

QCAD
A Framework for Total Quality Development
in an
Object Oriented Knowledge Based Engineering Environment

by

Yale Goldis

B.S., Mechanical Engineering
Tufts University, 1991

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

Master of Science in Mechanical Engineering

at the
Massachusetts Institute of Technology
February 1994

© 1994 Massachusetts Institute of Technology
All rights reserved

Signature of Author.....

Department of Mechanical Engineering
January 14, 1994

Certified by.....

Don P. Clausing
Bernard M. Gordon Adjunct Professor of
Engineering Innovation and Practice
Thesis Advisor

Accepted by

Ain Sonin
Chairman Departmental Graduate Committee
Department of Mechanical Engineering

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

MAR 08 1994

LIBRARIES

ARCHIVES

QCAD
A Framework for Total Quality Development
in an
Object Oriented Knowledge Based Engineering Environment

by

Yale Goldis

Submitted to the Department of Mechanical Engineering
on January 14, 1994 in partial fulfillment of the
requirements for the Degree of

Master of Science in Mechanical Engineering

Abstract

Conventional knowledge based engineering (KBE) systems connect engineers to engineers. These design systems integrate corporate engineering practices into a computerized framework. The intention is to quickly and efficiently create a product specification, leaving engineers time for other design tasks. KBE applications address conceptual design, geometric design, and part manufacturability, but do so in a segmented fashion.

This segmentation is reminiscent of the *throw it over the wall* approach to design. Each program optimizes its own task at the expense of the overall system. Many companies today use multifunctional product development teams and total quality development methods to break down barriers among departments. KBE design systems should follow suit.

To expand the horizons of computerized knowledge based engineering, this thesis develops the Quality Computer Aided Design (QCAD) system and tests it with both an easy to understand clamp model and a more complex copier model. The QCAD methodology consists of two parts: (1) a design process framework and (2) a computer program for use with the design framework. It is intended that a multifunctional product development team use QCAD to assist in developing both new and variational products from function designation to hardware definition to process specification.

The QCAD system was implemented in the ICAD design language (IDL). ICAD is the leading supplier of commercial knowledge based engineering software worldwide. IDL is a super set of common lisp, an object oriented programming language, and is geared toward modeling products in the ICAD user interface.

Comments are expressed about the benefits and drawbacks of combining the two design methods in a product development environment. Future improvements and research work are also suggested.

Thesis Advisor: Don P. Clausing

Title: Bernard M. Gordon Adjunct Professor of Engineering Innovation and Practice

Acknowledgments

First, I would like to thank my advisor, Professor Don Clausing, for his support, guidance, and assistance on this thesis. I appreciate the direction and freedom he gave me to explore this topic. I am most grateful to him for teaching me about total quality development and for giving me the opportunity to study cultural change with him. I am truly fortunate to have him as an advisor.

Second, I would like to thank Professor Ron Andrade, who is here on sabbatical, but gave his assistance with my thesis anyway. Moreover, I appreciate our countless philosophical discussions about life, the universe, and everything.

Third I would like to thank Diane de Alderete for her indispensable administrative support.

Fourth, I would like to thank all of the people at ICAD for their assistance with this project, including: Larry Rosenfeld for his support; Tom Smith, June Powers, and Phil Stockton for the training sessions; Mark Ouelette, Bill Brand, and Ralph Verrilli for their interviews; Mark Ouelette and Bob Phillips for the innumerable conversations and programming support.

Furthermore, I would like to thank the people at MIT for their assistance with this project, including: Professor George Chryssolorus and the students in his lab for the use of their workstation; Kevin Spratt for his assistance in configuring that workstation; and Añita Flynn in the artificial intelligence lab for her assistance with the initial installation of ICAD.

Lastly, I would like to thank my parents and friends for their patience and understanding.

I would also like to thank the Leaders for Manufacturing Program for its support of this work.

Table of Contents

1. Introduction.....	7
2. Review of Total Quality Development	12
2.1. Overview	12
2.2. Design Process Structure.....	13
2.3. Basic Quality Function Deployment.....	15
2.4. Pugh Concept Selection	20
2.5. Enhanced Quality Function Deployment	22
2.6. Taguchi System of Quality Engineering	24
3. Review of Computers in Design.....	31
3.1. Overview	31
3.2. Introduction to Knowledge Based Engineering.....	32
3.3. Conventional Design Assistants	34
3.3.1. Technically Focused Assistants	35
3.3.2. Quality Focused Assistants	36
3.4. Design History Systems.....	37
3.4.1. Design Representation Formats.....	37
3.5. Knowledge Based Design Systems	38
3.5.1. CAD Systems	39
3.5.2. ICAD	40
3.5.2.1. Representative ICAD Projects	41
3.5.3. Quality Design Systems	42
3.6. Summary.....	44
4. QCAD Definition.....	47
4.1. Overview	47
4.2. Initial Development	48
4.3. Basic QCAD Framework.....	55
4.4. Design Process in QCAD	59
4.4.1. Function Hierarchy	62
4.4.2. Hardware Hierarchy.....	66
4.4.3. Process Hierarchy	69
4.5. Computerized Elements of QCAD.....	71
4.5.1. Description of Clamp Product Model.....	72
4.5.2. Function Hierarchy	74
4.5.2.1. Function Identification.....	74
4.5.2.2. Constraints	75
4.5.2.3. QFD House of Quality Matrix	76
4.5.2.4. Function Diagram.....	79
4.5.2.5. Concept Generation.....	81
4.5.2.6. Concept Selection	82

4.5.2.7.	Concept Review	84
4.5.2.8.	Additional Comments	85
4.5.3.	Hardware Hierarchy.....	85
4.5.3.1.	Product Structure Diagram.....	86
4.5.3.2.	QFD Design Matrix.....	87
4.5.3.3.	Hardware Parameters	88
4.5.3.4.	Product Parameter Design	89
4.5.3.5.	Material and Process Selection	90
4.5.3.6.	Concept Review	91
4.5.3.7.	Additional Comments	92
4.5.4.	Process Hierarchy	93
4.5.4.1.	Bill of Materials.....	93
4.5.4.2.	Process Parameters.....	94
4.5.4.3.	Process Parameter Design	94
4.5.4.4.	QFD Process and Production Matrices	94
4.5.4.5.	Additional Comments	96
4.5.4.5.1.	Part Formation.....	99
4.5.4.5.2.	Part Finalization	100
4.5.4.5.3.	Instruction for Quality Assurance Check Points	100
4.5.4.5.4.	Process Feedback.....	100
4.5.5.	Miscellaneous	101
4.5.5.1.	Fault Trees	101
4.5.5.2.	Miscellaneous.....	102
5.	Discussion of QCAD Implementation	104
5.1.	Overview	104
5.2.	QCAD Case Studies.....	104
5.2.1.	Generic QCAD Structure.....	105
5.2.2.	Clamp Product Model.....	111
5.2.3.	Copier Product Model	123
5.2.4.	Larger Product Models	132
5.3.	Observations from QCAD Case Studies.....	133
5.3.1.	QCAD and TQD.....	133
5.3.1.1.	QCAD Improvements over TQD.....	133
5.3.1.2.	QCAD Difficulties with TQD	137
5.3.2.	QCAD and KBE.....	140
5.3.2.1.	QCAD Improvements over KBE	140
5.3.2.2.	QCAD Difficulties with KBE.....	142
5.3.3.	Summary	143
5.4.	QCAD in Product Development	145
5.4.1.	QCAD Uses in Product Development	146
5.4.2.	Additional QCAD Uses for Incremental Product Improvements ...	150

6. Conclusions and Future Work.....	152
6.1. Summary and Conclusions.....	152
6.2. Future Work	154
References.....	157
Appendix A.....	162
Appendix B	167
Appendix C.....	180

Chapter 1

Introduction

Computers are deeply embedded in our society. They are used in many applications, including magnetic resonance imaging, information processing, routine calculations, and human thought process modeling. It is hard to imagine life without computers. They have inundated all divisions of corporations, with applications in communication, data storage and retrieval, and analysis work. More specifically, in corporate product development, computers are mainly used for analysis, including design assistant functions and routine redesign tasks.

Design assistant programs perform specific tasks to help designers with calculations and sensitivity analyses. Some examples include finite element packages, math packages, and spreadsheets. A more general class of systems, which are used as design assistants, but are also used for routine redesign, include CAD¹ systems, expert systems, and knowledge based engineering systems. These systems use corporate design knowledge, engineering principles, and rules of thumb to explore designs and to make quick changes to designs. Some knowledge based and expert systems are even used to coordinate information, combining data from other design analysis programs and releasing processed information to those same programs. In general, computers have made available a means for quickly performing a pre-designated set of calculations and design tasks to speed up product development work and to make fewer mistakes.

Along these same lines, Xerox has identified three goals where computers can be used for improving product development (Heatley and Spear 1992).

- “1. Reduce the time required to develop and deliver electromechanical systems to the market.

¹ This definition of CAD, Computer Aided Design, includes the geometry modeling capabilities and the analysis features, including circuit analysis, computer aided manufacturing, parametric design, etc.

2. Improve quality of our products - consistency, accuracy, and manufacturability. This increases customer satisfaction with our products.
3. Reuse and incrementally improve existing designs, sub-systems, components.”

The Xerox quote not only identifies possible uses of computers in product development, but also exemplifies a commitment to satisfying customer wants and needs with robust products. The latter is a rare find in literature on computers in design. This perhaps stems from artificial intelligence research longing to model human thought processes (Edwards 1991) and companies wishing to develop products faster.

In general, using computers doesn't change the design process used in companies—computers reinforce it, standardize it, institutionalize it, and accelerate it. Expert system programmers are trained to extract design rules from engineers and code them into a program (Edwards 1991), whether or not they are relevant to satisfying the company's customers. Unless a company is already looking at customer wants and needs in designs, computers will not help satisfy those needs.

Computers may even make it harder for people to communicate and to improve products through collaborative effort. Computers tend to reinforce individualism (Moses 1990) and a narrow focus on a small area of the problem thus making them lose sight of the overall design and how their part interacts with other parts (Reddy 1991). The traditional product development process is characterized by intense rivalries among departments, with one department throwing a design *over the wall* to the next department (Clausing 1991). In such a situation, the computer system bought by one department may not even be compatible with the computer system purchased by another, further increasing the barriers in product development.

Instead of, or in addition to, using computers to accelerate the design process, some companies are using concurrent engineering techniques to overcome the barriers in traditional product development. Synonyms for concurrent engineering include simultaneous engineering, life-cycle engineering, and company wide quality control. The essence of these practices is to get people to interact, to see things from a different perspective and to design more cost-efficient, manufacturable, and serviceable products. Takeuchi and Nonaka (1986) compare this new approach to a rugby team moving a scrum down the field. The old approach is likened to that of a relay race.

For comparison, Figure 1.1 shows three typical product development processes. The Type A process is the traditional, sequential approach, with distinct barriers between phases. The Type B process, used by NASA, has some overlap, but is still dysfunctional. The Type C process is the concurrent process, with extensive overlap. Note that a concurrent process shortens the product development cycle time.

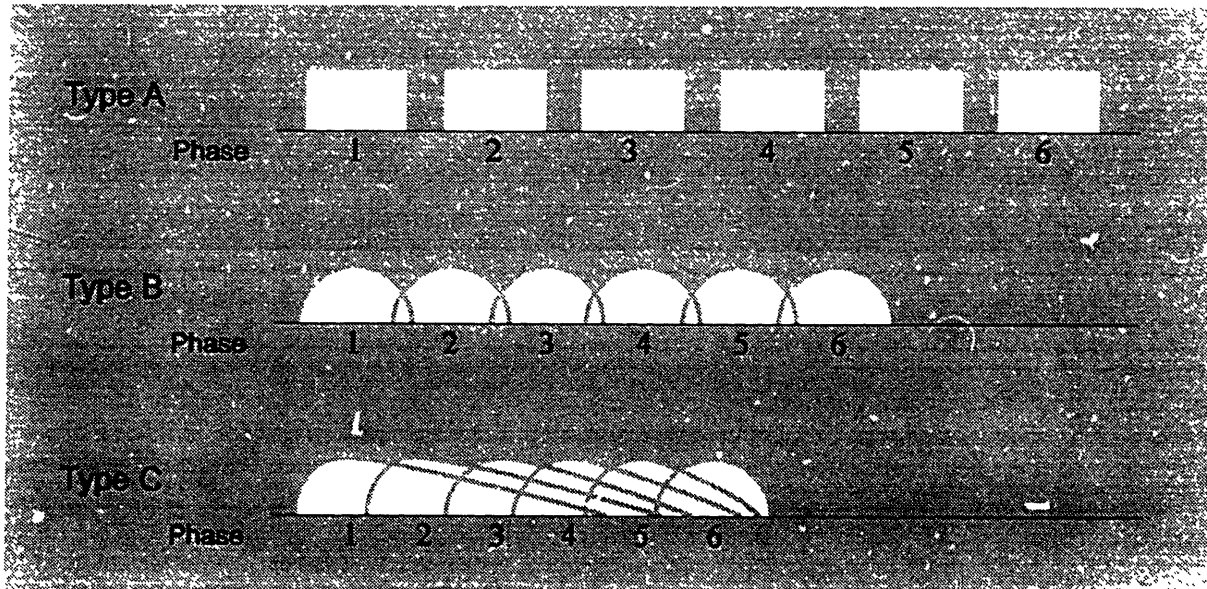


Figure 1.1 - Product Development Phases (Takeuchi and Nonaka 1986, Exhibit I)

In addition to concurrent engineering methodologies, Clausing (1994) has developed a Total Quality Development (TQD) approach to design, which promotes a

focus on quality, cost and delivery, an emphasis on customer satisfaction and an emphasis on competitive benchmarking. TQD uses visual connective methods and team experience to build understanding, consensus and a strong commitment to decisions.

Given that new paradigms are emerging in product development that are replacing the traditional, outdated processes, a new look should be taken at how computers are used in the design process. More specifically, how should computers be used with TQD and how should TQD be used with computers?

This thesis is an initial study of these questions. It assumes that computers should be used in product development and thus specifies and tests a framework for a Quality Computer Aided Design workspace called QCAD. QCAD facilitates the design and analysis of a product as it moves from total system concept definition to detailed part feature design and from function definition to process specification. Written in the ICAD Design Language (ICAD 1993), QCAD incorporates all of the features of the leading knowledge based engineering software in world with the quality design methods proven to facilitate enhanced product development. Traditional knowledge based engineering connects engineers to engineers—QCAD connects customers to the factory floor.

The remaining chapters follow the organization and description below:

Chapter 2: Review of Total Quality Development. This chapter presents the concepts and techniques of Total Quality Development. It starts out with a discussion of several common views of the design process and describes a generally accepted structure for the design process. The chapter then goes on to explain concepts and methods specific to Total Quality Development, including Enhanced Quality Function Deployment, Pugh Concept Selection, and the Taguchi system of quality engineering.

Chapter 3: Review of Computers in Design. As mentioned above, computers are used in many ways in the design process. This chapter describes several trends in the use of computers for product development to show some of the ideas that went into the development of QCAD. The chapter begins with a general introduction to knowledge

based engineering and then describes conventional design assistant programs and design history systems. The latter two types of systems are then used to help explain knowledge based engineering systems, with emphasis on ICAD KBE applications. The chapter also emphasizes programs that facilitate using the quality methods described in Chapter 2. The last part of the chapter brings together all of the presented information and shows how it relates to QCAD.

Chapter 4: QCAD Definition. Based on the discussion in the last two chapters, this chapter sets out to define the requirements for using computers in a total quality development environment. This chapter defines both a design process framework for a PDT to follow, and a specification for a computerized quality development system. The specification uses several examples to show where and how information is organized in QCAD. To facilitate explanation, the concepts in this chapter are explained using a simple clamp model.

Chapter 5: Discussion of QCAD Implementation. Given the definition for QCAD developed in Chapter 4, this chapter describes the implementation of the methodology with a simple clamp and a complex copier. The copier has both a broad and a deep range of sub-systems that challenge the computer's ability to decompose and recombine product information. The second part of the chapter extrapolates some learnings about the use of computers in design, as learned from the two case studies. The third part of the chapter describes how a product development team could use QCAD for design.

Chapter 6: Conclusions and Future Work. This chapter summarizes the thesis work and presents some of the notable results identified in the QCAD system specification and case studies. Also discussed are suggestions for future research on QCAD and in the use of computers in total quality development.

Chapter 2

Review of Total Quality Development

2.1. Overview

Products have been created to help man harness nature since prehistoric times. Many of these early designs were created from common sense, experience, and empirical data. Designs were then improved throughout the ages. Only recently, within the last two hundred years, has engineering design become more scientifically oriented, with designers searching vast collections of data on the understanding of nature (Krick 1965). A design process is needed to help understand how to select an appropriate solution to a problem.

Cross proposed two ways of looking at the design process, including descriptive and prescriptive methods (Cross 1989). Descriptive methods analyze how designers design and attempt to specify heuristics that model the design process. This is the model that knowledge engineers use to solicit information to put into an expert system. The information entered into the computer model attempts to imitate the process that a designer uses to develop a product.

Alternatively, prescriptive methods attempt to create a design process to be followed when designing a product. “Many of these prescriptive models have emphasized the need for more analytical work to precede the generation of solution concepts. The intention is to try to ensure that the design problem is fully understood, that no important elements of it are overlooked and that the ‘real’ problem is identified. There are plenty of examples of excellent solutions to the wrong problem! (Cross 1989)”

A generally accepted framework for a prescriptive design model includes function analysis, concept synthesis, and evaluation. The goal of function analysis is to define the design criteria and set the performance specifications in a way that represents the correct problem to be solved. Concept synthesis is used to generate solutions that match the requirements set by the function analysis. Lastly, an evaluation mechanism is provided to

help select among the design alternatives, before the design is fully specified, such that the final design has a high fidelity with the function specifications.

There are many prescriptive models for the design process (Asimow 1962; Krick 1965; Pahl and Beitz 1984; Cross 1989; Pugh 1991; etc.). Total Quality Development (TQD) is also a prescriptive method. It has a quality focused approach to design. TQD uses a concurrent process that emphasizes customer satisfaction, competitive benchmarking, robustness, and a focus on quality, cost, and delivery.

The following sections describe the fundamental parts of the TQD process. Other TQD tools and concepts that are used in the QCAD system are described as necessary in subsequent chapters. The reader is referred to Don Clausing's book on Total Quality Development (1994) for a more elaborate description of the methodology.

2.2. Design Process Structure

A design can be thought of at several levels of abstraction. The total system level describes the overall product, including its high level functions, hardware, and processes. The total system is broken into sub-systems, which are more detailed sections of the product. A design can also have sub-sub-systems and sub-sub-sub-systems, and so on, until the design is broken down to the piece part level. The piece part level describes the individual components that comprise a design. At the final level of detail are part features that describe critical² areas of a part.

Figure 2.1 is a partial hardware tree for a Xerox copier, which shows the basic hierarchical structure of a design. The dotted lines show places where other hardware pieces could go. Breaking apart a design is subjective and can be different for different designers. A graphic representation of the hardware is shown in Figure 2.2 to further explain the concept of abstraction levels.

² Critical parts are those parts that are crucial in satisfying a product's functional or safety requirements; those parts that define most of the product's costs and manufacturing lead time; and those parts that are expected to cause the most difficulties in manufacturing or design (Florusse and Clausing, 1992).

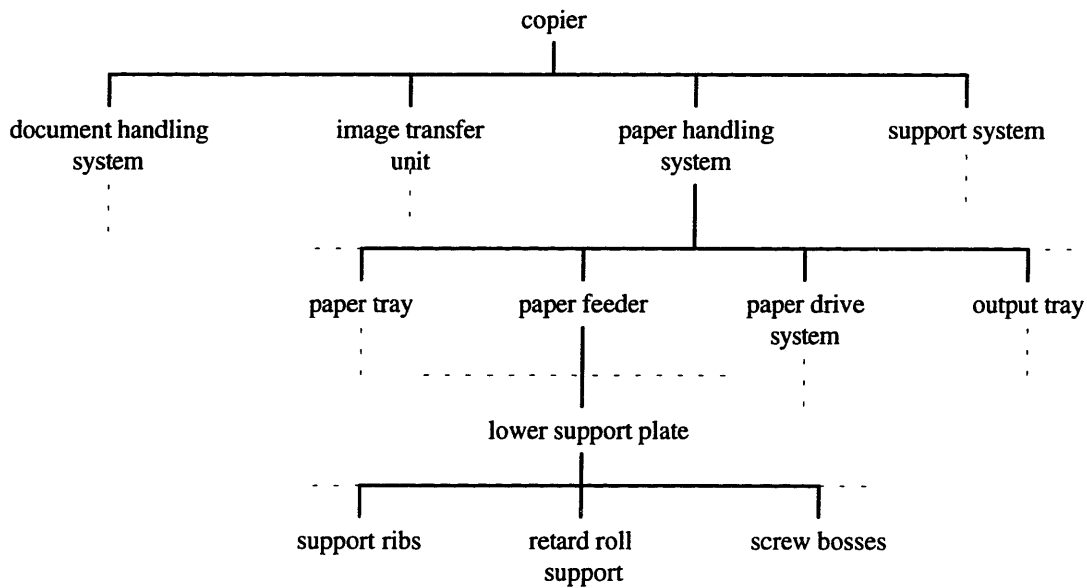
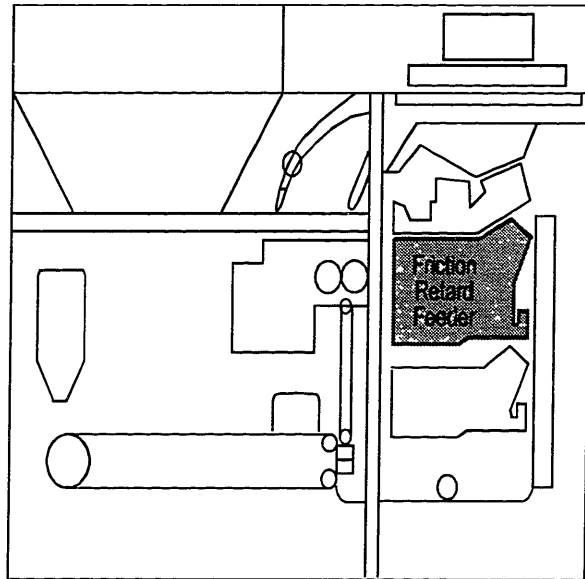


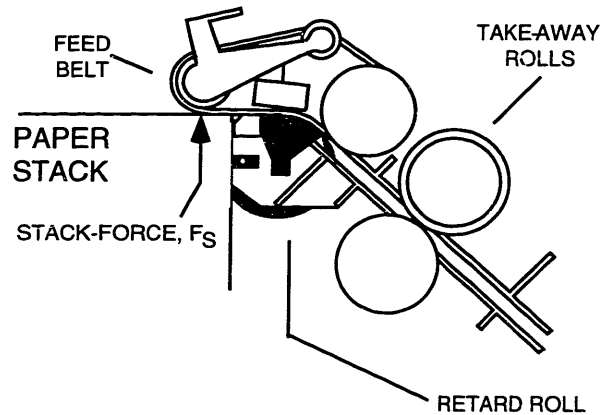
Figure 2.1 - Hardware Tree for Copier

The total system level is a copier. The copier can be broken into several sub-systems: original documents handler, image transfer unit, blank paper handling system and a copier support system. Each of these sub-systems have other sub-systems, for example, the paper feeder. A piece part would be the lower plate that supports some of the parts on the paper feeder. Part features include the flanges, bosses, and ribs on the support plate.

The natural flow of a design starts with a total system definition, then goes to sub-systems to piece parts and then to part features. The total system starts with a detailed list of customer wants and needs, also known as the voice of the customer, and deploys those characteristics to product expectations. The total system definition ends with selecting a concept for further refinement. The rest of the levels start with higher level expectations, translate them to expectations for the current level, and then select a concept for that level. These tasks are facilitated with quality function deployment and concept selection.



(a) Copier Total System Architecture
(Clausing and Pugh 1991, fig. 4)



(b) Paper Feeder Sub-system
(Clausing 1994, fig. 5.7)

Figure 2.2 - Copier Hardware

Deploying product expectations and selecting hardware and process concepts, does not mean that the design has the best product or process parameters, or the most economical tolerancing and on-line quality control procedures. Taguchi's system of quality engineering is used to optimize the product and process with respect to environmental conditions and customer satisfaction.

2.3. Basic Quality Function Deployment

Quality Function Deployment was formalized in 1972 by Akao, Mizuno, and Furukawa at the Mitsubishi Heavy Industries's Kobe shipyards in Japan (Akao 1990, xv). QFD is a method for translating customer needs to product expectations to hardware characteristics to process planning and to production planning. It does this through a series of matrices, shown in Figure 2.3. The matrices are called houses, with the first house being called the House of Quality. The QFD process is performed by a multifunctional product development team (PDT) and is used to immerse them in the

design to help them understand the design problem and to help identify possible hardware solutions.

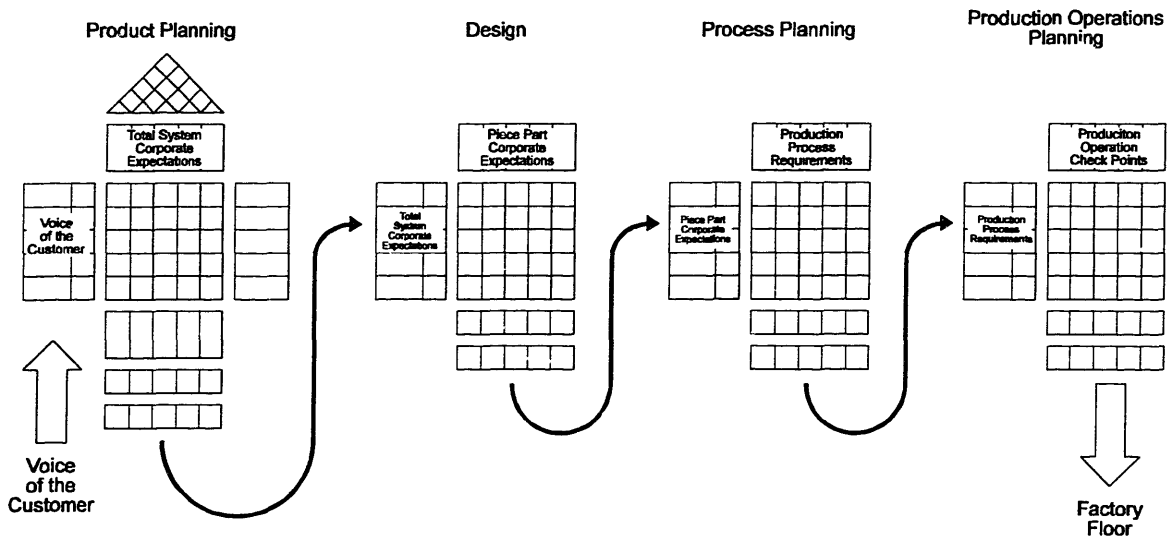


Figure 2.3 - Basic Quality Function Deployment
(Clausing and Pugh 1991, fig. 1 modified)

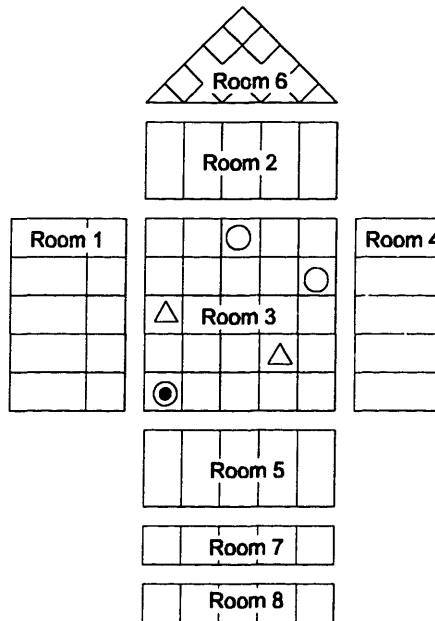


Figure 2.4 - The House of Quality (Hauser and Clausing 1988, Exhibit IX modified)

The House of Quality improves on current product development practices by relating development efforts to the voice of the customer, comparing the company's product with the best products on the market, and by creating a clear, commonly understood specification for the rest of product development. The House of Quality is shown in Figure 2.4 and its rooms are explained below.

