SELECTION AND EVALUATION OF JOINT TYPES AND JOINING PROCESSES FOR CONCURRENT ASSEMBLY/DISASSEMBLY-BASED DESIGN

by

Piyen Chang

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Industrial and Systems Engineering

APPROVED:

J. Shewchuk

Dr. John P. Shewchuk, Chairman

William G. Sullivan

Dr. William G. Sullivan

John E. Kobza

Dr. John E. Kobza

November 1996
Blacksburg, Virginia

Key words: Joining, Assembly, Disassembly, Design-for-Assembly, Design-for-Disassembly

c.2

LD
5655
V855
1996
C4357
c.2

SELECTION AND EVALUATION OF JOINT TYPES AND JOINING PROCESSES FOR CONCURRENT ASSEMBLY/DISASSEMBLY-BASED DESIGN

by

Piyen Chang

Dr. John P. Shewchuk, Chairman

Industrial and Systems Engineering

(ABSTRACT)

In designing products, Design-for-Assembly (DFA) has been successfully used for several decades to reduce lead times, processing times, and equipment overhead. Though the DFA approach results in products which are easy and efficient to assemble, such products may be difficult to disassemble and/or may adversely affect the environment. These environmental concerns resulted in the Design-for-Disassembly (DFD) approach, which stresses ease-of-disassembly and environmental compatibility. However, when applied independently of DFA, DFD underestimates the importance of assembly, and consequently can result in increased assembly time and cost. Design-for-Assembly may thus have negative repercussions on disassembly, and vice-versa. Consequently, in order to minimize assembly/disassembly time and cost and maximize component reusability, designers must implement DFA and DFD simultaneously when designing products. In this research, such an approach is developed. The approach, called Concurrent Assembly/Disassembly-Based Design (CAD^2), consists of simultaneously selecting joint types and joining processes for products, based upon both assembly and disassembly requirements. Two objectives are considered: the minimization of total assembly/disassembly time or cost. In addition, a 'penalty score' measure is developed to quantify the environmental impact (recyclability) associated with any solution. Total enumeration is used to solve these minimization problems. The CAD^2 approach is demonstrated and evaluated by comparing it with both DFA and DFD for a limited number of cases. The results indicate that the CAD^2 approach can give better solutions (total time and total cost) than either DFA or DFD.

Acknowledgments

I would like to express my sincere thanks to Dr. John P. Shewchuk who not only served as chairman on my committee but also provided expertise in the accomplishment of this thesis. By expending significant amounts of time to review and revise the work, he contributed to the writing of the final document.

I would also like to thank Dr. William G. Sullivan and Dr. John E. Kobza for serving on my committee. Their suggestions and guidance were most valuable to my work.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Design for Assembly

A primary goal in the manufacture of discrete products is to minimize cost yet maintain acceptable product quality. One avenue for achieving this goal is to improve the design of the product. However, when this idea was introduced, the emphasis was on the design and manufacture of the individual parts for better production: little attention was given to assembly [Boothroyd, 1982; Boothroyd, 1992]. This approach (emphasizing the design and manufacture of the individual parts) increased the lead time required for introducing new products into the marketplace, as parts could not easily be assembled and redesigns were subsequently required. In addition to increasing lead time, these redesigns also resulted in increased costs [Boothroyd, 1992].

Later, engineers found that in practice, assembly time is about half of the total production time [Andreasen, 1988]. To improve productivity, designers endeavored to reduce assembly costs by redesigning products for easy assembly. In particular, they focused on reducing the number of parts and simplifying those parts remaining. Simple parts are easy to manufacture and assemble, and fewer parts means reduced assembly time. Hence, designers were able to reduce the total production time tremendously. However, appropriate assembly processes were still not emphasized.

Finally, engineers realized that modifying the assembly process can greatly reduce product cost. Guidelines have appeared in handbooks for achieving efficient assembly designs: the implementation of such guidelines is known as "Design for Assembly" (DFA). Design for Assembly (DFA) considers assembly to be an important factor in determining product cost.

Since manufacture and assembly are complicated processes, qualitative information, instead of quantitative, is given to designers to use in the early stages of product design. These qualitative guidelines help designers identify efficient assembly processes. However, efficient assembly processes may cause problems for future disassembly processes. This problem is of ever-increasing concern today because of the growing demand for environmentally-friendly products, where ideally, the entire product can be disassembled for recovery, recycling, or reuse. For reducing product cost and obtaining environmentally-friendly products, DFA alone is insufficient, as it ignores disassembly processes completely. As designers know, many uncomplicated and inexpensive methods for assembling products may turn out to be inefficient, time-consuming, or costly processes with regard to disassembly. For example, spot welding is a fast and inexpensive joining process for sheet metal, yet it is almost impossible to separate parts joined in this manner without damaging them. In addition, disassembly costs in such cases may be several times more than assembly costs. Thus, DFA alone is not an appropriate method for designing products.

1.2 Design for Disassembly

In 1993, statistics show that nearly 350 million home and office appliances were disposed of [Dewhurst, 1993]. Each year, more than 10 million cars reach the end of their service lives. Furthermore, it is estimated that nearly 150 million personal computers will be discarded by 2005 [Chen, 1994].

For protecting our environment, existing products must be disassembled at the end of their life cycle and recycled. In addition, a growing concern for the environment has caused industries to focus on manufacturing environmentally-friendly products. It is expected that legislation will force manufacturers to take back a certain percentage of end-of-life products in the future. All of these reasons have made "Design for Disassembly" (DFD) a hot topic with two main research areas. The first concerns how to take products apart for profit; the second involves how to redesign products for ease of disassembly.

The first research area focuses on building economic models for balancing the cost of recycling processes and the profits from recycling. Several recycling alternatives exist: (i) reuse parts, (ii) remanufacture products, (iii) separate materials for recycling, and (iv) dispose of the remaining parts. These alternatives seem to include most of the primary points for recycling products, but using disposal costs for measuring the impact on the environment is still not an appropriate solution.

The second research area is redesigning parts for ease of disassembly. Since today's products are not always designed for easy disassembly, various problems may occur; e.g., some types of materials will cause problems for disposal, the geometry of parts can be difficult for disassembly, and certain fasteners can make products very difficult to disassemble. Clearly, a new approach will be necessary in the design of products to solve problems related to production. Therefore, rules and guidelines for product design that affect ease in disassembly are being developed. Some researchers are working to build quantitative models for evaluating the difficulty of redesigning products. These quantitative methods use index numbers to indicate results that can be easily calculated and evaluated, but the index number cannot give designers the actual time and cost needed to assemble and disassemble a product [Dewhurst, 1993; Kroll, 1994].

While DFD aims at selecting efficient disassembly processes in order to save more components for reuse/recycling and reduce disposal costs, the approach may result in the selection of inappropriate assembly processes. For example, designers may have several alternative joint types and assembly processes to choose from when the minimum time and cost for disassembly processes have been determined. If designers just focus on disassembly, they may select joint types which cause problems for assembly. Therefore, DFD alone is not an appropriate method for designing products.

1.3 Concurrent Assembly/Disassembly-Based Design

As shown in the previous sections, design-for-assembly may have negative repercussions on disassembly, and design-for-disassembly may have negative repercussions on assembly. Consequently, in order to minimize assembly/disassembly time and cost and maximize component reusability, designers must implement DFA and DFD simultaneously when they design a product.

One approach to consider when obtaining these objectives is the selection of joint types and joining processes to be used. Most of the time, there exists several ways to assemble parts and several corresponding assembly machines and tools for performing joining processes. In addition, product disassembly is the precondition of recycling, and how well a product can be demanufactured is based upon the types of joints and corresponding disassembly processes. Therefore, in designing products, a great deal of attention should be paid to the various types of joints and corresponding assembly and disassembly processes to be used. Furthermore, joint type selection (design function) is usually performed independently of joining process selection (processing planning function): the separation of functions may result in problems which cause redesigns to be required. Thus, these functions should be integrated, along with the cost-estimating function, in order to reduce both assembly/disassembly time and cost.

The objective of this thesis is to show how this concurrent assembly/disassembly-based design (CAD^2) approach can be developed and implemented. Specifically, the goal is to develop and demonstrate a method for simultaneous selection of joint types and joining processes for products, based upon consideration of <u>both</u> assembly and disassembly requirements.

1.4 Outline of Thesis

The remainder of this document is divided into four chapters:

Chapter 2 includes reviews and summaries of previous work concerning assembly and disassembly, along with the advantages and disadvantages of each approach. This chapter is divided into three sections: the first section covers design for assembly; the second, design for disassembly; and the third, design for both assembly and disassembly.

Chapter 3 describes the CAD^2 approach in terms of the modeling framework, objective functions, enumeration algorithm, and how the algorithm is implemented as software.

Chapter 4 describes the evaluation methodology. Four products are investigated in order to demonstrate that the CAD^2 approach is appropriate.

Chapter 5 presents a summary and identifies some areas for future work.

CHAPTER 2

LITERATURE REVIEW

Section 2.1 will introduce several research papers that deal with Design for Assembly. Section 2.2 presents the Design for Disassembly research papers. Section 2.3 introduces those research papers that have considered assembly and disassembly simultaneously.

2.1 Design for Assembly

Yoosufani, Ruddy, and Boothroyd [1983] evaluated the effect of part symmetry on manual assembly times. Their study revealed that geometrical design features will affect the time for grasp and orient operations. For evaluating the effect of part symmetry, the shape of parts were categorized into Alpha and Beta symmetries. Alpha symmetry depends on the angle through which a part must be rotated about an axis perpendicular to the axis of insertion. Beta symmetry depends on the angle through which a part must be rotated at the axis of insertion. The process was performed by experimentation: time values were measured by Methods Time Measurement (MTM). The results indicated that part symmetry can greatly reduce the assembly time for manual assembly. The importance of part symmetry increases with the degree of automation, as automated assembly systems are not as flexible as manual ones. However, the definitions of symmetry used for manual assembly cannot be used for handling parts automatically, because robots can only rotate parts in one direction (based on Alpha symmetry) after they are gripped. Despite this drawback, the paper gives designers a good example for when they are designing parts.

Boothroyd [1987] created a handbook which provides a selection of appropriate assembly methods for products. The methods of assembly processes are classified into three categories: manual assembly, special-purpose transfer machine assembly, and robotic assembly. When applying a method, the type of assembly process needs to be determined first by calculating the ratio of the total number of parts to the average number of parts for existing products, and finding the corresponding results from the charts provided in the handbook. Once the assembly method has been decided, the design efficiency needs to be calculated in order to compare the different designs. Design efficiency is obtained by using the theoretical minimum number of parts divided by the total time to assemble the currently-designed product. Thus, the higher the score, the better the design. Obviously, if the assembly time can be reduced, the design efficiency would be higher. Even though this handbook has been utilized successfully in industry, several points may result in disaster. First, in practice, to assemble a product may involve different types of processes. Therefore, it seems to be inappropriate to decide on a single type of assembly method for a product. Second, to obtain the ideal minimum number of parts is very subjective which can mislead designers to combine parts in an impracticable way. In many instances, the elimination of parts cannot be achieved because of other constraints, such as the cost of manufacture or the requirement of specialized equipment to manufacture the combined parts. In addition, all of these rules (obtained by MTM) are based on the assumption that a worker can only use one hand at a time: in practice, this is usually not true. The author claims that workers can use both hands at the same time and he suggests that the total time can be divided by 1.5. This is not an appropriate alternative for estimating assembly time because it is unlikely that the time savings resulting from using both hands are constant. Furthermore, no justification is presented for use of the given factor.

Andreasen [1988] presented a Design-for-Assembly approach by collecting, sorting and formulating the knowledge into a handbook. In this handbook, the assembly function has been divided into three sub-functions: handling, composing and checking. Rules and factors are generated for these three sub-functions. These rules can only provide the designers with basic concepts to design parts for assembly processes. Additionally, the rules and factors contain several drawbacks. First, factors in assembly are too broad for designers to follow so no one knows which factors would take first priority. Second, designers cannot estimate the cost and time of assembly processes by just having these rules in mind. Third, no rules will assist designers in selecting the alternative joint types and corresponding joining methods for joining processes.

Yamagiwa [1988] presented "An Assembly Ease Evaluation Method for Product Designers: DAC." Design for assembly cost-effectiveness (DAC) was developed to increase productivity at Sony Corporation. This method helps designers evaluate the assembly process during the design stage. Factors such as the shape of a part, the direction of assembly, and other factors are classified into thirty key words. The evaluation is based on these key words and a special form. One-hundred points are given for every assembly operation. The objective is to maximize the assembly DAC score: this is done by starting with the lowest DAC score and attempting to improve it. The method suggests two processes for when a low DAC score is encountered. The first is to change the structure of that operation. This requires a very experienced designer to perform the job in order to balance the line. The second is to move the lowest DAC to the last operation, i.e., putting the bottleneck station at the end of the assembly line.

Miyakawa [1990] introduced the Assemblability Evaluation Method (AEM). Part of AEM uses penalty scores to evaluate the difficulty for assembling a product by redesigning parts and processes to obtain lower penalty scores. The score is combined by using an assembly direction and a type of joint for assembly processes. Such an evaluation of the joint type by ease of assembly cannot meet today's recycling requirements because it is subjective and cannot be used to evaluate the end-of-life products. The penalty scores should consider assembly and disassembly methods simultaneously in order to provide more reliable indicators for future requirements.

Valentovic [1993] presented a new Design-for-Assembly method. He based it on Yoosufani's [1983] principle of using symmetrical parts in assembly, and investigated the kinematic problems of asymmetry. He used three-dimensional space for evaluating the degree of movement of parts instead of using Yoosufani's [1983] Alpha and Beta symmetries. The degree of freedom of a set of members was calculated to evaluate the assemblability. The results indicate only the difficulty of the design of a product.

2.2 Design for Disassembly

Remich [1991] presented the famous electric kettle called "Ukettle" [manufactured in United Kingdom] to describe the idea of design for disassembly. In this paper, titled "First Recyclable Appliance," he introduced a revolutionary product which could be disassembled just by pulling it apart. Design-for-Disassembly is the foundation of this special kettle. The whole product uses only two screws. The rest of the joining processes use "snap fits" to assemble. Obviously, at the end of its useful life, such a product can be recycled acceptably. The author suggests the following recycle design considerations: utilize compatible materials, use recyclable materials, minimize material count, minimize assembly operation, and design for uncomplicated separation. These key points can only give designers general guidelines when they design a product for disassembly; they do not give the designers an exact answer about the cost and time for and disassembly. Quantitative models are more appropriate for aiding designers in evaluating the assembly and disassembly processes than are qualitative guidelines, and will yield better results.

Dewhurst [1993] introduced two index numbers to evaluate the design efficiency for disassembly. The first index is the "Design for Service" (DFS) index, I_s. It measures the service efficiency, which is the efficiency for performing maintenance. The DFS Index is the service cost of the item divided by the service cost of the item and other costs (e.g., cost of items, labor cost, and cost of use of any special tools). If other costs can be reduced to zero, I_s will be one, which means that the service efficiency is high. On the other hand, increasing other factors will make I_s close to zero, which means that the service efficiency is low. The second index is I_r, the recycling efficiency. This index is obtained by using the value of reclaimed parts minus the cost of disposal and disassembly process cost, then dividing that number by the ideal maximum reclaimable value. The author claims that the efficiency of recycling can be evaluated by this index. Unfortunately, no sufficient example exists indicating how the index would work. In addition, the value could be negative when the reclaimable value is lower than the disposal cost and disassembly cost. He suggests nothing for designers to handle such a condition.

Kroll [1994] introduced a disassembly evaluation chart and efficiency index for evaluating the improvement of design for easier disassembly. First, the disassembly evaluation chart is prepared: the disassembly task type, disassembly direction and disassembly tool requirements are the main attributes. For calculating the efficiency index, several attributes were built into the evaluation chart: these include accessibility, position, force, and time. Attributes are rated according to degree of difficulty in assembly. The efficiency is calculated by using the difficulty of ideal design divided by the difficulty of current design. This paper simplifies Boothroyd's assembly method and changes it into a disassembly mode. Therefore, the drawbacks are similar to Boothroyd's method. First, to define the minimum ideal number of parts for a product might be difficult. Second, almost all the attributes appear in terms of index numbers, which are very subjective. Finally, improving the design of disassembly does not guarantee that the assemblability will be improved simultaneously.

Chen [1994] integrated the economic and environmental concerns of product design. She evaluated how recyclability can be investigated during the product design stage. The improvement is obtained by changing the material, structural layout, and interpart connections. The type and quantity of joints are considered in the paper. However, some corresponding variables are not included in the proposed mathematical model. Therefore, although this model includes most of the assembly, disassembly, and recycling variables, the results of the model are questionable.

VerGow [1994] evaluated the German fastener selection standard (VDI 2243) for recycling in different aspects of a process. The VDI 2243 presents the selection method by using a matrix. Thirteen alternatives were selected as column indicators (types of joints), and eight attributes were used as row indicators (strength, fatigue, and recycling efficiency). Each coordinate bin is presented by a qualitative symbol which indicates good, average, and bad. To fully evaluate the model provided by VDI 2243, the selection table is implemented as a Decision Support Problem (DSP). The DSP is based on multiple attributes to rank the alternatives: the good, average, and bad rating are indicated by the values 1, 2, and 3 respectively. Accordingly, the DSP will find the lowest numerical rating of the alternatives using a computer code. Uncertainty is built into the attributes in order to take the resulting variation into account. The advantage of this method is its simplicity and ability to be implemented by visual effect. Another advantage is that this method can list different attributes for selecting fasteners in order to meet different material and product recycling aspects of a process. Such an advantage could also be a disadvantage, however, because some fasteners may be ranked high, which may be incorrect for certain applications. If the selection of attributes and alternatives is based on applications, the method will be sufficient for designing the assembly and disassembly processes. Notwithstanding, to categorize applications into different attributes is a painstaking job. Another disadvantage of this method is that the qualitative information cannot represent the true story because it is too subjective.

Johnson and Wang [1995] aimed at improving the efficiency of the disassembly planning process and generating an optimal disassembly sequence in order to maximize the material recovery of a product after end-of-life cycle. The authors have structured such a methodology using a four-level disassembly analysis. Level one, "Feasibility study of material recovery opportunities (MRO)", estimates the cost of recovery and disposal by the weight of a product. Level two, "Optimal disassembly sequence generation", generates the disassembly sequence by studying the precedence relations between components; compatible materials were clustered into a subgroup. Level three, "Disassembly optimization", uses the sequence generated at the preceding level in order to optimize the cost. Finally, Level four, "Design For

Disassembly (DFD) guidelines", is used to support the above levels to varying degrees, depending on a company's commitment. Clustering some types of parts can save disassembly processes, but generating optimal sequences will be difficult when a product has many parts.

