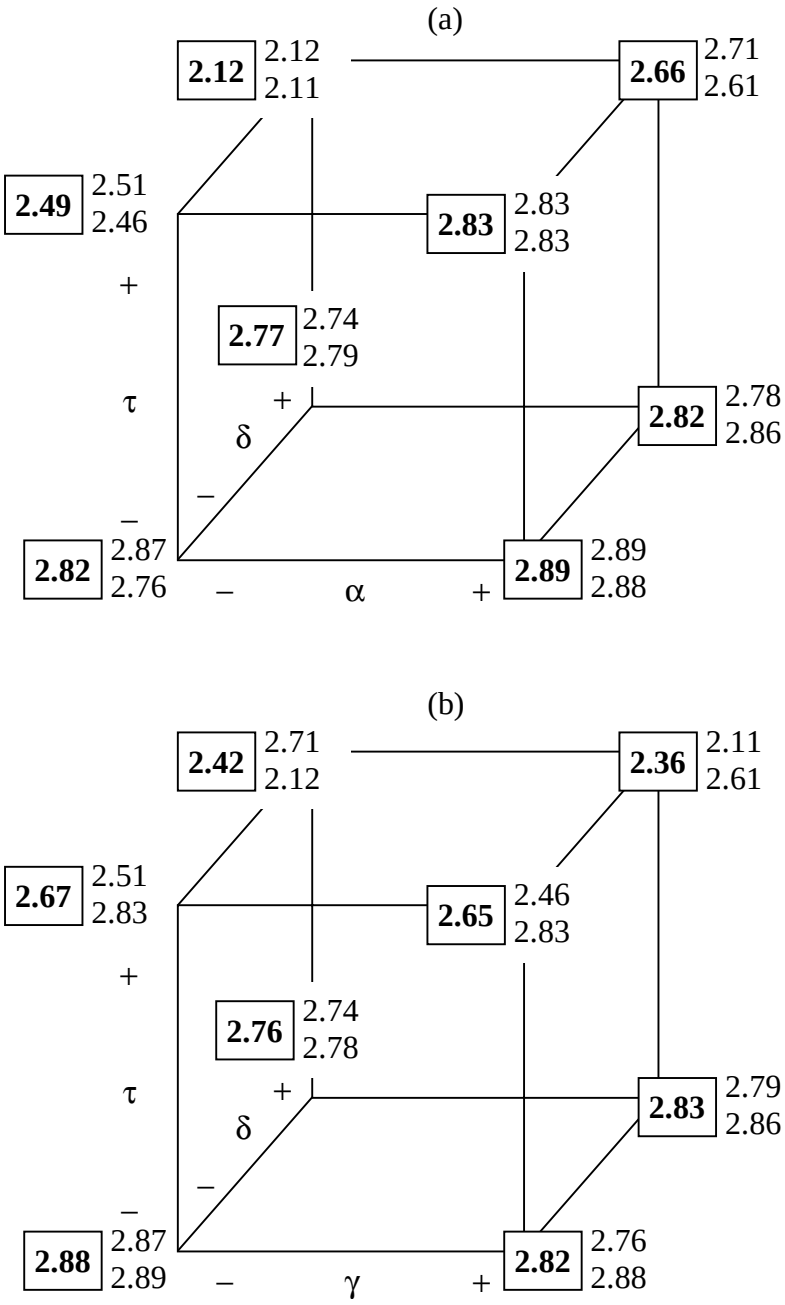


Figure 6.5. Interaction effects among (a) ατδ and (b) γτδ, environment #2.

Similar to what was discussed in environment #1, changing the level of one factor in environment #2 may require other variables to be reset as well. Figure 6.5 (b) indicates that if γ is forced to be set at a high level, to increase profit τ may be left at a

low level but δ will now have to be set at a high level. Although in this example the difference in profit between setting δ low or high is not substantial ($2.83M vs. $2.82M), this at least demonstrates that resetting one factor can affect the way other factors should be set to maximize profit.

The magnitude of effect for both significant interaction effects on the profit are similar and around $52K (Table 6.4 (a)). In addition, the ranges of the 90% confidence intervals for both interaction effects are somewhat similar. Therefore, both types of interaction effects have similar consequences on the profit and are weaker than the interaction effects in environment #1.

Analyzing the differences between environments #1 and #2, it appears that as the number of periods between new product introduction (*P*) and the length of PLC (*L*) increase, while demand capacity is constant, it becomes best to start with a small system and levels of other factors should remain as before. The same types of interaction effects ($\gamma\tau\delta$ and $\alpha\tau\delta$) still exist for the two environments. However, the magnitudes of interactions are weaker in environment #2 (i.e., closer to zero).

6.2.3 MANUFACTURING ENVIRONMENT #3

LB(*P*), UB(*P*) weeks	16, 24 weeks
LB(*L*), UB(*L*) weeks	21, 30 weeks
LB(*D*), UB(*D*) units	1900, 2800 units/product/week
LB(*T*), UB(*T*) hrs	2.3, 3.5 hours/week
Average capacity required, hours/week	$\cong$ 8700

Confidence intervals for environment #3 indicate that the highest-order interactions with statistical significance are for γ, α, τ and α, δ (Table 6.6). Because for this environment the highest-order significant effect that accounts for δ is a two-way interaction effect $\alpha\delta$, geometrical representation differs slightly than for three-way interaction effects. Using the same principles as before, the average profits corresponding to each interaction effect are displayed geometrically in Figure 6.6.

Considering the $\gamma\alpha\tau$ interaction, Figure 6.6 (a) indicates that profit is maximized when both γ and α are high and τ is low. As for α and δ, Figure 6.6 (b) indicates that profit is maximized when they are both high. Therefore, α is the only factor that interacts with all other factors --although it is worth noting that the $\alpha\delta$ interaction is not particularly strong when α is high. Hence, it is concluded that profit can be increased by setting γ, α, and δ high and only τ low. In other words, profit can be maximized in environment #3 by doing the following:

1. Start with a large system (γ = high).
2. Acquire a highly agile system (α = high).
3. Reconfigure quickly although at the expense of higher cost (τ = low).
4. Focus on purchasing more capacity from outside the system rather than reconfiguring from inside (δ = high).

Once all factors in environment #3 are appropriately set to increase profit, resetting either γ or δ does not lead any other factors to be reset as well. However, if α is reset in the opposite direction (i.e. forced to be low), then in order to maximize profit under this new constraint, only γ will have to be low. Similarly, if τ is forced to be high, then only γ will have to be changed to low level. Although confidence intervals reveal interaction effects among factors, they do not indicate the levels at which the interactions occur. As was indicated before, notice that although there is interaction effect between α and δ, the interaction effect between the two is week when α is high because the average profit remains relatively similar ($2.77M vs. $2.78M).

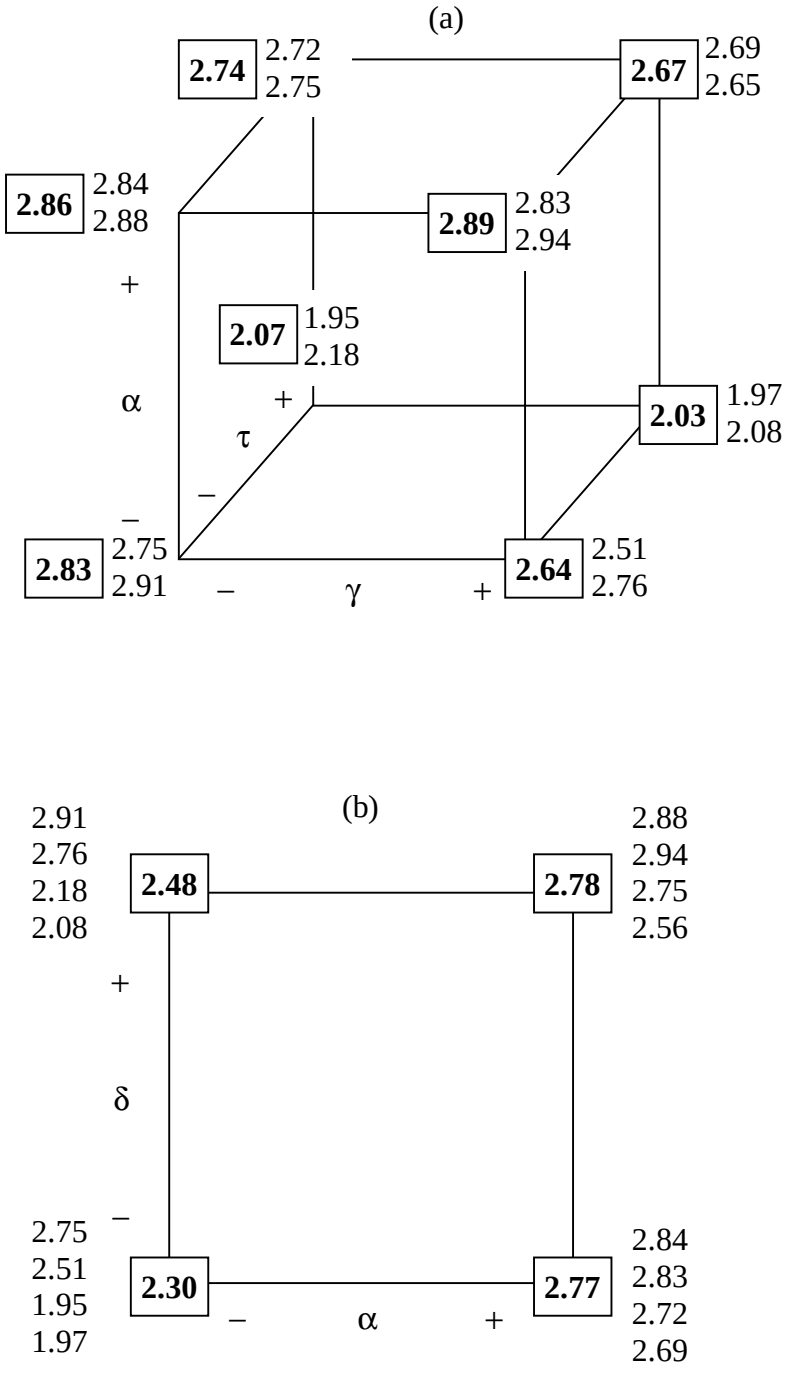


Figure 6.6. Interaction effects among (a) γατ and (b) αδ, environment #3.