Room 1 - Voice of the Customer. This room is the input to the house of quality. It contains the customer wants and needs. These requirements are found through qualitative interviews with customers and are left in qualitative terms. Also in room 1 is an importance rating for each customer input. This is usually a qualitative assessment by the product development team, however, a customer ranking of the requirements is preferable (Clausing 1994). More information on identifying customer needs can be found in a working paper by Griffin and Hauser (1991), titled *The Voice of the Customer*.

Room 2 - Product Expectations. The product expectations are the company's metrics for measuring product performance. These metrics are preferably continuous, quantitative, technical attributes that can be measured on a continuous scale. Since these attributes are in engineering language, they are easier to design for than the qualitative voice of the customer (Clausing 1994).

Room 3 - Relationship Matrix. This room is the translation part of quality function deployment. In this room, the PDT matches the voice of the customer in room 1 to the product expectations in room 2. Usually, the voice of the customer does not translate directly to the expectations in a one to one relationship. One voice of the customer can be measured by several expectations; or one expectation can measure several voices of the customer. The degree of correlation is indicated in room 3, with the following symbols: ⊙ strong; ○ moderate; and Δ weak. Boxes with no symbols indicate a really weak or no correlation between the rows and columns.

After the matrix is completed, it is checked for fidelity. Every row and column should have at least one strong interaction. If a row does not have a strong interaction,

then the product expectations do not strongly measure that customer voice, leaving the potential for a dissatisfied customer. If a column does not have a strong interaction, then the company may be inefficiently allocating resources to improve metrics that do not strongly affect customer satisfaction (Clausing 1994).

Rooms 4 and 5 - Competitive Benchmarking. These rooms show how the product is positioned with respect to other products on the market. Room 4 shows the competitive evaluation from the customer's perspective. Room 5 is a quantitative evaluation from the corporation's perspective, is identified from engineering tests. Both evaluations are compared to verify that the corporation and its customers have the same understanding of how the product is positioned in the market. These evaluations are also used in part to develop target values for the product expectations (Clausing 1994).

Room 6 - Correlation Matrix. This room, also known as the attic, shows the relationships among the product expectations. In many situations, improving one expectation may enhance or degrade another expectation. Room 6 shows both positive and negative interactions: ⊕ strong positive; ○ weak positive; ✖ strong negative; and ✕ weak negative. These interactions show the PDT where most of the technical effort will be focused. A strong negative interaction shows where providing more satisfaction for the customer on one dimension will decrease satisfaction on the other dimension. Resolving these interactions requires streamlining the current technology or creating a new technology. This room is also used in part to develop target values for the product expectations (Clausing 1994).

Room 7 - Technical Importance. The overall technical importance summarizes the relative importance ratings of the product expectations with respect to the customer requirements. The importance rating is calculated by multiplying the customer importance ratings by the relationships in the relationship matrix (room 3).

$$I_j = \sum_{i=1}^n W_i \times R_{ij}$$

where I_j is the overall importance rating for column j , n is the number of rows, W_i is the importance rating for row i , and R_{ij} is the relationship between row i and column j . The relationship symbols have the following weightings: $\odot = 9$; $\circ = 3$, and $\Delta = 1$. Other weighting systems can be used. The large difference in weights is to explicitly distinguish between the those product characteristics that really satisfy a customer voice and those that only partially satisfy a customer voice.

Sometimes a high score is obtained because that column was completed in more detail than the other columns. This effect can be countered by making sure that the relationship matrix is completed at a consistent level of detail or by normalizing the scores with respect to the number of relationships in the column.

$$\tilde{I}_j = \sum_{i=1}^n \frac{1}{N_j} \times W_i \times R_{ij}$$

where \tilde{I}_j is the normalized importance rating for column j and N_j is the number of indicated relationships in each column.

The columns with higher importance ratings show a greater relationship to customer satisfaction. These columns are the metrics that the PDT should pay the most attention to when developing this product.

Another section of this room indicates the technical difficulty of moving closer to the ideal value for this characteristic. The evaluations are done on a 5 point scale, with 5 being the most difficult.

The importance ratings, difficulty assessment, and the correlations (in room 6) are used as aids in planning the subsequent work (Clausing 1994).

Room 8 - Targets, Quantified Expectations. Target values and tolerances for each product expectation are recorded in room 8. These target values come from a PDT assessment of competitive products in rooms 4 and 5, importance ratings calculated in room 7, and correlations indicated in room 6. The basic idea for this room is to set target values to create a best in its class product (Clausing 1994).

2.4. Pugh Concept Selection

The Pugh Concept Selection method is a qualitative tool that immerses the product development team into the criteria and the concepts to facilitate the creation of new concept ideas. The Pugh Concept Selection matrix is shown in Figure 2.5.







Concepts						
Criteria						
A	+	-	+	-	-	D
B	-	S	+	+	S	A
C	+	-	+	+	-	T
D	S	S	+	-	-	U
E	-	+	-	S	S	M
$\Sigma+$	2	1	4	2	0	
ΣS	1	2	0	1	2	
$\Sigma-$	2	2	1	2	3	

Figure 2.5 - Pugh Concept Selection Process (Pugh 1981)

The concepts are displayed across the top, in the columns of the matrix. The criteria are listed in the rows. The criteria usually come from the columns of the House of Quality. However, other criteria can be used. The criteria are subjectively ranked in order of importance, from top to bottom. The rankings are not given weightings. The goal of

the method is to think about the design in qualitative terms, not to engage in mind numbing numerology.

After constructing the matrix, the next step is to choose a datum. The datum should be the design that is considered by the PDT to be the best. The rest of the concepts are then individually compared against the datum using a 3 point scale. The concept is either better than (+), worse than (-), or the same as (s) the datum in satisfying a criterion.

Once the matrix is completed, two activities of attacking the negatives and enhancing the positives are used to generate better designs. Concepts that perform worse than the datum on certain criteria can be combined with concepts that perform better than the datum to create new concepts. Likewise, concepts that perform better than the datum can benefit from other concepts that perform better than the datum.

To aid in selecting the best concept, the number of pluses, minuses, and sames are totaled at the bottom of the matrix. This step is just a formality because after completing the matrix, the PDT generally knows which is the best concept.

The team started the matrix with several concepts. After completing the matrix, some of the concepts were dropped, but a few more were added. Another matrix is completed, with perhaps even different criteria. The next round also eliminates some concepts, but adds a few more. This iteration cycle continues until the PDT converges on one dominant concept (Figure 2.6) (Clausing 1994).

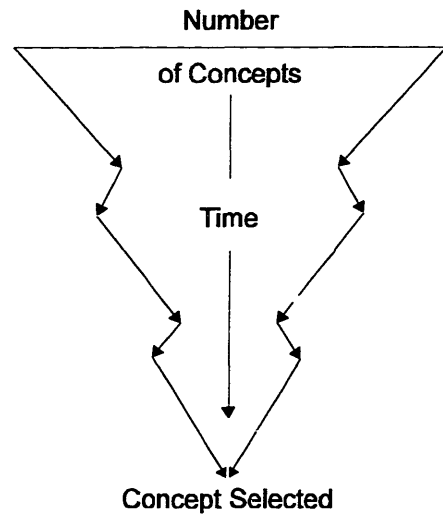


Figure 2.6 - Concept Iteration Cycle (Clausing 1994, fig. 3.9)

2.5. Enhanced Quality Function Deployment

Basic QFD is a serial process that deploys engineering characteristics from the total system to production. It is a relatively straight forward process, yet only works for simple, conceptually static designs. Enhanced Quality Function Deployment improves on these shortcomings of Basic QFD in two ways: (1) deployment through the levels and (2) the addition of Pugh Concept Selection.

The second phase of basic QFD translates total system expectations to piece part expectations. This is fine for small products, but not for products as large and complex as a car or a copier. Thus, as shown in Figure 2.7, EQFD translates expectations from the total system to sub-systems to piece parts. Each QFD matrix is essentially the same as described in Section 2.3, only the inputs and outputs have changed in abstraction level.

EQFD also integrates Pugh Concept Selection into the deployment from total system to piece parts to production. At every deployment, there is the opportunity to modify a product design. The concept selection matrix helps pick both hardware concepts and materials and processes to manufacture the hardware.

A product that uses the same concept for every product generation is conceptually static. Conversely, a product that changes every part and concept is 100% dynamic. In reality, a product generation that lies somewhere between the two extremes is best. The combination of the concept selection matrices and the explicit translation of requirements by QFD facilitates the development of appropriately dynamic products.

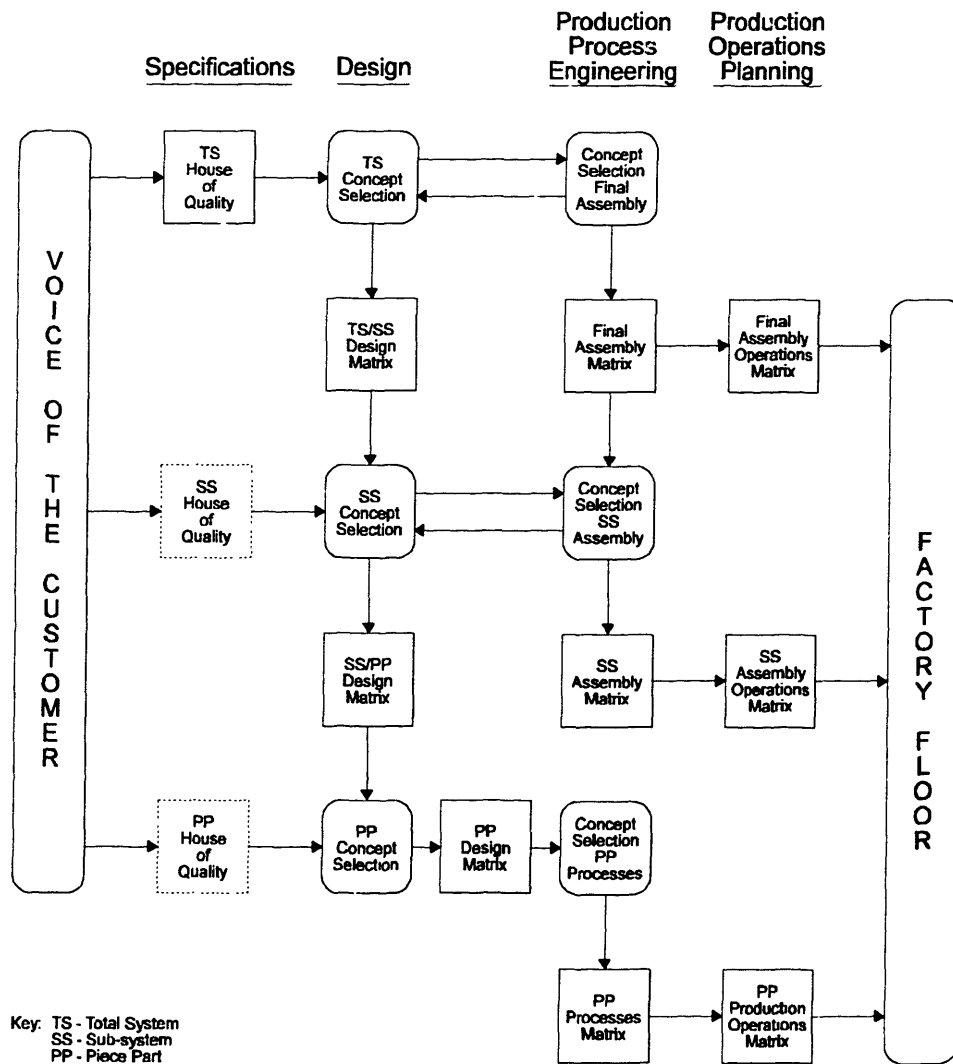


Figure 2.7 - Enhanced Quality Function Deployment
 (Sontow and Clausing 1993, modified from Clausing and Pugh 1991)

2.6. Taguchi System of Quality Engineering

Quality is measured in terms of how well a product satisfies the voice of the customer. In most situations, the customer's measure of quality is continuous. It does not stay zero until the product reaches the manufacturing specification limits and then skyrocket to infinity outside of those limits. Taguchi defines the customer's quality loss with a quadratic function, based on the deviation from the ideal value:

$$L(y) = k(y - m)^2, \text{ where } k = \frac{A_0}{\Delta_0^2}$$

This equation is also shown graphically in Figure 2.8. The customer quality loss increases as the product deviates from the ideal performance. It is not enough to mandate that the quality loss be reduced, because the tighter the manufacturing tolerance, the more costly the product is to produce. The sum of the quality loss cost and the manufacturing cost must be reduced to a minimal level.

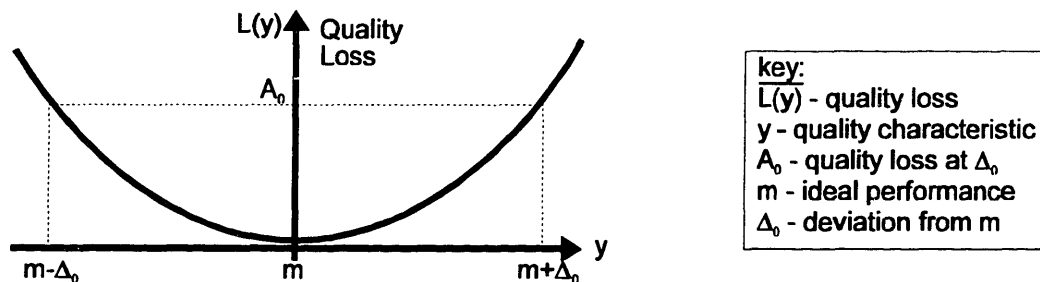


Figure 2.8 - Quality Loss Function (Phadke 1989, fig. 2.3)

There are three different types of quality characteristics: smaller is better, larger is better, and nominal is best. Figure 2.8 shows a nominal is best type of characteristic. The

other two characteristics are self-explanatory, where smaller is better strives for zero as a goal and larger is better moves toward infinity. There is another type of characteristic that is dynamic with respect to the input. For more information on characteristics, see Clausing (1994) and Phadke (1989).

The quality loss is a measure of the cost to society of a product that does not satisfy customer needs. The cost data is built into the constant k which is defined to incorporate all of the costs that are incurred after the product is sold to a customer. These costs can be a result of problems caused by an out of tolerance part, by environmental conditions, or by part deterioration.

There are four places where the company can affect the quality loss of a product. These places are shown in Figure 2.9.

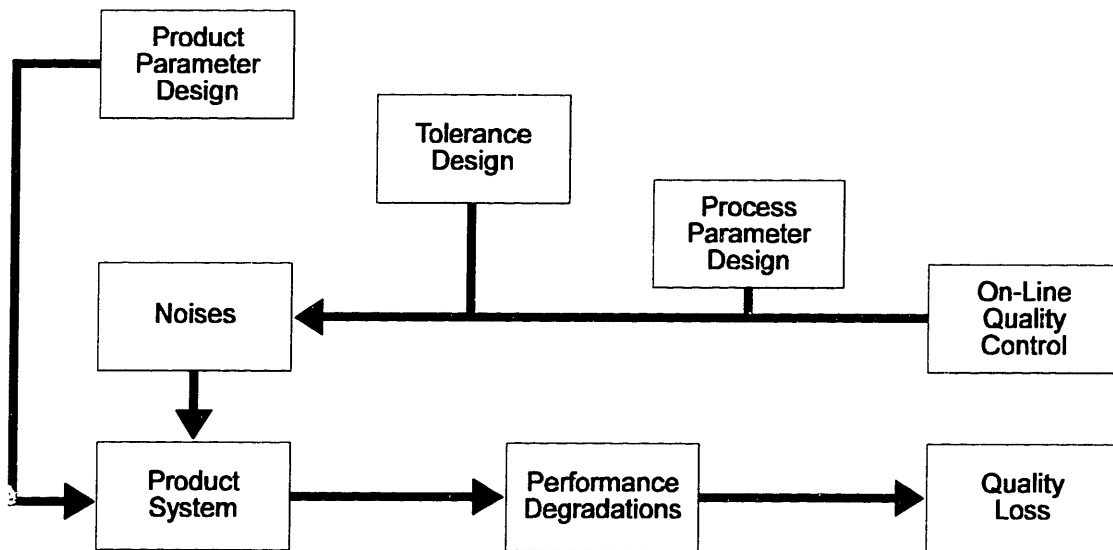


Figure 2.9 - Dr. Taguchi's System of Quality Engineering
(Clausing 1994, fig. 3.11)

Only one of these interaction points can help minimize product quality loss from all sources—the other three can only help minimize quality loss due to variations in

manufacturing processes. Once the product is shipped, the company cannot do anything to reduce the inherent quality loss (Taguchi 1986). Service operations will still affect the quality loss.

The quality loss should be measured against the actual use conditions of the product, not just the ideal laboratory conditions. Throughout its life, the product is subjected to a wide variety of noises, all of which try to degrade product performance and increase quality loss. A product that functions well in a wide variety of noises is said to be robust.

The first interaction point is in product parameter design. This event happens while the product is being designed, before it goes to the manufacturing floor. At this time, values for product parameters are selected that most enhance the successful fulfillment of the product's function. This can be done using design of experiments.

Clausing (1994) discusses seven steps for parameter design (see also Taguchi 1987):

- 1. Define objective; best functional metric.** A customer will evaluate a product on several different criteria. The most important criteria should be defined in a measurable way, on a continuous scale. These criteria, also known as quality characteristics, will be the metrics on which the design is evaluated.
- 2. Define feasible alternative design values.** In this step, the PDT selects several control factors that are believed to affect the objective. Control factors are design parameters, such as dimensions or material properties, that the PDT can change. Three feasible values for each control factor, which are neither too broad nor too narrow in range, are selected. The idea is to vary the control factors enough to get a feel for the design space in order to improve the design along the dimension of the quality characteristic.

3. **Select some options for evaluation.** Once the control factors and levels are defined, an experimentation plan for testing the product at the different factor levels needs to be selected. When optimizing a design, it has been common practice to change one control factor at a time and select the best value for each parameter. However, this technique does not model the interactions among control factors and can become time consuming if there are a large number of variables to change. Taguchi recommends changing multiple control factors at a time, following a design of experiments methodology.

		Control Factors				Noise Factors				
		1	2	3	4					
						1	1	2	2	1
						1	2	1	2	2
						1	2	2	1	3
Experiments	1	1	1	1	1	Results from Designed Experiments				SN Ratios
	2	1	2	2	2					
	3	1	3	3	3					
	4	2	1	2	3					
	5	2	2	3	1					
	6	2	3	1	2					
	7	3	1	3	2					
	8	3	2	1	3					
	9	3	3	2	1					

Figure 2.10 - Example of a Designed Experiment

Figure 2.10 shows a designed experiment, composed of fully crossed inner and outer arrays. The inner array, on the left side, is composed of four control factors, each with three different factor levels. The factors and levels are organized into nine different experiments. The '2' in the fourth row of column one means use the

second level of factor one in experiment number four. This inner array is called an $L_9(3^4)$ array because it has nine experiments, using four factors at three different levels.

4. **Impose noises.** Noises are changes in inputs to a product, or changes in environmental conditions in which the product is used, that are not controllable by the design engineers. A robust product is one that performs well despite the noises. To evaluate robust performance, the design team performs all of the inner array experiments against all of the noise combinations. In Figure 2.10, the outer, noise array is an $L_4(2^3)$ orthogonal array, with three factors at two levels. If experimental evaluation will be difficult, the noise array is usually replaced by two levels of a compound noise factor.

5. **Evaluate performance of selected options.** Every experimental design is measured with respect to the quality characteristic selected in step 1. To facilitate interpretation of the measurements, the data is transformed using signal to noise ratios. The signal to noise ratios are in the far right column of Figure 2.10. These ratios are specific to the type of quality characteristic used, that is, whether it is smaller is better, larger is better, or nominal is best:

$$\text{Smaller is better: } SN = -10 \log \frac{1}{n} \sum y^2$$

$$\text{Larger is better: } SN = -10 \log \frac{1}{n} \sum \frac{1}{y^2}$$

$$\text{Nominal is best: } SN = 10 \log \frac{\mu^2}{\sigma^2}$$

These are static SN ratios. In most cases a dynamic SN ratio will provide the greatest benefit (see Phadke 1989, 114).

Once the overall signal to noise ratios have been calculated, it is easy to calculate the signal to noise ratio for each design parameter. In Figure 2.10, each of the three levels of each the control factor was tested three times in the design. The SN ratio for each factor level is the average of the overall signal to noise ratios at that level. For example, the SN ratio for factor 2, level 2 is the average of the overall SN ratios for experiments 2, 5, and 8.

- 6. Select best design values.** The signal to noise ratios are defined in such a way that the largest SN ratio represents the best value for the control factor. Thus, the best design is the design where all of control factors are set to the values with the maximum signal to noise ratios.

- 7. Confirm robust performance.** In Figure 2.10, only 9 of the 81 possible design configurations were tested. This means that it is unlikely that the best design of the 9 already tested is actually the best overall design. This step is to confirm that the experimentally determined best design is actually the best design for the product. If everything goes well, the predicted signal to noise ratio for the best product should be close to the actual ratio determined in this confirmatory experiment.

The ideas presented in the rest of this section are not essential to understanding the QCAD methodology, but are briefly explained here for completeness. More information is available in Clausing (1994).

After product parameter design, the next place where the corporation can intervene to prevent quality loss is in tolerance design (Figure 2.9). The goal of tolerance design is to select the most economical precision level for the dimensions on critical parts.

The idea here is not to select the tolerance levels for the product, but to select a manufacturing process that is precise enough to minimize the total cost.

Usually, there is a higher manufacturing cost associated with tighter tolerances and in an effort to reduce costs, the company will want to loosen tolerances. The quality loss function, pictured in Figure 2.8, provides a way to quantitatively measure the costs associated with tighter and looser tolerances. Tighter manufacturing tolerances will cost \$x based on quantifiable manufacturing costs, while looser design tolerances will cost \$y based on the quality loss function. The production process is selected that minimizes both the manufacturing cost and the quality loss cost.

With the manufacturing process selected, the operating conditions for the process should be optimized to reduce manufacturing variation as much as possible. The process parameter design methodology, used in this interaction point is much the same as the product parameter design described above. A quality characteristic is selected; several control factors and their levels are determined; a designed experiment is performed; signal to noise ratios are calculated; and the optimal parameters are tested in a trial run. One main difference in the two methods is that product parameter design imposes noises on the experiments, while no noises are intentionally imposed in process design. It is assumed that the process is being tested under normal operating conditions and that any noises that would exist in the manufacturing environment are automatically present during experimentation.

The last interaction point for the company to reduce manufacturing cost and quality loss is in on-line quality control. Even if a process has been optimized using process parameter design, there will still be some variation in the output dimensions. The company makes a trade off between the cost of intervention and the quality loss of deviating from the target. Too much intervention has costs in terms of time and intervention. Too little intervention has costs in terms of customer quality loss. Trading off these two costs helps a company determine the least costly intervention plan.

Chapter 3

Review of Computers in Design

3.1. Overview

“There are a number of strands evident in current design automation research. These include attempts at understanding the design process itself, attempts at automating routine redesign, attempts at integrating knowledge of past designs and approaches towards the development of design assistants to work with the human design expert. We believe that attempts at automating the whole of the design process are futile, given the current level of technology. In fact, the benefit of ever doing this, we believe, is dubious (Hunt and Lee 1992)”

Current trends in the design process were discussed in Chapter 2, with primary emphasis on describing the total quality development process (Clausing 1994). TQD is a prescriptive design method to help designers: (1) create products that satisfy customer wants and needs, (2) benchmark with the best products in the industry, and (3) improve product robustness, quality, cost, and delivery.

The hypothesis is that computers are very efficient at performing quantitative tasks. They can be used to assist in and perform the detailed design work, to free time for engineers to concentrate on the up front conceptual work.

This chapter describes trends in the use of computers for product development. Of particular interest is computer software that helps designers use quality methods. The sections below are not meant to be mutually exclusive or collectively exhaustive. They are meant to show some of the ideas that went into developing QCAD.

The first section presents an introduction of knowledge based engineering, defining what it is and how it is used in design. This section is meant to put the rest of the chapter into perspective with respect to computer based design programs. Following that brief introduction, the next two sections take a step back to look at two types of systems used in design: (1) Conventional design assistants which have been around for years and are

still prominent in design, and (2) Design history systems which are somewhat in use, but are primarily still being developed. Section four combines these two ideas to describe several KBE systems, including the ICAD brand of KBE. The last section summarizes how the computer design concepts described in this chapter impact the understanding of what a computerized quality design system should do.

3.2. Introduction to Knowledge Based Engineering

Before discussing knowledge based engineering, some background in artificial intelligence is necessary. Artificial intelligence (AI) is concerned with modeling human thought processes (Edwards 1991, 2). AI has been divided into several areas: knowledge based systems, natural language understanding, pattern recognition, intelligent computer-assisted learning, speech recognition, and models of human cognition (see Figure 3.1).

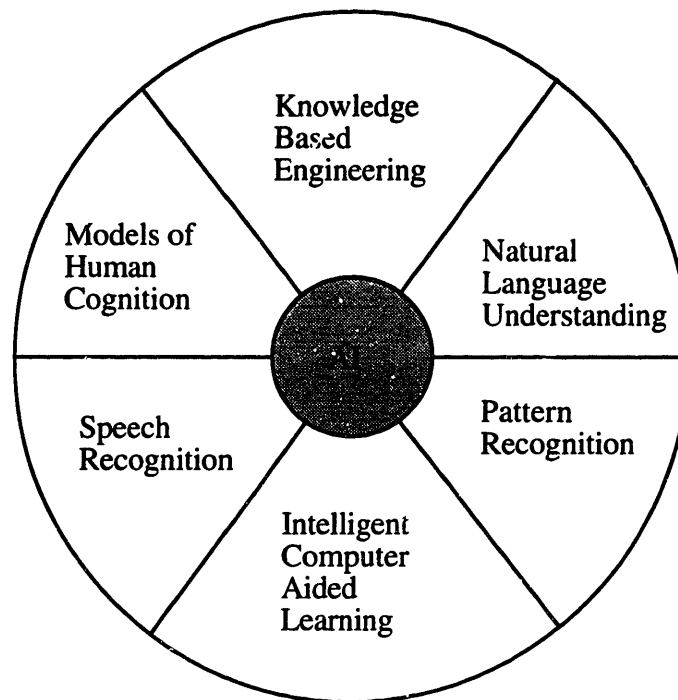


Figure 3.1 - Six Areas of AI (Edwards 1991, 6)

Knowledge based engineering (KBE) involves using computer systems to support or to automate tasks usually performed by human experts. These human experts use interpersonal skills, expertise and judgment, that has been acquired over several years in carrying out these tasks (Edwards 1991, 6).

Knowledge based engineering (KBE) in general is the use of knowledge, be it stored in a computer, a handbook, or someone's brain, to solve a problem. Problems can range in scope from adding two numbers to determining the piece part design from higher level design inputs to learning how to perform a task. John Edwards (1991) has defined six roles of KBE systems:

- *assistant* - help perform part of a larger task
- *critic* - reviews previously performed work
- *second opinion* - compares task results with the user
- *expert consultant* - offers advice based on input information
- *tutor* - trains the user to perform a task
- *automaton* - completes tasks independently and reports results

In product development, KBE systems have most commonly been used as design assistants. Design assistants can be used to perform a small set of tasks for a designer, such as helping to select material from a database, or can be used to perform a large task as in configuring an entire vending machine (Edwards 1991).

Expert consultant type systems are also common in product development work. An expert consultant, also known as an expert system, provides guidance in solving a problem. One example is a system that selects a machine tool after asking the designer a series of questions (Traughber 1986).

Critic systems can also be used in product development, however their use is not wide spread. One system currently being developed critiques designs from various points of view. The central idea of the model is to create a roundtable design review, where various experts discuss the proposed design and suggest improvements (Ishii and Krizan 1991).

The other three KBE roles are less common in product development tasks. The second opinion type systems examine a situation using different techniques or by using more information, thereby providing a second reference for decisions. An example of a second opinion system is used in stock market analysis. An analyst can look at a few leading indicators to make decisions, but computers can look at hundreds of indicators to make a decision.

The tutor role uses the computer to teach about an idea, and as such, can help train new designers, but is not meant to help develop products.

Lastly, the automaton system completely automates tasks, such that the task is automatically completed, with only a report left for the user to see. This is analogous to a coordinate measuring machine on a manufacturing line that just measures dimensions and outputs a report of the dimensions, perhaps emitting a signal when a part is out of tolerance (Edwards 1991).

3.3. Conventional Design Assistants

Design assistants are computer programs that perform a specific task as part of a larger task. The size of a design assistant's task can vary from performing a short set of calculations to developing a finite element mesh and recording a stress profile. In the conventional assistant programs, the designer knows the type of analysis to be performed and uses the program to run the calculations.