Harjula, Rapoza, Knight, and Boothroyd [1996] presented a paper where two major factors are considered for ease of disassembly: financial aspects and environmental aspects. Financial aspects include: (i) the special treatment for end-of-life items, (ii) the value or cost for each end-of-life item, (iii) labor cost for disassembly, and (iv) disposal cost. For environmental aspects, the effect of material, energy, and toxicity are considered. Several cases are evaluated. As a result, some suggestions for optimizing profit are presented, such as removing toxic items as early as possible. This removal will cause a less toxic effect for the remaining parts. In addition, parts with the highest yield should be removed first in order to reach the marginal value at an early disassembly stage. However, the disassembly sequence used in this paper assumes a reversal of the DFA assembly lists, and parts reduction is the main concept of Boothroyd's DFA. By using the above approach, the manufacturing cost of parts may be underestimated, and the results will be inaccurate.

Penev [1996] used graphs to present the disassembly processes. Each process is evaluated by different types of subprocesses. These five subprocesses are service, disassembly, dismantling, recycling, and disposal. Each subprocess has its own mathematical constraints in order to evaluate the regain of value added to goods and materials, as well as to protect the environment. The maximum profit among those subprocesses will be applied to such a disassembly process. Total regain will be added from every process. Separation of the disassembly process into five subprocesses (service, disassembly, dismantling, recycling, and disposal) is appropriate for evaluating disassembly. This is because each subprocess is characterized by different attributes. However, if the author could consider the effect of types of joints in his method, it would justify pre-selection of subprocesses, as most of the permanent type joints are not suitable for service and disassembly.

2.3 Design for Both Assembly and Disassembly

Boothroyd and Alting [1992] investigated several research papers about Design for Assembly (DFA) and Design for Disassembly (DFD). They pointed out that DFA results in reduced time-to-market and overhead, and they confirmed that the concurrent engineering concept is very important for design. Today, DFD influences recycling, because it can increase the reuse of parts and decrease the disposal of parts. Therefore, they concluded that only combined DFA and DFD can cover all life-cycle phases of the products.

Laperriere and Elmaraghy [1992] showed that given a graphical representation of a product with "n" components; the generation of an assembly sequence is equivalent to finding a sequence of "n-1" mutually exclusive cutsets. The graph consists of vertices which correspond to part, and edges which indicate the relationships between parts. To determine the possible sequence, the graph is split into two subassemblies. Then, the edges associated with the vertex are examined. They considered the three moving directions (x, y, and z) between parts, and assumed that the disassembly sequence was the reverse of the assembly sequence. This method will generate all the possible assembly sequences for a product. The degree of freedom of a sequence is translated into cost in order to evaluate the results of these sequences. If the size of parts is similar, one can use degree of freedom to evaluate the results. However, large parts may need more time to assemble and disassemble than small parts. The size of parts may cause inaccurate results for such an evaluation. In addition, some joining methods, such as welding, spot welding, and bonding, will have zero freedom. These joining operations will not be suitable for this model. In addition, it is hard to imagine calculating total time and cost for assembling and disassembling processes from this model.

CHAPTER 3

THE CAD^2 (CONCURRENT ASSEMBLY/DISASSEMBLY-BASED DESIGN) APPROACH

The Concurrent Assembly/Disassembly-Based Design (CAD^2) approach to product design consists of simultaneously selecting joint types and joining processes for a product, based upon both assembly and disassembly requirements. We are given the bill-of-materials for the end product, the lower-level items, and the requirement in joining the various items together (e.g. required joint strength): this is all part of the product design specification. Additionally, we are given a finite quantity of possible joints which can be used, and the possible resources (machines or human operators) and fixtures which can be used for performing the assembly/disassembly operations: these constitute the process. The goal is to simultaneously select the joints, resources, and fixtures, for both assembly and disassembly operations, in order to minimize the total time or total cost. These objectives may be conflicting: there is no guarantee that they can both be minimized simultaneously. Total cost usually dominates the other objective because of the overriding importance of reducing the cost of manufacturing operations. In the case where meeting due dates or maximizing throughput is of prime importance; however, total time may be more important. The tradeoffs among these two objectives will be highly situation-dependent: the ability to select between these two objectives will provide designers with flexibility when the CAD^2 approach is applied.

In addition to total time and total cost, we should also be concerned with the environmental impact any solution may cause. The cost of landfill is high and/or regulations force manufacturers to recycle end-of-life products in large quantities. If the quantity of disposed products can be reduced, the cost of landfill will be decreased. For determining the extent to which end-of-life products are reusable, and thus the extent to which a solution impacts the environment, penalty scores are used. The purpose of a penalty score is to reflect the information that total time and total cost cannot describe when evaluating the joining process. The penalty score is determined from two indices: (i) the maximum sustainable damage degree index and (ii) the expected damage degree index. The maximum sustainable damage degree index is given by designers when the joint type has been decided. The expected damage degree index is an index that is given for a joint. When joints are selected for operations, the penalty scores can be calculated. Thus, for any given solution (selection of joints, resources, and fixtures), the corresponding penalty score can be calculated.

We assume that a finite quantity of joint types and assembly/disassembly resources and fixtures exists. Thus, the problems of minimizing the total time and total cost associated with a given product are each combinatorial optimization problems. As the objective of this research is to develop and demonstrate the CAD^2 approach (and not necessarily to develop fast and efficient algorithms), total enumeration will be used to solve these optimization problems. In order to perform the required enumeration, an algorithm has been developed and implemented in software form. The CAD^2 design approach then consists of populating the required databases with the necessary product, joint, resource, and fixture data, and executing the program to find the minimum values of total time, total cost, and penalty scores.

This chapter consists of four sections. Section 3.1 presents the framework used for modeling joints, joining resources, fixturing devices and products. Section 3.2 describes the objective functions and defines the variables used in each. The enumeration algorithm developed is presented in Section 3.3. Finally, Section 3.4 describes the software implementation of the approach.

3.1 Modeling Framework

Joints

Joints are defined in terms of their class and type. A joint class consists of different types of joints which are thread fasteners, rivets, welding, or bonding. A joint type is the subgroup of a joint class. For example, screws and bolts are joint types which belong to the thread class. Joints can be continuous or discrete in nature. Continuous joints are those for substances capable of holding materials together by surface attachment: example of such joint types include epoxy bonds, silicone bonds, gas-tungsten arc welds (GTAW), or rod welds. Discrete joints are those which can be applied to solid parts and counted as pieces: examples of discrete joints include screws, bolts, rivets, spot welds, etc. We will use the indicator variable b_{jk} to denote the joint nature:

b_{jk} = 1, joint class j, type k is continuous.
0, joint class j, type k is discrete.

Each joint of a given class and type is specified in terms of twenty-two different variables. For joint class j, type k, these variables are as follows:

v_{jk} = size of a joint. It consists of x and y dimensions which indicate area, in units of inch^2 (the z dimension is the length of a joint).

s_{jkv} = joint strength for joints having size v.

u_{jk} = joint unit. One indicates each (i.e. discrete joints). Two indicates area, in units of inch^2 (continuous joints).

X_{jkv} = minimum required joining x dimension for size v joint, inches.

Y_{jkv} = minimum required joining y dimension for size v joint, inches.

f_{jk}	=	flexibility factor. Zero indicates the joint type is rigid (non-flexible), one indicates the joint type is flexible.
g_{jk}	=	part modification requirement. One indicates that the part needs to be modified. Zero indicates no modification is required.
c_{jkv}	=	required compressive force for assembling size *v* joint., lb.
t_{jkv}	=	required tensile force for assembling size *v* joint, lb.
q_{jkv}	=	required torque for assembling size *v* joint, in-lb.
r_{jkv}	=	1, tracing motion is required for assembling size *v* joint. 0, tracing motion is not required.
C_{jkv}	=	required compressive force for disassembling size *v* joint, lb.
T_{jkv}	=	required tensile force for disassembling size *v* joint, lb.
Q_{jkv}	=	required torque for disassembling size *v* joint, in-lb.
R_{jkv}	=	1, tracing motion is required for disassembling size *v* joint. 0, tracing motion is not required.
x_{jk}	=	*x* dimension of joint, inches.
y_{jk}	=	*y* dimension of joint, inches.
z_{jk}	=	*z* dimension of joint, inches. This indicates joint length.
D_{jk}	=	expected damage degree index. The index ranges from 1 to 10.
CJ_{jkv}	=	cost of a unit of joint, cents.

It is assumed that the suitability of a joint type for a given application is independent of the working environment associated with that application. Table 3.1 provides an example of how the above values might be specified for four joint classes and types. The first row represents thread class and screw type. The second row represents thread class with bolt joint type. The third row represents rivet class with mandrel joint type, and the fourth row represents snaps fit.

Table 3.1 Example Joint Data.

n	j	k	b_{jk}	s_{jk}	u_{jk}	X_{jkv}	Y_{jkv}	f_{jk}	g_{jk}	c_{jkv}	t_{jkv}	q_{jkv}	r_{jkv}	C_{jkv}	T_{jkv}	Q_{jkv}	R_{jkv}	x_{jk}	y_{jk}	z_{jk}	D_{jk}	CJ_{jkv}
1	1	1	0	100	1	0.5	0.5	0	1	0	95	0	0	50	0	75	0	0.2	0.2	0.5	6	2
2	1	2	0	65	1	0.5	0.5	1	1	50	0	70	0	50	0	70	0	0.2	0.2	1	1	4
3	2	1	1	85	2	1	1	0	0	0	0	0	1	0	0	0	1	0.3	0.3	1	10	15
4	3	1	1	65	3	0.2	0.2	0	0	0	0	0	1	0	0	0	1	1	1	1	1	4

Two other attributes used to specify joints are the joint length and joint area. Selecting the appropriate size for joints depends on the required joining depth and available joining area which in turn depend on the structure of the product. These attributes vary from product to product however, so they will be specified as part of the product data.

Resources

In order to perform assembly and disassembly, resources are required. Resources may be either human operators or machines. In either case, the resource will have associated with it certain capabilities and capacities which determine the types of joints the resource is capable of handling. These capabilities and capacities are specified in terms of nine different variables. For resource type *m*, these variables are as follows:

A_m	=	function accuracy.
		3, high accuracy.
		2, medium accuracy.
		1, low accuracy.
Fc_m	=	maximum compressive force, lb.
Ft_m	=	maximum tensile force, lb.

FQ_m	=	maximum torque, in-lb.
FR_m	=	1, resource can provide a tracing motion.
		0, resource cannot provide a tracing motion.
TS_m	=	setup time, seconds.
TP_m	=	process time, seconds.
Cl_m	=	labor cost, \$/hour.
CR_m	=	resource utilization cost, \$/hour.

Table 3.2 illustrates resource data for three different resources. The first row represents a machine which can provide 200 lb. compressive load. The second row represents a machine which can provide 300 lb. tensile load, and the third rows represents a machine which can provide a tracing motion.

Table 3.2 Example Resource Data.

m	A_m	Fc_m	Ft_m	FQ_m	FR_m	TS_m	TP_m	Cl_m	CR_m
1	1	200	0	0	0	45	12	15	10
2	2	0	300	0	0	35	30	15	8
3	3	0	0	0	1	90	75	18	20

Fixturing Devices

In addition to resources, fixtures are required for holding parts together during assembly/disassembly. Different shapes and sizes of products require different types of fixtures. Equally important, appropriately selecting a fixture will reduce the required time and cost for assembly/disassembly. Therefore, fixtures are considered as an important factor in assembly/disassembly. Fixtures are defined in terms of seven different variables. For fixture type *l*, these variables are as follows:

p_l	=	positional accuracy.
		1, low positional accuracy.
		2, medium positional accuracy.
		3, high positional accuracy.
h_l	=	holding force, lb.
d_l	=	holding force direction.
		1, external type.
		0, internal type.
hs_l	=	holding shape.
		1, flat.
		2, round.
		3, angled.
hA_l	=	holding area index.
TFU_l	=	time to fixture and unfixture a part, seconds.
CA_l	=	fixture utilization cost, $/hour.

Table 3.3 provides an example of how the above values might appear for three different types of fixtures. The first row shows that the fixture can resist force up to 200 lb. The second row show that the fixture can resist force up to 300 lb. The last row indicates the fixture can resist force up to 400 lb.

Table 3.3 Example Fixture Data.

l	p_l	h_l	d_l	hs_l	hA_l	TFU_l	CA_l
1	1	200	1	1	3	25	5
2	2	300	1	2	1	18	3
3	3	400	0	0	2	15	8

Products

Products are defined in terms of the entire set of operations required to take the product through its life cycle. Each product will consist of *N* assembly (joining) operations, followed by *M* disassembly (disjoining) operations. For each assembly operation, a set of requirements will exist. These specify the required functionality, strength, etc. of the joint and depend upon the product design. Joint requirements are specified in terms of thirteen different variables. For assembly operation *n*, these variables are as follows:

h_n	=	holding force direction.
		0, internal.
		1, external.
f_n	=	flexibility factor. One indicates a flexible joint type is required. Zero indicates a non-flexible joint type is required.
s_n	=	required joining strength, lb.
x_n	=	available joining *x* dimension, inches.
y_n	=	available joining *y* dimension, inches.
z_n	=	holding length, inches.
PS_n	=	product shape.
		1, flat.
		2, round.
		3, angled.
PZ_n	=	part size index.
		1, small.
		2, medium.
		3, large.
D_n	=	maximum sustainable damage degree index. The damage index ranges from 1 to 10.

J_n	=	required joining accuracy. 1, low. 2, medium. 3, high.
ST_n	=	solid thickness, inches.
MD_n	=	modification difficulty index. It ranks from one to ten. One indicates the lowest difficulty, ten the highest.

The cost of redesigning products to utilize different types of joints is not considered in the above model. An example of how the above values might be specified for three operations of a given product is presented in Table 3.4.

Table 3.4 Example Product Data.

n	h_n	f_n	s_n	x_n	y_n	z_n	PS_n	PZ_n	D_n	J_n	ST_n	MD_n
1	1	0	100	0.5	0.5	1	2	2	4	2	0.2	4
2	1	0	50	0.25	0.25	0.5	2	1	2	1	0.5	5
3	0	1	200	0.3	0.3	2	1	3	5	2	0.4	4

For each operation i, the joint class and joint type to be used must be specified.

Let

Ai	=	joint class for operation i.
Bi	=	joint type for operation i.

If the joint nature for a given operation is discrete (i.e., $b_{jk} = 0, j = Ai, k = Bi$), the quantity of fasteners must be specified. Similarly, if the joint nature is continuous ($b_{jk} = 1, j = Ai, k = Bi$), the joint area must be specified. A single variable will be used to indicate these items:

Nj_i	=	quantity of joints for operation i, $b_{jk} = 0$.
	=	area of joint for operation i, $b_{jk} = 1$.

The joint type, joint class, and above items must be specified to assure the joint requirements are all met.

Finally, for each operation, the resource type and fixture type used to both create the joint (in accordance with the joint requirements) and ultimately destroy the joint (demanufacture) must be specified. These are specified in terms of three different variables. For operation i, these variables are as follows:

Ci	=	joint dimension indicator for operation i.
Di	=	resource type for operation i.
Ei	=	fixture type for operation i.

3.2 Objective Function

As specified in the chapter introduction, two objective functions are utilized in the CAD^2 approach: total time and total cost. Each of these is developed as follows:

Total Time: The time required to perform each assembly/disassembly operation will depend upon four elements. These are the time required to fixture/unfixture the product, setup the machine, perform the operation, and modify parts. However, time to modify parts is not included in disassembly operations, as disassembly processes do not necessitate part modification.

Let

$$
\begin{aligned}
TT &= \text{total time required for assembling and disassembling one product.} \\
&= \text{(assembly time)} + \text{(disassembly time)} \\
&= \sum_{i=1}^{N} [TF{,}A_{Ei} + TS{,}A_{Ai,Bi,Ci,Di} + (NJ_i \cdot TJ{,}A_{Ai,Bi,Ci,Di}) + (NJ_i \cdot TM_{Ai,Bi,Ci})] \\
&+ \sum_{i=N}^{N+M} [TF{,}D_{Ei} + TS{,}D_{Ai,Bi,Ci,Di} + (NJ_i \cdot TJ{,}D_{Ai,Bi,Ci,Di})]
\end{aligned}
$$

where

$TF{,}A_l$ = time to fixture/unfixture a part on type *l* fixture for assembly, seconds.

$TF{,}D_l$ = time to fixture/unfixture a part on a type *l* fixture for disassembly, seconds.

$TS{,}A_{jkvm}$ = assembly setup time for joints of class *j*, type *k*, dimension *v*, using a type ***m*** resource, seconds.

$TS{,}D_{jkvm}$ = disassembly setup time for joints of class *j*, type *k*, dimension *v*, using a type ***m*** resource, seconds.

$TJ{,}A_{jkvm}$ = time to insert and to fasten a joint of class *j*, type *k*, dimension *v*, using a type ***m*** resource, seconds.

$TJ{,}D_{jkvm}$ = time to unfasten a joint of class *j*, type *k*, dimension *v*, using a type ***m*** resource, seconds.

TM_{jkv} = time to modify parts for utilizing a unit of joint of class *j*, type *k*, and dimension *v*, seconds (note: TM_{jkv} = thickness of parts $\times$ modifying time. In general, modifying time is set as 30 seconds per inch).

Total Cost: The cost associated with each assembly/disassembly operation will depend upon the required time in addition to the hourly labor rate, the hourly rate for utilizing machines, the hourly rate for using certain types of fixtures, and the cost of any required fasteners themselves (assembly operation only). We also assume that the costs of redesigning parts will not be included in the model. Thus, we can define

TC = total cost associated with assembling and disassembling one product.

= (assembly cost) + (disassembly cost)

$$= \sum_{i=1}^{N} \{\{[TF,A_{Ei} + TS,A_{Ai,Bi,Ci,Di} + (NJ_i \cdot TJ,A_{Ai,Bi,Ci,Di})] \cdot CL,A_{Bi}\}$$

$$+ (NJ_i \cdot CJ_{Ai,Bi,Ci})$$

$$+ (NJ_i \cdot TM_{Ai,Bi,Ci} \cdot CM,L_{Bi})$$

$$+ \{[TS,A_{Ai,Bi,Ci,Di} + (NJ_i \cdot TJ,A_{Ai,Bi,Ci,Di})] \cdot CT,A_{Di}\}$$

$$+ \{[TF,A_{Ei} + (NJ_i \cdot TJ,A_{Ai,Bi,Ci,Di})] \cdot CF,A_{Ei}\}\}$$

$$+ \sum_{i=N}^{N+M} \{\{[TF,D_{Ei} + TS,D_{Ai,Bi,Ci,Di} + (NJ_i \cdot TJ,D_{Ai,Bi,Ci,Di})] \cdot CL,D_{Bi}\}$$

$$+ \{[TS,D_{Ai,Bi,Ci,Di} + (NJ_i \cdot TJ,D_{Ai,Bi,Ci,Di})] \cdot CT,D_{Di}\}$$

$$+ \{[TF,D_{Ei} + (NJ_i \cdot TJ,D_{Ai,Bi,Ci,Di})] \cdot CF,D_{Ei}\}\}$$

where

CL,A_k	=	labor cost for assembling a joint of type k, \$/hour.
CL,D_k	=	labor cost for disassembling a joint of type k, \$/hour.
CJ_{jkv}	=	cost of a single fastener of joint class j, type k, and dimension v, cents.
CM,L_k	=	labor cost for modifying a part for utilizing joint type k (note: set at \$15/hour.)
CT,A_m	=	tooling cost for using a type m resource for assembly, \$/hour.
CT,D_m	=	tooling cost for using a type m resource for disassembly, \$/hour.
CF,A_l	=	cost of using a type l fixture for assembly, \$/hour.
CF,D_l	=	cost of using a type l fixture for disassembly, \$/hour.