Analysis of results between environments #2 and #3 show that as *P* and *D* (product demand per period) increase, δ should be changed from low to high. On the other hand, the levels of α and τ should remain high and low, respectively. The results also indicate that increasing *P* and *D* changes the significant interaction effects from $\alpha\tau\delta$ and $\gamma\tau\delta$ to $\gamma\alpha\tau$ and $\alpha\delta$.

In conclusion of this chapter, it is worth noting that the profits associated with each design point are obviously much higher than the costs involved (Table 6.7). This is expected in this situation because not all costs were considered. In fact, if those modeled costs were high or near the profit, then there would be room for concern. Recall that other costs, although very important, were assumed to be either fixed or unaffected by the decision variables of this model. This does not limit the validity of the model because this research neither addresses nor is concerned with the magnitude of the true profit. That is the reason why this research does not endeavor to search for optimal decision values; and the magnitude of interaction effect should in no way be confused with the magnitude of profit. Presence of significant interaction effects do not automatically indicate that the profit will have a positive value. Interaction effects are merely convenient means of deciding whether or not they have influence on profit.

In addition, statistical significance simply asserts that the difference in profit might change in one way or another, depending on the way decision variables are set. For instance, it was established in environment #1 that setting both γ and α high and both τ and δ low will yield the best possible profit. However, if other costs are too high, the best possible profit may still be a loss that only improved from being considerably negative to just below zero. In both cases profit can still maintain a negative value, but setting the decision variables the way just mentioned made it less severe.

CHAPTER 7

CONCLUSIONS, CONTRIBUTIONS, & FUTURE WORK

7.1 CONCLUSIONS

As was indicated before, the objective of this research is to identify a set of factors comprising system design/reconfiguration strategies, and provide some fundamental insight on how these factors should be set (low or high) for different agile manufacturing environments. With the aid of 2^k factorial design experiments, credible results have been obtained and the most significant findings are summarized below. It should be made clear once more that these findings are based upon particular models, assumptions, and data, and may not extend to other situations. The reader is advised to refer to Figures 6.1 (a), 6.2 (a), and 6.3 (a) when reading the following conclusions:

1. In environment #1, the initial system size exhibits interaction effects with (a) the level of reconfiguration time vs. cost and (b) with the level of reconfiguration vs. expansion. However, as the number of periods between new product introduction and the length of PLC increase (environment #2), the interaction remains statistically significant but the magnitude declines as confidence intervals get closer to zero and changing initial size (low vs. high) has little effect on profit. When further increasing the number of introduction periods and demand per unit type (environment #3), the interaction effect of initial system size declines further and its effect on profit also declines. Therefore, the initial system size becomes less relevant as the number of introduction periods, length of PLC, and demand per unit type increase.

2. The level of agility should be high and has consistently shown interaction effect with (a) the level of reconfiguration time vs. cost and (b) the level of reconfiguration vs. expansion. However, as the number of periods between new product introduction and the length of PLC increase, the interaction remains significant but the magnitude declines. When further increasing the number of introduction periods and also

demand per unit type, the magnitude of effect between agility and reconfiguration time vs. cost increases, and also the magnitude of effect between agility and reconfiguration vs. expansion increases but very little.

3. Reconfiguration should consistently be done quickly and in environment #1 has shown to interact with (a) initial system size, (b) the level of reconfiguration time vs. cost, and (c) with the level of reconfiguration vs. expansion. However, as the number of periods between new product introduction and the length of PLC increase, the interaction remains statistically significant but the magnitude declines as the confidence intervals get closer to zero. When further increasing the number of introduction periods and also demand per unit type, the interaction effect is maintained only with the initial system size and agility (i.e., it no longer interacts with reconfiguration vs. expansion).

4. In environments #1 and #2, reconfiguration should be from within the system rather than from outside, and it interacts with (a) initial system size, (b) level of agility, and (c) the level of reconfiguration time vs. cost. The magnitude of interaction becomes less in environment #2, and in environment #3 it only exhibits interaction effect with agility. However, in environment #3 the focus should be on purchasing more capacity from outside the system rather than reconfiguring from inside.

Though the above findings are for particular models, assumptions, and data, they do provide insights leading to some general conclusions. First of all, the results suggest that although the initial system size has some form of interaction effect with other factors in all three manufacturing environments, its effect on profit declines as the number of periods between new product introduction, length of PLC, and demand per unit type increase. The fact that initial system size does not have as strong an effect on profit as the other factors could be the reason why no general trends are exhibited regarding what the initial size should be. However, this was expected and does not render any surprises. In fact, if the simulation was performed for more than 500 periods, then the initial system size would have even less of an effect on profit. Another expected result is that high agility is most important when PLCs are short, which is due to the fact that new products

are coming into the system at a high rate and the system has to be reconfigured continuously as well. Furthermore, it is observed that high agility is also preferred when demand per product is high (high PLCs). It was mentioned in Chapter 2 that in today's environments PLCs are getting shorter and higher, and this research concludes that the shorter and higher PLCs become, the higher the level of agility should be and the more significant role it plays. One other conclusion regarding agility is that it always exhibits interaction effect with reconfiguration time vs. cost and the level of reconfiguration vs. expansion. That was expected because agility reflects the effort involved in making the change, and that affects the tradeoffs between reconfiguration time and cost and also how much capacity is reconfigured and readily available.

Parallel to what is deduced for agility, quick reconfiguration becomes more important when PLC is short and high (i.e., product in demand for a short periods of time but in high volumes during each period). This is probably due to the fact that if reconfiguration is slow then demand for that product will probably die-out by the time reconfiguration is complete. Therefore, reconfiguration should be complete while the product is still popular. A final conclusion is that reconfiguration is best when done from inside the system (rather than purchasing capacity from outside) in any of the following cases: (a) when the number of periods between new products is large (b) when PLCs are long, and (c) when demand per product is high. These appear to be characteristics of a stable manufacturing environment in which a product remains in high demand and for extended periods of time. Therefore, it appears that in a stable manufacturing environment available capacity should be made use of as much as possible instead of purchasing from the outside.

7.2 RESEARCH CONTRIBUTIONS

The major contributions of this research are as follows:

- A description of how agility is related to the time/cost of system configuration.
- An improved method for quantitatively modeling agile manufacturing environments (prior assumptions relaxed and new attributes introduced).

- A rough-cut approach for quantitatively modeling agile manufacturing systems. This research is amongst the first to tackle design/reconfiguration of agile manufacturing systems in this manner.
- A detailed cost model for quantifying the tradeoffs between different levels of agility and other decision variables.
- A coded simulation model with 2^k factorial design experiment that can be used for more experiments and further advancement of this research.

7.3 FUTURE WORK

The term *agility* has been defined in so many different ways and has often been confused with *flexibility*. There are several good publications that propose ways to quantify the level of flexibility; but none were encountered that attempt to do the same for agility. As was demonstrated before, agility and flexibility are not the same and this research is one of the first to model agility in a quantitative manner. It is time that agility be given more attention and serious studies be conducted to see how the speed and cost of change can affect system performance in different manufacturing environments.

Also, this research was limited in that only three different types of agile manufacturing environments were considered. With four attributes used to define an environment, many environments are possible, and it might be feasible to formulate additional environments and experiment with them. This should not require extensive work because the simulation model and procedures outlined in this thesis can be directly applied.

In addition, certain assumptions made during this research can be addressed. One significant assumption is that each product requires only one processing capability. It would be a good idea during future research to consider multi-processing requirements for each product. Another assumption made is that the overall system capacity can be reconfigured but not reduced. It would be interesting to consider the possibility of reducing system size whenever there is excess capacity. However, these more realistic assumptions will make the modeling process more complicated.

REFERENCES

Adamides, E. D., Responsibility-based manufacturing. *International Journal of Advanced Manufacturing Technology*, 1996, 11(6), 439-448.

Ashby, J. R. & Uzsoy, R., Scheduling and order release in a single-stage production system. *Journal of Manufacturing Systems*, 1995, 14(4), 290-306.

Azhar, T. & Leung, L., A multi-attribute product life cycle approach to replacement decisions: an application of Saaty's system-with-feedback method. *The Engineering Economist*, 1993, 38(4), 320-344.

Bayus, B. L., Are Product life-cycles really getting shorter. *Journal of Product Innovation Management*, 1994, 11(4), 300-308.

Bertrand, J. W. M. & Muntslag, D. R., Production control in Engineer-to-Order firms. *International Journal of Production Economics*, 1993, 30-31, 3-22.

Bhaskaran, K & Pinedo, M., *Handbook of Industrial Engineering*. New York, NY: John Wiley & Sons, Inc., 1992. pp. 2182-2198.

Booth, R., Agile manufacturing [management]. *Engineering Management Journal*, 1996, 6(2), 105-112.

Box, G. E. & Draper, N. R., *Empirical Model-Building and Response Surface*. New York, NY: John Wiley & Sons, 1987.

Box, G. E., Hunter, W. G., & Hunter, J. S., *Statistics for Experimenters*. New York, NY: John Wiley & Sons, 1978.

Boynton, A. C., Victor, B., & Pine, B. J. II, New competitive strategies: challenges to organizations and information technology. *IBM Systems Journal*, 1993, 32(1), 40-64.

Brannon, A., Schoenmakers M., Klapwijk, H., & Haley, K., Product value matrices help firms to focus their efforts. *International Journal of Management Science*, 1993 21(6), 699-708.

Bunce, P., & Gould, P., From lean to agile manufacturing. *IEE Colloquium, Conference Title: Next Generation manufacturing Future trends in manufacturing and Supply Chain Management*, 1996, 1-5.

Carravilla, M. A. & de Sousa, J. P., Hierarchical production planning in a MTO company: a case study. *European Journal of Operations Research*, 1995, 86(1), 43-56.

Cho, H., Jung, M., & Kim, M., Enabling technologies of agile manufacturing and its related activities in Korea. *Computers and Industrial Engineering,* 1996, 30(3), 323-334.

Cordero, R., Managing for speed to avoid product obsolescence: a survey of techniques. *Journal of Product Innovation Management*, 1991, 8(4), 283-294.

Cravens, D. W., Stragegic forces affecting marketing strategy. *Business Horizons*. September-October 1986, 77-86.