The design assistants discussed in this section do rely on some type of knowledge. However, the difference between the design assistants discussed in this section and the design assistants discussed in Section 3.5 pertains to the degree which the user can modify the design reasoning in the program. In this section, the user is limited to performing the analysis specified by the program. The user is free to choose a different program that better suits the analysis, or the user can change the inputs to the program, but the user cannot change the nature of the analysis.

This section is split into two parts: one describing technical, or engineering related, design assistants and the other describing quality related design assistants.

3.3.1. Technically Focused Assistants

Finite element analysis (FEA) is a design assistant type program. It performs, for example, stress or heat transfer analysis on a specified geometry. A user inputs a geometry, material properties, and boundary conditions. The program outputs stress contour lines, which the user analyzes to identify a new geometry design to minimize stresses (Swanson Analysis Systems 1993, Algor 1993). In this case, the FEA program performs a specific task, which is part of the larger task of specifying a component geometry.

There is an abundance of design assistant software available. For example, Mechanical Engineering magazine has an electronic bulletin board with over 4000 programs to assist mechanical engineering designers. These programs cover many facets of mechanical engineering work, from material selection to dynamics analysis to pressure vessel design (Jackson 1992).

Computer aided design (CAD) systems can also be thought of as design assistants. They store and manipulate geometric data. Designers can use the systems to specify parts and to check tolerances, without the expense of creating a costly prototype. In recent years, CAD systems have expanded their capabilities to include, electrical circuit analysis, and computer aided manufacturing modules (Cadence 1993; Head, Pietra, and Segal 1989; Parametric Technology 1993). There is more discussion of CAD design systems in Section 3.5.

Design assistants don't have to be programs that calculate complex engineering formulas. They can also be spreadsheets programmed to create charts to calculate the weight of an assembly, given different material properties. A designer can then use the information in the chart to identify an appropriate material.

3.3.2. Quality Focused Assistants

This section describes a set of design assistants that are not as technically oriented as those discussed in the previous section. These systems are less concerned with number crunching and creating a design, but are more concerned with manipulating numeric or text data to facilitate the use of quality design methods, such as those discussed in Chapter 2.

One commercially available design assistant helps with design of experiments. The Taguchi Expert software, by Quality Engineering Services (1992) helps designers select design arrays and compute signal to noise ratios. This software asks for user inputs and then leaves it up to the designer to evaluate the outputs.

Several systems are available to help a product development team use QFD matrices. For example, ITI markets a system called QFD/Capture, that helps designers store QFD matrices, calculate the importances, and generate reports (ITI 1992).

At Lamar University in Texas, Sriraman, Tosirisuk, and Chu (1990) programmed a series of QFD matrices from the voice of the customer through to process variables, using object oriented programming techniques. The idea behind the system is to create a compact database that relates customer needs to production quality control, to support manufacturing control activities.

Ford Motor Company has developed the Technical Information Engineering System (TIES), a computer program that propagates information by linking several layers of QFD charts together to facilitate total vehicle design (Vora, Jackson, and Veres 1989). This system helps maintain the integrity of the data base, while the designers use the program output to help conceptually design a product.

3.4. Design History Systems

The design assistant systems discussed in the last section, perform specialized design tasks to free the engineer's time for other duties. A design history system is meant to record the intent behind a design. Many decisions are made in the design process without engineering equations or decision matrices. The design history systems store the designs and the rationale behind them at the moment decisions are made. "A record of the design process may be used for: 1) verifying the design, 2) explaining the design, 3) determining the effect of modifications to the design, and 4) reusing all or part of a design in a different context (Ganeshan, Finger, and Garrett 1991)."

Several researchers are investigating design history systems (Bañares-Alcántara 1991; Chen et. al. 1990; Garcia and Howard 1991). Kuffner and Ullman (1990) are researching what information mechanical designers want when evaluating a design. In their study, three professional, mechanical design engineers were given the initial design specifications and a complete set of engineering drawings. The designers were then asked to modify the designs. The result from the exploratory study showed that 47% of the questions asked were concerned with how to fabricate a part, 22% with how one part is positioned with respect to another part, 20% with how a part works, and 11% with the purpose of a part. Kuffner and Ullman believe that an intelligent CAD tool would provide the best platform to store this additional design information (Kuffner and Ullman 1990).

3.4.1. Design Representation Formats

Design history systems are not only concerned with capturing and retrieving design information, but also with storing the information. Designs can be represented in terms of form or function (Finger and Dixon 1989).

Conventional CAD systems model geometric data. They use wire frame models to represent the shape, dimensioning, and positioning of a part. This information on its own, does not say much about a design, since a cylinder can represent either a hole or a knob.

Solid modelers provide more clarity of information, as the computer represents the graphic objects as solid pieces. If a cylinder is a hole, then it is an empty space in the model. However, representing geometry doesn't capture all of the design information (Finger and Dixon 1989).

The other approach, based on part behavior, describes the representation of design functions (Finger and Dixon 1989). Several techniques are available to represent functional behaviors: Pahl and Beitz (1984) have a function structure method, the U.S. Air Force has an IDEF methodology (United States Congress 1981), and Paynter (1960) has a bond graph representation.

Feature representations are an attempt to combine the two types of representations. Dixon (1988) defines a feature as an entity with both form and function. Most of the research on feature systems has been in the area of design and manufacturing (Finger and Dixon 1989). One system has been developed by Drake and Sela (1989) to use features to evaluate the manufacturability of a design.

Features represent some information about a design. Product models, incorporate even more information. Eastman (1981) said that computers should represent the semantic, geometric, and hierarchical features of a design. The term semantic refers to the consistency of information in a design. One example of semantic integrity, asks if the moment of inertia of a simply supported beam is large enough to carry the desired load? The reference to geometry is self explanatory, in that it refers to the physical characteristics of the product. Lastly, the term hierarchical, refers to a multi-layered representation of the relationships in a product. Eastman's grouping of information has now become known as a product model (Finger and Dixon 1989).

3.5. Knowledge Based Design Systems

Section 3.3 described technical and quality focused design assistants that help the designer with specific tasks. Section 3.4 described design representation systems which

model product information as it is being designed. This section takes more of a systems perspective on the design process by describing several knowledge based applications in terms of design representation methods and design assistant applications. The systems described in this section also have the common theme that they perform activities which normally require human expertise (Edwards 1991). Expertise, or knowledge, is different from data, in that data refers to information and knowledge refers to an understanding of that information (Ramamoorthy and Wah 1989).

The systems discussed in this section add knowledge, not just data, in the form of human expertise, or design intent, to the computerized design task. CAD systems model geometry. However, new programming capabilities enable designers to automate design tasks and incorporate some design knowledge. The ICAD system provides a programming language that allows designers to incorporate even more automation and design knowledge into a product model. Quality design systems provide yet another method to model design knowledge.

3.5.1. CAD Systems

CAD systems play a prominent role in product design and development. As described in Section 3.3.1, they are intrinsically design assistants, but they also serve as a central hub to interact with various design assistant programs. Several CAD systems are now equipped with programming languages to further increase CAD's role in the development process. Increasingly, CAD systems are being programmed to build product models that incorporate both geometric and non-geometric data into a design. It is this programming capability that creates the possibility for the user to code knowledge and thusly design intent into the model. Currently, there are two types of CAD design systems that are predominately available: parametric and variational (CIME 1989).

Parametric design systems link geometric features to design rules. These rules act like constraints on the design, such that when a part or dimension is changed, the

associated parts or dimensions are also changed. For example, if the length of a screw depends on the thickness of a plate, then as the plate thickness increases, the screw length increases. All of the design rules must be defined before using the system and there is little flexibility to use different rule sets while the system is running. As such, parametric systems are best suited for well understood designs (Chung and Schussel 1990).

Variational design systems also link geometric features to design rules. However, the product model does not have to be fully dimensioned before using the system. A designer can investigate the product model while it is not fully constrained. The advantage of this system is that the designer can change the nature of the problem in the middle of exploring a design. For example, a designer can look at the range of motion of a robot arm in one instant. While in the next instant, the designer can see the motion the robot arm makes to move along a specified trajectory (Chung and Schussel 1990).

3.5.2. ICAD

The previous section described CAD tools with a parametric or variational programming language. ICAD is a parametric programming language with a CAD interface. However, since ICAD has a programming language, the language can be used to write a constraint based design approach similar to variational geometry.

This section describes the ICAD system and its applications. Since this thesis uses the ICAD knowledge system to implement QCAD, an understanding of ICAD's applications and program structure will facilitate comprehension of the QCAD concept and implementation.

ICAD is the leading supplier of knowledge based engineering systems worldwide. The ICAD System™ is best used for design tasks that are similar, but different, which includes both products that require several iterations to optimize a design, and product platforms that are designed using the same set of rules.

In most companies, a design is specified based on years of experience with creating a particular product or process. In ICAD, this knowledge is stored in a product model, which contains a hierarchical set of rules to describe the product's geometry and design intent. The hierarchy is organized as a product structure tree, and generally breaks the product down into components. Rules, which are composed of attributes and equations, are embedded at every level of the tree. The attributes are variables that hold information, like part cost, yield stress, and material type. The equations can be engineering formulas or design heuristics (rules of thumb), which relate the attributes to the final part specification.

The rules in the product model can also link external programs, such as CAD systems, FEA programs, and design assistants, to the product model. The product model can supply inputs to or analyze outputs from those other programs to optimize a design. By linking corporate design practices, including both design rules and design programs, to the part geometry, ICAD provides an environment to capture the design intent of a product.

3.5.2.1. Representative ICAD Projects

The ICAD system has been used effectively in the following design situations: semi-custom, one-of-a-kind, generative tooling, generative process planning, and product configuration. A few representative projects as described in an ICAD brochure (ICAD 1992) are as follows:

Semi-Custom Products. There is a specific design process for boilers that rarely changes. However, a new boiler design to meet specific needs can take between 500 and 2,000 hours. ICAD is used to automate the boiler design process, including the design of several boiler components, the selection of several components from a catalog, and the generation of production intent drawings and a cost estimate; all of which are based on engineering rules.

One-of-a-Kind Products. Satellite antenna design is a highly iterative process with tradeoffs being made among weight, reliability, and volume. Engineers use an ICAD product model to prototype the antenna design such that design specifications can be changed to see how the overall design is affected.

Generative Tooling. A new injection molded part requires a customized mold base. The ICAD model is used to select and customize a standardized mold base using design rules, when given part geometries and production volumes.

Generative Process Planning. ICAD is used by the U.S. Navy in the RAMP (Rapid Acquisition of Manufactured Parts) project to create a manufacturing process plan, including operation sheets, from a standardized part description.

Product Configuration. In this situation, ICAD is used by salespeople to design products given customer requirements. The system prompts for specific inputs and outputs a fully specified product.

3.5.3. Quality Design Systems

Quality design systems are distinct from the set of quality design assistants, because they store information through the use of quality design methods for use in subsequent designs.

Several researchers have created, or are currently researching, quality focused design systems. Two quality design systems have already been completed at MIT. John Berg and Tom Knight, in two separate projects, programmed the total quality development process, including quality function deployment, Pugh concept selection, and robust design on a computer. John Berg programmed the system using the C language, while Tom Knight programmed the system on a Macintosh, using Hypercard. Both systems guide the designer through the use of the design process. The designer is able to record design information in the system. If the user gets stuck with how to perform a

certain part of the design process, the computer displays a help screen with relevant information (Berg 1988; Knight 1990).

Two other quality systems that are still in the design stage, both use QFD as a backbone. One is called Design Function Deployment (DFD) and is being pursued at City University in London. The DFD system is a shell system that stores design information and coordinates sets of design assistant modules. The design model in DFD has five milestones: (1) establish customer requirements, (2) generate different conceptual solutions, (3) develop detailed design of a few sub-systems, (4) define material and manufacturing processes for the design, and (5) establish production plans for each viable product. To progress through the milestones, the user can fill out a QFD matrix, look at a morphological chart, experiment with a solid geometric model, access a set of databases, or use a variety of design assistant tools, including finite element software and design for manufacturability analyzers (Jebb, et. al. 1992; Jebb, Sivaloganathan, and Evbuomwan 1993).

Another design system that is still under development does not have a name, but is referred to as a recursive model for the design process and is being pursued at the University of Michigan. The model contends that design is actually recursive and not iterative and that designers reason with sets of design options, not just one option at a time. The design model has designers perform three basic operations on a design to help explore and generate a solution for the design problem. The first type of operation involves setting up the problem. The second type of operation involves breaking the problem into parts. The third operation involves recombining the pieces after the problem has been broken apart. The recursive model proposes to use QFD, Taguchi methods, and design for methodologies to help with these operations. The system is intended to help designers and design teams document and organize the design process. It may also be possible to use the system to coordinate islands of automation (Ward 1990).

3.6. Summary

This chapter described three different ways computers are used in design in order to show some of the ideas that went into developing QCAD. The chapter started by describing design assistants that help with small, well defined portions of a design task. A designer, using one of the many thousands of available programs, enters specific information about the problem and asks the computer to calculate an answer. It is up to the user to understand and apply the results to enhance the design.

Next, the chapter discussed a general theory of design history systems and explored design representation formats. Neither design history systems nor design representation formats calculate anything about a design. They store information about the characteristics of the design. A design history system would store information about the steps a designer took to develop a product and why certain decisions were made along the way. A design representation format stores information about the design, such as the product function and geometry.

In the final section, the chapter described knowledge based engineering design systems, which both store a representation of the product and perform calculations to create or enhance a design. Since the product is represented in the computer, the procedures can be linked to different areas of the product model. When information is changed in one part of the model, the appropriate procedures process the information and change the design accordingly.

Mixed throughout the discussions in this chapter were descriptions of computer design systems that use the quality methods from Chapter 2. The quality design systems use the computer to not only crunch numbers, but also to store, retrieve, and organize information. Even though quality design systems were only discussed in the design assistant and knowledge based engineering sections, quality systems could also be classified as design history systems, because some of the quality methods provide a format for representing design intent.

This thesis is a cooperative project with ICAD. As such, the ICAD system was taken as a starting point for research. ICAD is a knowledge based design system that contains both a representation of the product and procedures to effect the design of the product. The discussion in Section 3.5.2 shows that the ICAD system currently has many applications in product development, ranging from one-of-a-kind designs to routine product configuration designs. Although one manufacturing related design task was mentioned, most of the ICAD applications concentrate on product geometry.

The quality design systems described in Section 3.5.3 are knowledge based engineering systems, in that they both store and manipulate product information. However, the quality design systems use quality structures to model product data, instead of geometry as used in ICAD. In general, the quality design systems use QFD to deploy and manipulate product information.

QFD takes a horizontal approach to product development, uniting customer needs with the factory floor. ICAD takes a vertical approach to product development, mainly concentrating on the hardware specification. One evidence of this is seen in the rigid product structure tree that ICAD uses to model a product (ICAD 1993). All of the design information feeds down from top to bottom. The tree structure facilitates the detailed hardware design, but limits the horizontal sharing of information.

QCAD combines a design methodology, like those used in the quality design systems with the processing capabilities of a knowledge based design system. The QCAD system also has a design representation that models a product from the total system voice of the customer to the part feature process specification. The QCAD methodology is flexible such that it is compatible with whatever quality methods or design assistant programs (see Section 3.3) the company already uses for design.

To get a better feel for QCAD, imagine a box with several holes in its sides. The product being modeled is inside the box. Holes are punched in the sides of the box to allow someone outside the box to see in. Each hole shows a different perspective on the

object inside. One hole shows a piece part manufacturing process, while another hole shows a total system functional requirement. A third hole shows a solid model of the product geometry. QCAD has several more views (see Section 4.3), each with a unique perspective on the product.

In addition to just looking at the product, the user can manipulate the product model. By providing different information at each hole in the box, the user can alter the design. The product representation in the box can combine knowledge based design rules with design assistant programs, to respond to the user's inputs. The information is stored in the system according to a structured design methodology (Section 4.5). The user, follows a similar methodology (Section 4.4) when supplying inputs to the system. Two examples of the QCAD system in action are discussed in Section 5.2.

Chapter 4

QCAD Definition

4.1. Overview

QCAD, Quality Computer Aided Design, is a method for linking quality focused design methods with computerized design software. It provides the capability to translate the voice of the customer to the factory floor. To facilitate this deployment, QCAD has two parts: (1) the design process used by a multi-functional product development team, and (2) the computer program used with that design process. The QCAD structure is specifically amorphous, such that it can be tailored to each application. It is up to the PDT to determine how to use the system to its best advantage.

The first section below describes the development of the QCAD concept. The next section explains the basic organizational framework that QCAD uses to model a product. The third section briefly explains the sequence of design activities used with QCAD. The design activity is a modified version of total quality development (Chapter 2), using additional design tools. The design process is the set of activities that the PDT performs without the assistance of the computer; although the computer may be used to retrieve information to aid the PDT in using the design tools. The last section explains the design process modules used in the computer portion of QCAD and describes how the modules are interrelated to create a computerized product model.

This chapter describes the QCAD theory, without mentioning specific details of the computer implementation. Chapter 5 explains the computer implementation of QCAD in the ICAD Design Language, using two case examples. The ICAD software and the product model are inseparable in QCAD and thus are explained together.

4.2. Initial Development

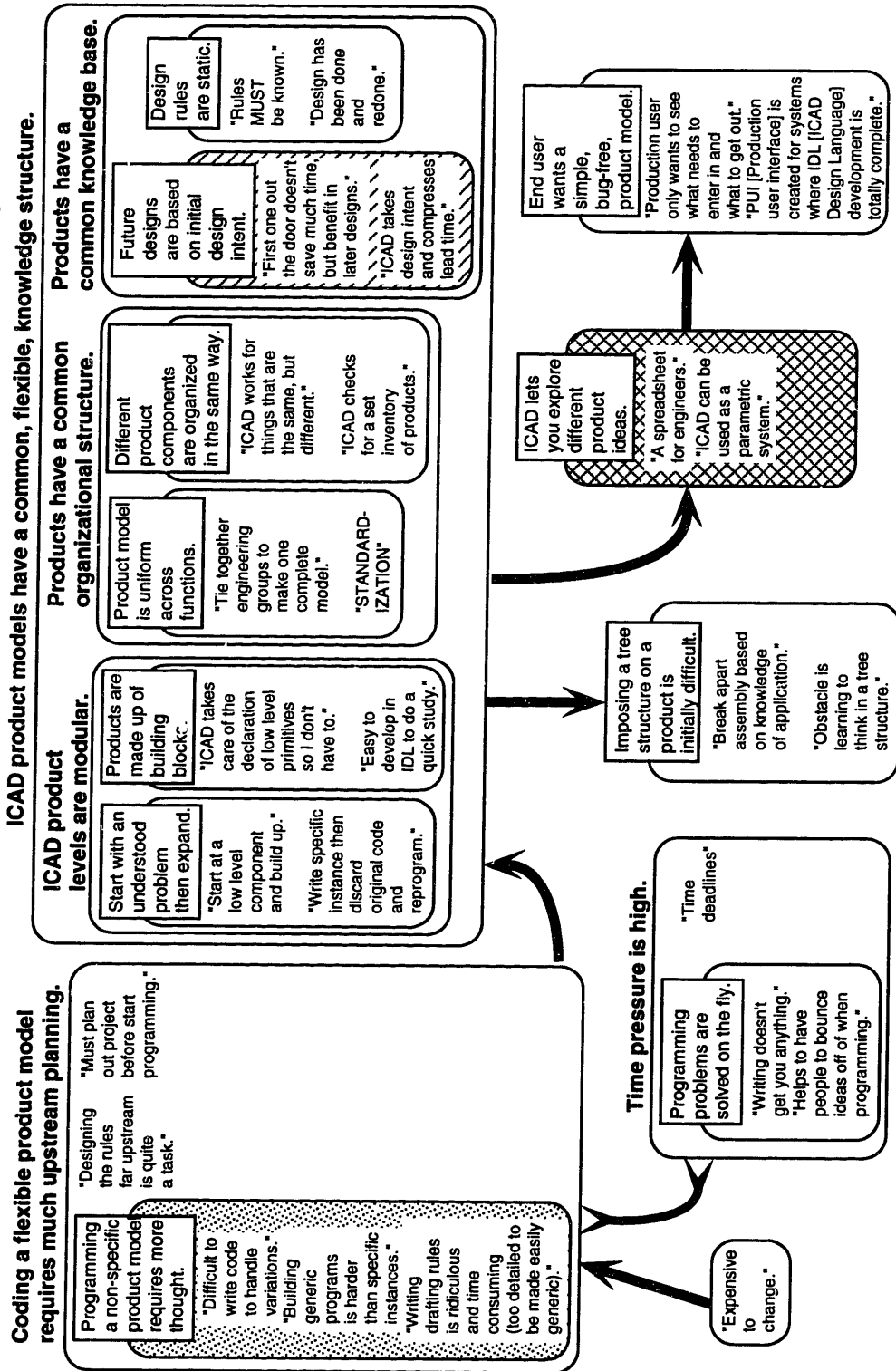
The literature search gave some information on what types of computer design systems are being used or are being researched today. However, only a few articles were found that discussed a designer's reactions to computer design systems (Ishii and Hornberger 1992). Since this thesis is a cooperative project with ICAD, interviews with some ICAD customers and employees supplied information about how ICAD's knowledge based system is used for product development.

Figure 4.1 shows the KJ, affinity diagram (Shiba, Graham, and Walden 1993) describing ICAD's use in design, based on the interviews. The KJ shows connections between different parts of a topic (See the question in the top right corner of Figure 4.1). The bottom level sentences are direct quotes from the interviews. Some of the more interesting quotes describe using ICAD to tie engineering groups together, mention that the first design out the door doesn't save much time, but the benefit is in later designs, and remark that it is difficult to write ICAD code to handle product variations.

Looking from left to right on the KJ diagram in Figure 4.1 shows the basic flow of how an ICAD program, or product model, is constructed. First, the product or sub-system to be modeled is selected and some of the details are sketched out. If the IDL programmer is not familiar with the product, this first stage includes a series of interviews to understand what knowledge is required for the product model. The second step involves programming and debugging the product model, including the specification of part geometries and coding the knowledge base. Lastly, there are two different ways to use the ICAD system, one is in programmer mode, the other is in end-user mode. The programmer mode provides a great deal of flexibility for the designer, as the designer can change the underlying product model while experimenting in the browser. The end-user mode is a little more constrained in that the user is expected to answer questions about the product model using a pre-programmed set of questions. The user does not change the programming in the product model.

What are the images of ICAD being used for product development?

With proper planning, ICAD provides a way to explore variations on a product.



May 20, 1993
MIT
Yale Goldis

Figure 4.1 - KJ of ICAD Use Environment

The final answer to the KJ was, “With proper planning, ICAD provides a way to explore variations on a product.” This raises the question about what constitutes proper planning. Proper planning includes finding and organizing the information that goes into the model. Thus, QCAD provides a framework, based on total quality development, for planning and programming the product model for use in the browser.

A little more intuition about how this could be achieved was discovered by using the House of Quality and Pugh Concept Selection. The rows of the House of Quality, shown in Figure 4.2a, come from a combination of the philosophy of TQD and from the more important interview voices about using ICAD. The rows were then translated to metrics for the columns, which are shown in Figure 4.2b. The completed House of Quality is shown in Figure 4.2c. Completing the House of Quality matrices gave some insight as to how the design process and computers could be used synergistically. ICAD is good at storing information, but TQD is good at organizing how the information should be stored. Additionally, the ICAD software is demand driven, which can be used to bring the relevant information into consideration when a decision is to be made.

The columns from the House of Quality were then used in a Pugh Concept Selection matrix, as shown in Figure 4.3. The rows in this matrix are the columns from the House of Quality. The matrix started with 8 different concepts, but the eighth concept changed dramatically over the course of completing the matrix. The eighth concept came to encompass all of the good features of the other ideas, while minimizing the bad features. The end result is that QCAD provides design tools and a way to organize those tools such that a variety of products can be modeled in the system.

Figure 4.2a - Rows for the House of Quality for QCAD³

Process Focuses on Meeting Expectations
 focus on robust performance
 Looks at Current Environment
 emphasis on competitive benchmarking
 emphasis on customer satisfaction

Resources are Pooled to Design a Product
 Information is Coordinated to Focus on the Decision to be Made
 bring relevant information to bear at an opportune time
 consider related decisions together
 production user should only see relevant information

Embodied Knowledge Works to Achieve One Goal
 strong commitment to decisions
 teamwork

Method to Attack Problem is Structured
 start with specific rules
 start with a small sub-system
 design process is structured

Products are Described Flexibly
 defparts are generic
 rules are easily generalizable
 designs are broken into building blocks

Tools Match the Product
 provide a collection of tools
 the program should allow the PUI to be easily added

Products are Amenable to the ICAD Environment
 Product Has a Simple Structure
 nodes have 2 - 10 leaves
 design can be broken into a tree structure

Products Are Variations on a Design
 different designs should have similar geometries
 design rules must be known
 product model must be demand driven

³ In Figures 4.2a-c, the row and column formatting represents a tree hierarchy.

- Every indention on a line represents a change in abstraction level. The lines that start farthest to the left are at the highest level of abstraction.
- The lines that start with upper case letters represent categories.
- The lines that start with lower case letters are the lowest level entries under each category.