Penalty Score: The penalty score will depend upon the joint types used for the joining operations. It is the summation of penalty scores from the individual joining operations. Because a product may have different designs, the number of joining operations may depend upon the design. In fact, fewer joining operations may possibly result in high penalty scores because each joint type has a high penalty score. On the other hand, if the design has more joining operations but with low penalty scores for each joint type, a high total penalty score may still result. Therefore, the penalty score is the summation of all the joining operations' penalty scores and reflects the overall recyclability of a product. We can thus define

$$PS = \text{penalty score associated with one product.}$$
$$= \sum_{i=1}^{N} (DJ_{Ai,Bi} - DP_i)$$

where

DJ_{jk}	=	expected damage degree index of using joint class *j*, type *k*. This index indicates the relative amount of damage which is expected to occur to the parts due to disassembly, and is a function of the joint type only. The index ranges from 1 (the lowest score) to 10 (the highest score).
DP_i	=	maximum sustainable damage degree index. This index indicates the maximum amount of damage which parts can sustain, as a result of disassembly, and still be reused. It is a function of not only the joint type, but also of the parts themselves and how they are to be reused. The damage degree index ranges from 1 (the lowest score) to 10 (the highest score).

Note that each of the above indices is relative, not absolute, and is somewhat subjective in nature. Also note that $DP_i > DJ_{Ai,Bi} \Rightarrow PS = 0$ (If the maximum sustainable damage degree index of a product is greater than the expected damage degree index of a joint, it means that the joining process is better than we required. Therefore, the penalty score will be set as zero).

3.3 Enumeration Algorithm

A five-stage enumeration algorithm has been developed, based upon the above modeling framework, for generating all possible solutions to the design problem. Each stage involves several steps. The algorithm is described as follows:

Stage (1) Select the required joint type from both product data and joint data. The detail procedures are as follows:

- Step 1: The joint type must provide a larger force than the required joining strength. Therefore, the number of joining components is calculated first. This value is obtained by dividing the required strength by the joint strength. The number of joining components is rounded up to the closest integer when discrete type of joints are selected.
- Step 2: The flexibility of the joining operation is identified. Bolts, screws, or rivets are the flexible joints in the database. For a joint type to be flexible, not more than one component for the joining operation can be used. Therefore, the appropriate dimensions of the joining component are identified in the step to meet the required strength.
- Step 3: In this step, the area available for the selected joining operation is checked. It may be possible that the available area is insufficient to perform the joining operation selected in steps 1 and 2. If the area is not sufficient, the next joint type which can be performed on the available area is selected.
- Step 4: When discrete joining components (bolts, rivets, etc.) are used, it is necessary that their length be more than the total thickness of the parts. This requirement is checked in step 4. However, this is not applicable to continuous joint types such as welding and bonding.

Stage (2) Select the resource for the assembly processes from product data and joint type selected from stage (1).

- Step 1: The resource is selected such that it meets the joining accuracy described in the design. Three indexes are provided in the modeling framework to describe the joining accuracy. Refer to Section 3.1 for details on the index of accuracy.
- Step 2: Force is necessary to perform most of the joining operations. This force can be in terms of compressive load, tensile load, torque, and tracing motion. It is necessary that the resource selected can provide this force. In this step the resource which can provide the necessary force is selected.

Stage (3) Select the required fixture for the assembly process from both the product data and joint type selected from stage (1).

- Step 1: The holding shape of a fixture needs to be the same as the shape of the product.
- Step 2: The clamping range of a fixture needs to be greater than the size of the product.
- Step 3: Since the holding direction is set either as external or internal types, the holding direction for clamping parts is investigated.
- Step 4: Accuracy is another consideration when selecting the fixture. The fixture which provides suitable accuracy for the product is selected.
- Step 5: A fixture is used to secure parts in position, and its purpose is to resist the applied force. Therefore, the clamping force of a fixture should be greater than the force used in a joining operation.

Stage (4) Select the required resource for disassembly from the product data and joint type selected from Stage (1). The procedure is similar to Stage (2), but the disassembly process is not always the reverse of assembly. Therefore, the resource for disassembly may be different form that for assembly. For example, the joining accuracy required for assembling a product is not required when a product is disassembled. Further, different tools are required for disassembly as the disassembly process may be different from the assembly process. Therefore, the following step is different from Stage (2).

- Step 1: Force is necessary to perform most of the disassembly operations. This force can be in terms of compress load, tensile load, torque, and tracing motion. It is necessary that the resource selected can provide this force. In this step the resource which can provide the necessary force is selected.

Stage (5) Select the required fixture for disassembly from product and selected joint type. The procedure is similar to stage (3). However, the disassembly process does not require accuracy. Therefore, a different type of fixture is selected for disassembly.

- Step 1. The holding shape is identified.
- Step 2. The available clamping range on a product is investigated.
- Step 3. Since the holding direction is set either as external or internal types, the holding direction for clamping parts is investigated.
- Step 4. A fixture is used to clamp parts in position to withstand the force necessary to disassemble. Therefore, the clamping force of a fixture should be greater than the force applied to disassemble.

Stage (6) Calculate TT, TC, and PS for the solution based on Stage (1) to Stage (5).

Stage (7) Update the solutions for TT and TC as required.

Repeat Stage (1) to Stage (7) until all possible solutions are evaluated.

3.4 Software Implementation

A computer program has been developed for implementing the above algorithm. A flow diagram of the program is shown in Figure 3.1. The program listing itself is presented as Appendix A. Separate data files are utilized for product, resource, and fixture data. The program first reads these files and stores them in a database. The input file is coded using dynamic memory allocation. Since there could be hundreds of joining processes, and each has thirteen fields, there may be shortages of computer memory if a static format is used. Each joining process is an individual data set, and each one will be read into the computer memory to search all the possible joint types in the database. The program will then calculate the quantity of joints possible for each joining operation. From the input data and the corresponding joint types, the program will select all the possible resources and fixtures for the assembly and disassembly processes. Finally, the minimum time and cost for CAD^2 will be calculated. After all the calculations have been performed, the occupied memory will be freed and reused by the next set of input data.

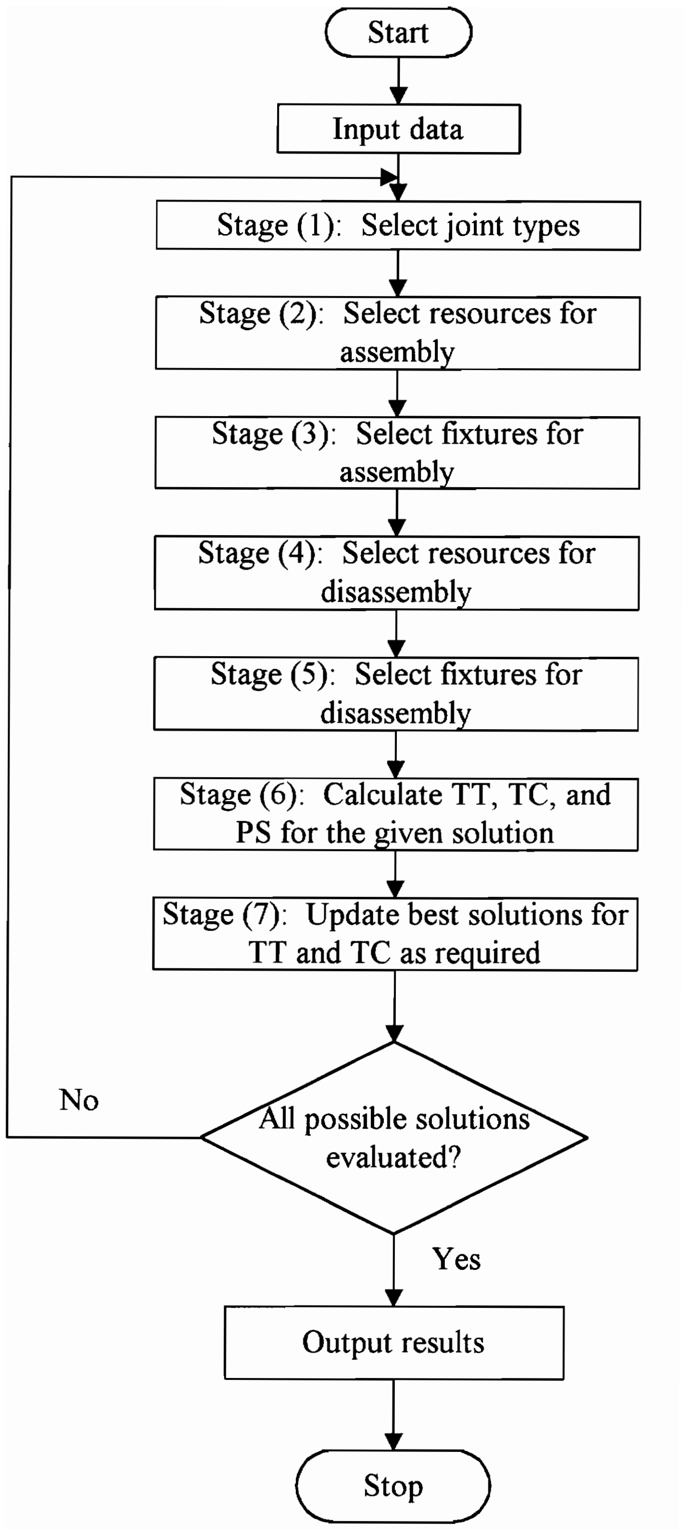
Start
Input data
Stage (1): Select joint types
Stage (2): Select resources for assembly
Stage (3): Select fixtures for assembly
Stage (4): Select resources for disassembly
Stage (5): Select fixtures for disassembly
Stage (6): Calculate TT, TC, and PS for the given solution
Stage (7): Update best solutions for TT and TC as required
All possible solutions evaluated?
No
Yes
Output results
Stop

Figure 3.1 Flow Diagram of CAD^2 Program.

CHAPTER 4
DEMONSTRATION AND EVALUATION OF CAD^2 APPROACH

The demonstration and evaluation of the CAD^2 approach will be performed by examining four products and comparing the results to those obtained using DFA and DFD. The purpose of evaluation is to show that the CAD^2 approach can give better results in terms of Total Time (TT), Total Cost (TC), and Penalty Scores (PS), and thus may be a better method for designing products.

Section 4.1 describes the evaluation methodology used for each product. The same joint, resource, and fixture data will be used for each of the four products examined: this data is presented in Section 4.2. The four products themselves, and the results of the evaluation for each, are presented in Section 4.3. Finally, Section 4.4 discusses the results.

4.1 Evaluation Methodology

Figure 4.1 shows the procedure used to compare the CAD^2 method with DFA and DFD to obtain Total Time (TT), Total Cost (TC), and Penalty Scores (PS). In order to compare DFA with CAD^2, we must somehow determine the disassembly processes to be used, as DFA considers assembly processes only. Additionally, for a given DFA solution of minimum time or cost, several disassembly solutions may be possible. Thus, we will consider both "best case" and "worst case" disassembly solutions to find best case and worst case results when DFA is applied. Moreover, if a given DFA solution of minimum time or cost is not unique (several solutions give the same minimum time or cost), the "best case" or "worst case" disassembly solution may effect a given DFA solution and select a different DFA solution for a best case or worst case result. In order to compare DFD with CAD^2, we must obtain a DFD solution first. Then, based upon this DFD solution, both "best case" and "worst case" assembly solutions will be evaluated and added the DFD solution in order to find best case and worst case results for the DFD approach. In a manner similar to the DFA comparison, if a given DFD solution of minimum time or cost is not unique, the "best case" or "worst case" assembly solution may effect a given DFD solution and select a different DFD solution for a best case or worst case result.

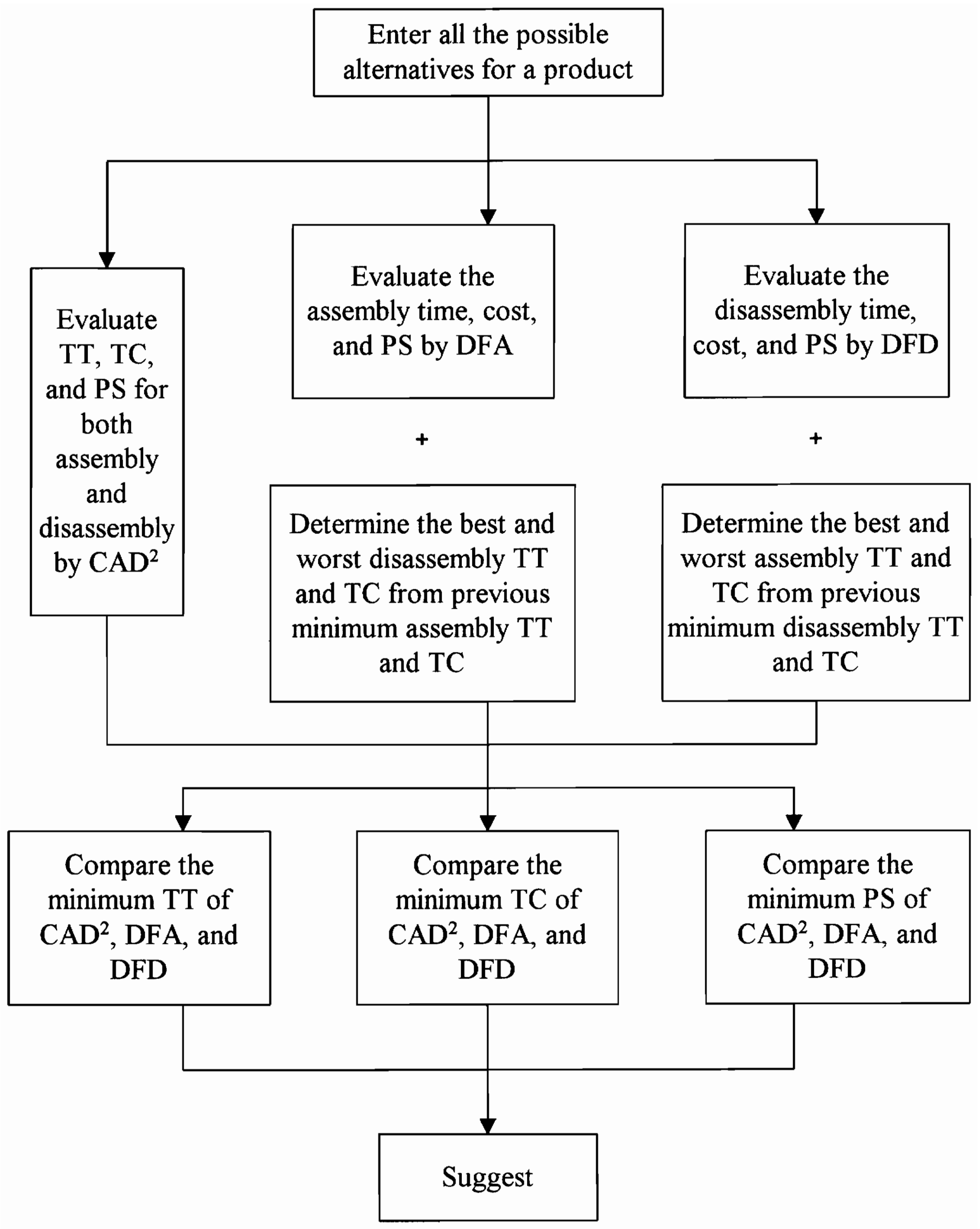
Enter all the possible alternatives for a product
Evaluate TT, TC, and PS for both assembly and disassembly by CAD²
Evaluate the assembly time, cost, and PS by DFA
Evaluate the disassembly time, cost, and PS by DFD
+
+
Determine the best and worst disassembly TT and TC from previous minimum assembly TT and TC
Determine the best and worst assembly TT and TC from previous minimum disassembly TT and TC
Compare the minimum TT of CAD², DFA, and DFD
Compare the minimum TC of CAD², DFA, and DFD
Compare the minimum PS of CAD², DFA, and DFD
Suggest

Figure 4.1 Procedure used to compare CAD^2 with DFA and DFD.

4.2 Design Data

The design data (joint data, resource data, and fixture data) used for each the four products is presented below. The data is based upon items found in the *Thomas Register of American Manufacturers* [Thomas, 1996] and the *Standard Handbook of Fastening and Joining* [Parmley, 1989]. For determining joining forces, it is assumed that all parts to be joined are mild steel. Cost data is based upon values of related products found at local vendors.

4.2.1 Joint Data

Since the purpose of this research is to demonstrate the CAD^2 approach, it includes forty-nine sets of joint data which accommodate most of the common joint types for assembly and disassembly processes. Table 4.2.1 shows the data defining the different joints considered. The first column numbers the sets. The second column denotes the joint class as follows — 1: thread joint, 2: rivet, 3: snap, 4: spot weld, 5: continuous weld, and 6: bonding. The third column names the joint types, which are subgroups of the joint classes. For example, the thread class has two types — 1: screw and 2: bolt. Searching speed is increased by specifying both joint class and joint type. One can refer to Section 3.1 for a description of the remaining columns.

4.2.2 Resources Data

Two types of resources are used in this research; human operators and machines. Human operators perform different types of jobs but with low strength, low accuracy, and longer processing times. This type of data is illustrated in the first six sets of data in Table 4.2.2. Machines can have different types of functions to indicate their performance. These functions are applications of compressive loads, tensile loads, torque, and tracing motion. Machines can perform jobs with high accuracy, high joining strength, and shorter processing times. However, the setup time will be longer and the machine overhead cost is high. Table 4.2.2 shows the resource data (the corresponding descriptions of the variables used are in Section 3.1).

4.2.3 Fixture Data

Nine sets of fixture data are used. This data, based upon Grant [1967], is shown in Table 4.2.3. The first fixture has medium accuracy, low clamping strength, round shape clamping ability, and low clamping range. The second fixture has high accuracy, high clamping strength, flat shape clamping ability and medium clamping range. The remaining fixture sets can be interpreted using the variable definitions given in Section 3.1.