Donaldson, C. F., Skelton, S., Lynch, T., & Lyons, M. H., Modeling product life-cycles from customer choice. *European Transactions on Telecommunications,* 1995, 6(4), 431-437.

Fine, C. & Freund, R., Optimal investment in product-flexible manufacturing capacity, Part I: *Economic analysis*, Sloan School of Management, M.I.T. Cambridge, MA, 1986, 1803-1863.

Forsythe, C., & Ashby, R. M., Human factors in agile manufacturing. *Ergonomics in Design*, 1996, 4(1), 15-21.

Franza R. & Gaimon, C., Flexibility and pricing decisions for high-volume products with short life cycles. *Working Paper,* July 1997.

Fujii, S., Morita, H., Tatsuta, Y., & Takata, Y., A basic study on high volume flexible manufacturing systems for agile manufacturing. *Proceedings of the 6th IFIP TC5/WG5.7 International Conference on Advances in Production Management Systems*, 1996, 39-44.

Gaimon, C. & Singhal, V., Flexibility and the choice of manufacturing facilities under short product life-cycles. *European Journal of Operational Research,* 1992, 60(2), 211-223.

Gillies, J., Nelder, G., & Ip-Shing, F., The applicability of agile manufacturing to small and medium-sized enterprises. *Middlesex Univ. Press, Conference Title: Proceedings of the Twelfth International Conference on CAD/CAM Robotics and Factories of the Future*, 1996, 538-543.

Graves, R. J., Agrawal, A., & Haberle, K., Estimating tools to support multi-path agility in electronics manufacturing. *IEEE, Conference Title: Seventeenth IEEE/CPMT International Electronics Manufacturing Technology Symposium "manufacturing Technologies-Present and Future"*, 1995, 38-48.

Griffin, A., Metrics for measuring product development cycle time. *Journal of Product Innovation Management*, 1993, 10(2), 112-125.

Hisrich, R. & Perters, M., *Marketing Decisions for New and Mature Products.* New York: Macmillan Publishing, 1991.

Hitchcock, M. F., Baker, A. D., & Brink, J. R., The role of hybrid systems theory in virtual manufacturing. *IEEE, Conference Title: Proceedings IEEE/IFAC Joint Symposium on Computer-Aided Control System Design,* 1994, 345-350.

Hormozie, A.M., Agile manufacturing. *Proceedings of the 37th International APICS Conference, APICS.*

Hutchinson, K. & Holland J., The economic value of flexible automation. *Journal of Manufacturing Systems,* 1982, 1(2), 215-227.

Hutchinson, K. & Sinha, D., A quantification of the value of flexibility. *Journal of Manufacturing Systems,* 1989, 8(1), 48-55.

Iacocca Institute, *21st century Manufacturing Enterprise Strategy,* Volumes 1 and 2, Lehigh University, PA, 1991, 216-218.

Kingsman, B. & Hendry, L., Customer inquiry management: part of a hierarchical system to control lead times in make-to-order companies. *Journal of operational Reseach Society,* 1993, 44(1), 61-70.

Kingsman, B., Worden, L., Hendry, L., Mercer, A., & Wilson, E., Integrated marketing and production planning in Make-to-Order companies. *International Journal of Production Economics,* 1993, 30-31, 53-66.

Kulatilaka, N., "Valuing the flexibility of flexible manufacturing systems." *IEEE Transactions on Engineering Management,* 1988, 53(4), 250-258.

Law, A. M. & Kelton, W. D., *Simulation Modeling & Analysis*. New York, NY: McGrow Hill, 1991. pp. 454-456.

Lawrence, R., Inside new product statistics: more, or less. . . new, or not? *Journal of Advertising Research* 33: RC3 – RC6, March-April 1993.

Lee Yong-Joo & Zipkin, P., Processing networks with inventories: sequential refinement systems. *Operations Research,* 1995, 43(6), 1025-1036.

Luss, H., "Operations Research and capacity expansion problems: a survey." *Operations Research,* 1982, 30, 907-947.

Mars, K. & Roberts, S. D., Implementing a Portable FORTRAN Uniform Generator. *Simulation,* 1983, 41, 135-139.

Millson, M. R., Raj, S. P., & Wilemon, D., A survey of major approaches for accelerating new product development. *Journal of Product Innovation Management,* 1992, 9(1), 53-69.

Norton, J. & Bass, F., Evaluation of technological generations: the law of capture. *Sloan Management Review,* Winter 1992, 66-77.

Richards, C. W., Agile manufacturing: beyond lean? *American Production and Inventory Control Society*, 1996, 37(2), 60-64.

Samadhi, T. M. & Hoang, K., Shared computer-integrated manufacturing for various types of production environments. *International Journal of Operations and Production Management*, 1995, 15(5), 95-108.

Sanderson, S. & Uzumeri, V., Strategies for new product development and renewal: design-based incermentalism. Center for Science and Technology Working Paper, Troy, New York: Rensselaer Polytechnic Institute, 1990.

Sheridan, J.S., Agile manufacturing: stepping beyond lean production. *Industry Week*, April 19, 1993, 30-46.

Shewchuk, J. P., A framework for modeling and design of agile discrete-part manufacturing facilities, *Proceedings of the 7th Annual Industrial Engineering Research Conference* (Banff, Canada), L. McGinnis and S. Reveliotis (Eds.), Institute of Industrial Engineers, Norcross, GA, 1998.

Shewchuk, J.P., Agile manufacturing: one size does not fit all. *Proceedings of the IFIP WG 5.7 Conference on Strategic Management of the Manufacturing Value Chain* (Troon, Scotland), Kluwer Academic Publishers, 1998, 143-150.

Stern, G., To outpace rivals, more firms step up spending on new product development. *Wall Street Journal*, October 28, 1992.

Wein, L. M., Dynamic scheduling of a multiclass MTS queue. *Operations Research*, 1992, 40(4), 724-735.

Xiaoyuan, Y., An object-oriented model of agile manufacturing system. *IEEE, Conference Title: Proceedings of the IEEE International Conference on Industrial Technology*, 1996, 171-175.

Voss, D., A new spring for manufacturing. *Journal of Business Strategy*, 1994, 15(1), 54-56.

Young, C. & Greene, A., *Flexible Manufacturing Systems.* New York, NY: AMA Membership Publication Division, 1986. p.35.