Figure 4.2b - Columns (metrics) for the House of Quality for QCAD³

- Degree Fidelity with the VoC
 - Taguchi methods included
 - External Input is Gathered
 - degree of competitive benchmarking
 - degree VoC is translated through system
- Resources are Combined
 - Information is Coordinated for Decisions
 - relevant information is gathered for decisions
 - related decisions are brought together
 - production user only sees relevant information
 - Embodied Knowledge Works to Achieve One Goal
 - strong commitment to decisions
 - teamwork
- Structured Method
 - start with specific rules
 - start with a small sub-system
 - structured process
- Flexible Product Descriptions
 - generic defparts
 - generalizable rules
 - number of building blocks in design
- Tools Match the Product
 - number of tools
 - ease which PUI can be added
- Products Fit with ICAD
 - Products Have a Simple Structure
 - average number of branches/node
 - design fits a tree structure
 - Different Products have the Same Design Intent
 - designs have similar geometries
 - design of uncertainty in design rules
 - product model is demand driven

Correlation Matrix Legend

Strong Positive	●	9
Positive	○	3
Negative	×	3
Strong Negative	⊗	9

Reliability Matrix Legend

Strong	●	9
Moderate	○	3
Weak	△	1

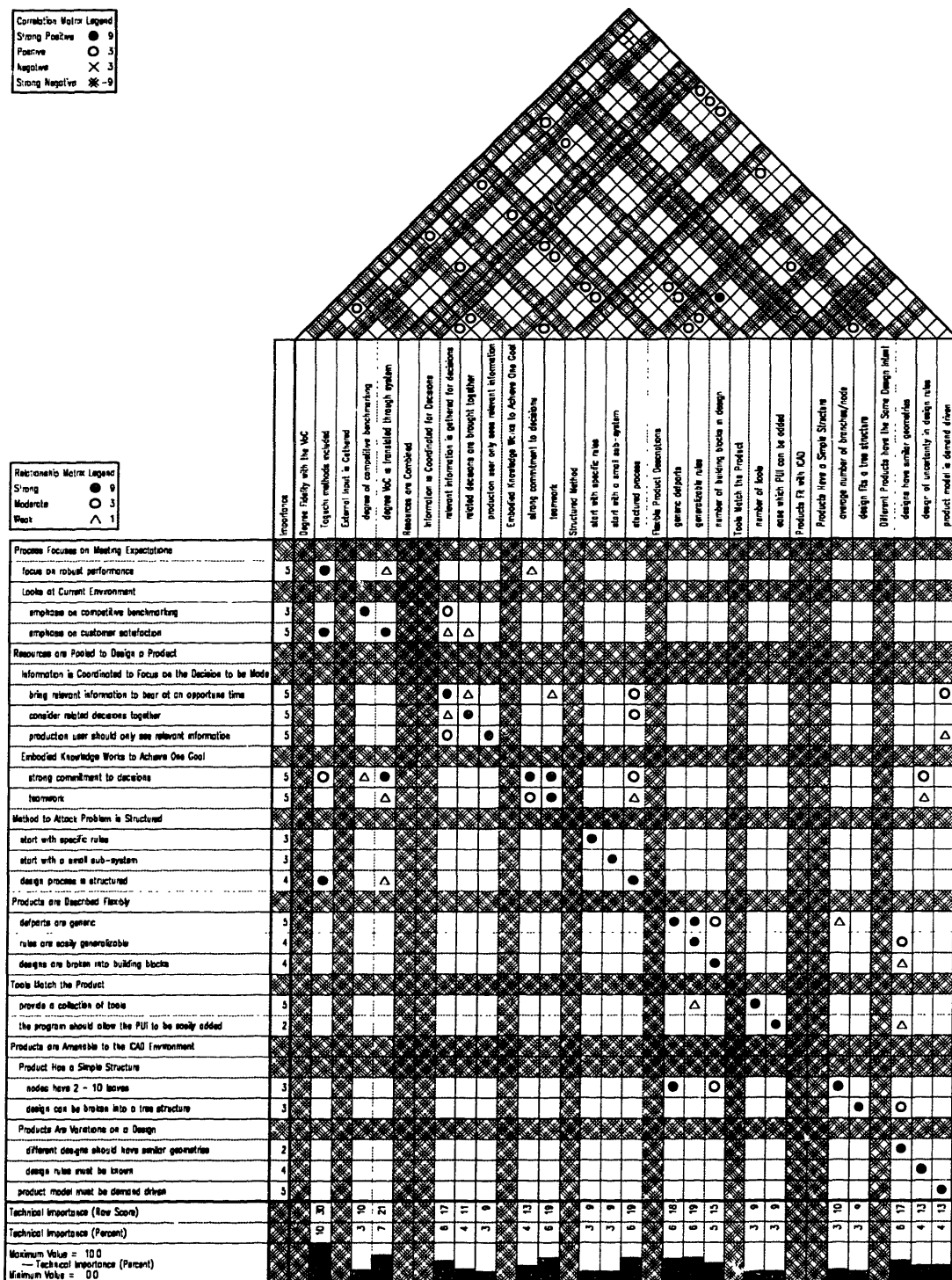


Figure 4.2c - House of Quality for QCAD³

Program development process in ICAD.	Use DSM and PSN for ICAD product models.	Function net in ICAD for product idea generation.	Use of ICAD for robustness experiments.	ICAD stores data for TQD.	TQD organizes data for ICAD.	Use downstream design knowledge farther upstream.	TQD developed part in ICAD.
Taguchi methods included	-	-	+	-	-	-	S
degree voice of the customer is translated	-	-	-	-	-	-	S
relevant information is gathered for decisions	S	S	S	-	S	S	S
teamwork	S	S	S	S	S	S	S
structured process	S	S	S	S	S	S	S
generic departs	S	S	S	S	S	S	S
generalizable rules	+	+	+	+	+	+	+
amount of design intent in subsequent products	S	S	S	S	S	S	S
number of building blocks in design	S	+	-	+	+	S	+
demand driven product model	S	-	S	-	S	S	S
degree of uncertainty in design rules	S	S	S	+	+	+	S
degree of competitive benchmarking	-	S	-	S	-	-	S
size of original part of the problem	+	+	+	S	S	S	S
number of tools	S	S	S	+	+	-	+
ease which PUI can be added	S	S	S	S	S	S	S
design fits a tree structure	S	S	S	S	S	S	S
average number of branches/node	S	S	S	S	S	S	S
$\Sigma+$	2	3	3	4	4	2	3
ΣS	12	11	11	9	10	11	14
$\Sigma-$	3	3	3	4	3	4	0

Key:	Design Structure Matrix
DSM	Product Structure Net
PSN	Production User Interface
PUI	Total Quality Development
TQD	

Figure 4.3 - Pugh Concept Selection

4.3. Basic QCAD Framework

QCAD has a basic backbone structure that coordinates information among different parts of the design process. The structure is generic such that a variety of design tools can be used with QCAD. However, the model is geared toward use with TQD.

Figure 4.4 shows the QCAD backbone structure. The structure is a network of interacting layers that represents a design. As described in Chapter 2, a design can be seen from several different levels of abstraction. These are listed from top to bottom on the left side of the diagram.

The top part of the diagram shows how a design would be specified, from initial problem definition through production. Both axiomatic design and quality function deployment propose ways to traverse a design from definition to production. Axiomatic design concentrates on the specific domains of a design in terms of the consumer domain, the functional domain, the physical domain, and the process domain (Suh 1990). QFD, on the other hand, concentrates on a set of planning matrices to translate customer voices to engineering expectations to hardware expectations to process requirements and to production requirements. The QCAD structure combines both methods of looking at a design with current methods of product model programming in ICAD. Combining the three methods together results in the three distinctions of functional requirements, hardware parameters and process parameters.

A functional requirement is the purpose of the product. It is the reason the product exists. For example, the purpose of one type of theater clamp is to position theater lights. A more detailed example would be the purpose of a screw is to hold two objects together.

The hardware is the physical description of the product being defined. It can be the whole product or a dimension of the product, depending on what level of abstraction is

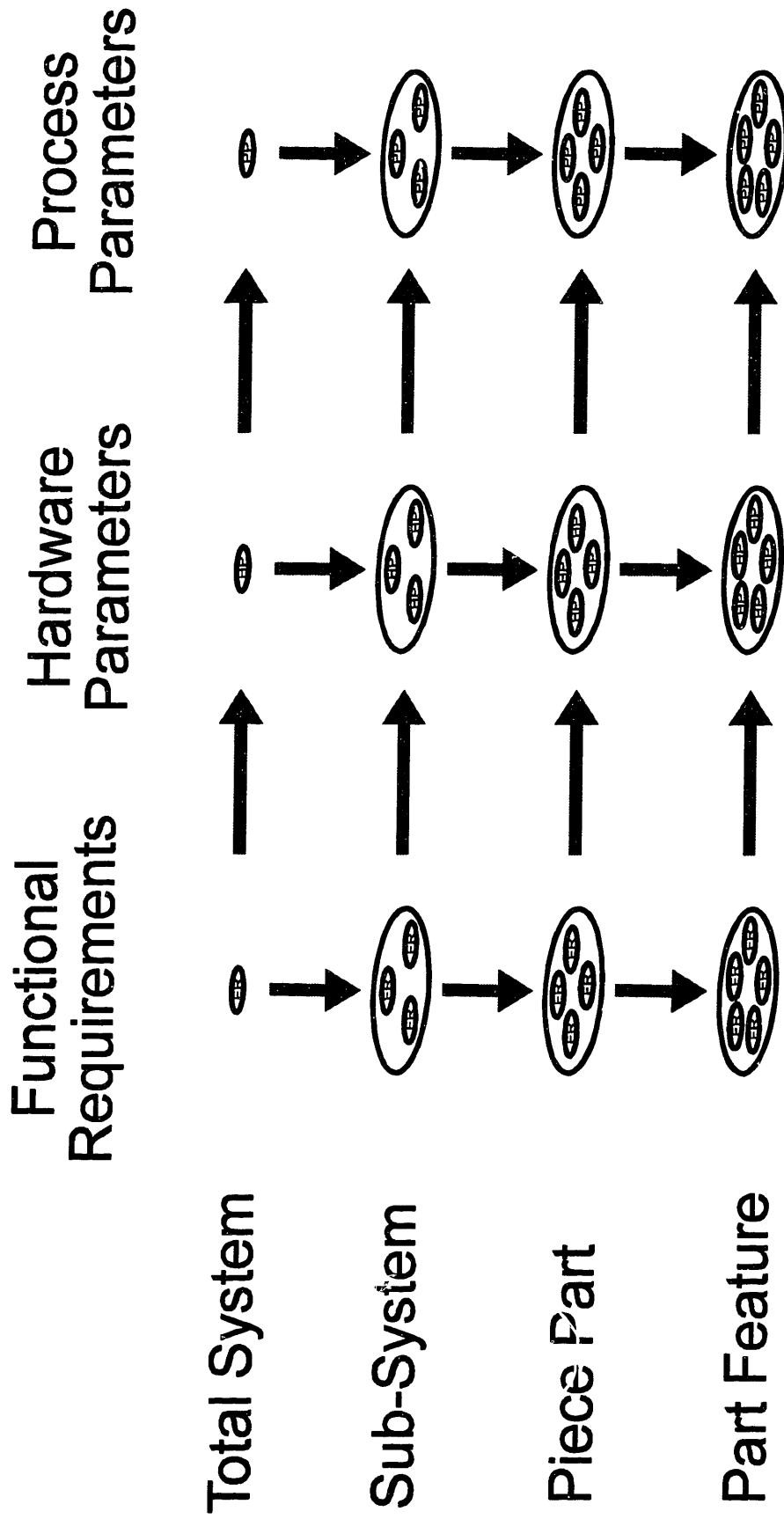


Figure 4.4 - QCAD Structure

desired. For example, the hardware to embody the function *position theater lights* is a c-clamp. At a lower level of abstraction, a hardware parameter is the length of the c-clamp.

Process refers to how the hardware is to be manufactured. At a high level of abstraction, the clamp is assembled. At a low level of abstraction, the clamp frame is sand cast.

Each small circle in Figure 4.4 is an object⁴. Each object can represent a function, a hardware piece, or a process, depending on which designation it is under. The larger circles represent layers of objects, such that each object in that layer can share information with the other objects on that layer. The total system layer contains only one object, because by definition, there is only one highest level function, hardware, or process definition.

Each of these three categories form hierarchies, with more abstract total system requirements at the top and more detailed, piece-part requirements at the bottom. Higher levels of abstraction represent conceptual design; lower levels represent detailed design. In Figure 4.4, this is represented by the arrows pointing down. Lower levels are organized by higher levels such that specific design information can flow through the hierarchy to where it is needed. The ideal is that objects on each layer can share information without going up and down the hierarchy. This feature is indicative of a layered structure.

The function, hardware, and process hierarchies are organized into different trees because of the varying nature of the information they represent. The function tree, being organized by functions is different from the hardware tree, which is organized by physical sub-systems. The process tree may, or may not, be different from the hardware tree. If the process tree is set up with assembly processes at the top and fabrication processes at the bottom, then the process tree will closely match the hardware tree. However, there

⁴ An object is the basic building block of a program, that contains specialized information and executable procedures. See Appendix B for more information on objects.

are other ways to think about processes, and as such, the process hierarchy is separate from the hardware hierarchy.

The three hierarchies are separate, but highly interdependent, such that one hierarchy constrains the choice of possible designs in the other hierarchies. The function definition guides the hardware development and the hardware choice guides the production process. This is represented in Figure 4.4 by the arrows pointing to the right.

All of the information flows from left to right and from top to bottom. A quick glance at the QCAD structure shows what other objects will be affected by a change in a single object.

The orderly flow of information in QCAD makes the design process look linear. In reality, design is amorphous, with designers working on several levels of abstraction and looking at several different criteria simultaneously. However, a well defined structure is necessary for the computer.

Sometimes a designer likes to experiment with different hardware configurations before selecting the exact nature of the function to be performed. The same is true with the hardware-process relationship. This capability is modeled into the QCAD structure by storing some hardware information in the function hierarchy and some process information in the hardware hierarchy.

Furthermore, implicit in Figure 4.4, are diagonal information flows from the top right objects to the bottom left objects. Going from right to left generally involves going down one level of abstraction. Staying on the same level would be an iteration or a feedback loop. For example, the type of hardware selected guides the functional requirements at the next level down (Clausing 1994). There are no explicit diagonal arrows in Figure 4.4 because the information that would normally flow in that direction is already stored in the function and hardware trees, respectively.

This explanation oversimplifies the design process into a sequential set of activities. In actuality, the design process is much more complicated. A designer may

start off in the function space and then move on to the hardware space, but will spend some time in the middle exploring the boundary between the two regions. The process can get even more complicated when considering that designers may also design on several levels of abstraction at a time (Moses 1990). However, the computer dislikes uncertainty, everything needs to be defined a priori. For this reason, in QCAD, (1) new designs are developed by the PDT with minimal computer use and then programmed into the computer as certain details of the design are frozen, and (2) some of the design information in the hierarchies are intermingled. Some elements of the hardware definition are stored in the function tree; some elements of the process definition are stored in the hardware tree (more on this in Section 4.5).

4.4. Design Process in QCAD

The previous section explained the organizational structure of a product in QCAD. This section describes the sequence of design activities that a multi-functional product development team would follow under the QCAD framework.

Before explaining the parts of the design process a few points should be expressed. First, for new product development, the PDT goes through the design process on paper first and then codes the information into the computer. Many of the quality design tools use visual connective methods to use the synergy of the PDT to develop bold new concepts. At the current state of computer technology, it is difficult to be entirely creative if constrained by the computer user interface (Hunt and Lee 1992). In subsequent designs, the computer may be used to help the PDT answer questions.

Second, the information recorded during the design process can be entered into the computer part of QCAD. However it is up to the design team to decide what information should be computerized and where in the structure it should be coded. For example, if the product being modeled is a copier, and the PDT is only interested in the paper feeder, the PDT can focus its efforts on the paper feeder sub-system and not worry about coding all

of the information for the rest of the copier. In this example, if the paper feeder is defined at the third level of abstraction, then the PDT does not have to worry about filling in the first two levels of abstraction. However, if the information is available, there is no problem incorporating information for other sub-systems, or for the total system, into the product model.

Third, the basic QCAD structure is generic. Many companies have their own versions of quality programs that they use to develop products. The QCAD structure provides a mechanism to use design information with the computer. If the PDT has other techniques that help them define a product, those techniques can be incorporated into QCAD.

Figure 4.5 shows the TQD tools superimposed over the QCAD structure. The rest of this section will briefly describe the role of each part of the design process. Section 4.5 will elaborate on how the design process will be used with the computer.

Each object in Figure 4.4, the QCAD structure, holds the information generated by the corresponding parts of the design process. The lines surrounding the design process tools are wavy, to show that the boundaries are not absolute. The PDT goes through the indicated design process steps in that general area of the product development cycle. The next three sections describe a general ordering of the indicated design process steps. Other variations on the design process steps are possible, and the team should tailor the process as desired.

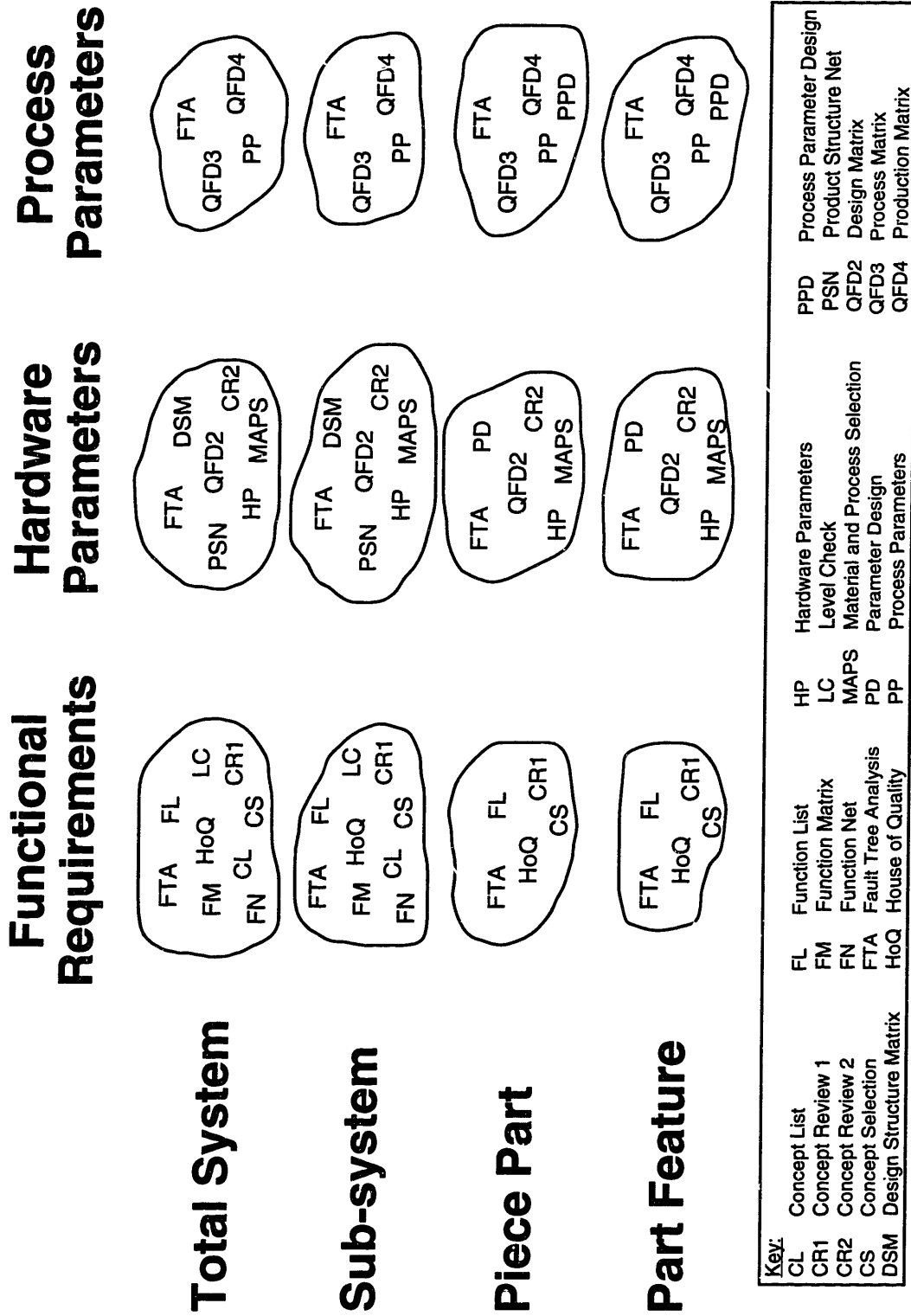


Figure 4.5 - Design Methods in QCAD

4.4.1. Function Hierarchy

This section describes the flow of information for the design process steps in the function hierarchy. Even though the flow is depicted as linear, some events may feedback information and other events may occur simultaneously.

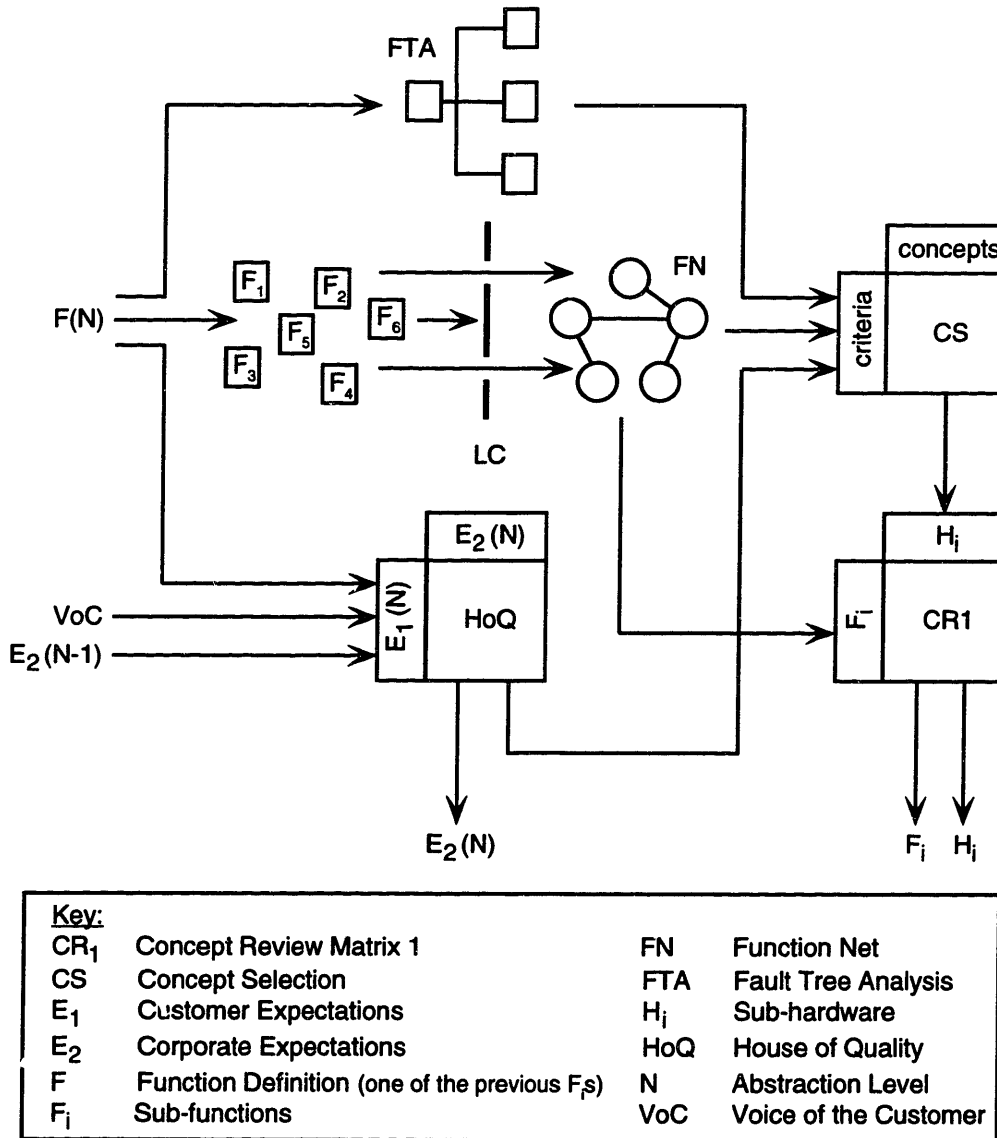


Figure 4.6 - Design Process in Function Hierarchy at Level N
(Clausing 1994 modified)

Notes for Figures 4.6, 4.7, and 4.8

Each figure represents one domain in Figure 4.5. As such, each figure represents one level of abstraction in one of the horizontal hierarchies.

Abstraction Level

To read the figures, replace N with the desired abstraction level. If N is at the sub-system level, then N-1 is the total system level and N+1 is the piece part level. If N is at the total system level, then N-1 does not exist. If N is at the part feature level, then N+1 does not exist.

Horizontal Domain

Each figure starts with a 'F,' 'H,' or 'P' definition, representing function, hardware, or process, respectively. The definition guides the application of the subsequent activities in that domain. F, H, and P are either initially specified, or are derived from previous design activities. The function definition flows from higher level function to lower level function. The hardware definition flows from same level function to same level hardware. The process definition flows from same level hardware to same level process. Implicit in the figures is a flow of the hardware definition down the hardware hierarchy and a flow of the process definition down the process hierarchy.

In Figures 4.7 and 4.8, the HP_i and the PP_i are different from H and P, in that HP_i and PP_i are parameters of the hardware and process, whereas H and P are the overall hardware and process definitions. Both HP_i and PP_i are domain dependent, i.e., they only flow down the corresponding hardware and process hierarchies.

Abstraction Level and Horizontal Domain

In addition to the function, hardware, and process criteria moving between domains, QFD expectations (E) move between domains. The different subscripts on the Es represent different expectations. In general, the expectations are defined in the columns of one QFD matrix and become rows of the next horizontal and vertical QFD matrices. There are two exceptions: (1) E_1 only goes into the first QFD matrix, and (2) E_4 only goes into the next vertical QFD matrix.

Keeping these caveats in mind, the design process follows the general pattern outlined in Figure 4.6, at all levels of abstraction. Each of the acronyms mentioned in Figure 4.6 are represented in this figure by a small icon. The icons approximate the visual format of the design tool, with the arrows representing primary information flows from and to the precise location on the design methods. Many other information transfer arrows could be drawn into the figure, but were removed to reduce clutter and to maximize drawing clarity. A case could be made for an arrow to be drawn from one design method to every other design method and even from and to different sections of the same design method.

The design process starts with an identified need, indicated by the boxed F on the left side of Figure 4.6. This need is the functional requirement that a product or part of a product is to satisfy. In the case of a theatrical lighting clamp, a high level need is to position theater lights. A low level need is to attach the theater light to the clamp.

The initial function definition, feeds into the first three design methods, which may be performed simultaneously or one at a time. At the top of the figure is the fault tree analysis (FTA), which is used to explore the function space to identify possible sub-functions and criteria for concept selection (Clausing 1994). The fault tree will also provide a base for the fault trees in the hardware and process hierarchies and in the lower level function hierarchy.

At the bottom of the figure is the first QFD matrix, also known as the house of quality (HoQ). It is used to translate qualitative customer needs into quantitative engineering expectations and to stimulate product concepts. The HoQ uses the function definition as a focus for the activity. For example, in the rows, customer needs are sought for a product that satisfies that functional need, and higher level engineering expectations, are used that further define the performance level for the function. The new set of product expectations are used as criteria in the concept selection matrix (CS) (Clausing 1994). See Sections 2.3 and 2.5 for a more detailed description of QFD. The 'E' variables, in the

rows and columns, represent expectations. Every deployment from a row to a column increases the subscript on the 'E.' The 'N' represents the abstraction level. 'N-1' is a higher abstraction level than 'N.'

In the middle of Figure 4.6 are a series of steps that refine and correlate the sub-functions. First, a set of possible sub-functions is generated. Then, the set is checked for abstraction using the level check (LC) procedure, described below. Last, the functions are arranged into a function net (FN), displaying the interrelationship of the functions (See Section 4.5.2.4.). The subscripts on the 'F's in the figure indicate that the function is a sub-function of the initial function. The 'i' subscripts on the 'F' and the 'H' indicate multiple sub-function and sub-hardware definitions.

The level check (LC) is used when defining functions, to make sure that all of the functions are at the same level of abstraction. In Figure 4.6, the level check is represented by a filter, allowing functions at the appropriate level of abstraction pass, rejecting the other sub-functions. This is done by asking two questions about the relationship between two functional requirements. The first question asks, "Does function B contribute to the performance of function A?" If yes, function B is a sub-function of function A. If no, ask the second question, "Does function A contribute to the performance of function B?" If yes, function A is a sub-function of function B. If the answer to both questions is no, then functions A and B are at the same level of abstraction (Sontow and Clausing 1993). A level check is only needed at higher levels of abstraction, where there are a large number of functions at differing abstraction levels. At lower levels, the level of abstraction of the functional requirements is clearer.

Starting with the set of functions, a function matrix (FM) and function net (FN) analysis is completed to organize the functions into a relationship diagram. The function matrix (not shown in Figure 4.6) is derived from the design structure matrix (Eppinger et al. 1990). The function net is a graphical representation of the function matrix. This step gives new insight into the functions to help generate hardware ideas and selection criteria.

This technique can be used at any level in the design process, but is especially insightful at higher levels of abstraction, where the function relationships may not be as clear.

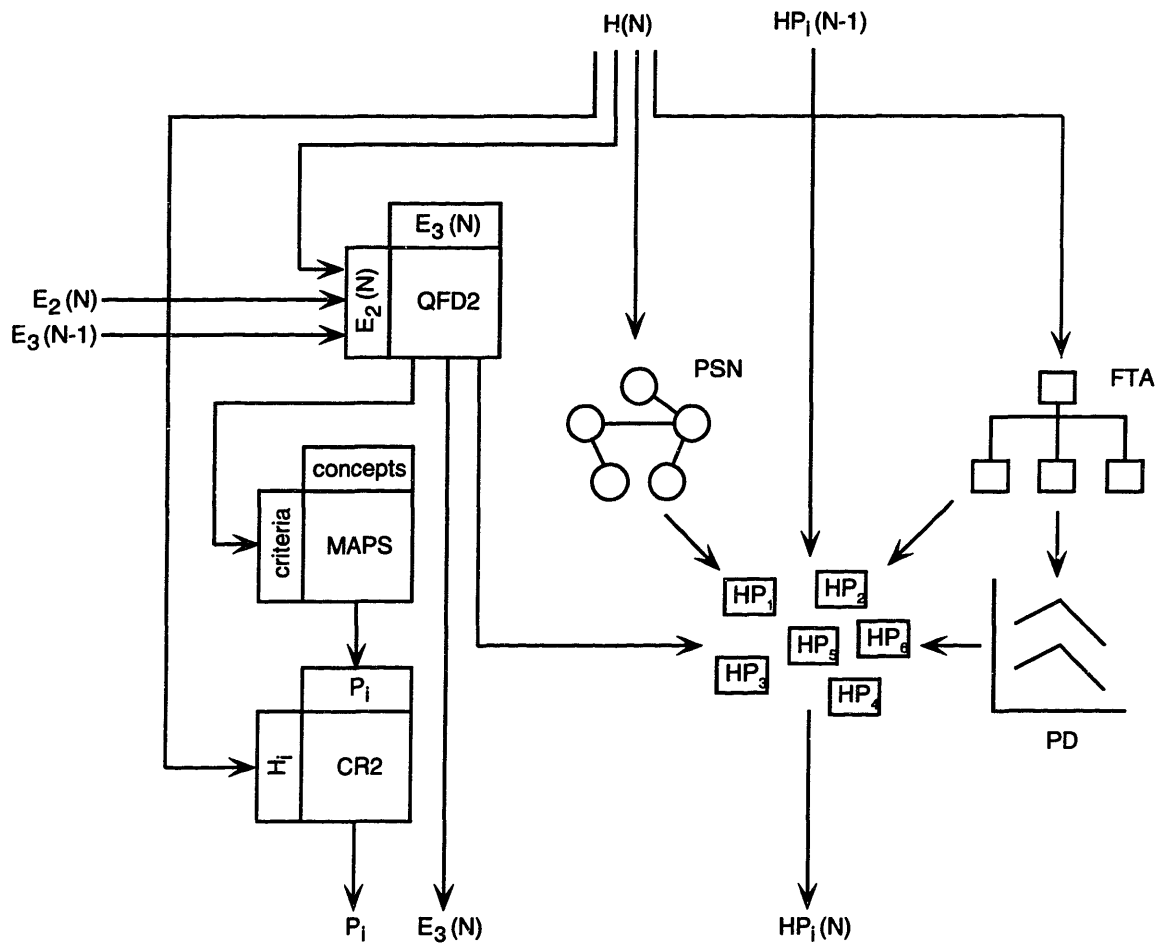
While working with the design methods thus far, members of the PDT have probably developed some hardware concepts. These concepts are documented in a concept list (CL) (not shown in Figure 4.6). If more concepts are desired, some authors (Pahl and Beitz 1984; Cross 1989) recommend brainstorming or a morphological analyses to generate concepts. However, Pugh (1981) recommends deriving new concepts using concept selection. Pugh concept selection (CS) is described in Section 2.4.