Table 4.2.1 Joint Data.

n	j	k	b_{jk}	s_{jk}	u_{jk}	X_{jkv}	Y_{jkv}	f_{jk}	g_{jk}	c_{jkv}	t_{jkv}	q_{jkv}	r_{jkv}	C_{jkv}	T_{jkv}	Q_{jkv}	R_{jkv}	x_{jk}	y_{jk}	z_{jk}	D_{jk}	CJ_{jkv}
1	1	1	0	75	1	0.3	0.3	0	1	10	0	50	0	5	0	50	0	0.1875	0.1875	0.5	1	4
2	1	1	0	75	1	0.3	0.3	0	1	10	0	50	0	5	0	50	0	0.1875	0.1875	1	1	5
3	1	1	0	75	1	0.3	0.3	0	1	10	0	50	0	5	0	50	0	0.1875	0.1875	2	1	7
4	1	1	0	160	1	0.5	0.5	0	1	12	0	55	0	5	0	55	0	0.25	0.25	0.5	1	6
5	1	1	0	160	1	0.5	0.5	0	1	12	0	55	0	5	0	55	0	0.25	0.25	1	1	8
6	1	1	0	160	1	0.5	0.5	0	1	12	0	55	0	5	0	55	0	0.25	0.25	2	1	10
7	1	2	0	85	1	0.3	0.3	0	1	10	0	50	0	5	0	50	0	0.1875	0.1875	0.5	1	5
8	1	2	0	85	1	0.3	0.3	0	1	10	0	50	0	5	0	50	0	0.1875	0.1875	1	1	6
9	1	2	0	85	1	0.3	0.3	0	1	10	0	50	0	5	0	50	0	0.1875	0.1875	2	1	8
10	1	2	0	185	1	0.5	0.5	0	1	12	0	55	0	5	0	55	0	0.25	0.25	0.5	1	7
11	1	2	0	185	1	0.5	0.5	0	1	12	0	55	0	5	0	55	0	0.25	0.25	1	1	10
12	1	2	0	185	1	0.5	0.5	0	1	12	0	55	0	5	0	55	0	0.25	0.25	2	1	15
13	1	2	0	350	1	0.55	0.55	1	1	14	0	60	0	7	0	60	0	0.375	0.375	0.5	1	9
14	1	2	0	350	1	0.55	0.55	1	1	14	0	60	0	7	0	60	0	0.375	0.375	2	1	25
15	1	2	0	350	1	0.55	0.55	1	1	14	0	60	0	7	0	60	0	0.375	0.375	4	1	45
16	2	1	0	60	1	0.3	0.3	1	1	0	30	0	0	30	0	220	0	0.1875	0.1875	0.5	4	7
17	2	1	0	60	1	0.3	0.3	1	1	0	30	0	0	30	0	220	0	0.1875	0.1875	0.75	4	9
18	2	1	0	60	1	0.3	0.3	1	1	0	30	0	0	30	0	220	0	0.1875	0.1875	1	4	11
19	2	1	0	120	1	0.5	0.5	1	1	0	60	0	0	50	0	245	0	0.25	0.25	0.5	4	8
20	2	1	0	120	1	0.5	0.5	1	1	0	60	0	0	50	0	245	0	0.25	0.25	0.75	4	11
21	2	1	0	120	1	0.5	0.5	1	1	0	60	0	0	50	0	245	0	0.25	0.25	1	4	12
22	2	2	0	70	1	0.25	0.25	1	1	90	0	0	0	30	0	220	0	0.1875	0.1875	0.5	4	5
23	2	2	0	70	1	0.25	0.25	1	1	90	0	0	0	30	0	220	0	0.1875	0.1875	0.75	4	7
24	2	2	0	70	1	0.25	0.25	1	1	90	0	0	0	30	0	220	0	0.1875	0.1875	1	4	8
25	2	2	0	140	1	0.375	0.375	1	1	160	0	0	0	50	0	245	0	0.25	0.25	0.5	4	6

Table 4.2.1 Joint Data (continued).

n	j	k	b_{jk}	s_{jk}	u_{jk}	X_{jkv}	Y_{jkv}	f_{jk}	g_{jk}	c_{jkv}	t_{jkv}	q_{jkv}	r_{jkv}	C_{jkv}	T_{jkv}	Q_{jkv}	R_{jkv}	x_{jk}	y_{jk}	z_{jk}	D_{jk}	CJ_{jkv}
26	2	2	0	140	1	0.375	0.375	1	1	160	0	0	0	50	0	245	0	0.25	0.25	0.75	4	8
27	2	2	0	140	1	0.375	0.375	1	1	160	0	0	0	50	0	245	0	0.25	0.25	1	4	10
28	2	2	0	160	1	0.5	0.5	1	1	180	0	0	0	80	0	275	0	0.375	0.375	0.5	4	8
29	2	2	0	160	1	0.5	0.5	1	1	180	0	0	0	80	0	275	0	0.375	0.375	0.75	4	10
30	2	2	0	160	1	0.5	0.5	1	1	180	0	0	0	80	0	275	0	0.375	0.375	1	4	11
31	3	1	0	30	1	0.5	0.5	0	1	5	0	0	0	0	10	0	0	0.125	0.125	0.3	1	9
32	3	1	0	30	1	0.5	0.5	0	1	5	0	0	0	0	10	0	0	0.125	0.125	0.5	1	10
33	3	1	0	30	1	0.5	0.5	0	1	5	0	0	0	0	10	0	0	0.125	0.125	0.8	1	11
34	3	1	0	40	1	0.6	0.6	0	1	6	0	0	0	0	10	0	0	0.1875	0.1875	0.3	1	9
35	3	1	0	40	1	0.6	0.6	0	1	6	0	0	0	0	10	0	0	0.1875	0.1875	0.5	1	10
36	3	1	0	40	1	0.6	0.6	0	1	6	0	0	0	0	10	0	0	0.1875	0.1875	0.8	1	11
37	3	1	0	60	1	0.7	0.7	0	1	8	0	0	0	0	10	0	0	0.25	0.25	0.3	1	9
38	3	1	0	60	1	0.7	0.7	0	1	8	0	0	0	0	10	0	0	0.25	0.25	0.5	1	10
39	3	1	0	60	1	0.7	0.7	0	1	8	0	0	0	0	10	0	0	0.25	0.25	0.8	1	12
40	4	1	0	80	1	0.25	0.25	0	0	30	0	0	0	0	120	0	0	0.125	0.125	0	8	7
41	4	1	0	120	1	0.35	0.35	0	0	30	0	0	0	0	140	0	0	0.1875	0.1875	0	8	7
42	4	1	0	160	1	0.5	0.5	0	0	30	0	0	0	0	180	0	0	0.25	0.25	0	8	8
43	5	1	1	300	2	0.2	0.2	0	0	0	0	0	1	0	0	100	1	0.2	0.2	0	10	25
44	5	1	1	400	2	0.375	0.375	0	0	0	0	0	1	0	0	100	1	0.375	0.375	0	10	25
45	5	2	1	200	2	0.2	0.2	0	0	0	0	0	1	0	0	100	1	0.2	0.2	0	10	15
46	5	2	1	275	2	0.25	0.25	0	0	0	0	0	1	0	0	100	1	0.25	0.25	0	10	15
47	6	1	1	80	2	0.55	0.55	0	0	2	0	0	1	0	120	0	0	0.55	0.55	0	9	5
48	6	2	1	60	2	0.55	0.55	0	0	2	0	0	1	0	140	0	0	0.55	0.55	0	9	4
49	6	3	1	50	2	0.55	0.55	0	0	2	0	0	1	0	100	0	0	0.55	0.55	0	9	3

Table 4.2.2 Resource Data.

m	A_m	Fc_m	Ft_m	FQ_m	FP_m	TS_m	TP_m	CI_m	CR_m
1	1	50	0	0	0	10	5	15	10
2	1	0	65	0	0	10	5	15	10
3	1	0	0	65	0	20	5	18	15
4	1	0	0	0	1	45	70	15	10
5	1	5	0	0	1	20	5	15	10
6	1	30	0	70	0	25	10	15	10
7	2	200	0	0	0	55	4	18	25
8	2	0	175	0	0	40	10	18	20
9	2	0	0	225	0	65	15	18	15
10	2	0	0	0	1	60	45	18	15
11	2	5	0	0	1	80	25	15	20
12	2	200	0	225	0	70	8	18	40
13	2	400	0	0	0	85	4	18	20
14	2	0	330	0	0	55	7	18	45
15	2	0	0	450	0	80	9	18	25
16	2	0	0	0	1	105	40	18	45
17	2	5	0	0	1	90	20	18	18
18	2	400	0	450	0	90	10	18	60
19	3	200	0	0	0	85	5	20	22
20	3	0	175	0	0	55	10	20	25

Table 4.2.2 Resource Data (continued).

m	A_m	Fc_m	Ft_m	FQ_m	FP_m	TS_m	TP_m	CI_m	CR_m
21	3	0	0	225	0	105	14	20	20
22	3	0	0	0	1	120	30	20	60
23	3	5	0	0	1	95	25	20	20
24	3	200	0	225	0	100	15	20	45
25	3	400	0	0	0	95	14	20	25
26	3	0	300	0	0	75	10	20	60
27	3	0	0	450	0	95	13	20	35
28	3	0	0	0	1	135	25	20	70
29	3	5	0	0	1	100	20	20	25
30	3	400	0	450	0	120	20	20	65
31	3	0	0	250	1	110	125	18	65
32	3	0	0	450	1	130	120	18	85

Table 4.2.3 Fixture Data.

l	p_l	h_l	d_l	hs_l	hA_l	TFU_l	CF_l
1	2	100	1	2	1	20	5
2	3	400	1	1	2	25	10
3	1	250	1	1	1	25	8
4	2	500	1	1	2	30	7
5	3	300	1	2	2	20	6
6	1	550	1	1	1	35	2
7	2	400	0	1	2	18	9
8	2	300	0	1	2	22	8
9	2	450	0	3	1	26	10

4.3 Products and Evaluation Results

To demonstrate the proposed research approach, four products are analyzed. These four products are a car seat, a squeegee, a computer chassis, and a water hose reel. It is assumed that the items to be joined in each product are made of mild steel. The purpose of selecting these four products is to demonstrate different joint types and joining processes. Each product will be described, and the analysis results will be presented.

4.3.1 Case #1 - Car Seat

The first example is the frame of a car seat, as illustrated in Figure 4.3.1.1. For evaluating the process, the assembly sequence is assumed. The assembly sequence and description of joining processes are shown in Figure 4.3.1.2. Table 4.3.1.1 shows the product data for the car seat. Refer to Section 3.1 for the variable definitions.

Table 4.3.1.2 presents the solutions obtained using the CAD^2, DFA, and DFD approaches. The CAD^2 results are analyzed first. When the objective is to minimize total time, the results show that the first and second joining operations will utilize screws. Although the time to fasten one screw is longer than to fasten one rivet, the disassembly time of one screw is much faster than drilling out a rivet. Therefore, the results indicate screws are the superior solution among these available joining processes. The third joining operation utilizes bolts. Bolts basically are stronger than screws because they hold the parts from two sides, if the screws and bolts are of the same size and material. In this operation, fewer bolts are required than screws, so bolts are selected. The fourth joining operation utilizes screws. The type of screw is selected from the second group of screws, which have larger diameters and higher strength. Since a much higher strength is required in this joining operation, using stronger screws will reduce the number of screws, which in turn reduces the operation time. In contrast, the first, second, and third operations cannot apply such a rule to select stronger

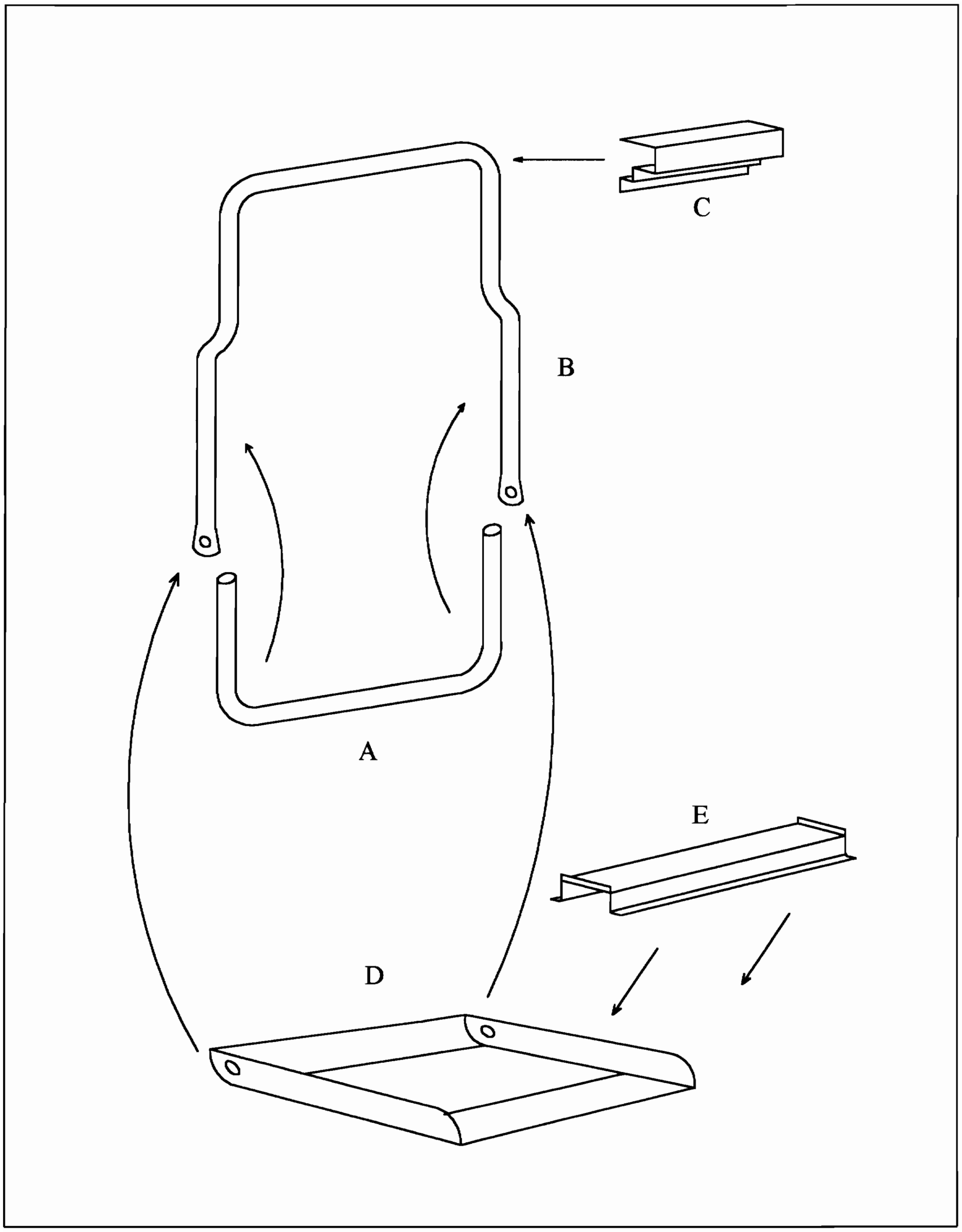

Figure 4.3.1.1 Frame assembly diagram, car seat example.

Operation 1:	Attach part A to part B.
Operation 2:	Attach part A to part B.
Operation 3:	Attach part C to part B.
Operation 4:	Attach part E to part D.
Operation 5:	Fasten the seat (part D and E) to the upright back frame (part A, B, and C).
Operation 6:	Fasten the seat (part D and E) to the upright back frame (part A, B, and C).

Figure 4.3.1.2 Joining process description, car seat example.

Table 4.3.1.1 Product data, car seat example.

n	h_n	f_n	s_n	x_n	y_n	z_n	PS_n	PZ_n	D_n	J_n	ST_n	MD_n
1	1	0	350	0.3	4	2	2	1	2	1	0.3	4
2	1	0	350	0.3	4	2	2	1	2	1	0.3	4
3	1	0	325	0.3	6	1.5	2	1	2	1	0.25	4
4	1	0	400	0.5	36	0.1	1	2	3	1	0.1	4
5	1	1	300	1.5	1.5	0.4	2	1	1	1	0.25	4
6	1	1	300	1.5	1.5	0.4	2	1	1	1	0.25	4

Table 4.3.1.2 Final solutions for each approach, car seat example.

Approach	Objective	Case	Life-cycle phase	Joint Sequence	Resource Sequence	Fixture Sequence
CAD^2	Time	*	Assembly	3-3-9-4-13-13	6-6-6-6-6-6	1-1-1-2-1-1
		*	Disassembly	3-3-9-4-13-13	6-6-6-6-6-6	1-1-1-2-1-1
	Cost	*	Assembly	3-3-9-4-13-13	6-6-6-6-6-6	1-1-1-4-1-1
		*	Disassembly	3-3-9-4-13-13	6-6-6-6-6-6	1-1-1-4-1-1
DFA	Time	Best	Assembly	43-43-43-42-13-13	10-10-10-1-6-6	1-1-1-2-1-1
		Best	Disassembly	43-43-43-42-13-13	31-31-31-14-6-6	1-1-1-2-1-1
		Worst	Assembly	43-43-43-42-13-13	10-10-10-1-6-6	1-1-1-2-1-1
		Worst	Disassembly	43-43-43-42-13-13	32-32-32-26-30-30	1-1-1-4-1-1
	Cost	Best	Assembly	46-46-46-42-13-13	4-4-4-1-6-6	1-1-1-4-1-1
		Best	Disassembly	46-46-46-42-13-13	31-31-31-14-6-6	1-1-1-4-1-1
		Worst	Assembly	46-46-46-42-13-13	4-4-4-1-6-6	1-1-1-4-1-1
		Worst	Disassembly	46-46-46-42-13-13	32-32-32-26-30-30	1-5-5-2-1-4
DFD	Time	Best	Assembly	3-3-9-31-13-13	6-6-6-1-6-6	1-1-1-2-1-1
		Best	Disassembly	3-3-9-31-13-13	6-6-6-2-6-6	1-1-1-2-1-1
		Worst	Assembly	3-3-9-31-13-13	30-30-30-30-30-30	1-1-1-4-1-1
		Worst	Disassembly	3-3-9-31-13-13	6-6-6-2-6-6	1-1-1-2-1-1
	Cost	Best	Assembly	3-3-9-31-13-13	6-6-6-1-6-6	1-1-1-4-1-1
		Best	Disassembly	3-3-9-31-13-13	6-6-6-2-6-6	1-1-1-4-1-1
		Worst	Assembly	9-9-9-31-13-13	30-30-30-30-30-30	5-5-1-2-5-5
		Worst	Disassembly	9-9-9-31-13-13	6-6-6-2-6-6	1-1-1-4-1-1

screws, because the minimum required area of the stronger screw is larger than the joining area available. In addition, the fixture type of the fourth operation, a flat type of fixture, is different from the previously-used fixture. The fifth and sixth operations utilize bolts, since flexible joints are required. In this case, although solid rivets are also an option, a single solid rivet is not strong enough for the given requirements. Therefore, bolts are selected. Finally, the selected resource type is human for each joining operation, because only low strength is needed in each case. The penalty scores are all zero because the parts will retain their original condition after disassembled when screws and bolts are used. When the objective is to minimize total cost, the results are the same as the results of minimizing total time. This is because that CAD^2 evaluates assembly/disassembly simultaneously, and only screws and snaps are suitable for these operations.