APPENDIX

```
/*

                    Design and Reconfiguration
                              of
                       Manufacturing Systems
                                in
                    Agile Manufacturing Environments


                    Written By: Shamil Daghestani
                    in partial fulfillment of the
                       Master of Science Degree
                                in
                    Industrial & Systems Engineering
                                at
                           VIRGINIA TECH
                           December 1998


*/

#include <stdlib.h>                       // libraries in the program
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <time.h>
#include <limits.h>

#define MODLUS  2147483647                // assign values
#define MULT1   24112
#define MULT2   26143
#define PI      3.1416


void main_2(void);
void initialize_1(void);      // subroutines contained in the program
void initialize_2(void);
void initialize_3(void);
void model_inputs(void);

float uniform_dist(float upper, float lower);
float uniform_design(int design);
float rand (int stream);
void randst (long zset, int stream);

void gen_plc_variables(void);
void gen_total_demand(void);

void gen_capabilities(void);
void capability_matrix(void);
void capabilities_assignment(void);
void gen_cost(void);
void production_func(void);
void function_1(void);
void cap_reserves(void);
void factor_values(void);
void profit_loop(void);
void replication(void);
void decision_variables_levels(void);
void prof_significance(void);
void cost_significance(void);
```

```
static long zrng[] =
{        0,          // strings of numbers used when generating randon no.
 7032729381, 29281629770, 844781392, 210317751907, 560370392, 9133576005,
 8173854928, 9840983048,   36347458, 218364794839, 8161990930, 712485156,
 1359872137, 9310492656, 1576265781, 90857146262,   456204515, 310594458,
 7897637496, 9679860498, 4496434910, 791038709689, 7134652875, 821335045,
};


     // define integers and float variables
     // most variables are globely defined because their values will be
     // needed in several subroutines

 int        // manufacturing environment
       w,s,i,j,k,p[102],l[102],d[102], strm, begin[102],end[102],
       variety_demand_in_period[102], sim_periods, sim_products,

           // manufacturing system
       v,c,x,y,Mx,m,Ap,try_capability,existing_capability[16],
       req_capability_for_product[50],  status_capability[16],
       best[10][102], temp_i, c1, c2, c3,quantity_unit,design,LoB_des,
       LoA_des,LoT_des,LoR_des,pdi,

           // not to be intialized
       reps, loop, factor[5][17], circ,level;

 float      // manufacturing environment
       t[102], ran1, period_low, period_up, life_low, life_up, demand_low,
       demand_up, time_low, time_up, p_mean, l_mean,
       d_mean, t_mean,

           // manufacturing system
       price_low, price_up,pr[102],PPd,PPd_temp, LoB, LoA, LoT, LoR,
       revenue[102], Z[102], Q[102], b[102], remaining_capacity[16][102],
       price[50][102],indicator,PW_profit, PW_cost,irate, period_profit,
       period_cost, Ck,Dk,Ak,Tk,Cmin,Cmax,Dmin,Dmax,Amin,Amax,Tmin,Tmax,
       total_idle, effort_cost[16],effort_time[16],extra_remaining[16],
       excess, hour_purchase[16], mean_capacity, uniform_value,oper_hours,
       pdf, portion, total_effort,

           // not to be initialized
       avg_AW, AW_profit[11], AW_cost[11], avg_AW_profit[17][11],
       avg_AW_cost[17][11], sqr_AW_profit, sqr_AW_cost, stdev_AW_profit,
       stdev_AW_cost, sum_AW_profit, sum_AW_cost, per_rem;


//--------------------------- main function -----------------------------
main()                        // main subroutine
{
 model_inputs();            // input variables
 main_2();
 prof_significance();
 cost_significance();
 printf("\a");
 exit(0);
 return(0);

}


//----------------------------- main_2 ----------------------------------
void main_2(void)
{
```

```
 for(i=0;i<=16;i++)   // initialize the matrix
   {
    for(j=0;j<=10;j++) avg_AW_profit[i][j] = avg_AW_cost[i][j]=0.0;
   }
 reps=strm=0;
 per_rem=0.0;
 initialize_1();
 initialize_2();
 factor_values();
 clrscr();
 reps = 10;                  // number of replications for each design point
 circ=0;
  for(level=1; level<=16; level++)
    {                          // do all sixteen design point combinations
    for(circ=1; circ<=10; circ++)
      {                        // 10 sets of main effects and interactions
     clrscr();
     per_rem += 0.625;   // display on screen the percentage of completion
     printf("\n\n\n\n\n\n\n\n\n\n\t\t\t    Precent Complete %3.2f", per_rem);
     initialize_2();     // initialize specific variables
     strm = circ;        // experiment with a different random number seed
     initialize_3();
     gen_plc_variables();
     gen_total_demand();

     for(loop=1; loop<=reps; loop++)
         {
         initialize_1();     // initialize spcific variables
         decision_variables_levels();
         gen_capabilities();
         capability_matrix();
         capabilities_assignment();
         profit_loop();
         }
       replication();
       initialize_2();
      }
    initialize_2();
   }
  return;
}


//----------------------------- profit loop -------------------------------
void profit_loop(void)
{
             // calculate PW profit and PW cost after each replication
  for(k=1;k<=sim_periods;k++)
     {
      period_profit = revenue[k]-Z[k]-Q[k];
      period_cost = Z[k]+Q[k];
      PW_profit += period_profit*(1/(pow(1+irate,k)));
      PW_cost += period_cost*(1/(pow(1+irate,k)));
     }
                 // now calculate AW profit and AW cost per period
  k=sim_periods;
  AW_profit[loop] = PW_profit*(irate*(pow(irate+1,k)))/((pow(1+irate,k)-1));
  AW_cost[loop] = PW_cost*(irate*(pow(irate+1,k)))/((pow(1+irate,k)-1));

}
```

```
//----------------------------- replication -----------------------------
void replication()
{

  for(k=1;k<=reps;k++)
     {sum_AW_profit += AW_profit[k];
      sum_AW_cost += AW_cost[k];
     }
     avg_AW_profit[level][circ] = (sum_AW_profit/reps);
     avg_AW_cost[level][circ] = (sum_AW_cost/reps);

  for(k=1;k<=reps;k++)
     {
     sqr_AW_profit += pow((AW_profit[k] - avg_AW_profit[level][circ]),2);
     sqr_AW_cost += pow((AW_cost[k] - avg_AW_cost[level][circ]),2);
     }
  stdev_AW_profit = sqrt((sqr_AW_profit)/(reps));
  stdev_AW_cost = sqrt((sqr_AW_cost)/(reps));

if (circ==1)  // save a sample of one set of main effects and interactions
   {          // in a file
    FILE *temp_res;     // open before saving
temp_res = fopen("c:\\tc\\output\\signif.out", "a");
      if(!temp_res){printf("\n\t Cannot save output file! Diskette may be \
write protected");
             exit(0);}
 fprintf(temp_res,"\n\t %2d   %7.0f   %7.0f   %7.0f   %7.0f",\
 level,avg_AW_profit[level][1],stdev_AW_profit,avg_AW_cost[level][1],\
 stdev_AW_cost);

  fclose(temp_res);    // close file after saving
   }

}


//--------------------------- intialize 2 ----------------------------
void initialize_2(void)
{
 PW_profit = PW_cost = sqr_AW_profit = sqr_AW_cost=0.0;
 sum_AW_profit = sum_AW_cost = stdev_AW_profit = stdev_AW_cost=0.0;
 for(k=0;k<=reps;k++) { AW_profit[k] = AW_cost[k]=0.0;}

}


//---------------------- model inputs function ---------------------------
void model_inputs (void)
{
 sim_periods=0.0;
 period_up=period_low=life_up=life_low=demand_up=demand_low=time_up=0.0;
 time_low=price_up=price_low=0.0;


   again_1:
   printf("\n\n\t\a  Enter number of simulation periods: ");
   scanf("%d",&sim_periods);

   printf("\n\n\t\a (1) Enter lower and upper limits for the number of ");
   printf("\n\t periods between new products (seperated by space): ");
   scanf("%f%f",&period_low, &period_up);

   printf("\n\n\t\a (2) Enter lower and upper limits for the number of ");
```

```
   printf("\n\t periods of product life (seperated by space): ");
   scanf("%f%f",&life_low, &life_up);

   if(sim_periods <= period_up+life_up)
      {printf("\n\n\a WARNING: Simulation periods too few!! Re-enter with
       more periods...\n");
       goto again_1;}

   again_2:
   printf("\n\n\t\a (3) Enter lower and upper limits for demand per ");
   printf("\n\t unit per period (seperated by space): ");
   scanf("%f%f",&demand_low, &demand_up);

   printf("\n\n\t\a (4) Enter lower and upper limits for production ");
   printf("\n\t time per unit for each type (seperated by space): ");
   scanf("%f%f",&time_low, &time_up);

 if((period_low>period_up || life_low>life_up || demand_low>demand_up
  || time_low>time_up) ||
 (period_low<0 || period_up<0 || life_low<0 || life_up<0 || demand_low<0
   || demand_up<0 || life_low<0 || life_up<0 || time_low<0.0 || time_up<0.0))
 { printf("\n\a -*-*-*- WARNING: Improper value(s) or variable(s)