The output from concept selection is an agreed upon hardware concept and an idea of how the concept will be designed in the hardware. As a final check before moving on to another phase of the design, the PDT evaluates the design with a concept review matrix. The concept review matrix matches the sub-functions with the hardware parts for the selected concept and helps reveal the most important sub-functions (F_i) and sub-hardware (H_i). Further sub-functions or sub-hardware may also be revealed by completing the matrix (Sontow and Clausing 1993).

4.4.2. Hardware Hierarchy

This section describes the flow of information for the design process steps in the hardware hierarchy. The same comments about reading the design process flow chart mentioned in Section 4.4.1, for the function hierarchy, also apply to the hardware design process, shown in Figure 4.7.

The hardware process also starts with a definition of the governing hardware, indicated by the boxed H at the top center of the figure. The H is one of the H_i s defined in the function hierarchy. Like the function definition, the hardware definition influences



Key:	
CR ₂	Concept Review Matrix 2
E ₂	Corporate Expectations
E ₃	Product Expectations
FTA	Fault Tree Analysis
H	Hardware Definition (one of the H _s)
HP _i	Hardware Parameters
MAPS	Material and Process Selection
N	Abstraction Level
PD	Product Parameter Design
PSN	Product Structure Net
QFD2	QFD Design Matrix

Figure 4.7 - Design Process in Hardware Hierarchy at Level N

three different design methods. At the top, left side of the figure is the second QFD matrix (QFD2). Its purpose is to translate engineering expectations into product expectations. While completing the matrix, some process concepts may develop that can be used in the material and process selection matrix (MAPS) (Clausing 1994).

The product expectations, from the columns of the QFD matrix are used as criteria for material and process selection. Some additional criteria may also be used in the evaluation. The MAPS matrix may follow the same format as the Pugh selection matrix, described in Section 2.4, or may follow a more analytical approach. Whichever the case, the output from the selection matrix is a process for manufacturing the hardware.

The manufacturing process is then compared with the hardware in a concept review matrix (CR2). The CR2 matrix is exactly like the CR1 matrix, in that it helps reveal the most important sub-hardware (H_i) and sub-processes (P_i). The concept review matrix is also used as a final check before moving on to the process phase of the design. Further sub-hardware or sub-processes may also be revealed by the matrix (Sontow and Clausing 1993).

The hardware hierarchy not only defines the production process, but also assigns values to the hardware parameters (HP_i), which are variables that store dimensions and material type for the hardware piece or sub-system. Note that higher level hardware parameters may constrain values for parameters at the current level. Current level parameters will then constrain parameters at the next lower level. Several design methods provide insight for setting the correct values for the hardware parameters.

The QFD2 matrix brings the customer perspective as deployed from the HoQ matrix. The product expectations in the columns have target values which the PDT defines to be the ideal value for several hardware parameters. In reality, the hardware parameters may not be able to achieve the target value, i.e. zero resistance for a wire (Clausing 1994). See Sections 2.3 and 2.5 for a more detailed description of QFD.

The design structure matrix (DSM) (Eppinger et al. 1990) and product structure net (PSN) are used in the same manner as the function matrix and function net, defined in Section 4.4.1, except that the DSM and PSN give new insight into the hardware parameters and provoke concepts for possible production processes (The DSM is not shown in Figure 4.7).

The fault tree analysis (FTA) is used to identify critical functional parameters, control factors, and noise factors for use in product parameter design (PD). Completing a fault tree may also identify critical hardware parameters and equations for calculating those parameter values (Clausing 1994).

Product parameter design is part of the Taguchi system of quality engineering (Taguchi 1986). It is used to create hardware systems that are robust against noises that degrade product performance. See Section 2.6 for more information on product parameter design.

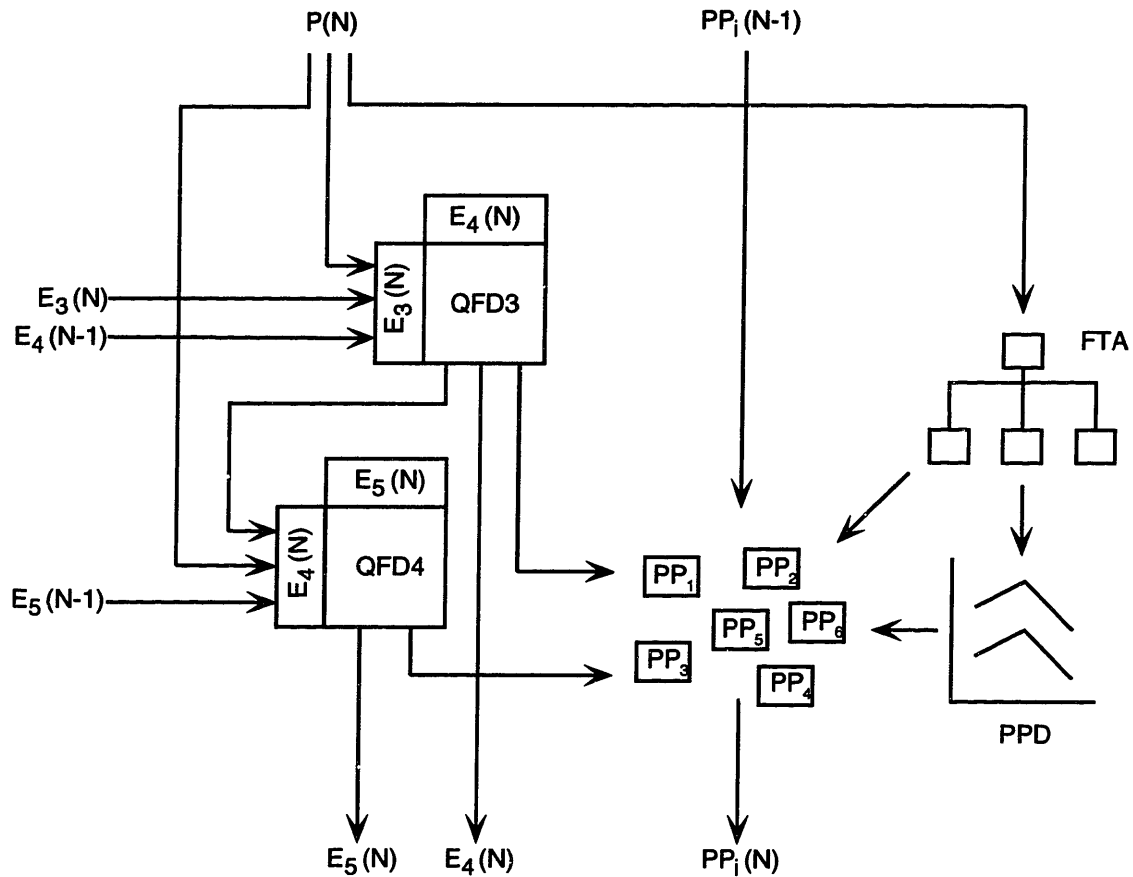
4.4.3. Process Hierarchy

This section describes the flow of information for the design process steps in the process hierarchy. The same comments about reading the design process flow chart mentioned in Section 4.4.1., for the function hierarchy, also apply to the design process in the process hierarchy, shown in Figure 4.8.

The process hierarchy design flow is similar to the design flow in the hardware hierarchy. The main difference is that the process hierarchy is the last category in the horizontal series of categories and thus only defines its own parameters.

Similar to the design process flows for the function and hardware hierarchies, the process hierarchy starts its design process with an overall process definition, indicated by the boxed P at the top left of Figure 4.8. At a high level of abstraction, the process definition can be assembly. At a low level, the process definition can be surface grinding. The P is one of the P_s defined in the hardware hierarchy.

The process definition feeds into the third and fourth QFD matrices (QFD3 and QFD4) and the fault tree analysis (FTA). No process structure net is needed for the



Key:	
E ₃	Product Expectations
E ₄	Process Requirements
E ₅	Production Requirements
FTA	Fault Tree Analysis
P	Process Definition (one of the P _s)
PP _i	Process Parameters
PPD	Process Parameter Design
QFD3	QFD Process Matrix
QFD4	QFD Production Matrix

Figure 4.8 - Design Process in Process Hierarchy at Level N

process hierarchy, as processes are usually arranged according to a time sequence of events (i.e. hole A is drilled before hole B; gizmo D fits into hole Q before gizmo F locks item Z into place).

QFD3 is the process planning matrix, which translates product expectations into process requirements. QFD4 is the production planning matrix, which translates process requirements into production requirements. Both matrices give insight into process and

production design and influence the values of process parameters (Clausing 1994). See Sections 2.3 and 2.5 for a more detailed description of QFD.

Process parameters (PP_i) contain detailed values that define how a production processes operates. For example, a high level parameter would be the assembly sequence. A low level parameter would be the mold temperature of a sand casting. Just like the hardware, higher level process parameters constrain lower level process parameters.

In addition to the QFD matrices, the fault tree and process parameter design (PPD) methods guide the definition of the process parameters. The fault tree is used to identify control factors and noise factors for use in product parameter design (PPD). Completing a fault tree may also identify critical process parameters and equations for calculating those parameter values (Clausing 1994).

Process parameter design is part of the Taguchi system of quality engineering (Taguchi 1986). It is used to optimize processes to manufacture robust products. See Section 2.6 for more information on product parameter design.

The Taguchi system also has procedures to find an economical tolerance design and an optimal intervention strategy for on-line quality control. Both methods are discussed in Section 2.6, but are not used in the current QCAD methodology.

4.5. Computerized Elements of QCAD

The previous section briefly described the elements of the design process used with the QCAD methodology and positioned those elements in the QCAD structure. This section elaborates on those design tools as they are used with the computer. Many examples are given using a theatrical lighting clamp. Some concepts from Chapter 2 are repeated in the examples. Section 5.2.2 shows the computer implementation of the clamp in QCAD.

This section is split into five parts. In the first part, the theater clamp model is described, since it will be used to explain QCAD concepts. The next three parts examine

the function, hardware, and process hierarchies, respectively. The last part examines concepts that apply to all three hierarchies.

4.5.1. Description of Clamp Product Model

This section describes a theatrical lighting clamp, which is a simple product that will be used to describe QCAD. QCAD can be used with more complex products, as will be shown in Chapter 5.

A theatrical lighting clamp is used to hang theater lights that weigh between 5 and 20 pounds on pipes in various places around a theater (for reference, the traditional cast iron clamp weighs about 1.5 pounds, which is negligible compared to the weight of the light). The pipes can be located up to 30 feet in the air, in any orientation, (i.e., horizontally, vertically, or diagonally) above both audience and cast members. Usually, a lighting technician will attach the clamp to the light on the ground, carry the light up to the catwalk, secure the light to a specific spot on a specific pipe, and then rotate the light to the desired position.

Figure 4.9 is an annotated picture of the clamp. The light secures to the clamp at the bottom of the barrel with a screw and a washer. To mount the clamp on a pipe, the frame screw is unscrewed as far as necessary to allow the pipe to snugly fit through the frame mouth. If the pipe is hung horizontally, the clamp and light can rest freely from the top of the frame, which is also known as the hook. The frame screw tightens down with a wrench against the pipe to stabilize both the clamp and the light and lock them in place. Once locked down, the light can be rotated by loosening the barrel set screw. The barrel is a cylindrical piece of aluminum, with a captivating lip, that is free to spin in the frame.

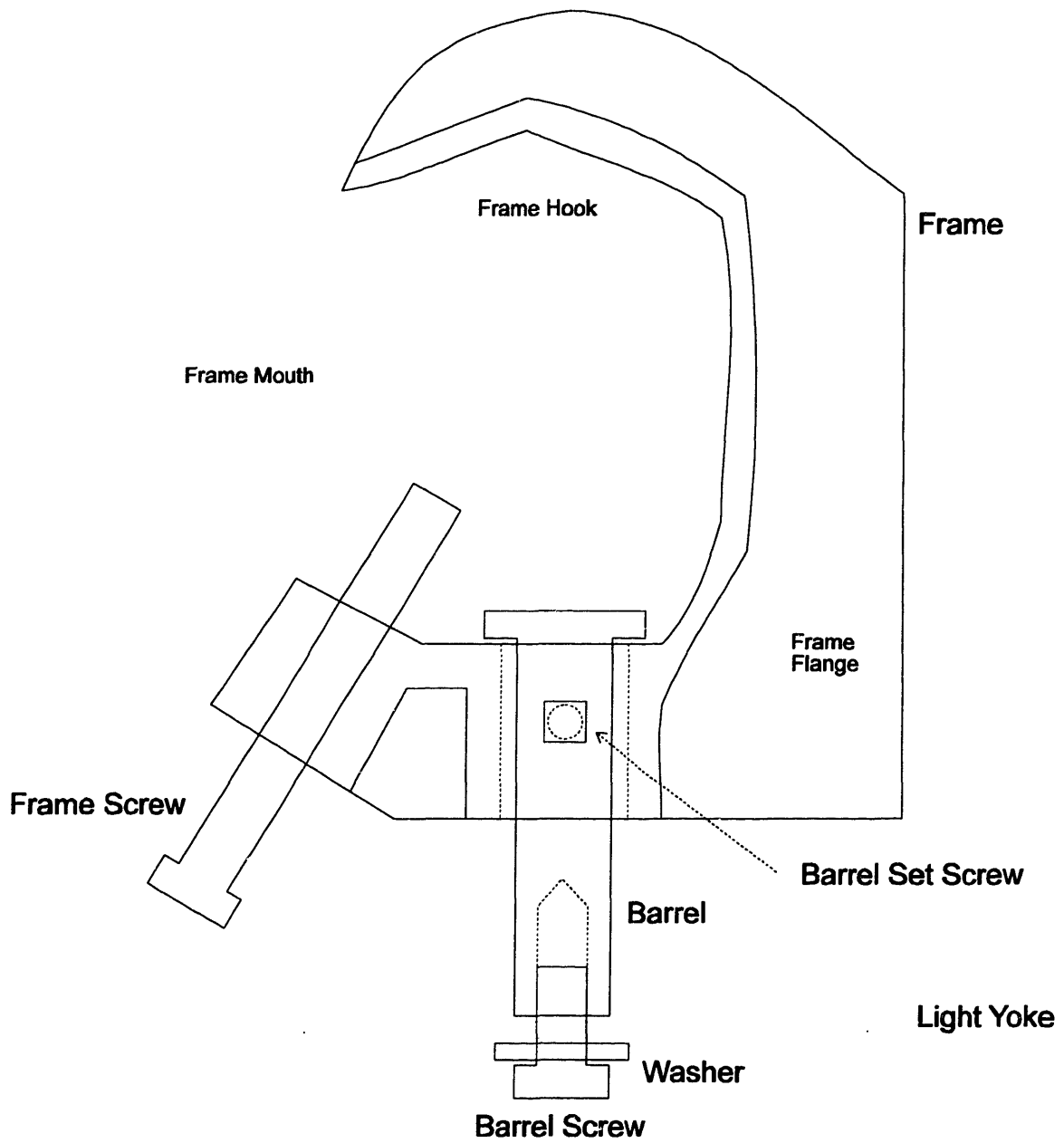


Figure 4.9 - Theatrical Lighting Clamp

4.5.2. Function Hierarchy

Each function is its own entity—each has its own name, a list of constraints, a QFD matrix, and a set of associated concepts. The first three are used by the designer to put the design problem into perspective, in order to create a set of hardware concepts that will satisfy the function. In this way, the function is the basic mechanism used to organize the storage and retrieval of several different hardware concepts.

QCAD enhances this feature of functions by dynamically linking them to hardware. Two advantages include: 1) if the design needs to change, the function can call up all of the hardware that satisfies that function, providing a holistic view of how a product can be modified, what parts need to be changed, and what the interactions are among those parts. 2) when redesigning a product, a designer can explore the design from the perspective of different functions to get a different understanding of the product which will help generate new ideas.

For reference, Figure 4.10 is the function hierarchy for the clamp. For clarity, part feature functions are not listed here.

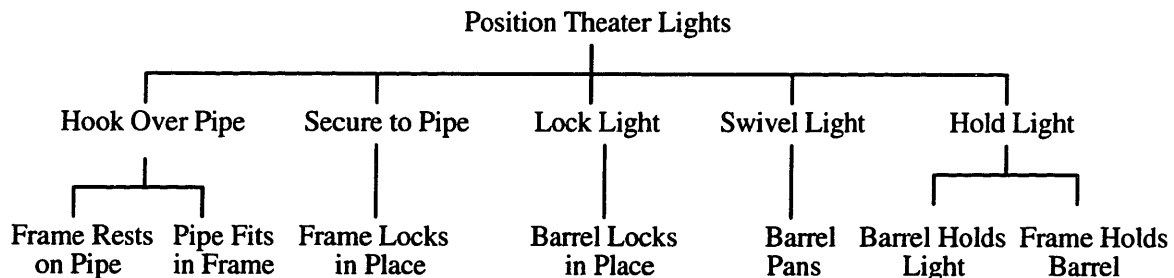


Figure 4.10 - Clamp Function Hierarchy

4.5.2.1. Function Identification

All products exist to fill a need. That need at the highest level of abstraction is called a total system functional requirement—it will be the focus of subsequent product development work. As such, the total system functional requirement (FR) is the node name at the top of the function hierarchy. The total system function is identified by a

company based on market research. For the clamp, it is defined as a device *to position theater lights*.

Lower level functions are defined by higher level hardware pieces. For example, when the PDT decided that the clamp would have a one piece frame, they created the functional requirement that the frame must fit over the pipe. If the frame was two pieces, there would be a different requirement that the pipe must fit in the frame. These FRs become the names for the lower level function nodes.

4.5.2.2. Constraints

Just defining the function name is not enough. Each function has some constraints on it due to the nature of the use environment or from customer needs identification. Probably the best way to think about constraints, as used in QCAD, is that they represent items that could be checklisted. They are either satisfied or not satisfied. Included in this category are must-be Kano (Shiba, Graham, and Walden 1993) customer requirements, government regulations, and safety requirements. The following constraints were found for the clamp:

1. reversibly lockable (can be opened and closed many times),
2. use only human power,
3. support any orientation of a light,
4. interface with existing standard theater equipment (pipes, lights, etc.), and
5. support human weight with a factor of safety.

In QCAD, two things can be done with these requirements. They can be left in an attribute list (an attribute is another name for a variable) that can be called at any point in the design process for guidance in decisions, or they can be converted into individual attributes that specify constraints on parts further down in the design process. Generally, a combination of both methods will be used.

For example, constraints 1, 2, and 3 would best be used in a textual list and contemplated when concept selection decisions are made. Constraints 1 and 2 constrain the concepts for selection. That is, welding cannot be used as a clamping concept because it is not reversible, nor can it be done solely under manual power. These constraints are textual constraints because they cannot be assigned a distinct numerical value, but do guide the design process.

Constraint 3 is also a textual constraint, but it actually refers to the types of engineering calculations that must be performed to validate a design. Likewise, constraint 1 could show that a fatigue calculation needs to be performed. (See Section 4.5.3.2 on hardware product parameters for more information).

A different type of constraint is a numerical constraint. Constraints 4 and 5 can be assigned to variables, with attributes such as :pipe-diameter = 2" and :design-weight = :human-weight * :safety-factor.

If QCAD is used to unite several PDTs, each team can use the system to identify constraints that must be satisfied. Changes made to constraints are propagated through the model, ensuring that all affected components are changed, or all design teams are informed of a change.

4.5.2.3. QFD House of Quality Matrix

Even though a function name and a set of constraints has been identified, there is more information to decipher about functions. Many designers are comfortable working with engineering metrics, but not with abstract requirements. The two can be correlated on paper, by a PDT using a QFD matrix (Figure 4.11). There are many reasons to use QFD, one of which is to submerge the designers in the design problem to help generate novel ideas.

Computers are good tools at making calculations and at making reports (ITI 1992), but do tend to stifle human creativity. The computer requires structured input of

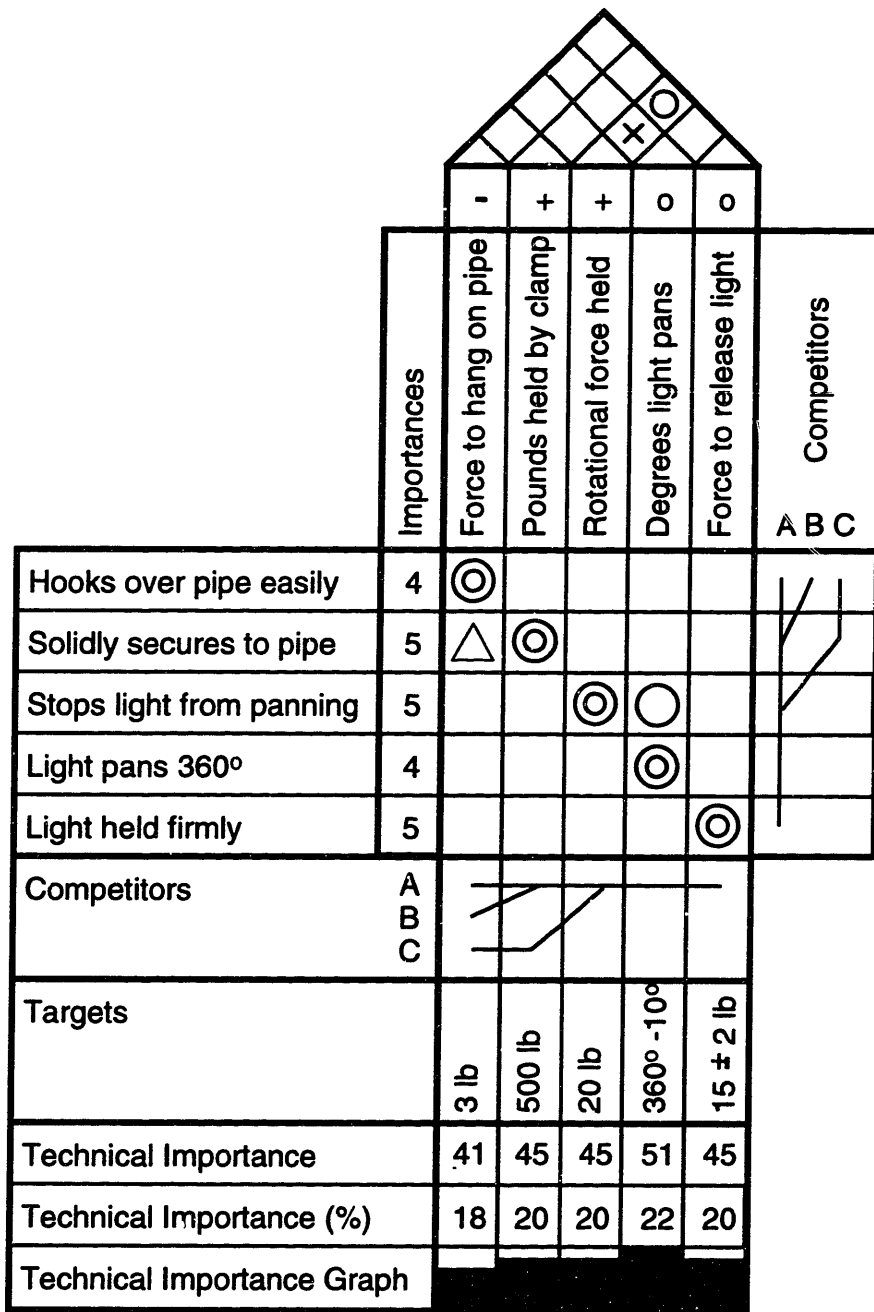


Figure 4.11 - House of Quality for the Clamp

information, which at the current state of technology is limiting for creativity (Hunt and Lee 1992).

The computer is good at crunching numbers and thus is good for use in sensitivity analysis. After a QFD matrix has been created off-line by a PDT and is input to a computer, both interactions and importance ratings can be changed to show their influence on the total design. Note that changing these parameters can help a person look at the design problem differently—not to mention that attaching a dynamically changing product to the QFD matrix takes the activity beyond the abstraction of conceptual design. This feature is more useful in subsequent designs, after some hardware has been defined.

The system of EQFD matrices in the product model are linked together using the technical importance ratings. Once a technical importance rating is calculated, it is linked to the corresponding metric. Whatever sub-systems use that metric, inherit that metric's importance rating. The importance ratings, however, are raw importances calculated by multiplying the row importances by the number of relations in the relationship matrix. This is fine if only one set of matrices is used. However, some engineering metrics may be misrepresented if multiple matrices are used and importance ratings from several different matrices are combined into one matrix. For example, the relationship matrix of one QFD matrix may be densely packed, while the relationship matrix of another may be sparsely packed. Thus, the technical importance ratings should be normalized to a scale from one to five, where five is the highest. The following formula may be used:

$$N_i = \frac{I_i \times 5}{I^+}$$

where N_i is the normalized importance rating for metric i , I_i is the raw importance rating for metric i , and I^+ is the largest of the importance ratings.

The QFD matrix also has design targets for the engineering metrics, which can be propagated to other design nodes. As described in Clausing (1994), target values can be allocated to sub-systems in two ways: 1) The values are additive and are distributed to

each sub-system; or 2) The values are uniform across all sub-systems. For example, cost is additive across sub-systems: A total system cost of \$50 can be split into \$25, \$15, and \$10 sub-systems. However, minimum fatigue life is uniform across the clamp parts.

QFD matrices at levels of abstraction below the total system are used as described above, except that the rows take information from the previous QFD matrix in addition to the voice of the customer.

4.5.2.4. Function Diagram

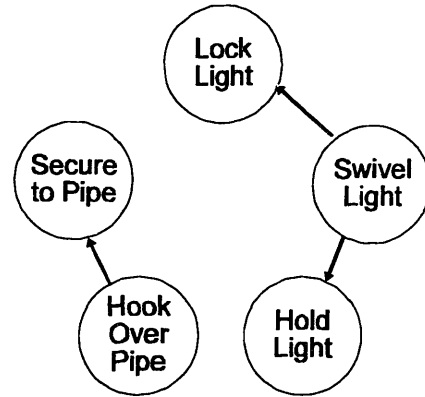
In addition to listing constraints and completing a QFD matrix for a function, a diagram of the function space should be used to show relationships among functions. There are many methods that give a structure to functional requirements. One of the more frequently mentioned techniques is a function structure developed by Pahl and Beitz (1984).

For this initial definition of QCAD, a function diagram will be used. If desired, future versions of QCAD can be upgraded to use a more complex function structure. A function diagram is a graphical representation of the interactions among sub-system functions, as shown in the Figure 4.12. Creating a function structure is a two step process: 1) Determine the information flows among parts by creating a function matrix (FM). 2) Translate the matrix into a diagram. Step one is performed by the PDT off line. Step two is done by the computer.