Considering the DFA approach and the objective of minimizing total time, the results of the first, second, and third joining operations show that GTAW welding is the best process. To assemble parts by GTAW is faster than to assemble parts by other methods because it can generate high strength on a small area, and a small area means less process time is required. The other reason that GTAW is selected is because it does not require "part modifying time," which can save a large amount of process time. "Part modifying time" is only required for screws, bolts, or rivets, as these require holes to be drilled before assembling. Furthermore, the result of the fourth joining process is spot welding. Since spot welding does not require "part modifying time" and the thickness of parts is within the limit, less than 0.125 inch [Budinski 1983], of spot welding, it becomes the best choice. Lastly, the fifth and sixth joining operations use bolts for the same reason as described in previous paragraph. When the objective is to minimize total cost, the results of the first, second, and third joining operations show that rod welding is the best choice. To assemble parts by rod welding is faster than to assemble parts by other methods (except GTAW) because it can generate high strength on a small area. Although the processing speed of rod welding is longer than GTAW, the cost of rod welding tools is less than GTAW. Hence, rod welding is the best choice for this operation. The result of the fourth joining operation is spot welding. The reason is the same as in the previous case.

The fifth and sixth joining operations use bolts because only bolts can meet the strength requirements.

Considering the DFD approach and the objective of minimizing total time, the results of the first and second joining operations show that screws are the best joint type. To disassemble screws from parts requires less time than other processes (except snap). However, the available area is not large enough to use snaps. The third joining operation utilizes bolts. Since bolts are stronger than screws and since one less bolt is required in this operation, using bolts takes less disassembly time. Therefore, bolts are the best choice for this operation. The fourth joining operation utilizes snaps, as the available area is large enough for this joint type. Although the required quantity of snaps is more than screws, it is much faster to disassemble a snap than to disassemble a screw. The fifth and sixth joining operations utilize bolts. From the two available solutions, bolts and solid rivets, solid rivets necessitate more steps to disassemble. Therefore, the best choice is to use bolts.

When the objective is to minimize total cost, the results of the first and second joining operations show that screws are the best solution for the best case. On the other hand, for the worst case, bolts are selected rather than screws. In these two cases, both require the same quantity of screws or bolts. However, the worst case will force the assembly process to select high cost fasteners, resources, and fixtures. Therefore, bolts are selected when the worst case of DFD is evaluated for the first two joining processes. The third joining operation uses bolts for both the best case and worst case. Since bolts are stronger than screws, fewer bolts are required in the joining operation which lowers the disassembly cost. The third operation is not the same as the first and second, as the quantity of screws is more than that of bolts. Therefore, bolts are the best solution for this operation. The fourth joining operation utilizes snaps, the same result found when the objective was to minimize total cost. The available area is large enough for applying snaps. Although the required quantity of snaps is more than that of screws, the disassembly of a snap is much faster than that of a screw. Therefore, it reduces process time and labor cost. For these reasons, snaps are used in the best case. Nevertheless, more snaps means a higher cost of modifying parts in the assembly process. Therefore, the minimum total cost of using snaps in DFD is about 40% higher than using screws in CAD^2. The fifth and sixth joining operations utilize bolts, for the same reason shown to minimize total time in the DFD method.

Table 4.3.1.3 presents the final objective function values and compare the CAD^2, DFA, and DFD approaches for the car seat example. The best total time of DFA is 26.28 minutes which is higher than 21.73 minutes — the result of CAD^2. The worst case of DFA is 30.28 minutes which is about 42% more than the total time of CAD^2. The best total cost of DFA is \$27.20 which is twice that of the total cost of CAD^2.

The best total time of DFD is 21.98 minutes which is higher than 21.73 minutes — the result of CAD^2. The worst case of DFD is 37.70 minutes which is about 73% more than the total time of CAD^2. The best total cost of DFD is \$12.49 is 11% higher than the total cost of CAD^2, and the worst total cost of DFD is almost four times more than the cost of CAD^2. The penalty scores only appear for DFA, because DFA ignores the disassembly process completely. It indicates that the recyclability is low when the DFA approach is applied. The penalty scores of CAD^2 and DFD are zero which imply that the product will have high degree of recyclability, because the selected joining processes for both of them have lower expected damage degree indices than maximum sustainable damage degree indices. Clearly, CAD^2 is the best approach among the three for this product.

Table 4.3.1.3 Final objective function values, car seat example.

Approach	Objective = min. total time		Objective = min. total cost	
	Total Time (minutes)	Penalty Scores	Total Cost (dollars)	Penalty Scores
CAD^2	21.73	0	11.24	0
DFA (Best)	26.28	29	27.20	29
DFA (Worst)	30.89	29	39.77	29
DFD (Best)	21.98	0	12.49	0
DFD (worst)	37.70	0	43.37	0

4.3.2 Case #2 - Squeegee

The second example is a squeegee, as is illustrated in Figure 4.3.2.1. The assembly sequence and description of joining process are shown in Figure 4.3.2.2. Table 4.3.2.1 shows the product data for the squeegee. For a description of the data, refer to Section 3.1.

Table 4.3.2.2 presents the solutions obtained using the CAD^2, DFA, and DFD approaches. The CAD^2 results are analyzed first. When the objective is to minimize total time, the CAD^2 approach uses snaps for joining operation. This is because that low strength is required in this joining operation and snaps are easy to assemble and disassemble. When the objective is to minimize total cost, the joining operation utilizes screws. Since a screw can generate more strength than a snap, fewer screws are required compared to the number of snaps in this joining operation. Therefore, screws are utilized.

Considering the DFA approach and the objective of minimizing total time, the result of the joining operation shows that riveting is the best joining operation. Assembly of parts using rivets is faster than assembling parts using screws. Compared to snaps, fewer rivets are required in this joining operation and this saves assembly time and the time to modify the parts. When the objective is to minimize total cost, the result of the joining operation shows that screws are the best choice, because they are cheaper.

Considering the DFD approach and the objective of minimizing total time, the results of the joining operation show that snaps are the best since they are easily disassembled. When the objective is to minimize total cost, the results of the joining operation show that snaps are the best for the same reason stated above.

Table 4.3.2.3 presents the final objective function values and compare the CAD^2, DFA, and DFD approaches for the squeegee example. The best total time of DFA is 2.85 minutes which is 82% higher than the result of CAD^2 — 1.57 minutes. The best total cost of DFA is the same as the total cost of CAD^2, but the worst case is almost five times greater than the total cost of CAD^2. The best total time of DFD is 1.57 minutes which is the same as the result of CAD^2. This only means that DFD is as good as CAD^2, and the result of DFD may possibly come from the worst case which can be greater than the result of CAD^2. The best total cost of DFD is \$1.02 which is 21% higher than the total cost of CAD^2, and the worst total cost of DFD is more than five times that of CAD^2. The penalty scores are only shown on DFA when the objective is to minimize the total time, because DFA ignores the disassembly process completely and using rivets. The penalty scores of CAD^2 and DFD are zero, because the selected joining processes for both of them have lower expected damage indices than maximum sustainable damage degree indices. Therefore, clearly, CAD^2 is the best approach among the three for this product.

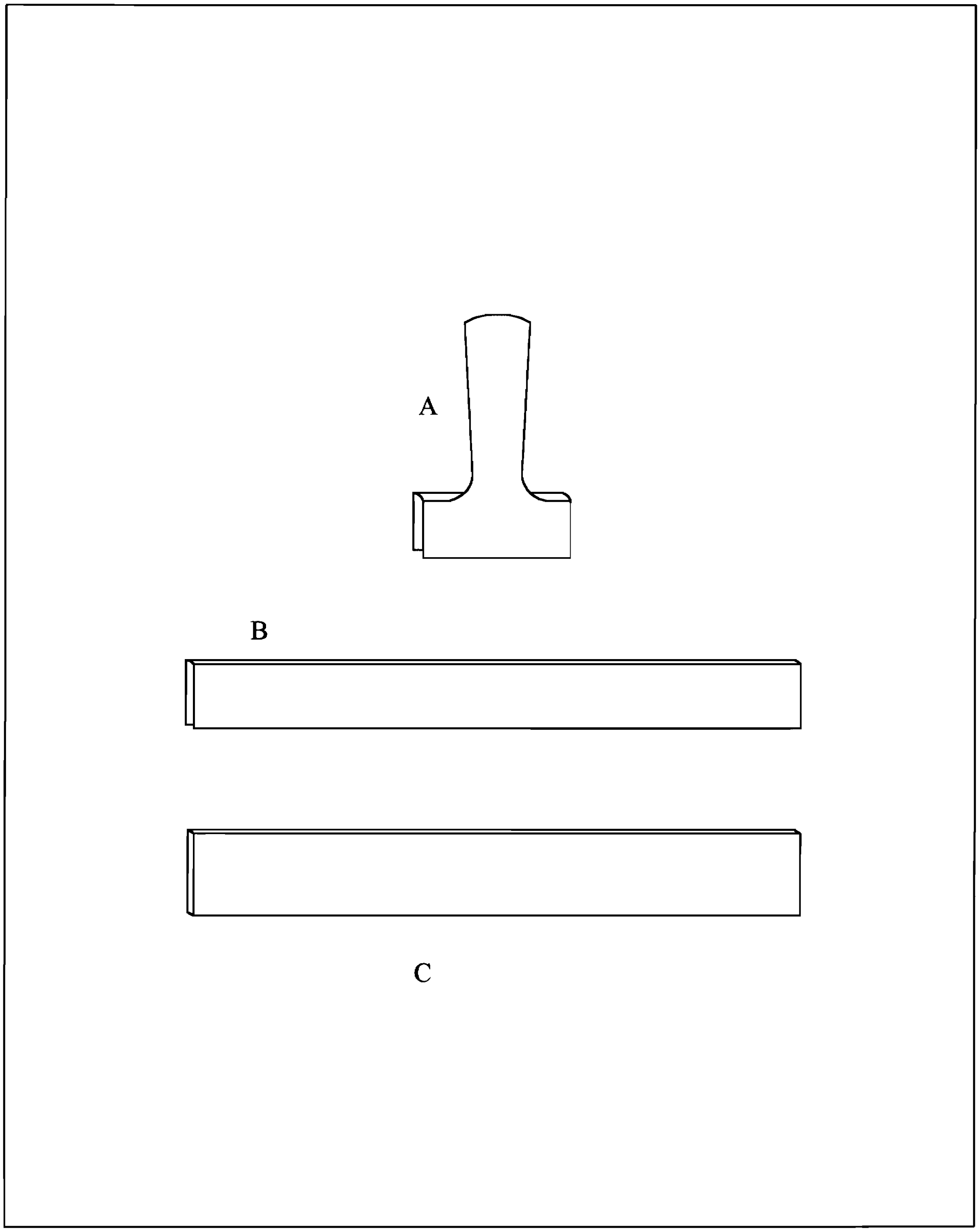

Figure 4.3.2.1 Assembly diagram, squeegee example.

Operation 1: Attach part A, part B, and part C together

Figure 4.3.2.2 Joining process description, squeegee example.

Table 4.3.2.1 Product data, squeegee example.

n	h_n	f_n	s_n	x_n	y_n	z_n	PS_n	PZ_n	D_n	J_n	ST_n	MD_n
1	1	0	150	0.6	12	0.2	1	1	2	1	0.1	4

Table 4.3.2.2 Final solutions for each approach, squeegee example.

Approach	Objective	Case	Life-cycle phase	Joint Sequence	Resource Sequence	Fixture Sequence
CAD^2	Time	*	Assembly	34	1	2
		*	Disassembly	34	2	2
	Cost	*	Assembly	4	6	6
		*	Disassembly	4	6	6
DFA	Time	Best	Assembly	19	2	2
		Best	Disassembly	19	18	2
		Worst	Assembly	19	2	2
		Worst	Disassembly	19	30	5
	Cost	Best	Assembly	4	6	6
		Best	Disassembly	4	6	6
		Worst	Assembly	4	6	6
		Worst	Disassembly	4	30	3
DFD	Time	Best	Assembly	34	1	2
		Best	Disassembly	34	2	2
		Worst	Assembly	34	30	6
		Worst	Disassembly	34	2	2
	Cost	Best	Assembly	34	1	6
		Best	Disassembly	34	2	6
		Worst	Assembly	34	30	1
		Worst	Disassembly	34	2	6

Table 4.3.2.3 Final objective function values, squeegee example.

Approach	Objective = min. total time		Objective = min. total cost	
	Total Time (minutes)	Penalty Scores	Total Cost (dollars)	Penalty Scores
CAD^2	1.57	0	0.84	0
DFA (Best)	2.85	2	0.84	0
DFA (Worst)	3.68	2	3.78	0
DFD (Best)	1.57	0	1.02	0
DFD (worst)	3.87	0	4.43	0

4.3.3 Case #3 - Computer Chassis

The third example is a computer chassis, as illustrated in Figure 4.3.3.1. The sequence and description of joining processes are shown in Figure 4.3.3.2. Table 4.3.3.1 shows the product data for a computer chassis. For a description of the data, refer to Section 3.1.

Table 4.3.3.2 shows the results obtained using the CAD^2, DFA, and DFD approaches. The CAD^2 results are analyzed first. When the objective is to minimize the total time, the results show that all the eleven joining operations need snaps. This is because snaps are easy to assemble and disassemble and low strength of the joints is adequate for computer chassis. However, when the objective is to minimize total cost, the final results show that screws are the best joining type. Since a screw can generate more strength than a snap, fewer screws are required in this joining operation. Consequently, the cost of fasteners will be reduced and the cost of modifying parts can be reduced. Therefore, all of the eleven operations utilize screws when the total cost is the objective.

Considering the DFA approach and the objective of minimizing total time, the results show all eleven operations need spot welding. Assembling parts by spot welding does not require time to modify them, and it is faster than to assemble parts using other types of joining operations. As a result, spot welding is the best choice to assemble the computer chassis when the objective is to minimize the total time and the DFA approach is used. In addition, spot welding is less expensive than other joining operations. Therefore, spot welding is also the solution for all the joining operations when the objective is to minimize the total cost.

Considering the DFD approach and the objective of minimizing total time, the results show that snaps are the best as they are very easy to disassemble. When the objective is to minimize the total cost, snaps are again the best solution. Since snaps are very easy to disassemble, they save large amount of process time and labor cost.

Table 4.3.3.3 presents the final objective function values and compares the CAD^2, DFA, and DFD approaches for the computer chassis example. The best total time of DFA is 21.08 minutes which is 31% higher than the result of CAD^2 — 16.13 minutes. The worst case of DFA is 28.42 minutes which is almost two times longer than the total time of CAD^2. The best total cost of DFA is $12.22 which is 30% more than the total cost of CAD^2, but the worst case is almost three times greater than the TC of CAD^2.

The best total time of DFD is 16.13 minutes which is the same as the result of CAD^2. This only means that DFD is as good as CAD^2 in this case. The best total cost of DFD is $10.10 which is 7% higher than the total cost of CAD^2, and the worst total cost of DFD is more than four times that of CAD^2. The penalty scores are shown on DFA only, as DFA ignores the disassembly process completely and uses spot welding. The penalty scores of CAD^2 and DFD are zero, because the selected joining processes for both of them have lower expected damage degree indices than maximum sustainable damage degree indices. Obviously, this example shows that CAD^2 is the best approach among the three for this product.

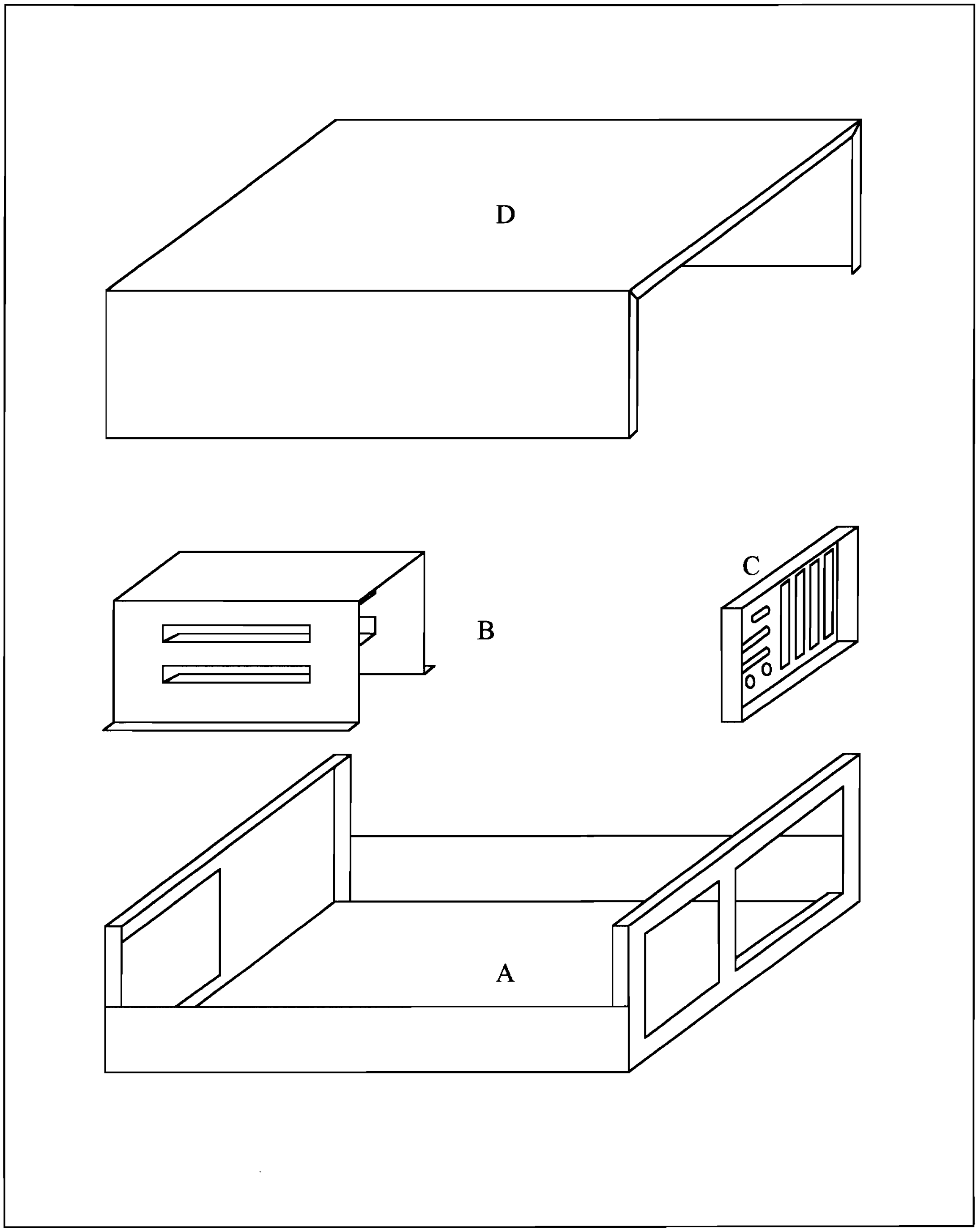

Figure 4.3.3.1 Assembly diagram, computer chassis example.