   entered!!..");
   printf(" Try Again -*-*-*-\n");
   goto again_2;
 }

   printf("\n\n\t\a Enter lower and upper limits for the price/unit ");
   scanf("%f%f",&price_low, &price_up);

   printf("\n\n\t\a Enter min. and max. cost for purchasing 1 hr of ");
   printf("\n\t capacity when agility is min: ");
   scanf("%f%f",&Cmin, &Cmax);

   printf("\n\n\t\a Enter min. and max. rate at which capacity increases ");
   printf("\n\t with agility: ");
   scanf("%f%f",&Dmin, &Dmax);

   printf("\n\n\t\a Enter min. and max. cost of acquiring 1 hr of capcity ");
   scanf("%f%f",&Amin, &Amax);

   printf("\n\n\t\a Enter min. and max. time (hr) of acquring 1 hr of
   capacity ");
   scanf("%f%f",&Tmin, &Tmax);

   printf("\n\n\t\a Enter percentage rate of decline in price as product
   ages ");
   scanf("%f",&PPd_temp);

   printf("\n\n\t\a Enter max. number of all possible processing capabilities
    ");
   scanf("%f",&Mx);

   printf("\n\n\t\a Enter the internal rate of return per period ");
   scanf("%f",irate);

   printf("\n\n\t\a Enter hrs of operation per period ");
   scanf("%f",&oper_hours);


  PPd = PPd_temp/100;
}
```

```
//----------------------- decision_variables_levels ------------------------
void decision_variables_levels (void)
{

 LoB_des = factor[1][level]; LoA_des = factor[2][level];
 LoT_des = factor[3][level]; LoR_des = factor[4][level];

}


//---------------------- gen_plc_variables function ------------------------
void initialize_3 (void)
{
 sim_products=temp_i=0;

 p_mean=l_mean=d_mean=t_mean=0.0;

   for(i=0;i<=101;i++)
     {p[i]=l[i]=d[i]=begin[i]=end[i]=variety_demand_in_period[i]=0;
      pr[i]=t[i]=0.0;
     }

   for(i=0;i<=49;i++) req_capability_for_product[i]=0;

   for(i=0;i<=49;i++)
      {
       for(j=0;j<=101;j++)
       {price[i][j]=0.0;}
      }

    for(i=0;i<=9;i++)
     {
     for(j=0;j<=101;j++)
       {best[i][j]=0;}
     }

       p_mean = (period_up+period_low)/2;  // PLC means
       l_mean = (life_up+life_low)/2;
       d_mean = (demand_up+demand_low)/2;
       t_mean = (time_up+time_low)/2;

}


//---------------------- gen_plc_variables function ------------------------
void gen_plc_variables (void)
{

   s=0;
   while(begin[s] <= sim_periods)
     {
      s++;
      p[s] = uniform_dist(period_up, period_low);//generate norml.distributed
value
      l[s] = uniform_dist(life_up, life_low);
      d[s] = uniform_dist(demand_up, demand_low);
      t[s] = uniform_dist(time_up,   time_low);

       if((p[s]<0) || (l[s]<0) || (d[s]<0) || (t[s]<0.0))
         {
 printf("\n\t\a Program Aborted: Some PLC random variable(s) below zero..");
 exit(0);
```

```
        }
    p[1] = 1;
    begin[s] = begin[s-1] + p[s]; // period when demand for prod. s starts
    end[s] = begin[s] + l[s]-1;   // period when demand for s ends
    req_capability_for_product[s] = 1 + (Mx * rand(strm));
    if(req_capability_for_product[s]>Mx) req_capability_for_product[s]=Mx;
    pr[s] = uniform_dist(price_up, price_low); // initial price for product s
     }
    sim_products=s;    // establishes the number of products simulated

}


//----------------------- gen_total_demand function -----------------------
void gen_total_demand()
{

 float temp_var[50][102];           // this matrix only needed in this
                                 // subroutine
  for(i=0;i<=49;i++)
     {
      for(j=0;j<101;j++)
        {temp_var[i][j]=0.0;}
      }

 for(i=1;i<=sim_products;i++)              // look for units produced in
   {                                       // each period
    for(j=begin[i]; j<=end[i]; j++)
       {
       if(j>sim_periods) continue;
       variety_demand_in_period[j] += 1;
       price[i][j] = (pr[i]*pow(PPd,(j-begin[i])));
       temp_var[i][j] = price[i][j]/t[i];
              //selling price divided by prod. time
       }
   }
                           //sort by descending order-- highest first

  for(j=1;j<=sim_periods;j++) //look for most profitable units in each period
     {
      for(k=1;k<=sim_products;k++)
        {
         indicator=0.000001;
         for(i=1;i<=sim_products;i++)
           {
            if(temp_var[i][j]>indicator)
              {indicator=temp_var[i][j];
              temp_i=i;
              }
           }
         temp_var[temp_i][j] = -1.0;
         best[k][j]=temp_i;
         if(k >= variety_demand_in_period[j]) break; //go to next period
        }
     }

}


//-------------------- generating capabilities function --------------------
void gen_capabilities(void)
{
```

```
  LoB = uniform_design(LoB_des);
  LoA = uniform_design(LoA_des);
  LoT = uniform_design(LoT_des);
  LoR = uniform_design(LoR_des);

       m = (LoB*Mx)+1;
       if(m>Mx) m = Mx;
       else if(m==0) m=1;

  for(s=1; s<=m; s++) // generates the ACQUIRED amount of capabilities
     {
      again_5:
      for(;;)         // open loop keeps looking for unique capabilities
        {           //
          try_capability = 1 + (Mx * rand(strm));
          for(k=1; k<=s; k++)
             {
            if(try_capability == existing_capability[k]) goto again_5;
            else if(k >= s)
                {
                 existing_capability[s] = try_capability;
                 if(s >= m) break;
                }
             }
          break; //break out of open each time a unique capability is found
        }
     }
}


//-------------------- generate capability_matrix -------------------------
void capability_matrix (void)
{
                  //generate random capacity parameter
  mean_capacity = ((l_mean/p_mean)*d_mean*t_mean)/m;

 for(v=1; v<=m; v++)      //gen. random req. capabilities,capacity, etc.
  {
  gen_cost();
  b[v] = mean_capacity;  // generate random capacity (hrs)
  if(b[v] < 0){printf("\n\a Error...random capacity below zero"); exit(0);}
  Q[1] += b[v]*hour_purchase[v];
  Z[1] += b[v]*effort_cost[v];

  for(c=1; c<=sim_periods; c++) remaining_capacity[v][c] = b[v];

  }

}


//------------------------------ gen_cost --------------------------------
void gen_cost(void)
{

  Ck = Cmin + ((Cmax - Cmin) * rand(strm));
  Dk = Dmin + ((Dmax - Dmin) * rand(strm));
  Ak = Amin + ((Amax - Amin) * rand(strm));
  Tk = Tmin + ((Tmax - Tmin) * rand(strm));
  hour_purchase[v] = Ck*(exp(Dk*LoA));
  effort_time[v] = LoT*(Tk*(1-LoA));
  effort_cost[v] = Ak*(1-LoA)-((Ak/Tk)*effort_time[v]);
}
```

```
//------------------------ capabilities_assignment -------------------------
void capabilities_assignment(void)
{

  for(j=1; j<=sim_periods; j++)       // use the appropriate capabilities
    {
     cap_reserves();
     i=0;
     again_6:
     i++;               //see if required capab. already exists in system
     if(i<=variety_demand_in_period[j])
       {
       w=best[i][j];
       for(k=1; k<=m; k++)
         {                      // product with highest profit that period
         if(req_capability_for_product[w]==existing_capability[k])
            {
            if(d[w]*t[w] <= remaining_capacity[k][j])
              { production_func();    // produce without making any changes
               goto again_6;
              }
            if(d[w]*t[w] > remaining_capacity[k][j])
              {
              function_1();
              production_func();
              goto again_6;
               }
            }
         }
              // get here if required capab. does not exist
              // check if at least one capacity is idle
       m++;     // creat a new capability
       existing_capability[m] = req_capability_for_product[best[i][j]];
       status_capability[m] = 1;
       k=m;
       v=m;
       gen_cost();   // generate all costs associated with the new capability
       function_1();
       production_func();
       goto again_6;

       }
    }
}


//-------------------------- Production_func -----------------------------
void production_func(void)
{
                       // produce after after capacities are assinged
                       // fulfill demand as much as possible but no more

     quantity_unit = 0;
     quantity_unit = remaining_capacity[k][j]/t[w]; // check for adequate
     if(quantity_unit < d[w])                       // amount of capacity
       { remaining_capacity[k][j] -= quantity_unit*t[w]; }
     else if(quantity_unit >= d[w])
       { remaining_capacity[k][j] -= d[w]*t[w];
        quantity_unit = d[w];
       }
  revenue[j] += price[w][j]*quantity_unit;
```

```
}


//----------------------------- function_1 -------------------------------
void function_1(void)
{

          // ratio indicating percentage in shortage of capacity to be
  float y2; // fulfilled
       y2=1.0; // percentage will be set at 100% throughout the model
  float just_got_capacity,reconfig_capacity,look_more_capacity,
       purchase_capacity,addi_acquired_capacity;
  addi_acquired_capacity = y2*(d[w]*t[w]-remaining_capacity[k][j]);