A function matrix, much like a design structure matrix (Eppinger et al. 1990), shows the direction of information flows among functions. Figure 4.12a is the FM for the high level clamp functions. The sub-system functions are listed in both the rows and columns of the matrix. An X in the matrix shows that the definition of a hardware concept for the function in the column will affect the hardware concept for the function in the row.

	Hook Over Pipe	Secure to Pipe	Lock Light	Swivel Light	Hold Light
Hook Over Pipe	x				
Secure to Pipe	x	x			
Lock Light			x	x	
Swivel Light				x	
Hold Light				x	x

a. Function Matrix



b. Function Diagram

Figure 4.12 - Function Diagram for the Clamp

For example, changing the concept of how the light swivels, affects the concept for how the light locks in place. Right now, the swivel mechanism concept is a barrel that goes through the clamp frame. The barrel is locked in place with a side screw. If the swivel mechanism concept was changed to a ball in socket, the light locking mechanism would have to change.

The function diagram is a graphical representation of the function matrix. As shown in Figure 4.12b, it is easier to understand than the matrix. The circles are the sub-system functions and the arrows show the direction of influence.

An alternate way of looking at the matrix, or diagram, is that the functions in the columns affect the corresponding functions in the rows. Thus, if a change is made to one sub-system function concept, the designer can find other sub-systems that will be affected. Alternatively, the matrix can be used to find what function concepts should be considered when making changes. This method of storing information, when recorded in the computer for easy access, helps the designer minimize design mistakes.

Even though the benefit of this module is more apparent when redesigning a product, it can still be effective for original designs. A product structure diagram at the

high level shows interactions between lower level sub-systems that design teams should investigate to fully understand the product.

4.5.2.5. Concept Generation

While submerged in the function space via customer interviews and QFD matrices, many concepts are generated by the PDT. A few of the more feasible ones are carried through into development. A list of generated concepts is maintained in QCAD to store ideas. These ideas form part of a design history showing what the original designers were thinking as part of the design. The concept list contains information like a concept name, a description of its intended functionality, and other comments about the design. In subsequent designs, the list can point to ideas for evaluation and to reasons why certain designs are not feasible and should not be pursued. All considered designs should be put in this list, including both those being used and those that aren't. A sample concept list is shown below.

<u>ID Letter</u>	<u>Concept</u>	<u>Description</u>	<u>Comments</u>
A	solid frame	a solid frame connects both locking mechanisms	current solution
B	hinged frame	a hinged frame with locking clasp	hinge not robust
C	solid frame with double lock mechanism	solid frame with interconnected locking mechanism	further evaluation required

The concept list is not intended to help first round designers create novel products. It is intended to record a design history and to trigger ideas for subsequent round designers. This capability can be augmented by continually adding concepts to the list. Remember that each function is its own object and can be called no matter where the

function is in the hierarchy. Thus, wherever the function goes, the associated generated concepts go with it.

4.5.2.6. Concept Selection

In TQD, the next step is to select a design based on a set of criteria. Usually, the criteria used are the metrics, (columns) from the QFD matrix. The design team starts off with about 30 concepts at the beginning and after several iterations narrows the choices down to two or three for further work. These two or three alternatives are coded into a concept selection matrix on the computer. The ideas that don't make it can be annotated in the concept generation list.

Two methods can be used in the computer to model concept selection. The first method would be to create a concept selection matrix in the computer as shown in Figure 4.13. In this figure, the criteria importance ratings are dynamically linked to the columns of the QFD matrix. The relationships in the matrix come directly from the PCS matrix, where +'s mean that the product is superior in that category, S's mean that the product is the same in that category, and -'s mean that the product is inferior in that category. The datum concept has a column filled with S's.

The best concept is then the one with the highest overall importance. This *mind numbing numerology* is a deviation from PCS and is only being used for computer programming purposes—a PDT using QCAD should not forget this distinction. With that said, the overall importance ratings are computed from the weighted sum of the relationships in the column:

$$\sum_{i=1}^n W_i \times R_{ji} = I_j$$

where n is the number of evaluation criteria, W_i is the criteria importance rating for criterion i , R_{ji} is the relationship (+, S, -) for concept j on criterion i , and I_j is the overall

	Importance	Concept		
		A	B	C
Force to hang on pipe	4	S	-	S
Pounds held by clamp	4.5	S	-	S
Rotational force held	4.5	S	S	S
Degrees light pans	5	S	S	-
Force to release light	4.5	S	S	S
Overall Importance		0	-9.5	-5

Legend
- = -1
S = 0
+ = 1

Figure 4.13 - Concept Selection in QCAD for the Clamp

importance rating for concept j . The concept with the highest score is selected for further refinement. Notice that if the voice of the customer importance ratings or the relationships are changed in the QFD matrix, the column importances will change, and the selected concept may change. Moreover, the importance ratings are linked throughout the model, which could result in a wholesale change in the product. Additionally, as more concepts are added to the model, the more powerful the product model becomes. There is also a brief discussion of this kind of sensitivity analysis in the QFD matrix section above.

If the designer wants to work on a specific design regardless of the importance ratings, QCAD allows the user to choose a concept for refinement.

The second method is to use a heuristic to select a concept. Sometimes a concept selection matrix is not required to select a concept. Instead, a rule of thumb is used to decide between ideas. For example, if the clamp is to hold more than 500 pounds, use concept A; if the clamp is to hold between 50 and 100 pounds use concept B.

In general, the concept selection feature of QCAD is more useful for modifying existing designs, rather than creating new designs.

4.5.2.7. Concept Review

As a kind of concept review, the PDT, without the help of the computer, should mark relationships between the required functions and the parts in the selected concept. The matrix pictorially represents the allocation of functions, making it easier for a PDT to decide if the interactions are too complex to be dealt with efficiently. Once the PDT is satisfied with the level of interactions, the matrix is programmed into the computer (Figure 4.14). For subsequent product designs, the matrix may already exist in the computer and may be reused. A new matrix will be required for each different product architecture.

This matrix also guides what information is sent to which sub-systems or piece-parts. Constraints and QFD columns are allocated to certain functional requirements, which are passed to the next system via the concept review matrix. Not all functions are critical and require a QFD matrix. In such a case, constraint information and target values can be ear marked for specific hardware concepts.

	clamp support	securing mechanism	swivel mechanism	light holding mechanism
hook over pipe	X			
secure to pipe	X	X		
lock light			X	
swivel light			X	
hold light				X

Figure 4.14 - Concept Review Matrix for the Clamp

The computer uses the matrix to link functions to hardware. In the product model, when a function is selected, it will be possible to list and view all of the hardware pieces that satisfy that function, facilitating engineering changes. This feature is also the mechanism for seeing the product from different points of view, in terms of which parts perform what functions.

The concept review matrix can also be used to help decide where improvements to a design should be made. It shows which functions have few interactions and would be easy to change and those that have more complex hardware interactions.

4.5.2.8. Additional Comments

The use of the function hierarchy goes beyond just the use of modules. Listing out the functions at each node name can be helpful in understanding the product. For example, it is possible to look up and down the function tree for side-effect functions, as in those functions that are necessary because of the chosen hardware architecture, but are not used directly in satisfying the overall product function. Finding ways to reduce these side effect functions should improve the product design.

4.5.3. Hardware Hierarchy

As discussed above, the function hierarchy is the starting point for each level of design planning; the function space is defined and then searched for different hardware concepts. After selecting hardware for the product, design decisions are made in the hardware tree. Higher level design decisions in the hardware include general allocation of space to sub-systems and recording issues involved when using those sub-systems. The product structure diagram helps organize thoughts about the hardware design, which reflect the underlying product parameters.

At lower levels in the hardware hierarchy, the parameters become more concrete, referring to specific dimensions, material and process selection, and parameter optimization through design of experiments. For reference, Figure 4.15 is the hardware hierarchy for the clamp. For clarity, hardware part features are not listed here.

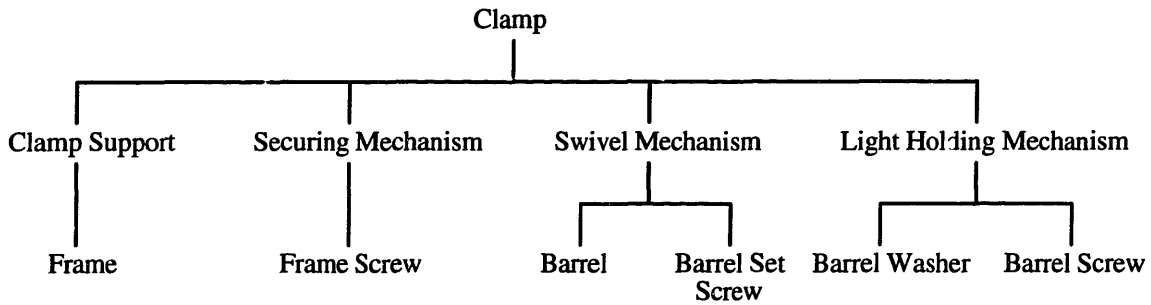
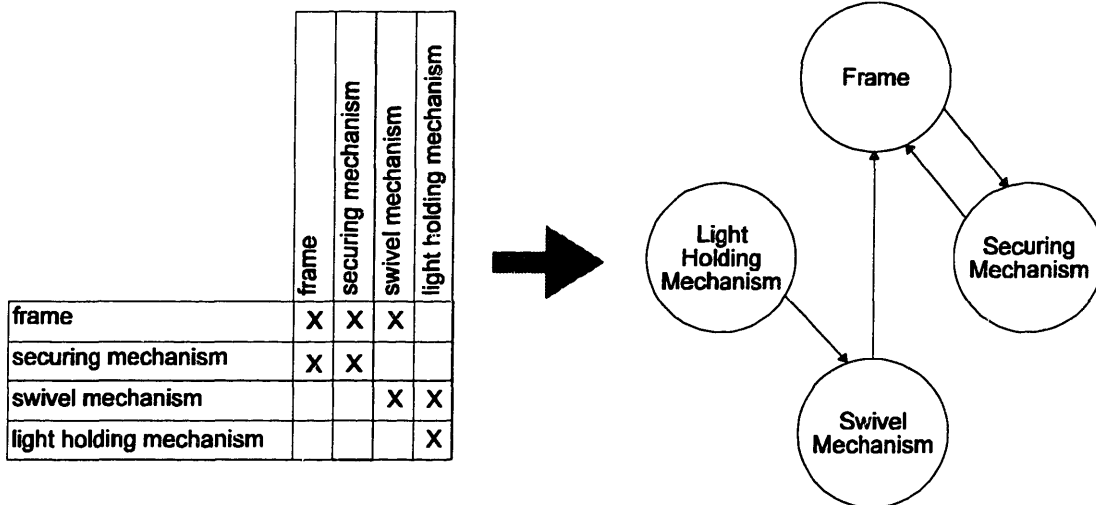


Figure 4.15 - Clamp Hardware Hierarchy

4.5.3.1. Product Structure Diagram

Creating a product structure diagram is a two step process. 1) Determine the information flows among parts by creating a design structure matrix (DSM). 2) Translate the matrix into a diagram. Step one is performed by the PDT off line. Step two is done by the computer.

A design structure matrix (Eppinger et. al. 1990) shows the direction of information flows in a set of design activities. Figure 4.16a. is the DSM for the high level clamp hardware. Note that the hardware sub-systems are listed in both the rows and columns of the matrix. An X in the matrix shows that the hardware in the row requires design information from the corresponding hardware piece in the column.



a. Design Structure Matrix

b. Product Structure Diagram

Figure 4.16 - Product Structure Diagram for the Clamp

The product structure diagram is a graphical representation of the design structure matrix. As shown in Figure 4.16b, it is easier to understand than the matrix. The circles are the sub-systems as identified in the function hierarchy and the arrows show the direction of information transfer.

An alternate way of looking at the matrix, or diagram, is that the hardware in the columns affect the corresponding hardware in the rows. Thus if a change is made to one sub-system, the designer can find what other sub-systems will be affected. Alternatively, the matrix can be used to find what hardware should be considered when making changes. These two benefits, when recorded in a computer for easy access, help the designer minimize design mistakes.

Even though the benefit of this module is more apparent when redesigning a product, it can still be effective for original designs. A product structure diagram at the high level shows interactions between lower level sub-systems that design teams should investigate to fully understand the product.

4.5.3.2. QFD Design Matrix

The goal of the second QFD matrix is to relate hardware design parameters to the engineering metrics defined in the first QFD matrix (see Figure 4.17). Both the rows and importance ratings in this matrix come from the columns of the first matrix. However, the function and hardware hierarchies are organized differently, meaning that there may not be a one to one correlation between the function object where the first QFD matrix is and the hardware object where the second matrix is. What happens instead is that the engineering metrics for the rows come from the QFD matrices in all of this hardware's associated function nodes. The QFD matrix then picks off the engineering metrics that relate to this hardware object.

	Importances	Volume	Frame Thickness	Frame Width	Frame Height	Barrel Support Thickness
Weight	3	⊙	△	△	△	△
Bending Moment Held	5		⊙		⊙	
Torsion Held	5		⊙	⊙		
Force Held by Frame	4					⊙
Force Held by Barrel	4					⊙
Importances		27	93	48	48	75
Targets		10 in ³	.2 in	1.5 in	1 in	.4 in

Figure 4.17 - QFD Design Matrix

The operation for the rest of the matrix is the same as described in Section 4.5.2.3. for the House of Quality. The importances are calculated and normalized; the target values can be assigned to attributes; and the design parameters in the columns are propagated, with their importance rating to the next QFD matrix.

4.5.3.3. Hardware Parameters

Hardware parameters are the core of the product model. They are variables that store information like dimensions, material used, and geometric architectures. At high levels of abstraction, the hardware parameters store sub-system target values as designated from the previous QFD matrix. This information can include costs, reliability targets, and relative

geometric proportions. For example, the high level geometric proportions for the clamp frame would be a bounding box with dimensions 1.5" X 4" X 6".

At a lower level of abstraction, the hardware parameters store the actual dimensions and materials used. For example, the length of the frame screw is 3 inches and is made from carbon steel.

Hardware parameters generally come from engineering equations. The constraints specified in the function tree are considered in this module. The textual constraints that weren't used in selecting configuration designs will specify engineering analyses to be considered for piece-part design. The numerical constraints give limit values for use in equations that solve for product parameters. The textual constraints are more of an information management tool that will have greater impact in initial product development, but can be used to a lesser extent in subsequent product designs. Numerical constraints can be useful for either type of design. Just changing the value of a constraint can automatically change the entire design.

Some designers, however, like to use rules of thumb for certain dimensions, especially those that can't easily be modeled with engineering equations. QCAD, like many other knowledge based systems, can help collect and standardize these rules of thumb to make design practices uniform throughout the product.

4.5.3.4. Product Parameter Design

Another way to standardize a design and to reduce the number of thumbs, is to use design of experiments. Product parameter design uses statistics and common sense about the underlying physics to optimize a design with respect to a quality characteristic. The quality characteristic measures the quality of a product or process with respect to customer expectations (Phadke, 1989).

Computers have been used with Taguchi experimentation in two ways: 1) they can be used to model the product, with the simulation being used for experimentation; and 2) they can be

used to guide a designer through the process and to perform the analysis of variance calculations (Quality Engineering Services 1992).

QCAD can be used for both. In the first scenario, the QCAD product model can be created with modifiable control factors and can be used to output experimental results. For the clamp, the frame height, frame width, frame thickness, and hook arc length can be specified as inputs (the frame lip is the short extension where the frame screw goes through the frame). The clamp strength, weight, and ease of use can be output to a report. The design simulation could be linked to the analysis described below, but it would make using QCAD more involved than necessary. It is easy enough to perform the experiments, write that data to a file, and read it back in later. QCAD is flexible, if it is desired to link the robust experimentation and report functions, it can be done.

In the second scenario, QCAD can accept experimental results from a design team, calculate the analysis of variance, and set hardware parameters equal to the optimized results.

4.5.3.5. Material and Process Selection

As the name states, the material and process selection module is used to help a PDT select materials and processes for piece-parts. The use of this module is the same as that for concept selection described in the section on QCAD functions. Material and process selection tends to be more numerically oriented than conceptual design selection. Thus, the use of heuristics to designate manufacturing processes may be more appropriate than a selection matrix.

Figure 4.18 is an example of a material and process selection matrix. The criteria in the rows come from the second QFD matrix. Some additional criteria is also used in the matrix to further distinguish among concepts in terms of cost and production feasibility. The numbers in this matrix are added for use with the computer. When completing this matrix originally, the PDT subjectively evaluates the concepts without the numbers.

	Importance	Iron Sand Casting	Aluminum Sand Casting	Iron Die Casting	Steel Machining
Volume	1.5	S	-	S	S
Frame Thickness	5	S	S	S	S
Frame Width	2.5	S	S	S	S
Frame Height	2.5	S	S	S	S
Barrel Support Thickness	4	S	S	S	S
Production Volume	4	S	S	-	-
Tolerances	2.5	S	S	+	-
Production Cost	5	S	S	-	-
Weight	1	S	+	S	S
Overall Importance		0	-5	-6	-11

Legend
- = -1
S = 0
+ = 1

Figure 4.18 - Material and Process Selection Matrix

The output of this module is a production process that will become a node name in the process hierarchy. In general, there will be a one to one correlation between the hardware nodes and the production process nodes.

4.5.3.6. Concept Review

This concept review matrix works the same way as the concept review matrix in the function hierarchy. The only difference is that this matrix compares hardware to processes,

whereas the other matrix compared functions to hardware. Figure 4.19 is an example of this concept review matrix.

	make clamp support	buy securing mechanism	make swivel mechanism	buy light holding mechanism
clamp support	X			
securing mechanism		X		
swivel mechanism			X	
light holding mechanism				X

Figure 4.19 - Concept Review Matrix

4.5.3.7. Additional Comments

Unlike the function hierarchy that stored a list of all viable, generated hardware concepts, there is no such list production processes. In general, a generated list of processes that can be used to manufacture a product can become rather extensive very quickly. Thus only those choices considered, or explicitly not considered, should be stored in a documented list.

In addition to designating manufacturing processes in the hardware hierarchy, the PDT creates a list of functions that the hardware sub-system is required to perform. These sub-systems are sent back to the next level down in the function hierarchy along with the design parameters and constraints list.

Lastly, parts can be called by any functional requirement or higher level sub-system hardware node. As more and more parts are created, the product structure becomes more and more flexible, which will be more useful in redesigns of the same product.

4.5.4. Process Hierarchy

The process parameter node looks to the hardware node for the defining manufacturing process. The higher level nodes, that reference hardware assemblies will further define an assembly process. The lower level nodes will further define manufacturing operations.

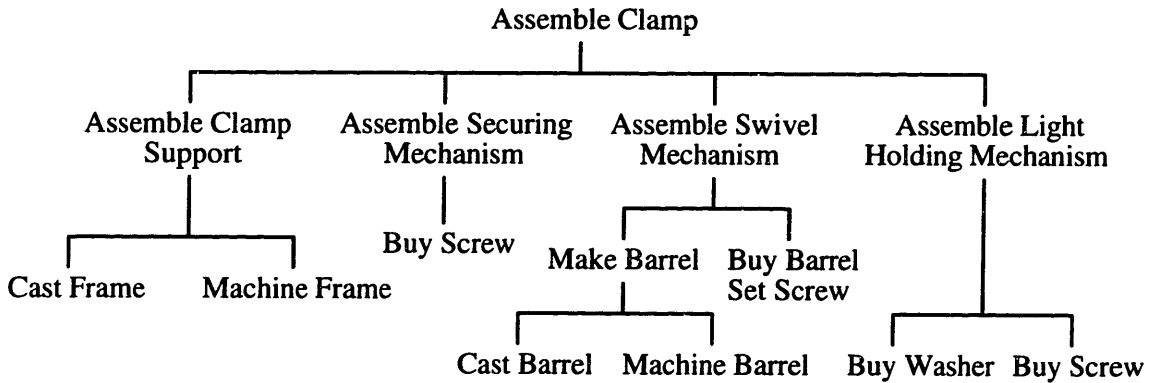


Figure 4.20 - Process Hierarchy for the Clamp

The process hierarchy for the clamp is shown in Figure 4.20. The parts are broken down like an MRP (Materials Requirement Planning) system with the overall process steps occurring from left to right and from bottom to top (The manufacturing floor does not have to use an MRP system to use this product model). Each node will have parameters that further define the process. Some nodes may have QFD process planning and production planning matrices and/or a list of optimal process parameters determined from a design of experiments.

4.5.4.1. Bill of Materials

For all of the assembly process nodes, a bill of materials is generated that itemizes the parts in that assembly. For example, the bill of materials for the total system includes the clamp

support, securing mechanism, swivel mechanism, and the light holding mechanism; each of quantity one. The swivel mechanism would include one barrel and one barrel set screw.

QCAD is also able to generate an itemized bill of materials for all of the piece-part hardware components.

4.5.4.2. Process Parameters

A process parameter contains detailed values about the production process. For a high level production process, one parameter would be the assembly sequence. At lower levels, the parameter could include the speeds and feeds of a milling operation. Other more generic parameters include lead time and tolerances.

4.5.4.3. Process Parameter Design

Taguchi design of experiments is generally not simulated on computer for production processes. It is usually performed on the production floor on the actual machine under normal operating conditions. The parameter design module lists the optimal operating parameters for the production process.

4.5.4.4. QFD Process and Production Matrices

The third QFD matrix on process planning, translates hardware characteristics to process requirements and is used the same way as the QFD matrix described in the hardware section (see Figure 4.21). This matrix is used to understand how different process requirements relate to the functionality of the product with respect to the voice of the customer. The target values are recorded as process parameters. A sensitivity analysis of the importance ratings and the relationships will dynamically effect the importances of certain process parameters.

	Importances	Cast Metal Temperature	Cast Metal Pour Velocity	Cast Cooling Time	Frame Hole Drilling Speed	Frame Hole Tapping Speed	Barrel Screw Hole Drilling Speed	Barrel Screw Hole Tapping Speed
Volume	1.5	◎	◎	◎				
Frame Thickness	5	◎	○		◎	◎		
Frame Width	2.5	◎	○					
Frame Height	2.5	◎	○					
Barrel Support Thickness	4	◎	○				◎	◎
Importances		369	55.5	13.5	45	45	36	36
Targets		500 °F	30 in ³ /s	1 hour	1 in/s	.25 in/s	1 in/s	.25 in/s

Figure 4.21 - QFD Process Planning Matrix

The fourth QFD matrix on production planning translates process requirements to production requirements for use in determining production operation sheets and control charts (see Figure 4.22) (Krininger and Clausing 1991). The rows and importance ratings are from the columns of the third QFD matrix.

Both of these matrices concentrate on process and production operations. QCAD doesn't specifically use either of these matrices in assisting a designer to develop a product. However, they store additional information in the system, for completeness.

	Importances	Furnace Temperature	Ladle Pour Rate	Environmental Temperature	Spindle1 Feed Rate	Tap1 Feed Rate	Spindle2 Feed Rate	Tap2 Feed Rate
Cast Metal Temperature	5	◎						
Cast Metal Pour Velocity	2		◎					
Cast Cooling Time	.5			◎				
Frame Hole Drilling Speed	1.5				◎			
Frame Hole Tapping Speed	1.5					◎		
Barrel Screw Hole Drilling Speed	1						◎	
Barrel Screw Hole Tapping Speed	1							◎
Importances		45	18	4.5	13.5	13.5	9	9
Targets		625 °F	30 in ³ /s	1 hour	1 in/s	.25 in/s	1 in/s	.25 in/s

Figure 4.22 - QFD Production Planning Matrix

4.5.4.5. Additional Comments

The process and production planning phases of EQFD have been further developed by Florusse and Clausing (1992) as part of the Rapid Acquisition of Manufactured Parts (RAMP) project at the South Carolina Research Authority (SCRA). The revised procedure was developed through a cooperative project with MIT, ICAD, and the SCRA.

One goal of the new procedure was to address several concerns that United States manufacturing companies expressed about the third QFD matrix. The most predominant concerns are listed below (Krininger and Clausing 1991):

- “ • unclear implementation of QFD in the product development process,
- undefined integration of other methods, e.g. quality planning methods, in the QFD procedure,
- vaguely described role of blue collar workers in phases III and IV,
- weak management support,
- inadequate documentation of former production processes,
- insufficient QFD training, and
- inappropriate documentation of planning results.”

The revised methodology consists of four main steps, which provide a detailed framework for process planning in EQFD. The procedure emphasizes concurrent development among the design of sub-systems, piece parts, and process planning (Florusse and Clausing 1992):

1. Select the appropriate material and process in terms of cost, function, safety, precision, and possible engineering bottlenecks;
2. Translate part characteristics from the second QFD matrix to process parameters for critical part values;
3. Transfer critical process parameter checks to production; and
4. Assess production difficulties and suggest possible resolutions for the future.

Figure 4.23 shows how these steps integrate with the design process. The next four sections elaborate on how the revised process could be integrated into the QCAD system.

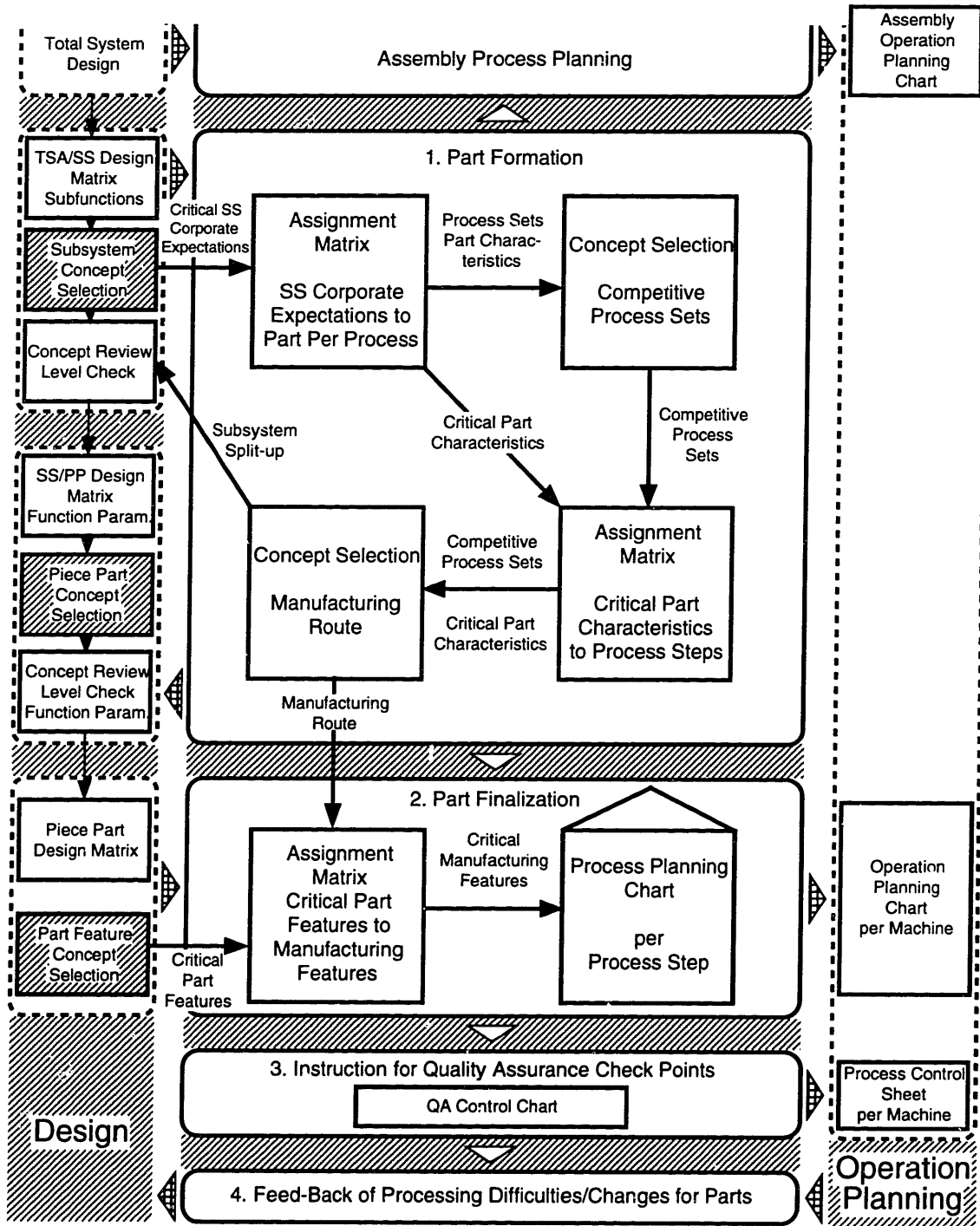


Figure 4.23 - A Detailed Framework for EQFD Process Planning
 (Sontow and Clausing 1993, according to Florusse and Clausing 1992)

4.5.4.5.1. Part Formation

Part formation is concerned with the appropriate selection of sub-system materials and manufacturing processes. As such, this step explores sub-system hardware configurations for different materials and manufacturing processes. This part formation step performs the same function as the material and process selection step described in Section 4.5.3.5. of the hardware hierarchy. QCAD users wishing to follow this revised methodology should substitute the four matrices in this part formation step for the material and process selection matrix in the hardware hierarchy. The rest of this section gives a brief description of the four matrices in this step. For more information, see Florusse and Clausing (1992) and Sontow and Clausing (1994).