Operation 1:	Attach first corner of part A
Operation 2:	Attach second corner of part A
Operation 3:	Attach third corner of part A
Operation 4:	Attach fourth corner of part A
Operation 5:	Attach first side of part B to part A
Operation 6:	Attach second side of part B to part A
Operation 7:	Attach upper side of part C to part A
Operation 8:	Attach lower side of part C to part A
Operation 9:	Attach upper side of part D to part A
Operation 10:	Attach left side of part D to Part A
Operation 11:	Attach right side of part D to part A

Figure 4.3.3.2 Joining process description, computer chassis example.

Table 4.3.3.1 Product data, computer chassis example.

n	h_n	f_n	s_n	x_n	y_n	z_n	PS_n	PZ_n	D_n	J_n	ST_n	MD_n
1	1	0	80	0.5	2	0.1	1	2	1	1	0.1	4
2	1	0	80	0.5	2	0.1	1	2	1	1	0.1	4
3	1	0	80	0.5	2	0.1	1	2	1	1	0.1	4
4	1	0	80	0.5	2	0.1	1	2	1	1	0.1	4
5	1	0	75	0.5	8	0.1	1	2	1	1	0.1	4
6	1	0	75	0.5	8	0.1	1	2	1	1	0.1	4
7	1	0	75	0.5	8	0.1	1	2	1	1	0.1	4
8	1	0	75	0.5	8	0.1	1	2	1	1	0.1	4
9	1	0	75	0.5	24	0.1	1	2	1	1	0.1	4
10	1	0	75	0.5	5	0.1	1	2	1	1	0.1	4
11	1	0	75	0.5	5	0.1	1	2	1	1	0.1	4

Table 4.3.3.2 Final solutions for each approach, computer chassis example.

Approach	Objective	Case	Life-cycle phase	Joint Sequence	Resource Sequence	Fixture Sequence
CAD^2	Time	*	Assembly	31-31-31-31-31-31-31-31-31-31-31	1-1-1-1-1-1-1-1-1-1-1	2-2-2-2-2-2-2-2-2-2-2
		*	Disassembly	31-31-31-31-31-31-31-31-31-31-31	2-2-2-2-2-2-2-2-2-2-2	2-2-2-2-2-2-2-2-2-2-2
	Cost	*	Assembly	1-1-1-1-1-1-1-1-1-1-1	6-6-6-6-6-6-6-6-6-6-6	2-2-2-2-2-2-2-2-2-2-2
		*	Disassembly	1-1-1-1-1-1-1-1-1-1-1	6-6-6-6-6-6-6-6-6-6-6	2-2-2-2-2-2-2-2-2-2-2
DFA	Time	Best	Assembly	40-40-40-40-40-40-40-40-40-40-40	1-1-1-1-1-1-1-1-1-1-1	2-2-2-2-2-2-2-2-2-2-2
		Best	Disassembly	40-40-40-40-40-40-40-40-40-40-40	8-8-8-8-8-8-8-8-8-8-8	2-2-2-2-2-2-2-2-2-2-2
		Worst	Assembly	40-40-40-40-40-40-40-40-40-40-40	1-1-1-1-1-1-1-1-1-1-1	2-2-2-2-2-2-2-2-2-2-2
		Worst	Disassembly	40-40-40-40-40-40-40-40-40-40-40	26-26-26-26-26-26-26-26-26-26-26	4-4-4-4-4-4-4-4-4-4-4
	Cost	Best	Assembly	40-40-40-40-40-40-40-40-40-40-40	1-1-1-1-1-1-1-1-1-1-1	2-2-2-2-2-2-2-2-2-2-2
		Best	Disassembly	40-40-40-40-40-40-40-40-40-40-40	8-8-8-8-8-8-8-8-8-8-8	2-2-2-2-2-2-2-2-2-2-2
		Worst	Assembly	40-40-40-40-40-40-40-40-40-40-40	1-1-1-1-1-1-1-1-1-1-1	2-2-2-2-2-2-2-2-2-2-2
		Worst	Disassembly	40-40-40-40-40-40-40-40-40-40-40	26-26-26-26-26-26-26-26-26-26-26	4-4-4-4-4-4-4-4-4-4-4
DFD	Time	Best	Assembly	31-31-31-31-31-31-31-31-31-31-31	1-1-1-1-1-1-1-1-1-1-1	2-2-2-2-2-2-2-2-2-2-2
		Best	Disassembly	31-31-31-31-31-31-31-31-31-31-31	2-2-2-2-2-2-2-2-2-2-2	2-2-2-2-2-2-2-2-2-2-2
		Worst	Assembly	31-31-31-31-31-31-31-31-31-31-31	30-30-30-30-30-30-30-30-30-30-30	4-4-4-4-4-4-4-4-4-4-4
		Worst	Disassembly	31-31-31-31-31-31-31-31-31-31-31	2-2-2-2-2-2-2-2-2-2-2	2-2-2-2-2-2-2-2-2-2-2
	Cost	Best	Assembly	31-31-31-31-31-31-31-31-31-31-31	1-1-1-1-1-1-1-1-1-1-1	2-2-2-2-2-2-2-2-2-2-2
		Best	Disassembly	31-31-31-31-31-31-31-31-31-31-31	2-2-2-2-2-2-2-2-2-2-2	2-2-2-2-2-2-2-2-2-2-2
		Worst	Assembly	31-31-31-31-31-31-31-31-31-31-31	30-30-30-30-30-30-30-30-30-30-30	4-4-4-4-4-4-4-4-4-4-4
		Worst	Disassembly	31-31-31-31-31-31-31-31-31-31-31	2-2-2-2-2-2-2-2-2-2-2	2-2-2-2-2-2-2-2-2-2-2

Table 4.3.3.3 Final objective function values, computer chassis example.

Approach	Objective = min. total time		Objective = min. total cost	
	Total Time (minutes)	Penalty Scores	Total Cost (dollars)	Penalty Scores
CAD^2	16.13	0	9.42	0
DFA (Best)	21.08	7	12.22	7
DFA (Worst)	28.42	7	27.45	7
DFD (Best)	16.13	0	10.10	0
DFD (worst)	39.69	0	45.78	0

4.3.4 Case #4 - Water Hose Reel

The fourth example is a water hose reel, as illustrated in Figure 4.3.4.1. The assembly sequence and the description of joining processes are shown in Figure 4.3.4.2. Table 4.3.4.1 shows the product data for the water hose reel. For a description of data, refer to Section 3.1.

Table 4.3.4.2 presents the solutions obtained using the CAD^2, DFA, and DFD approaches. The CAD^2 results are analyzed first. When the objective is to minimize total time, the result shows that the first joining operation will utilize a bolt. Since a four-inch flexible type of joint is required, a special type of bolt is the only available fastener for this joining operation. The second joining operation utilizes GTAW. Since the required joining strength is very large, the available area is very small, and it is faster than rod welds. The third and fourth joining operations are similar. Screws are the best solution for these two operations because the available joining areas for them are large enough for screws. The fifth and sixth joining operations require joints of high strength in a very small area. Therefore, GTAW is the only suitable joining operation. The seventh and eighth joining operations utilize screws of type II, which have higher strength than the first type. Therefore, fewer screws of the second type are utilized in these two joining operations which can reduce joining time. The ninth and tenth joining processes utilize bolts rather than screws because bolts have higher strength, and fewer bolts are required in these two operations. The penalty scores are assigned to joining operations two, five, and six, because these joining operations utilize GTAW and are assigned high penalty scores. The other joining operations have zero penalty score because these joining operation utilize screws or bolts. The results show that the same joining operations are needed when the objective is to minimize the total time or to minimize the total cost.

Considering the DFA approach and the objective of minimizing total time, the DFA approach utilizes a bolt for the first joining operation. It is the same for the CAD^2 approach, as this operation requires a long and flexible type of joint. The second joining operation utilizes GTAW; the reason is the same as that given for CAD^2. The results of the third and fourth

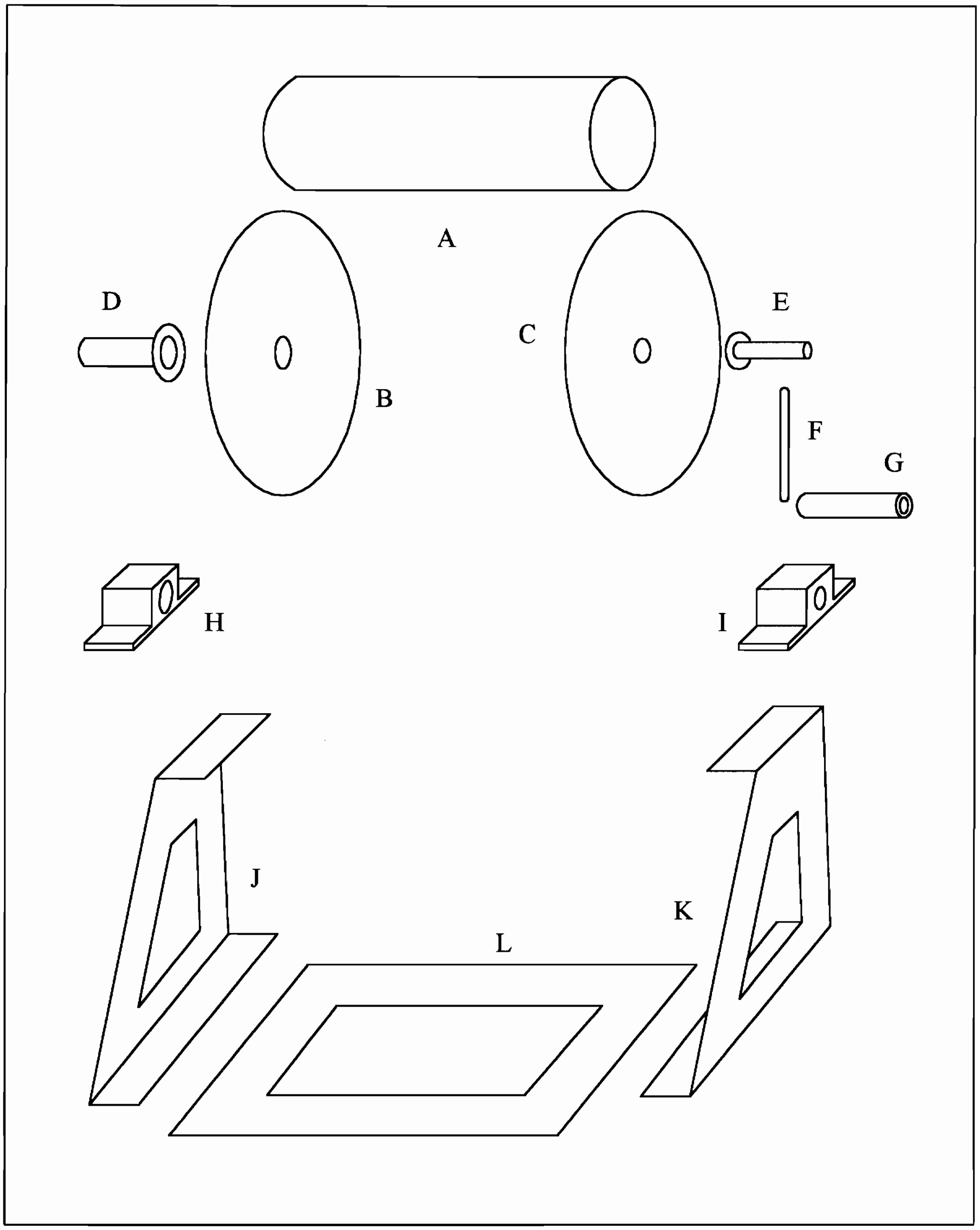

Figure 4.3.4.1 Assembly diagram, water hose reel example.

Operation 1:	Attach part G to part F
Operation 2:	Attach part F to part E
Operation 3:	Attach part E to part C
Operation 4:	Attach part D to part B
Operation 5:	Attach part A to part C
Operation 6:	Attach part A to part B
Operation 7:	Attach part H to part J
Operation 8:	Attach part I to part K
Operation 9:	Attach part K to part L
Operation 10:	Attach part J to part L

Figure 4.3.4.2 Joining process description, water hose reel example.

Table 4.3.4.1 Product data, water hose reel example.

n	h_n	f_n	s_n	x_n	y_n	z_n	Ps_n	Pz_n	D_n	J_n	ST_n	MD_n
1	1	1	350	0.6	0.6	4	2	1	5	1	0.1	3
2	1	0	350	0.6	0.6	4	2	1	5	1	0.1	3
3	1	0	400	0.5	3	0.3	2	1	2	1	0.3	3
4	1	0	400	0.5	4	0.3	2	1	2	1	0.3	3
5	1	0	375	0.2	15	0.2	2	2	2	1	0.2	4
6	1	0	375	0.2	15	0.2	2	2	2	1	0.2	4
7	1	0	300	1.5	2	0.3	1	2	3	1	0.3	3
8	1	0	300	1.5	2	0.3	1	2	3	1	0.3	3
9	1	0	370	1.5	15	0.3	1	2	3	1	0.3	3
10	1	0	370	1.5	15	0.3	1	2	3	1	0.3	3

Table 4.3.4.2 Final solutions for each approach, water hose reel example.

Approach	Objective	Case	Life-cycle phase	Joint Sequence	Resource Sequence	Fixture Sequence
CAD^2	Time	*	Assembly	15-43-4-4-43-43-4-4-10-10	6-10-6-6-10-10-6-6-6-6	1-1-1-1-5-5-2-2-2-2
		*	Disassembly	15-43-4-4-43-43-4-4-10-10	6-31-6-6-31-31-6-6-6-6	1-1-1-1-5-5-2-2-2-2
	Cost	*	Assembly	15-43-4-4-43-43-4-4-10-10	6-4-6-6-4-4-6-6-6-6	1-1-1-1-5-5-2-2-2-2
		*	Disassembly	15-43-4-4-43-43-4-4-10-10	6-31-6-6-31-31-6-6-6-6	1-1-1-1-5-5-2-2-2-2
DFA	Time	Best	Assembly	15-43-19-19-43-43-19-19-13-13	6-10-2-2-10-10-2-2-5-5	1-1-1-1-5-5-2-2-2-2
		Best	Disassembly	15-43-19-19-43-43-19-19-13-13	6-31-18-18-31-31-18-18-8-8	1-1-5-5-5-5-2-2-2-2
		Worst	Assembly	15-43-19-19-43-43-19-19-13-13	30-10-2-2-10-10-2-2-5-5	1-1-1-1-5-5-2-2-2-2
		Worst	Disassembly	15-43-19-19-43-43-19-19-13-13	6-32-30-30-32-32-30-30-26-26	1-1-5-5-5-5-4-4-4-4
	Cost	Best	Assembly	15-46-4-4-43-43-4-4-10-10	6-4-6-6-4-4-6-6-6-6	1-1-1-1-5-5-2-2-2-2
		Best	Disassembly	15-46-4-4-43-43-4-4-10-10	6-31-6-6-31-31-6-6-6-6	1-1-1-1-5-5-2-2-2-2
		Worst	Assembly	15-46-4-4-43-43-4-4-10-10	30-4-30-30-4-4-6-6-6-30	1-1-1-1-5-5-2-2-2-2
		Worst	Disassembly	15-46-4-4-43-43-4-4-10-10	6-32-6-6-32-32-30-30-30-6	5-5-5-5-5-5-2-2-2-2
DFD	Time	Best	Assembly	15-43-4-4-43-43-4-4-37-37	6-10-6-6-10-10-6-6-1-1	1-1-1-1-5-5-2-2-2-2
		Best	Disassembly	15-43-4-4-43-43-4-4-37-37	6-31-6-6-31-31-6-6-2-2	1-1-1-1-5-5-2-2-2-2
		Worst	Assembly	15-43-4-4-43-43-4-4-37-37	30-32-30-30-32-32-30-30-30-30	1-1-1-1-5-5-2-2-4-4
		Worst	Disassembly	15-43-4-4-43-43-4-4-37-37	6-31-6-6-31-31-6-6-2-2	1-1-1-1-5-5-2-2-2-2
	Cost	Best	Assembly	15-43-4-4-43-43-4-4-37-37	6-4-6-6-4-4-6-6-1-1	1-1-1-1-5-5-2-2-2-2
		Best	Disassembly	15-43-4-4-43-43-4-4-37-37	6-31-6-6-31-31-6-6-2-2	1-1-1-1-5-5-2-2-2-2
		Worst	Assembly	15-43-10-10-43-43-10-10-37-37	30-32-30-30-32-32-30-30-30-30	5-5-5-5-5-5-2-2-2-2
		Worst	Disassembly	15-43-10-10-43-43-10-10-37-37	6-31-6-6-31-31-6-6-2-2	1-1-1-1-5-5-2-2-2-2

joining operations utilize rivets. Since rivets can assemble parts faster than screws and small areas are available in these two joining operations, rivets become the best solution. The fifth and sixth joining operations use GTAW because it is the only available joint type for generating high strength in a very small area. The seventh and eighth joining operations utilize rivets, as rivets can assemble parts faster than screws. The ninth and tenth joining operations utilize epoxy. This is because the available areas are large enough for the epoxy to generate required strength. In addition, applying epoxy does not require time to modify parts. Even though the above mentioned joint types can assemble parts faster than their alternatives, their penalty scores are high. This is because they create difficulties in the disassemble. When the objective is to minimize total cost. The results of the first joining operation show that a bolt is utilized because of the special requirement. The second joining operation utilizes rod welding because it is inexpensive than GTAW. The third and fourth joining operations utilize screws. Since screws only need human operators with common power tools, the machine setup time is low when comparing with welding process. On the other hand screws are less expensive than rivets. The fifth and sixth joining operations utilize GTAW, because only small areas are available in these two joining operations. The seventh and eighth joining operations utilize screws. The reason is as the third and fourth joining operations. The ninth and tenth joining operations utilize bolts. Although larger areas are available, utilizing fewer bolts are cheaper than applying large quantity of epoxy. Further fewer bolts are required when compared to screws. Therefore, bolts are the best choice for these operations.

Considering the DFD approach and the objective of minimizing total time, the results of the first eight joining operations remain the same as that of CAD^2. The last two joining operations use snaps, because large available areas allow more snaps to generate the required strength and snaps are easy to disassemble. When the objective is to minimize total cost, the results of the best case are the same as minimizing the total cost as the DFA approach. On the other hand, for the worst case, the third, fourth, seventh, and eighth joining operations utilize bolts. In these joining operations, both require the same quantity of screws or bolts. However, the worst case forces the assembly process to select high cost components, resources and, fixtures. Therefore, bolts are selected for the third, fourth, seventh, and eighth joining operations when the worst case of DFD is evaluated.

Table 4.3.4.3 presents the final objective function vales and compare the CAD^2, DFA, and DFD approaches for the water hose reel example. The best total time of DFA is 43.30 minutes which is higher than 37.87 minutes — the result of CAD^2. The worst case of DFA is 50.91 minutes which is about 34% more than the total time of CAD^2. The best total cost of DFA is $32.80 which is higher than the total cost of CAD^2, and the worst case is almost twice that of the total cost of CAD^2. The best total time of DFD is 38.41 minutes which is higher than 37.87 minutes — the result of CAD^2. The worst case of DFD is 60.90 minutes which is about 61% more than the total time of CAD^2. The best total cost of DFD is $33.85 which is higher than the total cost of CAD^2, and the worst total cost of DFD is three times more than the cost of CAD^2. The penalty scores are assigned to each approach because some joining processes utilize welding which will decrease the recyclability. There is no significant difference in the penalty scores among the three approaches. Again, obviously, CAD^2 is the best approach among the three for this product.