  if(addi_acquired_capacity <= total_idle)
    { reconfig_capacity = LoR*addi_acquired_capacity;
      purchase_capacity = (1-LoR)*addi_acquired_capacity;
      look_more_capacity = reconfig_capacity;
    }

  else if(addi_acquired_capacity > total_idle)
    {
      reconfig_capacity = LoR*total_idle;
      purchase_capacity = addi_acquired_capacity-reconfig_capacity;
      look_more_capacity = reconfig_capacity;
    }

  for(c2=1; c2 <= m; c2++)
    {
    if(status_capability[c2] != 1)
      {
      if(remaining_capacity[c2][j] >= look_more_capacity)
       {
       for(c3=j; c3<=sim_periods; c3++)
          {remaining_capacity[c2][c3] -= look_more_capacity;}
       total_idle -= look_more_capacity;
       total_effort = effort_time[c2]*look_more_capacity;
       pdf = total_effort/oper_hours;
       pdi = pdf;

       portion = (pdf-pdi)*look_more_capacity;
       remaining_capacity[k][j+pdi] += portion;
       for(c3=j+pdi+1; c3<=sim_periods; c3++)
          {remaining_capacity[k][c3] += look_more_capacity;}
       Z[j] += look_more_capacity*effort_cost[c2];// spread over periods
       goto skip_1;
       }

      if(remaining_capacity[c2][j] < look_more_capacity)
       {
       look_more_capacity -= remaining_capacity[c2][j];
       just_got_capacity = remaining_capacity[c2][j];
       total_idle -= just_got_capacity;
       for(c3=j; c3<=sim_periods; c3++) remaining_capacity[c2][c3] = 0.0;
       total_effort = effort_time[c2]*just_got_capacity;
       pdf = (total_effort/oper_hours);
       pdi = pdf;    // convert the float number to a integer
       portion = (pdf-pdi)*just_got_capacity;
       remaining_capacity[k][j+pdi] += portion;
       for(c3=j+pdi+1; c3<=sim_periods; c3++)
          {remaining_capacity[k][c3] += just_got_capacity;}
       Z[j] += just_got_capacity*effort_cost[c2];
```

```
      }
     }
   }
     // if all capacity from idle machines is already taken; get excess
     // capacity from busy machines
   if(look_more_capacity >= 0.00001)
    {
    for(c2=1; c2<=m; c2++)
      {
      if(c2==k) continue;
      if(status_capability[c2] == 1)
       {
       if(extra_remaining[c2] >= look_more_capacity)
        {
        for(c3=j; c3<=sim_periods; c3++)
           {remaining_capacity[c2][c3] -= look_more_capacity;}
        extra_remaining[c2] -= look_more_capacity;
        total_idle -= look_more_capacity;
        total_effort = effort_time[c2]*look_more_capacity;
        pdf = (total_effort/oper_hours);
        pdi = pdf;
        portion = (pdf-pdi)*look_more_capacity;
        remaining_capacity[k][j+pdi] += portion;
        for(c3=j+pdi+1; c3<=sim_periods; c3++)
           {remaining_capacity[k][c3] += look_more_capacity;}
        Z[j] = look_more_capacity*effort_cost[c2];
        goto skip_1;
        }

      if(extra_remaining[c2] < look_more_capacity)
        {
        look_more_capacity -= extra_remaining[c2];
        just_got_capacity = extra_remaining[c2];
        total_idle -= just_got_capacity;
        for(c3=j; c3<=sim_periods; c3++) remaining_capacity[c2][c3] = 0.0;
        total_effort = effort_time[c2]*just_got_capacity;
        pdf = total_effort/oper_hours;
        pdi = pdf;    // convert the float to integer
        portion = (pdf-pdi)*just_got_capacity;
        remaining_capacity[k][j+pdi] += portion;
        for(c3=j+pdi+1; c3<=sim_periods; c3++)
           {remaining_capacity[k][c3] += just_got_capacity;}
        extra_remaining[c2] = 0.0;
        Z[j] += just_got_capacity*effort_cost[c2];
         }
       }
      }
    }

  skip_1:
  Q[j] += hour_purchase[k]*purchase_capacity;

  total_effort = effort_time[k]*purchase_capacity;
  pdf = total_effort/oper_hours;
  pdi = pdf;    // convert the float number to an integer
  portion = (pdf-pdi)*purchase_capacity;
  remaining_capacity[k][j+pdi] += portion;
  for(c3=j+pdi+1; c3<=sim_periods; c3++)
     {remaining_capacity[k][c3] += purchase_capacity;}
  Z[j] += purchase_capacity*effort_cost[k];

}
```

```
//--------------------------- cap_reserves -----------------------------
void cap_reserves(void)
{
       /* reserve all necessary capacities of capabilities at the
          begining of each period so that they will not be taken away
          later during that period as each product (starting from highest
          priority) is being considered
       */

    total_idle = 0.0;   //reset total reserved capacity for period j
              //create a matrix to preserve the current "remaining_capacity" for
now
    for(c2=1; c2<=m; c2++)
       {
       extra_remaining[c2] = remaining_capacity[c2][j];
       status_capability[c2] = 0;
       }

    for(c1=1; c1<=variety_demand_in_period[j]; c1++)
      {
      for(c2=1; c2<=m; c2++)
       {if(req_capability_for_product[best[c1][j]]==existing_capability[c2])
          {
           status_capability[c2]=1;
           excess = extra_remaining[c2] - d[best[c1][j]]*t[best[c1][j]];
           if(excess >= 0.0)
             {
              extra_remaining[c2] -= d[best[c1][j]]*t[best[c1][j]];
             }
           else
             {
              extra_remaining[c2] = 0.0;
             }
          }
       }
      }

    for(c2=1; c2<=m; c2++) total_idle += extra_remaining[c2];

}


//--------------------------- float uniform_dist ---------------------------
float uniform_dist(float upper, float lower)
{
   ran1 = rand(strm);
   return (lower + ((upper - lower) * ran1));
}


//--------------------------- float uniform_design --------------------------
float uniform_design(int design)
{
   if(design == -1) uniform_value = 0.5*rand(strm);
   else if(design == +1) uniform_value = 0.5+(0.5*rand(strm));
   return (uniform_value);
}


//--------------------------- factor_values ------------------------------
void factor_values(void)
{
```

```
 for(i=0;i<=4;i++)                          // reset factor values
    {                                       // for the next replication
     for(j=0;j<=16;j++) factor[i][j] = 0; // of mean effects and interactions
    }

 factor[1][1] =-1; factor[1][2] =1; factor[1][3] =-1;
 factor[1][4] =1; factor[1][5] =-1; factor[1][6] =1;
 factor[1][7] =-1; factor[1][8] =1; factor[1][9] =-1;
 factor[1][10] =1; factor[1][11] =-1; factor[1][12] =1;
 factor[1][13] =-1; factor[1][14] =1; factor[1][15] =-1;
 factor[1][16] =1; factor[2][1] =-1; factor[2][2] =-1;
 factor[2][3] =1; factor[2][4] =1; factor[2][5] =-1;
 factor[2][6] =-1; factor[2][7] =1; factor[2][8] =1;
 factor[2][9] =-1; factor[2][10] =-1; factor[2][11] =1;
 factor[2][12] =1; factor[2][13] =-1; factor[2][14] =-1;
 factor[2][15] =1; factor[2][16] =1; factor[3][1] =-1;
 factor[3][2] =-1; factor[3][3] =-1; factor[3][4] =-1;
 factor[3][5] =1; factor[3][6] =1; factor[3][7] =1;
 factor[3][8] =1; factor[3][9] =-1; factor[3][10] = -1;
 factor[3][11] = -1; factor[3][12] = -1; factor[3][13] = 1;
 factor[3][14] = 1; factor[3][15] = 1; factor[3][16] = 1;
 factor[4][1] = -1; factor[4][2] = -1; factor[4][3] = -1;
 factor[4][4] = -1; factor[4][5] = -1; factor[4][6] = -1;
 factor[4][7] = -1; factor[4][8] = -1; factor[4][9] = 1;
 factor[4][10] =1; factor[4][11] =1; factor[4][12] =1;
 factor[4][13] =1; factor[4][14] =1; factor[4][15] =1;
 factor[4][16] =1;

}


//-------------------------- prof_significance ----------------------------
void prof_significance(void)
{
               // calculate profit significance and confidence intervals

  float avg_sig1, avg_sig2, avg_sig3, avg_sig4, avg_sig12, avg_sig13,
     avg_sig14, avg_sig23, avg_sig24, avg_sig34, avg_sig123,
     avg_sig124, avg_sig134, avg_sig234, avg_sig1234,
   sqr_sig1, sqr_sig2, sqr_sig3, sqr_sig4, sqr_sig12, sqr_sig13,
   sqr_sig14, sqr_sig23, sqr_sig24, sqr_sig34, sqr_sig123,
   sqr_sig124, sqr_sig134, sqr_sig234, sqr_sig1234,
   stdev_sig1, stdev_sig2, stdev_sig3, stdev_sig4, stdev_sig12,
   stdev_sig13, stdev_sig14, stdev_sig23, stdev_sig24,stdev_sig34,
   stdev_sig123, stdev_sig124,  stdev_sig134, stdev_sig234, stdev_sig1234,
   dn_1, up_1, dn_2, up_2, dn_3, up_3, dn_4, up_4, dn_12, up_12, dn_13, up_13,
   dn_14, up_14, dn_23, up_23, dn_24, up_24, dn_34, up_34, dn_123, up_123,
   dn_124, up_124, dn_134, up_134, dn_234, up_234, dn_1234,up_1234,

   sig1[11], sig2[11], sig3[11], sig4[11], sig12[11], sig13[11],
   sig14[11], sig23[11], sig24[11], sig34[11], sig123[11], sig124[11],
   sig134[11], sig234[11], sig1234[11];

     avg_sig1=avg_sig2=avg_sig3=avg_sig4=avg_sig12=avg_sig13=0.0;
     avg_sig14=avg_sig23=avg_sig24=avg_sig34=avg_sig123=0.0;
     avg_sig124=avg_sig134=avg_sig234=avg_sig1234=0.0;
   sqr_sig1=sqr_sig2=sqr_sig3=sqr_sig4=sqr_sig12=sqr_sig13=0.0;
   sqr_sig14=sqr_sig23=sqr_sig24=sqr_sig34=sqr_sig123=sqr_sig124=0.0;
   sqr_sig134=sqr_sig234=sqr_sig1234=0.0;
      stdev_sig1=stdev_sig2=stdev_sig3=stdev_sig4=stdev_sig12=0.0;
      stdev_sig13=stdev_sig14=stdev_sig23=stdev_sig24=stdev_sig34=0.0;