The first matrix, the assignment matrix, in the top, left corner of Figure 4.23, is used to explore the feasible material and process options to fabricate a sub-system. For each feasible option, the PDT formulates different piece part configurations and determines critical sub-system characteristics related to the piece parts.

The second matrix, the concept selection matrix, in the top right corner, is used to make a rough selection among the manufacturing processes, against a set of technical criteria including technical feasibility, material cost, and production cost.

The third matrix, another assignment matrix, correlates critical part characteristics against the remaining manufacturing processes. In this matrix, the remaining processes are further defined by designating the types of machines used in the manufacturing process.

The fourth matrix, another concept selection matrix, is used to select the manufacturing process for the piece parts. A new set of criteria is used in this matrix to help the PDT think about critical part characteristics, manufacturing characteristics, engineering bottlenecks, and the interdependence of part characteristics.

4.5.4.5.2. Part Finalization

The goal of this step is to translate critical piece part hardware parameters to manufacturing features and then to process steps. The assignment matrix facilitates the first translation, while the process planning chart facilitates the second translation. Both of these steps occur after a manufacturing process has been selected and as such would be stored in the process hierarchy in QCAD.

4.5.4.5.3. Instruction for Quality Assurance Check Points

This step is concerned with setting appropriate quality control check point positions and measurements. Several tools can be used in this step. Among them are quality assurance charts, SPC charts, and Taguchi optimization methods. These tools are used after the manufacturing process is selected and would be stored in the process hierarchy in QCAD. However, one must be careful of putting too much information into the QCAD system, such that the user would get confused by all of the available data. It may be more appropriate to keep some of the quality assurance information in another database that can be referenced by QCAD.

4.5.4.5.4. Process Feedback

This final step is concerned with providing feedback on the production operations specified in the previous three steps. There are many ways to monitor manufacturing processes to see if they are performing as expected. These include using SPC charts to monitor the flow of material through a production line, and using reports to document machine failures. Both types of information would be stored in the process hierarchy in QCAD. However, it may be more appropriate to document the feedback information in another database that can be referenced by QCAD to keep excess information from confusing the user. QCAD does provide a way to store failure information, in a fault tree, which is relatively easy for a designer to use. The fault tree is described in Section 4.5.5.1.

4.5.5. Miscellaneous

4.5.5.1. Fault Trees

The fault tree is separate from the other three hierarchies because faults, or reasons why the design doesn't work, can occur anywhere in the set of hierarchies. A production process operation may cause a hardware dimension to go out of tolerance, which will then interface incorrectly with other hardware pieces, which then won't fulfill a desired product function. A detailed list of faults can be very useful when trying to determine why the product doesn't work.

Fault trees can become complex very quickly, making paper fault trees long and hard to decipher. An abbreviated fault tree for the clamp is in Figure 4.24. The dotted lines show where more faults are listed.

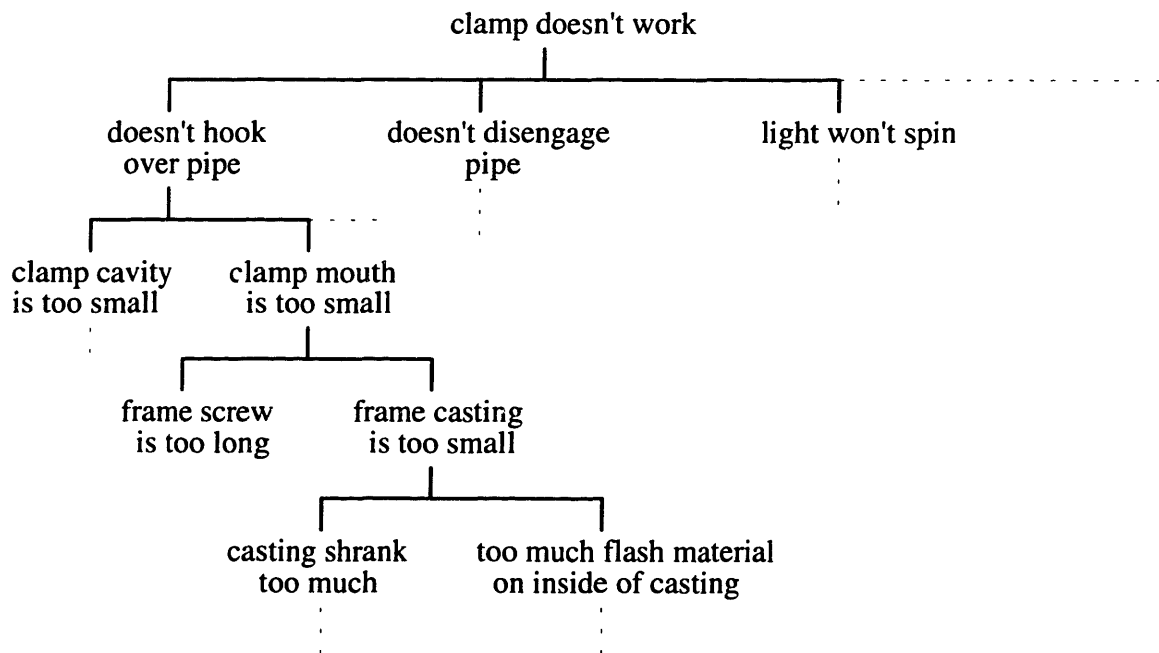


Figure 4.24 - Fault Tree for the Clamp

In QCAD, each node has a fault list that describes causes for why that part would fail. QCAD then has a command that will search the entire product model for fault data, correlate them into a tree structure, and generate a report. The search process could also be limited by key

search parameters, narrowing the size of the generated tree, making it easier for people to decipher.

The computerized fault tree can also help with subsequent product designs, especially if a remark is stored every time a failure occurs in a sub-system. When the product design comes up for review, a designer can call up the faults with the most remarks and study them to create a more robust product.

4.5.5.2. Miscellaneous

In addition to finding fault attributes anywhere in the product model, other parameters such as costs can be compiled, in the same manner, from anywhere in the tree.

In large product models, it can be difficult to keep track of all of the possible places where an error might occur and what triggered those errors to occur. In QCAD, it is possible to specify attributes anywhere in the product model that look for discrepancies and report errors. One type of error could be that a screw doesn't fit into its corresponding hole. Another could be that a manufacturing process was selected that the company does not own. The error locations work in the same way as the fault trees in that they are distributed throughout the product model. However, the error locations are linked directly to attribute values. Once a product model is instantiated, the user can search for the locations with an error. Through the list of errors, the user can locate the object that caused the error and look through the attribute values to discover the cause.

The above discussions talked about the design process proceeding in a straight line from start to finish. Even though this may be the ideal design case, in practice, there are feedback loops. When backtracking in QCAD, the designer just has to move the cursor to the previous node and resume working. If the designer is new or doesn't know what the previous node is, each node has a list of both of its calling nodes to facilitate moving around in the design. For example, the clamp barrel piece-part has two parents, one from the function hierarchy (swivel light) and one from the hardware hierarchy (swivel mechanism).

After going up the hierarchy, the designer can get a broad view of the sub-system by looking back down. The function node, swivel light, would show the user the frame, the barrel, and the barrel set screw. The hardware node, swivel mechanism, would show the user the barrel and the barrel set screw. These different views of the system also help a designer make changes to interfacing parts, to help minimize mistakes.

Furthermore, QCAD is intended to be a flexible way to represent and design a product on a computer. The product representation includes functional features, hardware features, and process features. The flexible structure is expandable and customizable. The basic QCAD framework was presented with several TQD modules. However, users should not feel restrained to using only these modules, other modules that the PDT finds beneficial in design can also be hung onto the QCAD structure.

Likewise, not all functions, hardware pieces, and processes are critical and need to be designed using every available TQD tool. The modules described above can be selectively used, as necessary.

Chapter 5

Discussion of QCAD Implementation

5.1. Overview

Chapter four defined a structure to enable the integration of total quality development design methods with a computerized knowledge based engineering system. The methodology includes a design process that the design team follows without the assistance of the computer and a computer framework that supplements the design process.

The first section of this chapter describes the implementation of QCAD in the ICAD Design Language (IDL) and explains how both the clamp and copier products can be used with QCAD. The second section describes insights gained from creating the system and from modeling both products. The third section describes how a product development team could use QCAD for design.

5.2. QCAD Case Studies

The QCAD system was developed on a Sun SparcStation 1+, with 64 MB RAM and a 1.05 GB hard drive running SunOS 4.1.3. All programming was done under ICAD, Version 4.0 running the Unigraphics II solid modeler, parasolids, Version 4.2.81. No changes to the basic IDL program were necessary to implement QCAD, however, some modifications would be necessary for a commercialized version of the system.

The IDL product models for the case studies presented in this section are stored on the disk in the pocket on the back cover. Instructions for using the product models are in Appendix A.

A brief overview of the most important ICAD concepts is given in Appendix B. Both object oriented programming and IDL specific concepts are discussed.

As described in Appendix B, there are two methods of interacting with the ICAD system: one through the programming language interface and the other through the browser. As such, the QCAD implementation had two objectives: (1) make product models easy to implement in the programming language, yet (2) make it easy to explore product variations in the browser. In the initial stages of the research, the product models were programmed in a specific manner, with limited parametric capability. As the QCAD concepts proved viable for a specific case, the product model was reprogrammed to be more generic, enabling more flexible use of the product model. The generic structure is described in Section 5.2.1.

5.2.1. Generic QCAD Structure

An ICAD product model organizes defparts into a tree structure, with a root node, branches, and leaves. For the QCAD implementation, the root node is the QCAD defpart. It organizes the highest level information in the product model. As such, the QCAD defpart contains the selected function to be examined, the hardware embodiment of that function, and the production process used to create that hardware. The QCAD defpart is the foundation for the function, hardware, and process branches of the tree. These branches are large and could be considered as trees in their own right. Chapter 4 described these three branches as hierarchies.

A fourth branch, stemming from the QCAD defpart is a miscellaneous defpart that takes care of administrative functions for the product model. The miscellaneous defpart is also the starting node for the fault tree, or fault branch in this case.

Figure 5.1 shows a QCAD defpart, with its four children, in the QCAD browser. This figure displays the underlying QCAD structure without a product, hence no graphical image is displayed in the graphics viewport, on the left side of the picture. In the figure, the four branches are referred to as children, because only one defpart in each branch of

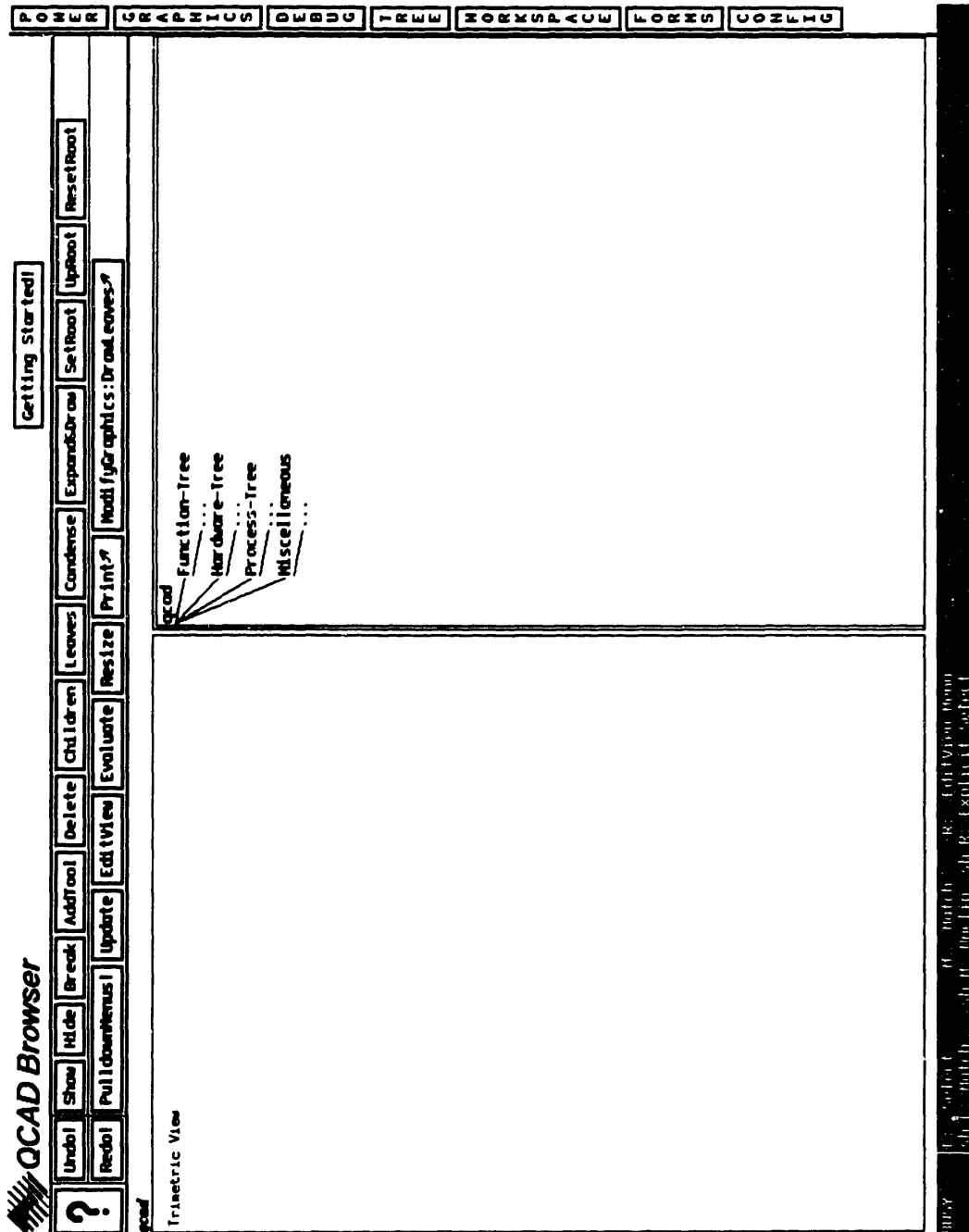


Figure 5.1 - QCAD Browser

the tree is displayed. In all, five defparts are shown: the root, QCAD defpart, the total system function defpart, the total system hardware, the total system process defpart, and the miscellaneous defpart. Of course, these defparts are the beginning of the four branches, as is indicated by the dots.

Although displayed separately, the four hierarchies are interconnected, as shown in Figure 5.2. All of the hierarchies are coordinated through the QCAD defpart, represented by the large rectangle. Each of the QCAD defpart's four children are represented as four large boxes that are half in and half out of the QCAD rectangle. This signifies that the QCAD defpart defines its children and passes to them some defining attributes. After the information is passed down, the children become parents and define new attributes and children. In Chapter 4, there is a discussion about iterating between the function and hardware trees to further define sub-functions; and a discussion about iterating between the hardware and process trees to further define sub-hardware. The arrows in Figure 5.2 represent this iteration.

The arrows going from left to right show that the hardware defpart gets information from the function defpart and that the process defpart gets information from the hardware defpart. The two arrows in the QCAD outline are directly managed by the QCAD defpart. These arrows are at the total system level of the product model.

The arrows outside the larger rectangle of the QCAD defpart are not actively managed by the QCAD defpart. The definition of the total system defpart guides which sub-system defparts will be created. The smaller boxes represent the sub-system level of the product model. Each sub-system is a child of the larger total system defpart. In object oriented programming, each child inherits attributes from its parent, which in this case is the total system defpart. However, in QCAD, as the curved arrows show, information flows not only from the parent, but also from the previous design step.

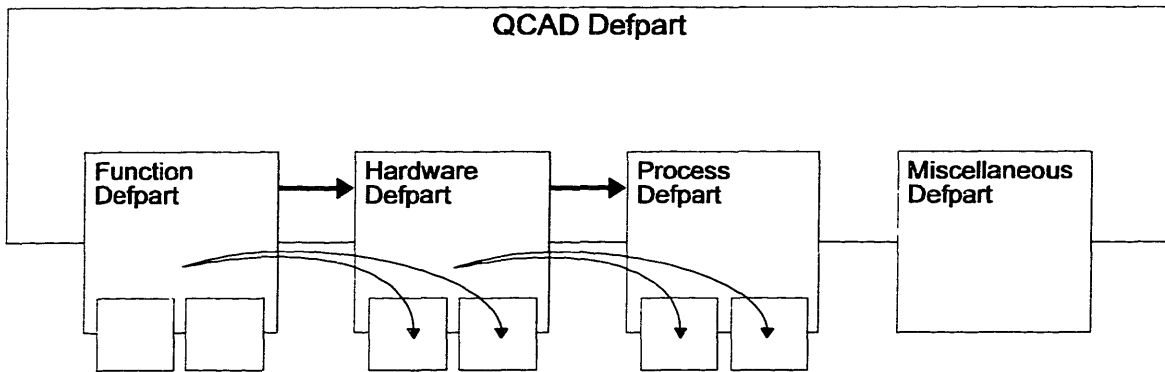


Figure 5.2 - QCAD Defpart Interconnectedness

The function tree has information about both functions and hardware. Thus, it is able to define its children without specifically referencing the hardware tree. The hardware tree however looks to the function tree for its definition. The process tree looks to the hardware tree for its definition. At the highest level, this definition is performed by the QCAD defpart. At lower levels of abstraction, this definition is performed by the corresponding function or hardware defpart.

The miscellaneous defpart works a little bit differently. The information stored in the miscellaneous defpart, such as the bill of material and fault tree, can come from any defpart in the function, hardware, or process trees. As such, all of the information from the three hierarchies is passed up to the QCAD defpart and sent to the miscellaneous defpart, as shown in Figure 5.3. In the figure, imagine that the function, hardware, and process trees are several layers deep and that the information can come from the bottom-most defpart up to the QCAD defpart and over to the miscellaneous defpart. The miscellaneous defpart is half in and half out of the QCAD box because the information processing is performed in the miscellaneous defpart, and not in the QCAD defpart.

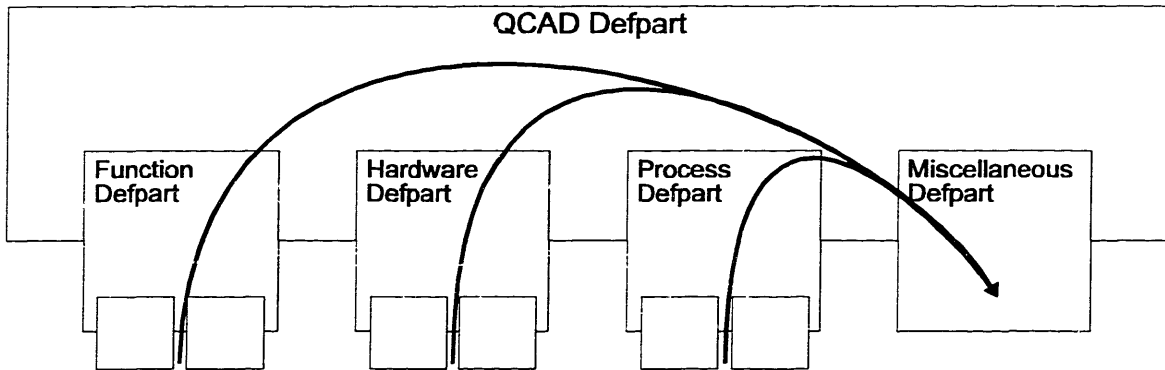


Figure 5.3 - QCAD Miscellaneous Defpart Interconnectedness

<u>Function Defpart</u>	<u>Hardware Defpart</u>	<u>Process Defpart</u>	<u>Miscellaneous Defpart</u>
<ul style="list-style-type: none"> • Function Matrix • Function Net • House of Quality • Constraint Attribute • Fault Attribute • Error Attribute • Concept List • Pugh Concept Selection • Concept Review Matrix 	<ul style="list-style-type: none"> • Design Structure Matrix • Product Structure Net • QFD Design Matrix • Hardware Attribute • Taguchi Attribute • Fault Attribute • Error Attribute • Material and Process List • Material and Process Selection • Concept Review Matrix 	<ul style="list-style-type: none"> • QFD Process Matrix • QFD Production Matrix • Process Attribute • Taguchi Attribute • Cost Attribute • Fault Attribute • Error Attribute 	<ul style="list-style-type: none"> • Numeric Constraint Attribute • Textual Constraint Attribute • Bill of Materials • Total Cost • Error Location • Fault Tree

Table 5.1 - QCAD Defpart Contents

Table 5.1 lists the design process components that were included with each type of defpart in the QCAD system. This table is comparable to Figure 4.5, which shows an overlay of the design process on the QCAD structure. The table is provided to show the defpart type where design components and attributes are defined. An attribute is a variable that takes on a value. A component refers to a design method that involves several attributes. For more information on a specific component or attribute, see Chapter 4 or Sections 5.2.2. and 5.2.3.

Furthermore, Table 5.1 lists a wide variety of design components and attributes to be used in the system. These capabilities had to be implemented in such a way that they would not be a burden to the programmer. As such, the QCAD system uses mixins, functions, and methods to assist in developing a product model.

A mixin is an easy way to customize a defpart. The defpart takes on all of the properties of the mixin, but can modify those properties if necessary. Each defpart in the function, hardware, and process trees uses a function, hardware, or process mixin respectively. These mixins ask for user inputs and perform the calculations for the design processes listed in the table above. Each defpart is customized with specific information used by the mixin. The mixin provides a structure for adding design information to the defparts. The design process steps in each mixin will only be used if the result from that design process step is requested by the system or by the user.

The mixins provide a solid base on which to build each defpart. However, some customizations cannot be created with mixins alone. Each child in the three hierarchies is selected from a set of possible children. Each child has different parameters that it requires for its definition. To facilitate the selection of children for each hierarchy, the QCAD system uses functions that compare the child's name to a list of desired children. If there is a match, the child is created and the information is passed down. If the child is not on the list, it is not created. For example, if there are ten possible children for a hardware defpart, the child selection function will be used to search a list of hardware designated by

the function hierarchy to create only the children that are both on the list and defined for the hardware defpart.

Lastly, methods are used to assign constraint and Taguchi attribute values to hardware and process attributes. The method is actually defined in the hardware and process mixins, which cuts down on the clutter in the actual hardware and process defparts. Both the constraint and Taguchi attribute lists are specified in certain locations of the product model, which the methods access. The constraint information is collected from the function hierarchy and stored in the miscellaneous defpart. The Taguchi attributes are stored in the individual defpart. Once the appropriate list of attributes is found, the method compares the designated attribute to the list and, if found, assigns the value to that attribute.

5.2.2. Clamp Product Model

This section describes the clamp product model developed in the QCAD system. Chapter 4 has a description of the clamp product and presents many examples of the design methods using the clamp model. Figure 5.4 shows the QCAD browser, with a picture of the clamp on the left and the four inter-connected hierarchies on the right. The top branch of the tree is the function tree; the second is the hardware tree; the third is the process tree; and the last is the miscellaneous tree. On the actual computer screen, each tree is a different color.

The levels of abstraction go from left to right, with the total system level on the left and the piece parts on the right. The part feature hardware nodes are not shown.

The clamp product model in QCAD was expanded from a hardware only product model. A QCAD model does not have to be created in this way. However, it is assumed that many ICAD users already have working hardware product models. This was a test to make sure that the QCAD system could be retrofit on top of existing hardware product models.

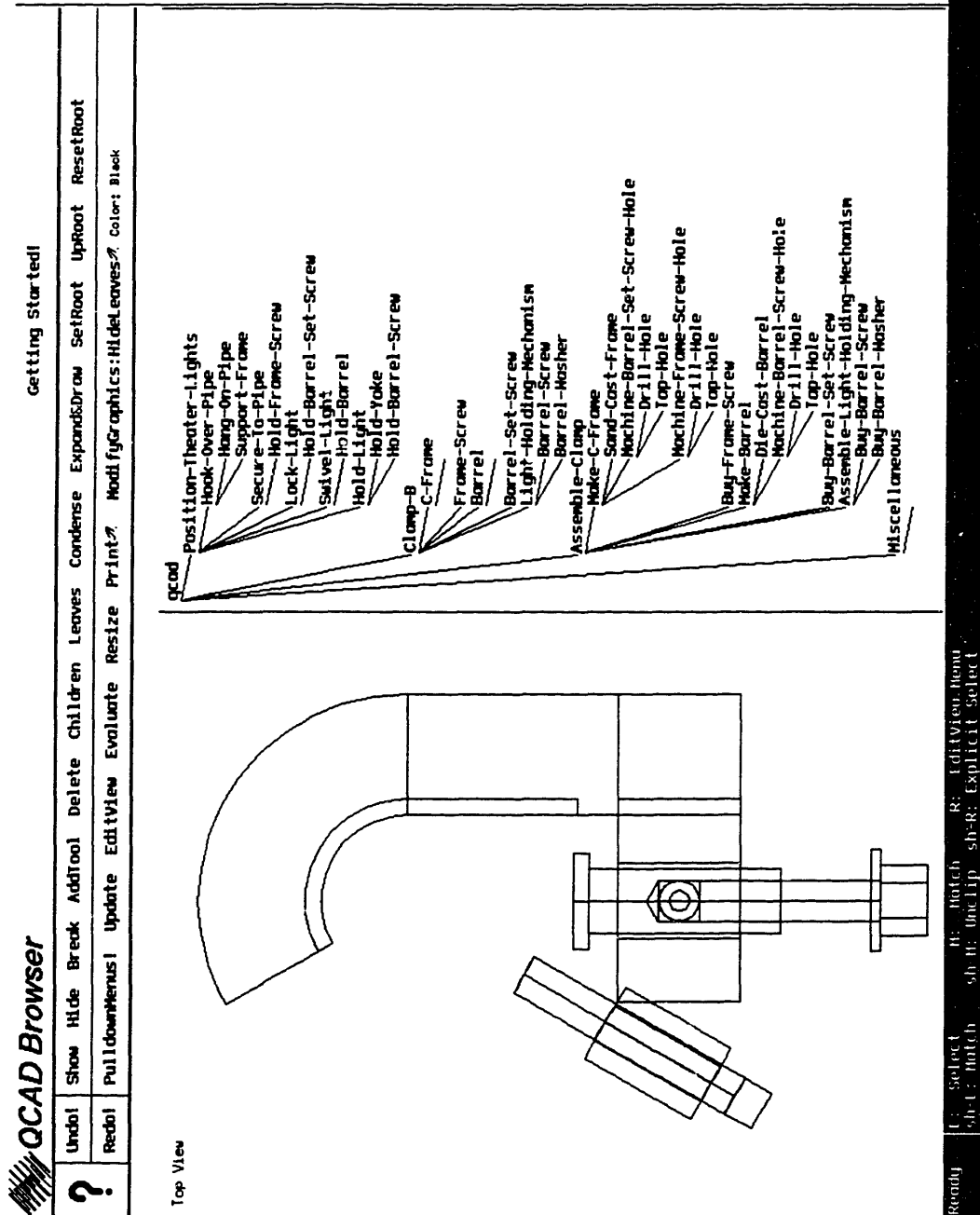


Figure 5.4 - QCAD Clamp Product Model

Figure 5.5 shows an example of geometric parametric capability. This is simulated by using only the hardware tree to create the change. In the hardware tree, there are formulas that calculate the height of several components based on the size of the pipe that the clamp will fit around. The graphics viewport in the figure shows what happens to the height of the clamp when the pipe diameter is changed from 2.1 inches to 4 inches. Compare the picture of the clamp in Figure 5.5 with the picture in Figure 5.4. On the right side of Figure 5.5, a modifiable attributes inspector window is shown in the middle of the hardware tree. The window displays all of the attributes, associated with the clamp defpart, that the user can change while the product model is instantiated. Note the new value for the :max-pipe-diameter attribute.