Table 4.3.4.3 Final objective function values, water hose reel example.

Approach	Objective = min. total time		Objective = min. total cost	
	Total Time (minutes)	Penalty Scores	Total Cost (dollars)	Penalty Scores
CAD^2	37.87	21	32.51	21
DFA (Best)	43.30	29	32.80	21
DFA (Worst)	50.91	29	61.98	21
DFD (Best)	38.41	21	33.85	21
DFD (worst)	60.96	21	80.55	21

4.4 Discussion of Results

The CAD^2 approach appears to be an improved method for selecting joint types and joining processes during product design. The summary of final objective function values for four products is on Table 4.4. For each of the four products, CAD^2 resulted in the lowest total time and total cost because it gives the optimal solution based upon the given data. The DFD approach eliminated those joint types that would increase the difficulty of disassembly. However, DFD may underestimate the time and cost of assembly processes by using inappropriate resources or fixtures that can increase the cost and time for assembly/disassembly processes. Therefore, the results of DFD were close to the results of CAD^2 in some cases. On the other hand, DFA gave the worst results for all the cases, because fast and inexpensive assembly processes can create difficulties in disassembly. These difficulties involve long part separation times, damaged parts which must be repaired in order to be reused, and permanently-damaged parts which must be recycled or disposed of. These accumulate to make DFA as the worst approach of the three.

Table 4.4 The summary of final objective function values for four cases.

	CAD^2		DFA		DFD	
	Total time (minutes)	Total cost (dollars)	Total time (minutes)	Total cost (dollars)	Total time (minutes)	Total cost (dollars)
Case 1	21.73	11.24	26.28	27.20	21.98	12.49
Case 2	1.57	0.84	2.85	0.84	1.57	1.02
Case 3	16.13	9.42	21.08	12.22	16.13	10.10
Case 4	37.87	32.51	43.30	32.80	38.41	33.85

The available joining area and the required joint strength are the most important factors when deciding on suitable joint types. For example, for joint types having small area and requiring high strength, welding is desirable. However, welding creates difficulties as it is meant to be permanent. For small joining area and low joint strength, most of the common joint types can be used. Snap fits meet the requirements of CAD^2 most of the time. However, the cost may be slightly higher than that of screws or bolts because snaps have low strength thus more snaps may be required. Snap fits are not designed for close fits: to obtain such fits, small tolerances are required and the manufacturing cost of the snap fits will be very high. Nevertheless, the snap fit is one of the best joint types, according to the CAD^2 approach. In other instances, bonding can be applied to parts when a large area is available. Assembly time may be relatively small, but a relatively large force will be required to disassemble the bonded parts. Also, no part modification is required. Therefore, it might be superior over other joining types when considering assembly only and where large joining areas are available.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

Design for Assembly is the traditional process to reduce the time and cost during product development. Design for Disassembly, on the other hand, focuses on maintenance, recycling, reuse, and disposal. Industries have accepted and successfully used DFA and DFD in some situations. However, focusing only on assembly or disassembly can cause unexpected problems.

A new approach aimed at overcoming the limitations of DFA and DFD has been developed in this research. The approach, called CAD^2 (Concurrent Assembly/Disassembly-Based Design), works by selecting both joint types and joining processes simultaneously, based upon both assembly and disassembly considerations. Designers input product, joint, resource, and fixture data, and the approach yields minimum-time and minimum-cost solutions, as well as the corresponding penalty score in each case. The use of multiple objectives provides flexible guidelines to assist designers in evaluating the time/cost trade-offs which may occur, whereas the penalty scores indicate the adverse environmental effects associated with different solutions. The CAD^2 approach extensively reduces designing time because it integrates the design, process planning, and cost-estimating functions. Because it uses complete enumeration to find the optimal solutions, it is better suited to uncomplicated products.

The CAD^2 approach has two major advantages over previous DFA/DFD approaches. The first is that there is no subjectivity in determining possible solutions (selections of joints, resources, and fixtures), as all data used is quantitative. This is considered to be a major improvement over previous works which rely on subjective indicators of assembly and/or disassembly alternatives [e.g., Dewhurst 1993; VeGow 1994]. The second is that the CAD^2 approach results in optimal (minimum-time or minimum-cost) solutions, whereas previous approaches based upon the use of design rules and factors [e.g., Boothroyd 1987, Andreasen 1988] yield heuristic solutions. Thus, the design problem is shifted from one of choosing which heuristic solution to employ to one of determining the tradeoff between time and cost which is most suitable for a given application.

Several avenues exist for future work. Efficient algorithms can be developed for solving the total time and total cost minimization problems. Environmental objectives, such as disposal costs and costs of recycling different types of materials, may also be incorporated into the model. A decision-support system (DSS) may be built to guide the designer in implementing the design, process planning, and cost-estimating functions performed in the CAD^2 approach. To properly assist the designer in evaluating time/cost tradeoffs, the program should be expanded upon to plot time vs. cost for some subset of solutions close to, and including, the optimal solutions. The designer can then select, based upon the plot, the solution which provides the most desirable time/cost tradeoff. The product structure can be further refined in order to produce more accurate results: this may involve modeling joining angles and a more detailed breakdown of the types of forces required for joining. Finally, the assumptions made in the model (suitability of joint types independent of working environment, single material available for products, and no cost for redesigning parts to utilize different joint types) can all be relaxed and these variables incorporated into the model. Each of these tasks will result in either a more valid model (and hence better solutions), faster solution generation, or easier implementation of the approach. Such improvements would undoubtedly make the CAD^2 approach even more attractive to product designers.

REFERENCES

Andreasen, M. M., Kahler, S., Lund, T., and Swift, K., 1988, *Design for Assembly*, Second edition, New York: Springer-Verlag.

Boothroyd, G., 1982, *Automatic Assembly*, New York: Marcel Dekker, Inc.

Boothroyd, G., and Dewhurst, P., 1987, *Product Design for Assembly*, Wakefield, RI: Boothroyd Dewhurst Inc.

Boothroyd, G. and Alting, L., 1992, "Design for Assembly and Disassembly," *Annals of the CIRP*, Vol. 41, No. 2, pp. 625-636.

Budinski, K., 1983, *Engineering Materials — Properties and Selection*, Reston, VA: Reston Publishing Company.

Chen, R. W., Navin-Chandra, D., and Prinz, F. B., December 1994, "A Cost-Benefit Analysis Model of Product Design for Recyclability and its Application," *IEEE Transaction on Components, Packaging, and manufacturing Technology-Part A*, Vol. 17, No. 4, pp. 502-507.

Dewhurst, P., September 1993, "Product Design for Manufacture: Design for Disassembly," *Industrial Engineering*, pp. 26-28.

Grant, E. H., 1967, *Jig and Fixtures — Non-standard Clamping Devices*, New York: McGraw-Hill Company.

Harjula, T., Rapoza, B., Knight, W. A., and Boothroyd, G., 1996, "Design for Disassembly and the Environment," *Annals of the CIRP*, pp. 109-114.

Johnson, M. R. and Wang, M. H., 1995, "Planning product disassembly for material recovery opportunities," *International Journal of Production Research*, Vol. 33, No. 11, pp. 3119-3142.

Kroll, E., Beardsley, B., Parullan, A., and Berners, D., 1994, "Evaluating Ease-of-Disassembly for Product Recycling," *Material and Design Technology*, PD-Vol. 62, pp. 165-172.

Laperriere, L. and Elmaraghy, H. A., 1992, "Planning of Products assembly and Disassembly," *Annals of the CIRP*, Vol. 41, No. 2, pp. 5-9.

Miyakawa, S., Ohashi, T., and Iwata, M., May 1990, "The Hitachi New Assemblability Evaluation Method (AEM)," *Transactions of NAMRI/SME*, pp. 352- 359.

Parmley, O. R., 1989, *Standard Handbook of Fastening and Joining*, Second edition, New York: McGraw-Hill, Inc.

Penev, K. D. and de Ron, A. J. 1996, "Determination of a disassembly strategy," *International Journal of Production Research*, Vol. 34, No. 2, pp. 495-506.

Remich, N. C. Jr., June 1991, "First Recyclable Appliance," *Appliance Manufacturer*, pp. 57- 60.

Thomas Publishing Company, 1996, *Thomas Register of American Manufacturers*, 86th edition, New York: Thomas Publishing Company.

Valentovic, E., and Chal, J., 1993, "A New Design for Assembly Method," *International Conference on Engineering Design*, The Hague, August 17-19, pp. 1017-1023.

VerGow, Z., and Bras, B., 1994, "Recycling Oriented Fasteners: A Critical Evaluation of VDI 2243's Selection Table," *Advances in Design Automation*, DE-Vol. 69-2, pp. 341-349.

Yamagiwa, Y., December 1988, "An Assembly Ease Evaluation Method for Product Designers: DAC," *Techno Japan*, Vol. 21, No. 12, pp. 26-29.

Yoosufani., Z., Ruddy, M., and Boothroyd, G., 1983, "Effect of Part Symmetry on Manual Assembly Times," *Journal of Manufacturing Systems*, Vol. 2, No. 2, pp. 189-194.