      stdev_sig123=stdev_sig124=stdev_sig134=stdev_sig234=stdev_sig1234=0.0;
   dn_1=up_1=dn_2=up_2=dn_3=up_3=dn_4=up_4=dn_12=up_12=dn_13=up_13=dn_14=0.0;
```

```
   up_14=dn_23=up_23=dn_24=up_24=dn_34=up_34=dn_123=up_123=dn_124=up_124=0.0;
   dn_134=up_134=dn_234=up_234=dn_1234=up_1234=0.0;

 for(i=0;i<=reps;i++){
   sig1[i] = sig2[i] = sig3[i] = sig4[i] = sig12[i] = sig13[i] = 0.0;
   sig14[i] = sig23[i] = sig24[i] = sig34[i] = sig123[i] = 0.0;
   sig124[i] = sig134[i] = sig234[i] = sig1234[i] = 0.0;}

  for(i=1;i<=reps;i++)        // i = replication number
   {
   for(j=1;j<=16;j++)   // j = level
    {
   sig1[i] +=factor[1][j]*avg_AW_profit[j][i]/8;
   sig2[i] +=factor[2][j]*avg_AW_profit[j][i]/8;
   sig3[i] +=factor[3][j]*avg_AW_profit[j][i]/8;
   sig4[i] +=factor[4][j]*avg_AW_profit[j][i]/8;
   sig12[i] +=factor[1][j]*factor[2][j]*avg_AW_profit[j][i]/8;
   sig13[i] +=factor[1][j]*factor[3][j]*avg_AW_profit[j][i]/8;
   sig14[i] +=factor[1][j]*factor[4][j]*avg_AW_profit[j][i]/8;
   sig23[i] +=factor[2][j]*factor[3][j]*avg_AW_profit[j][i]/8;
   sig24[i] +=factor[2][j]*factor[4][j]*avg_AW_profit[j][i]/8;
   sig34[i] +=factor[3][j]*factor[4][j]*avg_AW_profit[j][i]/8;
   sig123[i] +=factor[1][j]*factor[2][j]*factor[3][j]*avg_AW_profit[j][i]/8;
   sig124[i] +=factor[1][j]*factor[2][j]*factor[4][j]*avg_AW_profit[j][i]/8;
   sig134[i] +=factor[1][j]*factor[3][j]*factor[4][j]*avg_AW_profit[j][i]/8;
   sig234[i] +=factor[2][j]*factor[3][j]*factor[4][j]*avg_AW_profit[j][i]/8;
sig1234[i]+=factor[1][j]*factor[2][j]*factor[3][j]*factor[4][j]*avg_AW_profit[j
][i]/8;
    }
   }

   for(i=1;i<=reps;i++)
     {
      avg_sig1 += sig1[i]/reps;
      avg_sig2 += sig2[i]/reps;
      avg_sig3 += sig3[i]/reps;
      avg_sig4 += sig4[i]/reps;
      avg_sig12 += sig12[i]/reps;
      avg_sig13 += sig13[i]/reps;
      avg_sig14 += sig14[i]/reps;
      avg_sig23 += sig23[i]/reps;
      avg_sig24 += sig24[i]/reps;
      avg_sig34 += sig34[i]/reps;
      avg_sig123 += sig123[i]/reps;
      avg_sig124 += sig124[i]/reps;
      avg_sig134 += sig134[i]/reps;
      avg_sig234 += sig234[i]/reps;
      avg_sig1234 += sig1234[i]/reps;
     }

  for(i=1;i<=reps;i++)
     {
      sqr_sig1 += pow((avg_sig1 - sig1[i]),2);
      sqr_sig2 += pow((avg_sig2 - sig2[i]),2);
      sqr_sig3 += pow((avg_sig3 - sig3[i]),2);
      sqr_sig4 += pow((avg_sig4 - sig4[i]),2);
      sqr_sig12 += pow((avg_sig12 - sig12[i]),2);
      sqr_sig13 += pow((avg_sig13 - sig13[i]),2);
      sqr_sig14 += pow((avg_sig14 - sig14[i]),2);
      sqr_sig23 += pow((avg_sig23 - sig23[i]),2);
      sqr_sig24 += pow((avg_sig24 - sig24[i]),2);
      sqr_sig34 += pow((avg_sig34 - sig34[i]),2);
      sqr_sig123 += pow((avg_sig123 - sig123[i]),2);
```

```
      sqr_sig124 += pow((avg_sig124 - sig124[i]),2);
      sqr_sig134 += pow((avg_sig134 - sig134[i]),2);
      sqr_sig234 += pow((avg_sig234 - sig234[i]),2);
      sqr_sig1234 += pow((avg_sig1234 - sig1234[i]),2);
     }

   stdev_sig1 = sqrt(sqr_sig1/reps);
   stdev_sig2 = sqrt(sqr_sig2/reps);
   stdev_sig3 = sqrt(sqr_sig3/reps);
   stdev_sig4 = sqrt(sqr_sig4/reps);
   stdev_sig12 = sqrt(sqr_sig12/reps);
   stdev_sig13 = sqrt(sqr_sig13/reps);
   stdev_sig14 = sqrt(sqr_sig14/reps);
   stdev_sig23 = sqrt(sqr_sig23/reps);
   stdev_sig24 = sqrt(sqr_sig24/reps);
   stdev_sig34 = sqrt(sqr_sig34/reps);
   stdev_sig123 = sqrt(sqr_sig123/reps);
   stdev_sig124 = sqrt(sqr_sig124/reps);
   stdev_sig134 = sqrt(sqr_sig134/reps);
   stdev_sig234 = sqrt(sqr_sig234/reps);
   stdev_sig1234 = sqrt(sqr_sig1234/reps);

                          // 90% confidence intervals

      dn_1 = avg_sig1-1.833*(stdev_sig1/sqrt(reps));
      up_1 = avg_sig1+1.833*(stdev_sig1/sqrt(reps));
      dn_2 = avg_sig2-1.833*(stdev_sig2/sqrt(reps));
      up_2 = avg_sig2+1.833*(stdev_sig2/sqrt(reps));
      dn_3 = avg_sig3-1.833*(stdev_sig3/sqrt(reps));
      up_3 = avg_sig3+1.833*(stdev_sig3/sqrt(reps));
      dn_4 = avg_sig4-1.833*(stdev_sig4/sqrt(reps));
      up_4 = avg_sig4+1.833*(stdev_sig4/sqrt(reps));
      dn_12 = avg_sig12-1.833*(stdev_sig12/sqrt(reps));
      up_12 = avg_sig12+1.833*(stdev_sig12/sqrt(reps));
      dn_13 = avg_sig13-1.833*(stdev_sig13/sqrt(reps));
      up_13 = avg_sig13+1.833*(stdev_sig13/sqrt(reps));
      dn_14 = avg_sig14-1.833*(stdev_sig14/sqrt(reps));
      up_14 = avg_sig14+1.833*(stdev_sig14/sqrt(reps));
      dn_23 = avg_sig23-1.833*(stdev_sig23/sqrt(reps));
      up_23 = avg_sig23+1.833*(stdev_sig23/sqrt(reps));
      dn_24 = avg_sig24-1.833*(stdev_sig24/sqrt(reps));
      up_24 = avg_sig24+1.833*(stdev_sig24/sqrt(reps));
      dn_34 = avg_sig34-1.833*(stdev_sig34/sqrt(reps));
      up_34 = avg_sig34+1.833*(stdev_sig34/sqrt(reps));
      dn_123 = avg_sig123-1.833*(stdev_sig123/sqrt(reps));
      up_123 = avg_sig123+1.833*(stdev_sig123/sqrt(reps));
      dn_124 = avg_sig124-1.833*(stdev_sig124/sqrt(reps));
      up_124 = avg_sig124+1.833*(stdev_sig124/sqrt(reps));
      dn_134 = avg_sig134-1.833*(stdev_sig134/sqrt(reps));
      up_134 = avg_sig134+1.833*(stdev_sig134/sqrt(reps));
      dn_234 = avg_sig234-1.833*(stdev_sig234/sqrt(reps));
      up_234 = avg_sig234+1.833*(stdev_sig234/sqrt(reps));
      dn_1234 = avg_sig1234-1.833*(stdev_sig1234/sqrt(reps));
      up_1234 = avg_sig1234+1.833*(stdev_sig1234/sqrt(reps));

  time_t lt;                                  // for time display
FILE *results;
results = fopen("c:\\tc\\output\\prosig.out", "w");
      if(!results){printf("\n\t Cannot save output file! Diskette may be \
write protected");
             exit(0);}

  lt = time(NULL);                     // display time/date in the file
```

```
  fprintf(results,ctime(&lt));

                      // save the results in a file

  fprintf(results,"\t * * * P R O F I T  S I G N I F I C A N C E * * *\n");
  fprintf(results,"\n\t    1   %8.0f   %8.0f   %8.0f",dn_1,avg_sig1,up_1);
  fprintf(results,"\n\t    2   %8.0f   %8.0f   %8.0f",dn_2,avg_sig2,up_2);
  fprintf(results,"\n\t    3   %8.0f   %8.0f   %8.0f",dn_3,avg_sig3,up_3);
  fprintf(results,"\n\t    4   %8.0f   %8.0f   %8.0f",dn_4,avg_sig4,up_4);
  fprintf(results,"\n\t   12   %8.0f   %8.0f   %8.0f",dn_12,avg_sig12,up_12);
  fprintf(results,"\n\t   13   %8.0f   %8.0f   %8.0f",dn_13,avg_sig13,up_13);
  fprintf(results,"\n\t   14   %8.0f   %8.0f   %8.0f",dn_14,avg_sig14,up_14);
  fprintf(results,"\n\t   23   %8.0f   %8.0f   %8.0f",dn_23,avg_sig23,up_23);
  fprintf(results,"\n\t   24   %8.0f   %8.0f   %8.0f",dn_24,avg_sig24,up_24);
  fprintf(results,"\n\t   34   %8.0f   %8.0f   %8.0f",dn_34,avg_sig34,up_34);
  fprintf(results,"\n\t  123   %8.0f   %8.0f
%8.0f",dn_123,avg_sig123,up_123);
  fprintf(results,"\n\t  124   %8.0f   %8.0f
%8.0f",dn_124,avg_sig124,up_124);
  fprintf(results,"\n\t  134   %8.0f   %8.0f
%8.0f",dn_134,avg_sig134,up_134);
  fprintf(results,"\n\t  234   %8.0f   %8.0f
%8.0f",dn_234,avg_sig234,up_234);
  fprintf(results,"\n\t 1234   %8.0f   %8.0f
%8.0f",dn_1234,avg_sig1234,up_1234);

  fclose(results);   // close the file
}


//-------------------------- cost_significance ---------------------------
void cost_significance(void)
{
        // calculate cost significance and confidence intervals

  float avg_sig1, avg_sig2, avg_sig3, avg_sig4, avg_sig12, avg_sig13,
     avg_sig14, avg_sig23, avg_sig24, avg_sig34, avg_sig123,
     avg_sig124, avg_sig134, avg_sig234, avg_sig1234,
   sqr_sig1, sqr_sig2, sqr_sig3, sqr_sig4, sqr_sig12, sqr_sig13,
   sqr_sig14, sqr_sig23, sqr_sig24, sqr_sig34, sqr_sig123,
   sqr_sig124, sqr_sig134, sqr_sig234, sqr_sig1234,
   stdev_sig1, stdev_sig2, stdev_sig3, stdev_sig4, stdev_sig12,
   stdev_sig13, stdev_sig14, stdev_sig23, stdev_sig24,stdev_sig34,
   stdev_sig123, stdev_sig124, stdev_sig134, stdev_sig234, stdev_sig1234,
   dn_1, up_1, dn_2, up_2, dn_3, up_3, dn_4, up_4, dn_12, up_12, dn_13, up_13,
   dn_14, up_14, dn_23, up_23, dn_24, up_24, dn_34, up_34, dn_123, up_123,
   dn_124, up_124, dn_134, up_134, dn_234, up_234, dn_1234,up_1234,

   sig1[11], sig2[11], sig3[11], sig4[11], sig12[11], sig13[11],
   sig14[11], sig23[11], sig24[11], sig34[11], sig123[11], sig124[11],
   sig134[11], sig234[11], sig1234[11];

     avg_sig1=avg_sig2=avg_sig3=avg_sig4=avg_sig12=avg_sig13=0.0;
     avg_sig14=avg_sig23=avg_sig24=avg_sig34=avg_sig123=0.0;
     avg_sig124=avg_sig134=avg_sig234=avg_sig1234=0.0;
   sqr_sig1=sqr_sig2=sqr_sig3=sqr_sig4=sqr_sig12=sqr_sig13=0.0;
   sqr_sig14=sqr_sig23=sqr_sig24=sqr_sig34=sqr_sig123=sqr_sig124=0.0;
   sqr_sig134=sqr_sig234=sqr_sig1234=0.0;
      stdev_sig1=stdev_sig2=stdev_sig3=stdev_sig4=stdev_sig12=0.0;
      stdev_sig13=stdev_sig14=stdev_sig23=stdev_sig24=stdev_sig34=0.0;
      stdev_sig123=stdev_sig124=stdev_sig134=stdev_sig234=stdev_sig1234=0.0;
   dn_1=up_1=dn_2=up_2=dn_3=up_3=dn_4=up_4=dn_12=up_12=dn_13=up_13=dn_14=0.0;
   up_14=dn_23=up_23=dn_24=up_24=dn_34=up_34=dn_123=up_123=dn_124=up_124=0.0;
```

```