Appendix A

Program Listing

```
/*  Thesis.c
 *
 *  Objective  -  Calculate Total Time(TT), Total Cost(TC), and Penalty Scores(PS)
 *                by searching evaluating the corresponding data from the databases.
 */

#include <stdio.h>
#include <stdlib.h>
#define MAX_SET 49      /* Quantity of joints in database */
#define MAX_RES 32      /* Quantity of resources in database */
#define MAX_FIX 9       /* Quantity of fixtures in database */
#define R 13            /* Quantity of fields in a product file */

void inputjo(void);
void inputres(void);
void inputfix(void);
void flex(void);
void nonflex(void);
void calculation(void);
void save(void);
void print(void);



/* Define fields of joints */
struct joint {
        int class;
        int type;

        int physical_type;
        int strength;
        int unit;
        float min_x;
        float min_y;
        int flex_factor;
        int part_change;
        int as_compress;
        int as_tensile;
        int as_torque;

        int as_tracing;
        int dis_compress;
        int dis_tensile;
        int dis_torque;
        int dis_tracing;

        float joint_size_x;
        float joint_size_y;
        float joint_size_z;
        int damage_index;
        float fastener_cost;

};
struct joint p[MAX_SET];

/* Define fields of resources */
struct resource {
        int res_type;
        int res_accuracy;
        int res_compress;
        int res_tensile;
        int res_torque;

        int res_tracing;
        int setup_time;
        int process_time;
        int labor_cost;
        int machine_over;
};
struct resource r[MAX_RES];

/* Define fields of fixtures */
struct fixture {
        int fix_type;
        int pos_accuracy;
        int holding_force;
        int hold_direction;
        int hold_shape;
        int hold_area_index;
        int fixture_time;
        int fixture_over;
};
struct fixture f[MAX_FIX];

/*  Define fields of products  */
struct node {
        float oper_num;
        struct node *next;
};

int j, k, m, n, o, J1, J2, J3, J4, J5, J6, J7, J8, J9, J10,
   K1, K2, K3, K4, K5, K6, K7, K8, K9, K10, M1, M2, M3,
   M4, M5, M6, M7, M8, M9, M10, N1, N2, N3, N4, N5, N6,
   N7, N8, N9, N10, O1, O2, O3, O4, O5, O6, O7, O8, O9, O10,
   PS, CAD_T_PS, CAD_C_PS, DFA_T_PS, DFA_C_PS, DFD_T_PS, DFD_C_PS,
   CAD_TT_PS, CAD_TC_PS, DFA_TT_PS, DFA_TC_PS, DFD_TT_PS,
DFD_TC_PS;

float TM, CM, as_t, as_c, dis_t, dis_c, TA, TD, CA, CD,
    product[R], num_joint, process_as, process_dis, num_joint_discrete,
    TA_MIN, TA_MAX, TD_MIN, TD_MAX,
    CA_MIN, CA_MAX, CD_MIN, CD_MAX,
    CAD_T, CAD_C, CAD_T_TEM, CAD_C_TEM,
    DFA_T_MIN, DFA_T_MAX, DFA_C_MIN, DFA_C_MAX,
    DFD_T_MIN, DFD_T_MAX, DFD_C_MIN, DFD_C_MAX,
    CAD_TT, CAD_TC,
    DFA_TT_MIN, DFA_TT_MAX, DFA_TC_MIN, DFA_TC_MAX,
    DFD_TT_MIN, DFD_TT_MAX, DFD_TC_MIN, DFD_TC_MAX;

void main(void)
{
        FILE        *fpro;

        int         join_process;
        int        s, t, i ,g, q;

        struct joint *fpjo;
        struct resource *fpres;
        struct fixture *fpfix;
struct node  *list, *lead,
             *follow, *add_node(void);
void         delete (struct node *listptr),
             connect(struct node *old, struct node *new);

CAD_TT = 0.0;
CAD_TC = 0.0;
CAD_TT_PS = 0;
CAD_TC_PS = 0;
DFA_TT_MIN = 0.0;
DFA_TT_MAX = 0.0;
DFA_TC_MIN = 0.0;
DFA_TC_MAX = 0.0;
DFA_TT_PS  = 0;
DFA_TC_PS  = 0;
DFD_TT_MIN = 0.0;
DFD_TT_MAX = 0.0;
DFD_TC_MIN = 0.0;
DFD_TC_MAX = 0.0;
DFD_TT_PS  = 0;
DFD_TC_PS  = 0;


fpjo = (struct joint *)calloc(MAX_SET, sizeof(struct joint));
if(fpjo == NULL)
{
  printf("\nAllocation memory failure.");
  exit(1);
}
inputjo();

fpres = (struct resource *)calloc(MAX_RES, sizeof(struct resource));
if (fpres == NULL)
{
  printf("\nAllocation memory failure.");
  exit(2);
}
inputres();

fpfix = (struct fixture *)calloc(MAX_FIX, sizeof(struct fixture));
if(fpfix == NULL)
{
  printf("\nAllocation memory failure.");
  exit(3);
}
inputfix();


if((fpro = fopen("product2.txt", "r")) == NULL)
{
  printf("Input file could not be opened\n");
  exit(4);
}

fscanf(fpro, "%d", &join_process);
for(s=0 ;s<join_process ;s++)
{
  TA = 9999.9;
  TD = 9999.9;
  CA = 9999.9;
  CD = 9999.9;
  CAD_T_TEM = 9999.9;
  CAD_C_TEM = 9999.9;

  CAD_T = 0.0;
  CAD_C = 0.0;
  CAD_T_PS  = 0;
  CAD_C_PS  = 0;
  DFA_T_MIN = 0.0;
  DFA_T_MAX = 0.0;
  DFA_C_MIN = 0.0;
  DFA_C_MAX = 0.0;
  DFA_T_PS  = 0;
  DFA_C_PS  = 0;
  DFD_T_MIN = 0.0;
  DFD_T_MAX = 0.0;
  DFD_C_MIN = 0.0;
  DFD_C_MAX = 0.0;
  DFD_T_PS  = 0;
  DFD_C_PS  = 0;

  for(t=0;t<R;t++)
  {
  if(t==0)  /* scan first joining operation from product.txt */
     {
         list = add_node();
```

```
        list->next = NULL;
        follow = list;
        fscanf(fpro, "%f", &list->oper_num);
        product[t] = list->oper_num;
        printf("\n%f", list->oper_num);
   }

else
   {
        lead = add_node();
        connect(follow, lead);
        follow = lead;

        fscanf(fpro, "%f", &lead->oper_num);
        product[t] = lead->oper_num;
        printf("\t%f", lead->oper_num);

      }
}

if (product[2] == 1)
      {
       flex();
      }
                          /*  Checking the flex factor */
                          /*  One is flexible joint; zero is not. */
else               /*  Evaluate the none flexible joint.  */
      {
      nonflex();
      }


 /*  Accumulating data  */

 CAD_TT     = CAD_TT     + CAD_T;
 CAD_TT_PS  = CAD_TT_PS  + CAD_T_PS;
 CAD_TC     = CAD_TC     + CAD_C;
 CAD_TC_PS  = CAD_TC_PS  + CAD_C_PS;

 DFA_TT_MIN = DFA_TT_MIN + DFA_T_MIN;
 DFA_TT_MAX = DFA_TT_MAX + DFA_T_MAX;
 DFA_TT_PS  = DFA_TT_PS  + DFA_T_PS;
 DFA_TC_MIN = DFA_TC_MIN + DFA_C_MIN;
```

```
            DFA_TC_MAX = DFA_TC_MAX + DFA_C_MAX;
            DFA_TC_PS  = DFA_TC_PS  + DFA_C_PS;

            DFD_TT_MIN = DFD_TT_MIN + DFD_T_MIN;
            DFD_TT_MAX = DFD_TT_MAX + DFD_T_MAX;
            DFD_TT_PS  = DFD_TT_PS  + DFD_T_PS;
            DFD_TC_MIN = DFD_TC_MIN + DFD_C_MIN;
            DFD_TC_MAX = DFD_TC_MAX + DFD_C_MAX;
            DFD_TC_PS  = DFD_TC_PS  + DFD_C_PS;
            save();


            print();
           delete(list);
        }
        fclose(fpro);
        free(fpjo);
        free(fpres);
        free(fpfix);
}

/*  Input joint data  */
void inputjo(void)
{
  FILE *file;
  int i;

        if((file = fopen("joint3.txt", "r")) == NULL)
        {
                printf("The file does not exist");
        }
        else
        {
          for (i=0; i<MAX_SET; i++)
          {
            fscanf(file, "%d %d %d %d %d %f %f %d %d %d %d %d\n",
            &p[i].class, &p[i].type, &p[i].physical_type, &p[i].strength, &p[i].unit,
            &p[i].min_x, &p[i].min_y, &p[i].flex_factor, &p[i].part_change,
            &p[i].as_compress, &p[i].as_tensile, &p[i].as_torque);

            fscanf(file, "%d %d %d %d %d %f %f %f %d %f\n",
            &p[i].as_tracing, &p[i].dis_compress, &p[i].dis_tensile,
            &p[i].dis_torque, &p[i].dis_tracing, &p[i].joint_size_x,
            &p[i].joint_size_y, &p[i].joint_size_z,
            &p[i].damage_index, &p[i].fastener_cost);
          }
        }
        fclose(file);
}

/*  input resources data  */
void inputres(void)
{
  FILE *fileres;
  int g=0;
        if((fileres = fopen("resint3.txt", "r")) == NULL)
        {
          printf("\nThe file does not exist.");
        }
        else
        {
          for(g=0; g<MAX_RES; g++)
          {
            fscanf(fileres, "%d %d %d %d %d %d %d %d %d %d\n",
            &r[g].res_type, &r[g].res_accuracy, &r[g].res_compress,
            &r[g].res_tensile, &r[g].res_torque, &r[g].res_tracing,
            &r[g].setup_time, &r[g].process_time, &r[g].labor_cost,
            &r[g].machine_over);
          }
        }
        fclose(fileres);
}


/*  input fixture data  */
void inputfix(void)
{
  FILE *filefix;
  int q=0;
        if((filefix = fopen("fixture3.txt", "r")) == NULL)
        {
          printf("\nThe file does not exist.") ;
        }
        else
        {
          for(q=0; q<MAX_FIX; q++)
```

```
            {
              fscanf(filefix, "%d %d %d %d %d %d %d %d\n",
              &f[q].fix_type, &f[q].pos_accuracy, &f[q].holding_force,
              &f[q].hold_direction,
              &f[q].hold_shape, &f[q].hold_area_index,
              &f[q].fixture_time, &f[q].fixture_over);
            }
          }
          fclose(filefix);
}


/*      Allocates the space for a node.     */
struct node *add_node(void)
{
      struct node *temp_ptr;

      temp_ptr = (struct node *) malloc ( sizeof(struct node) );
      if (temp_ptr == NULL) {
                  printf ("out of memory for nodes\n");
                  exit(1);
          }
      return (temp_ptr);
}



/*  Link new node to the last node.  */

void connect (struct node *old_node, struct node *new_node)
{
      new_node->next = NULL;
        old_node->next = new_node;
}

void flex()
{
  j = 0;
  label:
  while( j < MAX_SET )
  {
    if(( product[2] == p[j].flex_factor)
        &&( product[3] <= p[j].strength)
```

```
&&(((product[4]>= p[j].min_x)&&(product[5]>= p[j].min_y))
 ||((product[4]>= p[j].min_y)&&(product[5]>= p[j].min_x)))
 &&(product[6] <= p[j].joint_size_z))
  {
      k = 0;
      label1:
      while( k < MAX_RES)
      {
       if(( product[10] <= r[k].res_accuracy)
         &&( p[j].as_compress <= r[k].res_compress)
           &&( p[j].as_tensile <= r[k].res_tensile)
               &&( p[j].as_torque <= r[k].res_torque))
                {

                m = 0;
                label2:
                while ( m < MAX_RES)
                {
                if((p[j].dis_compress <= r[m].res_compress)
                &&(p[j].dis_tensile <= r[m].res_tensile)
                &&(p[j].dis_torque <= r[m].res_torque))
                {


                 n=0;
                 label3:
                 while (n < MAX_FIX)
                 {
                 if((product[7] == f[n].hold_shape)
                   &&(product[8] <= f[n].hold_area_index)
                   &&(product[1] == f[n].hold_direction)
                   &&(product[10] <= f[n].pos_accuracy)
                   &&((p[j].as_compress <= f[n].holding_force)
                   &&(p[j].as_tensile <= f[n].holding_force)
                   &&(p[j].as_torque <= f[n].holding_force)))
                 {
                      o = 0;
                      while(o < MAX_FIX)
                      {
                      if((product[7] == f[o].hold_shape)
                        &&(product[8] <= f[o].hold_area_index)
                        &&(product[1] == f[o].hold_direction)
                        &&((p[j].dis_compress <= f[o].holding_force)
                                                   &&(p[j].dis_tensile <= f[o].holding_force)
                                                   &&(p[j].dis_torque <= f[o].holding_force)))
                                                  {
                                                  num_joint = 1;
                                                  calculation();
                                                  if((o+1) == MAX_FIX)  break;
                                                  else {o++; continue;}
                                                  }
                                                  else {o++; continue;}
                                                  }
                                                  n++;
                                                  goto label3;
                                             }
                                             else {n++; continue;}
                                             }
                                             m++;
                                             goto label2;

                                           }
                                           else {m++; continue;}
                                         }
                                         k++;
                                         goto label1;
                                     }
                                     else {k++;continue;}
                                }
                                j++;
                                goto label;
                          }
                        else {j++; continue;}

   }
 }

void nonflex()
{
  j = 0;
  label4:
  while( j < MAX_SET )
  {
    if( product[2] <= p[j].flex_factor)
        {
        num_joint = 0.0;
        if(p[j].physical_type == 0)  /* If the joint is discrete type, num_joint should be
integer. */

{       num_joint_discrete = (int)(((float)product[3]/(float)p[j].strength)+0.9);/* Round
up to nearset integer.*/
        num_joint = (float)num_joint_discrete;
}
        if((p[j].physical_type == 0)&&(p[j].flex_factor == 1)&&(num_joint<=1))
         {
         j++;
         goto label4;
         }

        if(p[j].physical_type == 1)  /* If the joint is continuous type, num_joint would be
float type. */

        num_joint = ((float)product[3]/(float)p[j].strength);
        if(num_joint >=1)  /* Less than one will not meet the minimum requirement */
         {

         /* Checking the total required joint area is great then the available product area.*/
         if(((product[4]>= p[j].min_x*num_joint)&&(product[5]>= p[j].min_y))
           ||((product[4]>= p[j].min_x)&&(product[5]>= p[j].min_y*num_joint)))
           {

          /* Checking the length of joint is available or not. */
          if(p[j].physical_type == 1 || (p[j].physical_type == 0
          && p[j].joint_size_z == 0 && product[6]<=0.2)
          || (p[j].physical_type == 0 && product[6] <= p[j].joint_size_z))
            {
                k = 0;
                label5:
                while( k < MAX_RES)
                {
                 if( product[10] <= r[k].res_accuracy)
                   {
                   if( (p[j].as_compress <= r[k].res_compress)
                          && ( p[j].as_tensile <= r[k].res_tensile)
                          && ( p[j].as_torque <= r[k].res_torque)
                          && ( p[j].as_tracing == r[k].res_tracing))
                            {
                            if((p[j].as_compress == 0 && r[k].res_compress ==0 )||
                             ((p[j].as_compress < r[k].res_compress)
                                  && (p[j].as_compress > 0)))
                                {

                              m = 0;
                              label6:
                              while ( m < MAX_RES)
                              {
                              if(product[10]<= r[m].res_accuracy)
                              {
                              if ((p[j].dis_compress <= r[m].res_compress)
                              && (p[j].dis_tensile <= r[m].res_tensile)
                              && (p[j].dis_torque <= r[m].res_torque)
                              && (p[j].dis_tracing == r[m].res_tracing))
                              {
                              if((p[j].dis_compress == 0 && r[m].res_compress == 0)||
                                ((p[j].dis_compress<
r[m].res_compress)&&(p[j].dis_compress>0)))
                              {
                               n=0;
                               label7:
                               while (n < MAX_FIX)
                               {
                               if((product[7] == f[n].hold_shape)
                                 &&(product[8] <= f[n].hold_area_index)
                                 &&(product[1] == f[n].hold_direction)
                                 &&(product[10] <= f[n].pos_accuracy)
                                 &&((p[j].as_compress <= f[n].holding_force)
                                 &&(p[j].as_tensile <= f[n].holding_force)
                                 &&(p[j].as_torque <= f[n].holding_force)
                                 &&(p[j].as_tracing <= f[n].holding_force)))
                                 {

                               /* Disassemby doesn't need fixture accuracy. */
                                 o = 0;
                                 while(o < MAX_FIX)
                                 {
                                 if((product[7] == f[o].hold_shape)
                                      &&(product[8] <= f[o].hold_area_index)
                                      &&(product[1] == f[o].hold_direction)
                                      &&((p[j].dis_compress <= f[o].holding_force)
                                      &&(p[j].dis_tensile <= f[o].holding_force)
                                      &&(p[j].dis_torque <= f[o].holding_force)
                                      &&(p[j].dis_tracing <= f[o].holding_force)))
```

```
                         {

                             calculation();
                             if((o+1)== MAX_FIX)
                             break;
                             else { o++; continue;}
                         }
                         else { o++; continue;}
                         }
                         n++;
                         goto label7;
                        }
                        else {n++; continue;}
                        }
                        m++;
                        goto label6;
                       }
                       else {m++; continue;}
                       }
                       else {m++; continue;}
                       }
                       else {m++; continue;}
                      }
                      k++;
                      goto label5;
                  }
                  else {k++;continue;}
                  }
                  else {k++;continue;}
                 }
               else {k++; continue;}
               }
               j++;
               goto label4;
       }
      else {j++; continue;}
      }
     else {j++; continue;}
     }
    else {j++; continue;}
    }
else {j++; continue;}
```

```
  }
}


void calculation()
{
float modify_time = 30.0;
float modify_labor_cost = 15.0;

TM = (product[11]*num_joint*modify_time*p[j].part_change);
CM = (TM*(p[j].part_change)*modify_labor_cost)/3600;

if(p[j].joint_size_z == 0)
{
  process_as  = r[k].process_time;
  process_dis = r[m].process_time;
}

else
{
  process_as  = (p[j].joint_size_z * r[k].process_time);
  process_dis = (p[j].joint_size_z * r[m].process_time);
}

as_t  = (f[n].fixture_time + r[k].setup_time
     + (num_joint* process_as)
     + TM);                        /* Unit is in seconds  */

dis_t = (f[o].fixture_time + r[m].setup_time
     + (num_joint* process_dis));  /* Unit is in seconds. */

as_c  = (as_t*r[k].labor_cost/3600) + (num_joint*p[j].fastener_cost/100)
     + ((r[k].setup_time+(num_joint* process_as))*r[k].machine_over/3600)
     + ((f[n].fixture_time+(num_joint* process_as))*f[n].fixture_over/3600)
     + CM;

dis_c = (dis_t*r[m].labor_cost/3600)
     + ((r[m].setup_time+(num_joint* process_dis))*r[m].machine_over/3600)
     + ((f[o].fixture_time+(num_joint* process_dis))*f[o].fixture_over/3600);


if(product[9] < p[j].damage_index)
  {
   PS =( p[j].damage_index - product[9]);

  }
else
  PS = 0;


 if(as_t < TA)    /*  If the assembly time is less than previous one,
                      program will update TA, TD_MIN, and TD_MAX. */
 { TA = as_t;
   TD_MIN = dis_t;
   TD_MAX = dis_t;
   J1 = J2 = j;
   K1 = K2 = k;
   M1 = M2 = m;
   N1 = N2 = n;
   O1 = O2 = o;
   DFA_T_MIN = TA + TD_MIN;
   DFA_T_MAX = TA + TD_MAX;
   DFA_T_PS  = PS;

}
 else if(as_t == TA)  /* If the assembly time is the same as before,
                         program will reset TD_MIN and TD_MAX. */
 {
   if(dis_t < TD_MIN)
    {
     TD_MIN = dis_t;
     J1 = j;
     K1 = k;
     M1 = m;
     N1 = n;
     O1 = o;
     DFA_T_MIN = TA + TD_MIN;

    }
   if(dis_t > TD_MAX)
    {
     TD_MAX = dis_t;
     J2 = j;
     K2 = k;
     M2 = m;
     N2 = n;
    O2 = o;
    DFA_T_MAX = TA + TD_MAX;
   }
   DFA_T_PS = PS;
}


if(as_c < CA)
{
  CA = as_c;
  CD_MIN = dis_c;
  CD_MAX = dis_c;
  J3 = J4 = j;
  K3 = K4 = k;
  M3 = M4 = m;
  N3 = N4 = n;
  O3 = O4 = o;
  DFA_C_MIN = CA + CD_MIN;
  DFA_C_MAX = CA + CD_MAX;
  DFA_C_PS  = PS;
}
else if(as_c == CA)
{
  if(dis_c < CD_MIN)
   {
    CD_MIN = dis_c;
    J3 = j;
    K3 = k;
    M3 = m;
    N3 = n;
    O3 = o;
    DFA_C_MIN = CA + CD_MIN;
   }

  if(dis_c > CD_MAX)
   {
    CD_MAX = dis_c;
    J4 = j;
    K4 = k;
    M4 = m;
    N4 = n;
    O4 = o;
    DFA_C_MAX = CA + CD_MAX;
  }
  DFA_C_PS = PS;
}


if(dis_t <TD)
{
 TD = dis_t;
 TA_MIN = as_t;
 TA_MAX = as_t;
 J5 = J6 = j;
 K5 = K6 = k;
 M5 = M6 = m;
 N5 = N6 = n;
 O5 = O6 = o;
 DFD_T_MIN = TA_MIN + TD;
 DFD_T_MAX = TA_MAX + TD;
 DFD_T_PS  = PS;
}
else if (dis_t == TD)
{
 if(as_t < TA_MIN)
  {
   TA_MIN = as_t;
   J5 = j;
   K5 = k;
   M5 = m;
   N5 = n;
   O5 = o;
   DFD_T_MIN = TA_MIN + TD;

  }
 if(as_t > TA_MAX)
  {
   TA_MAX = as_t;
   J6 = j;
   K6 = k;
   M6 = m;
   N6 = n;
   O6 = o;
   DFD_T_MAX = TA_MAX + TD;
  }
  DFD_T_PS = PS;
```

```
}


if(dis_c < CD)
{
 CD = dis_c;
 CA_MIN = as_c;
 CA_MAX = as_c;
 J7 = J8 = j;
 K7 = K8 = k;
 M7 = M8 = m;
 N7 = N8 = n;
 O7 = O8 = o;
 DFD_C_MIN = CA_MIN + CD;
 DFD_C_MAX = CA_MAX + CD;
 DFD_C_PS  = PS;

}
else if(dis_c == CD)
{
 if(as_c < CA_MIN)
  {
   CA_MIN = as_c;
   J7 = j;
   K7 = k;
   M7 = m;
   N7 = n;
   O7 = o;
   DFD_C_MIN = CA_MIN + CD;
  }
 if(as_c > CA_MAX)
  {
   CA_MAX = as_c;
   J8 = j;
   K8 = k;
   M8 = m;
   N8 = n;
   O8 = o;
   DFD_C_MAX = CA_MAX + CD;
  }
  DFD_C_PS = PS;
}
```

```
 if((as_t + dis_t) < CAD_T_TEM)
 {
  CAD_T_TEM = as_t + dis_t;
  CAD_T = CAD_T_TEM;
  J9 = j;
  K9 = k;
  M9 = m;
  N9 = n;
  O9 = o;
  CAD_T_PS = PS;

 }

 if((as_c + dis_c) < CAD_C_TEM)
 {
  CAD_C_TEM = as_c + dis_c;
  CAD_C = CAD_C_TEM;
  J10 = j;
  K10 = k;
  M10 = m;
  N10 = n;
  O10 = o;
  CAD_C_PS = PS;
 }

}

/*	Deleting nodes from the memory.	*/

void delete (struct node *listptr)
{
	struct node *temp1, *temp2;

	temp1 = listptr;
	while (temp1->next != NULL) {
		temp2 = temp1->next;
		free ((void *)temp1);
		temp1 = temp2;
	}
	free ((void *)temp1);
	listptr = NULL;
}
```

```
void print()
{
            printf("\n\t%f\t\t\t%f", CAD_T/60.0, CAD_C);

            printf("\n\t%f", DFA_T_MIN/60.0);  /* In minutes  */
            printf("\t%f", DFA_T_MAX/60.0);
            printf("\t%f", DFA_C_MIN);
            printf("\t%f", DFA_C_MAX);

            printf("\n\t%f", DFD_T_MIN/60.0);
            printf("\t%f", DFD_T_MAX/60.0);
            printf("\t%f", DFD_C_MIN);
            printf("\t%f\n", DFD_C_MAX);
            printf("\n\t%f %f %f %f\n", as_c, CA, CD_MIN, CD_MAX);
            printf("\t%f %f %f %f\n\n", dis_c, CD, CA_MIN, CA_MAX);

            printf("\t%d %d   %d %d %d %d   %d %d %d %d\n",
                      J9, J10, J1, J2, J3, J4, J5, J6, J7, J8);
            printf("\t%d %d   %d %d %d %d   %d %d %d %d\n",
                      K9, K10, K1, K2, K3, K4, K5, K6, K7, K8);
            printf("\t%d %d   %d %d %d %d   %d %d %d %d\n",
                      M9, M10, M1, M2, M3, M4, M5, M6, M7, M8);
            printf("\t%d %d   %d %d %d %d   %d %d %d %d\n",
                      N9, N10, N1, N2, N3, N4, N5, N6, N7, N8);
            printf("\t%d %d   %d %d %d %d   %d %d %d %d\n",
                      O9, O10, O1, O2, O3, O4, O5, O6, O7, O8);

            printf("%f %d %d %f %f\n", product[9], j,
                   p[j].damage_index, TM, CM);
            printf("%d %d %d %d %d %d\n", CAD_T_PS, CAD_C_PS,
                     DFA_T_PS, DFA_C_PS, DFD_T_PS, DFD_C_PS);

}

void save()
{
FILE *output;
        output = fopen("results.txt", "w");
        if( output == NULL )
          {
           printf("\nResults.txt file cannot be opened.");
           exit(4);
        }
       else
        {
          fprintf(output,"\n%12s %f","    CAD_TT =",CAD_TT/60.0);
          fprintf(output,"\t%12s %f","    CAD_TC =",CAD_TC);
          fprintf(output,"\n\n%12s %f","DFA_TT_MIN =", DFA_TT_MIN/60.0);
          fprintf(output,"\t%12s %f","DFA_TC_MIN =", DFA_TC_MIN);
          fprintf(output,"\n%12s %f","DFA_TT_MAX =", DFA_TT_MAX/60.0);
          fprintf(output,"\t%12s %f","DFA_TC_MAX =", DFA_TC_MAX);

          fprintf(output,"\n\n%12s %f","DFD_TT_MIN =", DFD_TT_MIN/60.0);
          fprintf(output,"\t%12s %f","DFD_TC_MIN =", DFD_TC_MIN);
          fprintf(output,"\n%12s %f","DFD_TT_MAX =", DFD_TT_MAX/60.0);
          fprintf(output,"\t%12s %f","DFD_TC_MAX =", DFD_TC_MAX);

          fprintf(output,"\n\n%12s %d","CAD_TT_PS  =", CAD_TT_PS);
          fprintf(output,"\n%12s %d","CAD_TC_PS  =", CAD_TC_PS);
          fprintf(output,"\n%12s %d","DFA_TT_PS  =", DFA_TT_PS);
          fprintf(output,"\n%12s %d","DFA_TC_PS  =", DFA_TC_PS);
          fprintf(output,"\n%12s %d","DFD_TT_PS  =", DFD_TT_PS);
          fprintf(output,"\n%12s %d","DFD_TC_PS  =", DFD_TC_PS);

               }
}□
```

VITA

Piyen Chang was born on December 13, 1962, in Taichung, Taiwan. He received his A.A. degree in Industrial Management at Tamsui Oxford College, Tamsui, Taiwan, in June 1985. He then served in the Taiwan's Army and discharged in 1987. In January 1989, he enrolled in Industrial Technology Department at Central Missouri State University and received his B.S. in August 1991. In August 1991, he was admitted as a graduate student of Industrial Engineering at University of Nebraska-Lincoln, and received his M.S. degree in December 1993. In January 1994, he enrolled in Industrial and Systems Engineering Department at Virginia Polytechnic Institute and State University to pursue his another M.S. degree in Manufacturing Systems Engineering.

[illegible]