   dn_134=up_134=dn_234=up_234=dn_1234=up_1234=0.0;

 for(i=0;i<=reps;i++){
   sig1[i] = sig2[i] = sig3[i] = sig4[i] = sig12[i] = sig13[i] = 0.0;
   sig14[i] = sig23[i] = sig24[i] = sig34[i] = sig123[i] = 0.0;
   sig124[i] = sig134[i] = sig234[i] = sig1234[i] = 0.0;}

  for(i=1;i<=reps;i++)        // i: replication number
   {
   for(j=1;j<=16;j++)         // j: level
    {
   sig1[i] +=factor[1][j]*avg_AW_cost[j][i]/8;
   sig2[i] +=factor[2][j]*avg_AW_cost[j][i]/8;
   sig3[i] +=factor[3][j]*avg_AW_cost[j][i]/8;
   sig4[i] +=factor[4][j]*avg_AW_cost[j][i]/8;
   sig12[i] +=factor[1][j]*factor[2][j]*avg_AW_cost[j][i]/8;
   sig13[i] +=factor[1][j]*factor[3][j]*avg_AW_cost[j][i]/8;
   sig14[i] +=factor[1][j]*factor[4][j]*avg_AW_cost[j][i]/8;
   sig23[i] +=factor[2][j]*factor[3][j]*avg_AW_cost[j][i]/8;
   sig24[i] +=factor[2][j]*factor[4][j]*avg_AW_cost[j][i]/8;
   sig34[i] +=factor[3][j]*factor[4][j]*avg_AW_cost[j][i]/8;
   sig123[i] +=factor[1][j]*factor[2][j]*factor[3][j]*avg_AW_cost[j][i]/8;
   sig124[i] +=factor[1][j]*factor[2][j]*factor[4][j]*avg_AW_cost[j][i]/8;
   sig134[i] +=factor[1][j]*factor[3][j]*factor[4][j]*avg_AW_cost[j][i]/8;
   sig234[i] +=factor[2][j]*factor[3][j]*factor[4][j]*avg_AW_cost[j][i]/8;
   sig1234[i]
+=factor[1][j]*factor[2][j]*factor[3][j]*factor[4][j]*avg_AW_cost[j][i]/8;
    }
   }

   for(i=1;i<=reps;i++)
     {
      avg_sig1 += sig1[i]/reps;
      avg_sig2 += sig2[i]/reps;
      avg_sig3 += sig3[i]/reps;
      avg_sig4 += sig4[i]/reps;
      avg_sig12 += sig12[i]/reps;
      avg_sig13 += sig13[i]/reps;
      avg_sig14 += sig14[i]/reps;
      avg_sig23 += sig23[i]/reps;
      avg_sig24 += sig24[i]/reps;
      avg_sig34 += sig34[i]/reps;
      avg_sig123 += sig123[i]/reps;
      avg_sig124 += sig124[i]/reps;
      avg_sig134 += sig134[i]/reps;
      avg_sig234 += sig234[i]/reps;
      avg_sig1234 += sig1234[i]/reps;
     }

  for(i=1;i<=reps;i++)
     {
      sqr_sig1 += pow((avg_sig1 - sig1[i]),2);
      sqr_sig2 += pow((avg_sig2 - sig2[i]),2);
      sqr_sig3 += pow((avg_sig3 - sig3[i]),2);
      sqr_sig4 += pow((avg_sig4 - sig4[i]),2);
      sqr_sig12 += pow((avg_sig12 - sig12[i]),2);
      sqr_sig13 += pow((avg_sig13 - sig13[i]),2);
      sqr_sig14 += pow((avg_sig14 - sig14[i]),2);
      sqr_sig23 += pow((avg_sig23 - sig23[i]),2);
      sqr_sig24 += pow((avg_sig24 - sig24[i]),2);
      sqr_sig34 += pow((avg_sig34 - sig34[i]),2);
      sqr_sig123 += pow((avg_sig123 - sig123[i]),2);
      sqr_sig124 += pow((avg_sig124 - sig124[i]),2);
```

```
      sqr_sig134 += pow((avg_sig134 - sig134[i]),2);
      sqr_sig234 += pow((avg_sig234 - sig234[i]),2);
      sqr_sig1234 += pow((avg_sig1234 - sig1234[i]),2);
     }

   stdev_sig1 = sqrt(sqr_sig1/reps);
   stdev_sig2 = sqrt(sqr_sig2/reps);
   stdev_sig3 = sqrt(sqr_sig3/reps);
   stdev_sig4 = sqrt(sqr_sig4/reps);
   stdev_sig12 = sqrt(sqr_sig12/reps);
   stdev_sig13 = sqrt(sqr_sig13/reps);
   stdev_sig14 = sqrt(sqr_sig14/reps);
   stdev_sig23 = sqrt(sqr_sig23/reps);
   stdev_sig24 = sqrt(sqr_sig24/reps);
   stdev_sig34 = sqrt(sqr_sig34/reps);
   stdev_sig123 = sqrt(sqr_sig123/reps);
   stdev_sig124 = sqrt(sqr_sig124/reps);
   stdev_sig134 = sqrt(sqr_sig134/reps);
   stdev_sig234 = sqrt(sqr_sig234/reps);
   stdev_sig1234 = sqrt(sqr_sig1234/reps);

                          // 90% confidence intervals

      dn_1 = avg_sig1-1.833*(stdev_sig1/sqrt(reps));
      up_1 = avg_sig1+1.833*(stdev_sig1/sqrt(reps));
      dn_2 = avg_sig2-1.833*(stdev_sig2/sqrt(reps));
      up_2 = avg_sig2+1.833*(stdev_sig2/sqrt(reps));
      dn_3 = avg_sig3-1.833*(stdev_sig3/sqrt(reps));
      up_3 = avg_sig3+1.833*(stdev_sig3/sqrt(reps));
      dn_4 = avg_sig4-1.833*(stdev_sig4/sqrt(reps));
      up_4 = avg_sig4+1.833*(stdev_sig4/sqrt(reps));
      dn_12 = avg_sig12-1.833*(stdev_sig12/sqrt(reps));
      up_12 = avg_sig12+1.833*(stdev_sig12/sqrt(reps));
      dn_13 = avg_sig13-1.833*(stdev_sig13/sqrt(reps));
      up_13 = avg_sig13+1.833*(stdev_sig13/sqrt(reps));
      dn_14 = avg_sig14-1.833*(stdev_sig14/sqrt(reps));
      up_14 = avg_sig14+1.833*(stdev_sig14/sqrt(reps));
      dn_23 = avg_sig23-1.833*(stdev_sig23/sqrt(reps));
      up_23 = avg_sig23+1.833*(stdev_sig23/sqrt(reps));
      dn_24 = avg_sig24-1.833*(stdev_sig24/sqrt(reps));
      up_24 = avg_sig24+1.833*(stdev_sig24/sqrt(reps));
      dn_34 = avg_sig34-1.833*(stdev_sig34/sqrt(reps));
      up_34 = avg_sig34+1.833*(stdev_sig34/sqrt(reps));
      dn_123 = avg_sig123-1.833*(stdev_sig123/sqrt(reps));
      up_123 = avg_sig123+1.833*(stdev_sig123/sqrt(reps));
      dn_124 = avg_sig124-1.833*(stdev_sig124/sqrt(reps));
      up_124 = avg_sig124+1.833*(stdev_sig124/sqrt(reps));
      dn_134 = avg_sig134-1.833*(stdev_sig134/sqrt(reps));
      up_134 = avg_sig134+1.833*(stdev_sig134/sqrt(reps));
      dn_234 = avg_sig234-1.833*(stdev_sig234/sqrt(reps));
      up_234 = avg_sig234+1.833*(stdev_sig234/sqrt(reps));
      dn_1234 = avg_sig1234-1.833*(stdev_sig1234/sqrt(reps));
      up_1234 = avg_sig1234+1.833*(stdev_sig1234/sqrt(reps));

                        // for time display
FILE *results;
results = fopen("c:\\tc\\output\\prosig.out", "a");
      if(!results){printf("\n\t Cannot save output file! Diskette may be \
write protected");
             exit(0);}
                        // save the results in a file
  fprintf(results,"\n\n\t * * *  C O S T   S I G N I F I C A N C E * * *\n");
  fprintf(results,"\n\t    1   %8.0f   %8.0f   %8.0f",dn_1,avg_sig1,up_1);
```

```
  fprintf(results,"\n\t    2   %8.0f   %8.0f   %8.0f",dn_2,avg_sig2,up_2);
  fprintf(results,"\n\t    3   %8.0f   %8.0f   %8.0f",dn_3,avg_sig3,up_3);
  fprintf(results,"\n\t    4   %8.0f   %8.0f   %8.0f",dn_4,avg_sig4,up_4);
  fprintf(results,"\n\t   12   %8.0f   %8.0f   %8.0f",dn_12,avg_sig12,up_12);
  fprintf(results,"\n\t   13   %8.0f   %8.0f   %8.0f",dn_13,avg_sig13,up_13);
  fprintf(results,"\n\t   14   %8.0f   %8.0f   %8.0f",dn_14,avg_sig14,up_14);
  fprintf(results,"\n\t   23   %8.0f   %8.0f   %8.0f",dn_23,avg_sig23,up_23);
  fprintf(results,"\n\t   24   %8.0f   %8.0f   %8.0f",dn_24,avg_sig24,up_24);
  fprintf(results,"\n\t   34   %8.0f   %8.0f   %8.0f",dn_34,avg_sig34,up_34);
  fprintf(results,"\n\t  123   %8.0f   %8.0f
%8.0f",dn_123,avg_sig123,up_123);
  fprintf(results,"\n\t  124   %8.0f   %8.0f
%8.0f",dn_124,avg_sig124,up_124);
  fprintf(results,"\n\t  134   %8.0f   %8.0f
%8.0f",dn_134,avg_sig134,up_134);
  fprintf(results,"\n\t  234   %8.0f   %8.0f
%8.0f",dn_234,avg_sig234,up_234);
  fprintf(results,"\n\t 1234   %8.0f   %8.0f
%8.0f",dn_1234,avg_sig1234,up_1234);

  fclose(results);

}


//------------------------------ initialize_1 ------------------------------
void initialize_1(void)
{
            //------- initialize integers --------
 w=s=i=j=k=0;
 v=c=x=y=m=Ap=try_capability=c1=c2=c3=0;
 quantity_unit=design=0;LoB_des=LoA_des=LoT_des=LoR_des=pdi=0;

  for(i=0;i<=15;i++) existing_capability[i]=status_capability[i]=0;


              //------- initialize floats --------

  ran1=LoB=LoA=LoT=LoR=indicator=0.0;
  PW_profit=PW_cost=period_profit=period_cost=Ck=Dk=Ak=Tk=0.0;

  total_idle=mean_capacity=uniform_value=0.0;
  excess=pdf=portion=total_effort=0.0;

  for(i=0;i<=101;i++)
     { revenue[i]=0.0;
       Z[i]=Q[i]=b[i]=0.0;
     }

  for(i=0;i<=15;i++)
   { effort_cost[i]=effort_time[i]=extra_remaining[i]=hour_purchase[i]=0.0;}

  for(i=0;i<=15;i++)
     {
     for(j=0;j<=101;j++)
       {remaining_capacity[i][j]=0.0;}
     }

 i=j=0;

}
```

```
//-------------------- Generate the next random number---------------------

float rand(int stream)
{
       long zi, lowprd, hi31;

       zi      = zrng[stream];
       lowprd  = (zi & 65535) * MULT1;
       hi31    = (zi >> 16) * MULT1 + (lowprd >> 16);
       zi      = ((lowprd & 65535) - MODLUS) +
                 ((hi31 & 32767) << 16) + (hi31 >> 15);
       if (zi < 0) zi += MODLUS;
       lowprd  = (zi & 65535) * MULT2;
       hi31    = (zi >> 16) * MULT2 + (lowprd >> 16);
       zi      = ((lowprd & 65535) - MODLUS) +
                 ((hi31 & 32767) << 16) + (hi31 >> 15);
       if (zi < 0) zi += MODLUS;
       zrng[stream] = zi;
       return ((zi >> 7 | 1) + 1)/ 16777216.0;
}

// Set the current zrng for stream "stream" to zset.

void randst (long zset, int stream)
{
       zrng[stream] = zset;
}

//---------------------------End of  program----------------------------
```

Vita

Shamil Daghestani was born in Rochester, Minnesota, in April 29, 1970. He obtained both B.S. and M.S. degrees in Industrial and Systems Engineering from Virginia Polytechnic Institute and State University. During his internships, Shamil worked for ABB (Asea Brown Boveri) and Lucent Technologies. Shamil also held teaching assistantship positions during his college career. In 1998, he started working full-time as a Supply Chain Management Engineer at Viasystems Technologies in Richmond, Virginia.