In the QCAD system, there are two ways to change the :max-pipe-diameter attribute, depending on the desired effect on the rest of the product model. The maximum pipe diameter attribute is a numeric constraint. It was initially defined in the function tree. If the user wants to change the value of that attribute for the entire product model, then it should be changed in the function tree. If the user wants the change to affect only the hardware and process trees, then the change should be made in the hardware tree as described above. In Figure 4.4, the QCAD structure, it is easy to identify what parts of the product model will be affected by a change in another part of the product model—a change in one object affects the definition of the objects below it and to the right.

Figure 5.6 shows the House of Quality for the clamp. The matrix is stored in the *position theater lights* function defpart. It appears on the screen through a modifiable attributes inspector window in the same way that the maximum pipe diameter attribute was displayed in the previous example. The window in the middle of the screen shows the contents of the QFD1-matrix attribute. The QFD matrix is a list of lists. As displayed, the first list is the voice of the customer with importance ratings; the second list is the engineering characteristics; the third list is the relationship matrix; the fourth list is the

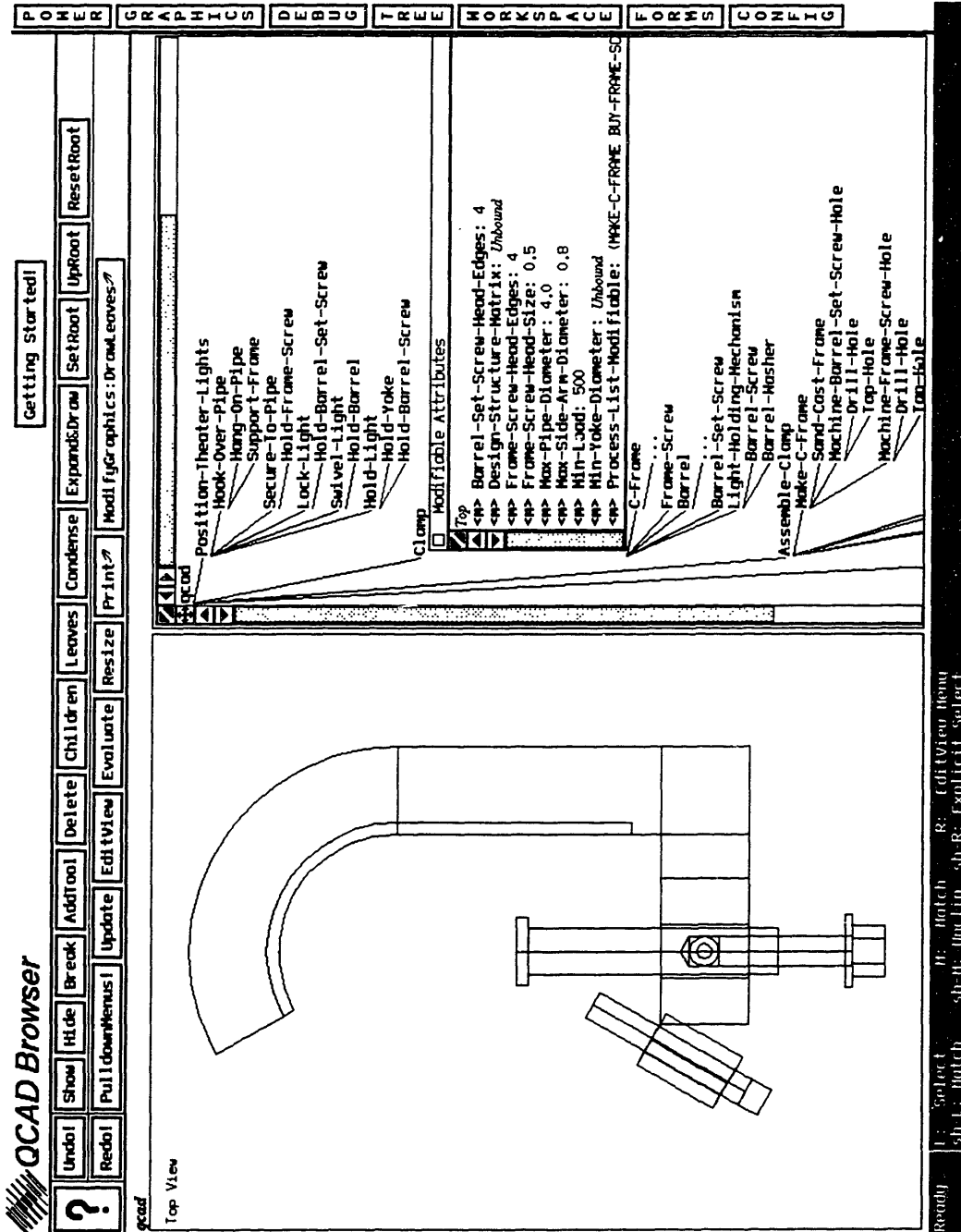


Figure 5.5 - Parametric Clamp Instance

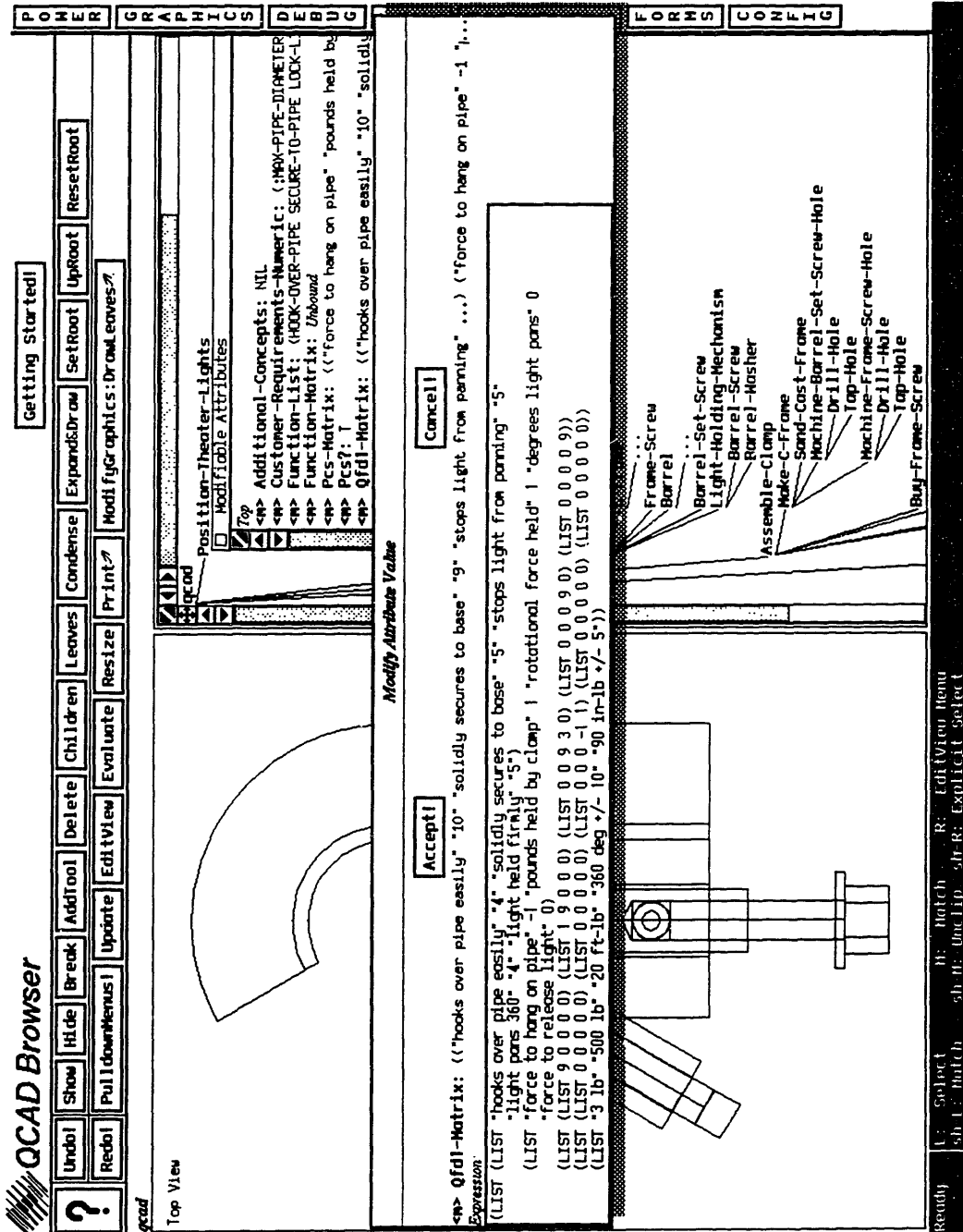


Figure 5.6 - Clamp House of Quality

correlation matrix; and the fifth list contains the target values. This window allows the user to change any part of the matrix. The importance ratings for the voice of the customer are stored nominally in the :qfd-matrix. A different attribute stores the actual voice of the customer in order to facilitate programming and to simplify use by the end user. The product model also calculates the column importance ratings by multiplying the voice of the customer importances by the values in the relationship matrix. The final importance ratings can be viewed by the user, but not changed.

Figure 5.7 shows the parametric capability using the QCAD routines in ICAD. This example shows the use of QFD and concept selection matrices in QCAD. By changing a few customer importance ratings the entire product changes. This is an example of the type of dramatic change in hardware that is possible with the QCAD system. Before, the hardware was a clamp, now it is a floor mount. This change happened at the total system level as a result of a change in customer priorities. Changes are also possible at the sub-system and piece part levels.

This past example is an example of a sensitivity analysis. The voice of the customer importance ratings were modified and the hardware changed. There are a number of different attributes in the QCAD system that can create a concept change, which is different from a parametric value change, like that shown in Figure 5.5. Among the attributes that can affect a concept change are the voice of the customer importances, the QFD relationship matrices, the concept selection row weights, the concept selection matrix relationships, the material and process selection matrices, and the concept review matrices.

The miscellaneous defpart creates the opportunity for another type of sensitivity analysis, based on total cost. The cost is determined by the material and process used to create and assemble hardware pieces. All of the cost attributes are defined in the process tree. The total cost attribute in the miscellaneous defpart adds up all of the costs in the process tree.

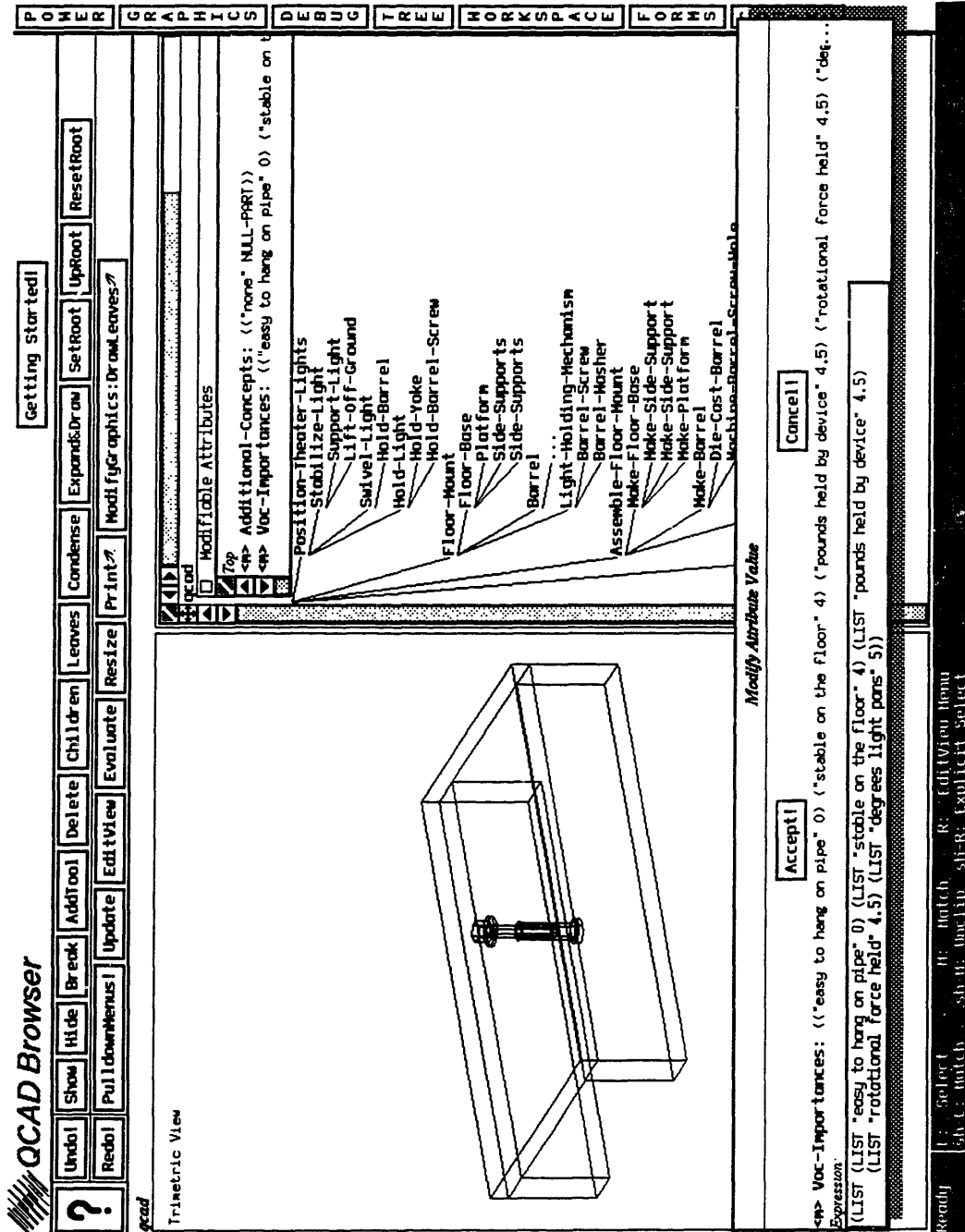


Figure 5.7 - Floor Mount

Figure 5.8 shows an example of the total cost attribute. For an iron sand cast clamp frame, the clamp has a total cost of close to \$29. The inspector window under c-frame on the right side of the screen shows that the frame material is iron; The process tree identifies a sand cast frame; And the inspector window under the miscellaneous defpart shows the total cost.

In Figure 5.9, a different material and process is selected and the total cost goes up to about \$77. For this sensitivity analysis, the material and process selection matrix was over-ridden by the user. In the top right corner of the figure, a choice attribute box shows the user what materials and processes can be selected. In this case, the user chose the steel machining option, resulting in a larger cost. Note that the process matrix now shows a machined clamp frame instead of a sand cast frame.

The frame pictured on the left side of the figure did not change with the new material and process selection. It could have changed. There is a way to link the hardware to the selected material and process, however, for purposes of demonstrating the functionality of the QCAD system, it was not deemed necessary to create a detailed representation of the hardware. More detailed hardware can be developed in the ICAD system.

Figure 5.10 shows more detail of the miscellaneous defpart. The function, hardware, and process trees are condensed to create room for the miscellaneous defpart inspector window. In the top of the window is a list of the bill of materials for the clamp. All six parts are listed there. The bill of materials is one list of a set of routines that search all or part of the function, hardware, and process trees to collect information for a report. For consistency, all of the search and report generation routines are located in the miscellaneous defpart.

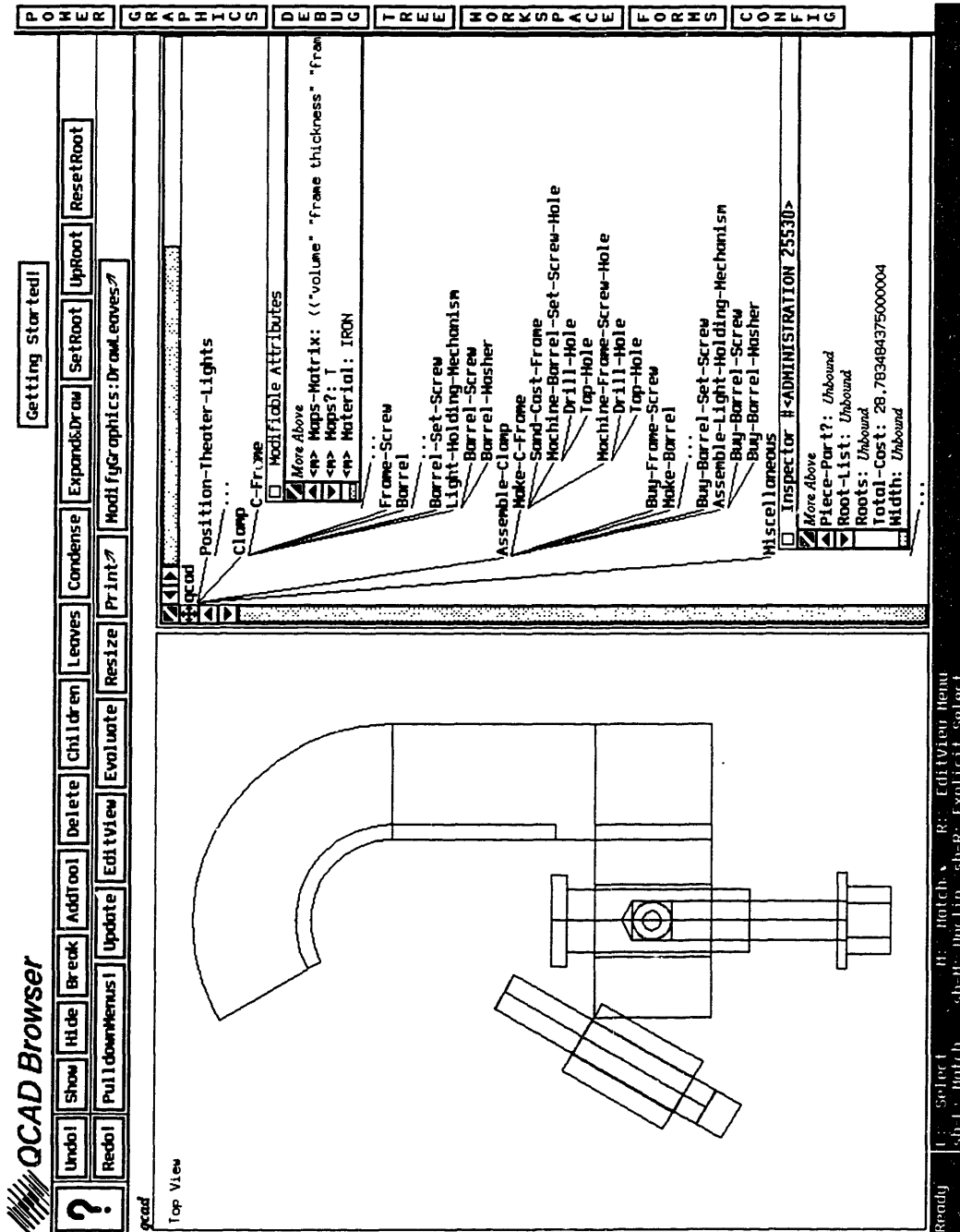


Figure 5.8 - Sand Cast Clamp Cost

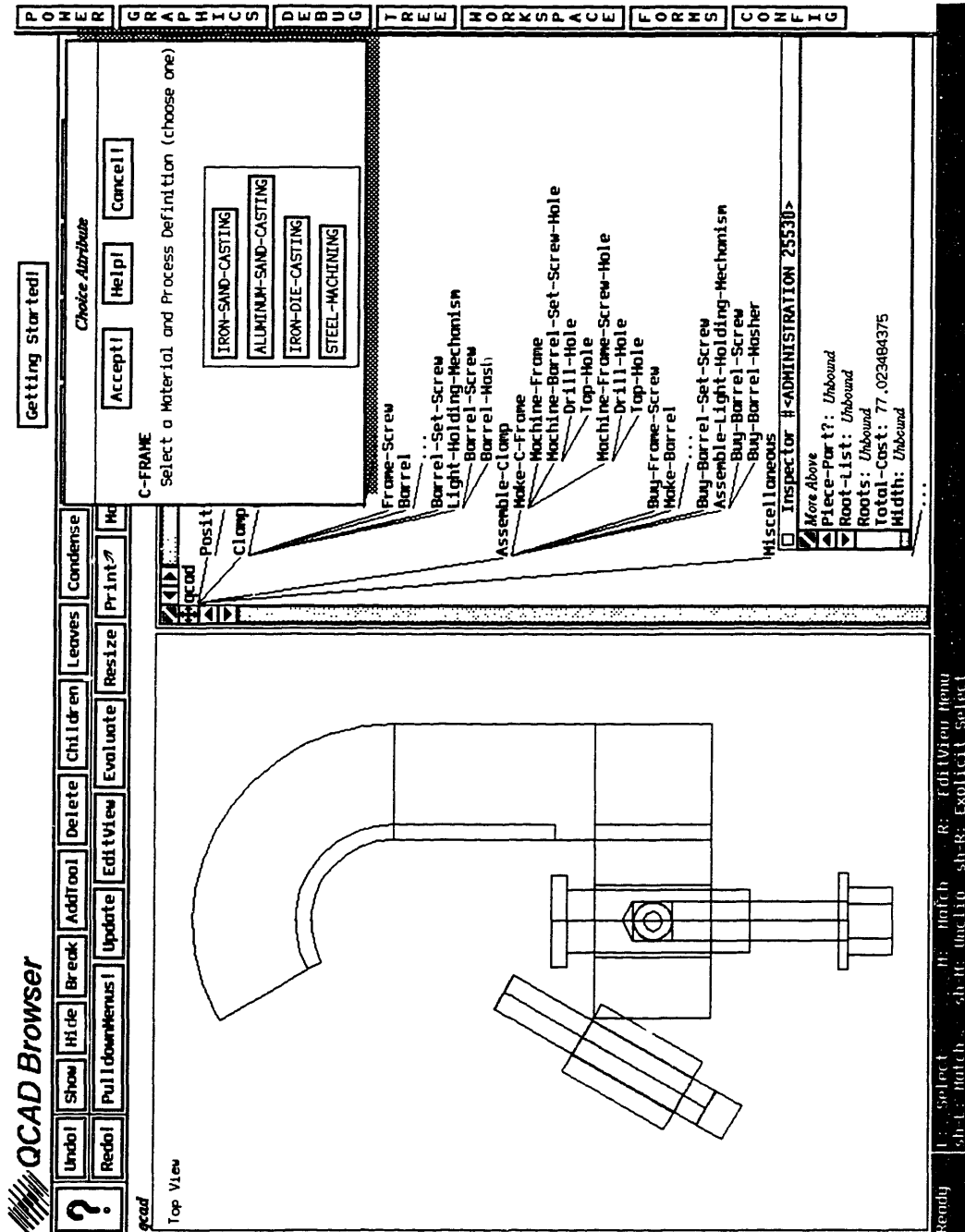


Figure 5.9 Machined Clamp Cost

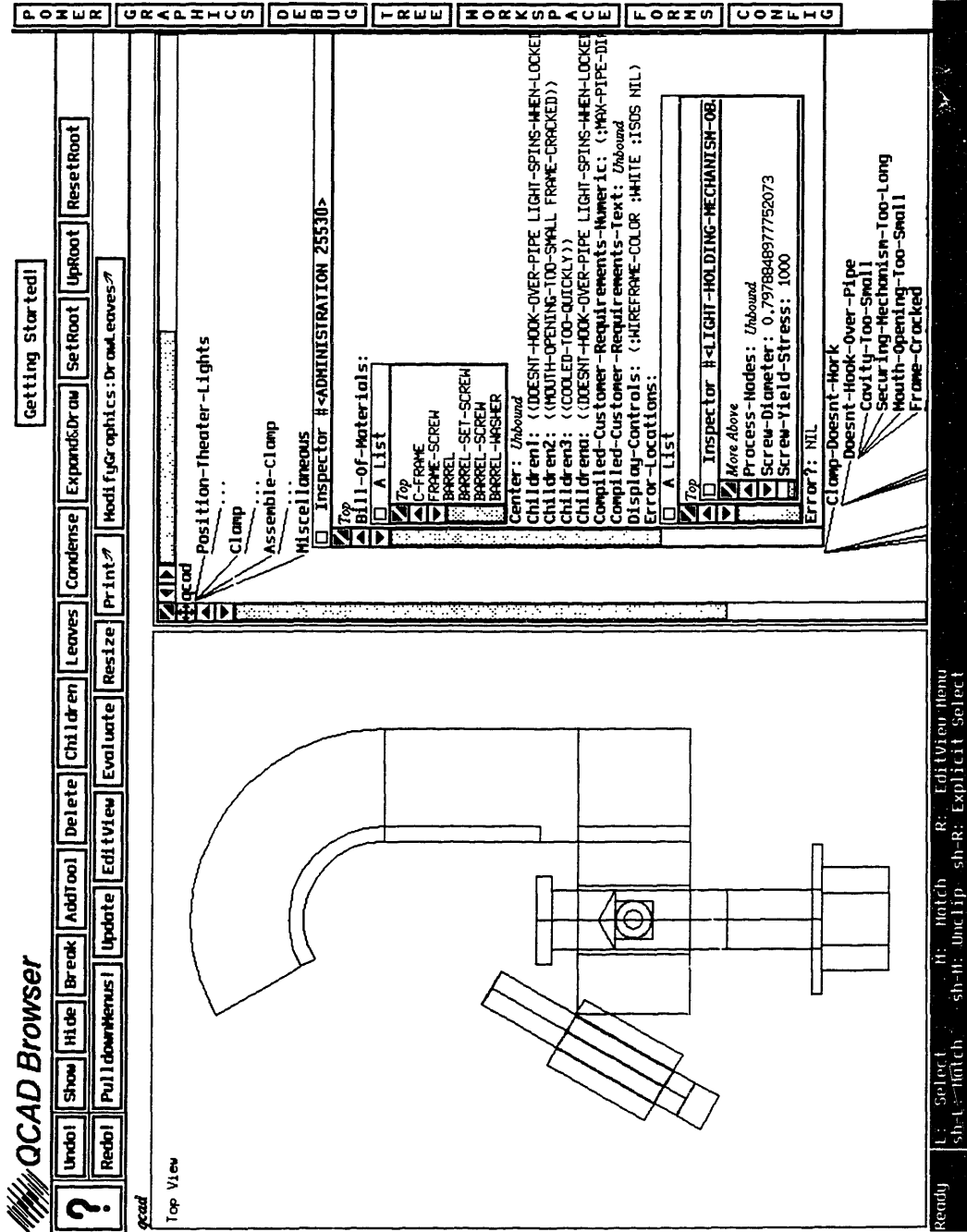


Figure 5.10 - Miscellaneous Defpart

A little farther down in the inspector window of Figure 5.10 is the error locations attribute. As described in Section 4.5.5.2, there are attributes throughout the product model that signal errors. In the ICAD system, it makes more sense to instantiate the entire product model before looking at product specific errors. The alternative is to stop the product model instantiation process to signal the error. However, once stopped, the error must be fixed and the model reinstated, which can be a very time consuming process for large product models. Additionally, letting the model fully instantiate facilitates parametric exploration of the product model, by letting erroneous instances be created.

The following is an example of how the error tracking notification system works. In Figure 5.10, an error is identified by the error-locations attribute. It signals that the error is in the light holding mechanism defpart. An inspector window is opened in the error-locations attribute that looks at the light holding mechanism defpart. After some exploration of the attribute values, the error is found. The barrel screw diameter is larger than the hole that the barrel that it fits into. The screw is .8 inches wide, whereas the hole is .5 inches wide. A knowledge of the product model indicates that the large bolt diameter is caused by the small value for the yield stress. Now that the error has been identified, it can be corrected. Note that it is difficult to find the problem by just looking at the graphic picture and that some error indicator is necessary.

On the bottom, right of Figure 5.10 is the start of a fault tree for the clamp. The rest of the tree runs off the bottom of the screen, but can be scrolled into view. The fault tree is created by linking fault data embedded in the three hierarchies of the product model. The miscellaneous defpart collects the fault attributes and organizes them into a tree structure.

5.2.3. Copier Product Model

The clamp is a rather simple product, good for describing the QCAD methodology and the computer implementation. However, most products are more complex than the clamp, with more parts, more sub-systems, and many more interactions among the parts and sub-systems. Even part features are affected by the choice of geometry and material for other interfacing parts. The idea of this case study is to show that the QCAD methodology works for a complex product.

The copier example has the same functionality as the clamp model, even though it has roughly 1000 times as many hardware piece parts. The same methodology was used to create the copier as the clamp. No rework in QCAD capabilities was necessary to implement the copier. However, since it was the second case study using the ICAD design language, the code is a little more compact.

Figure 2.2a shows the total system architecture for a Xerox 1075 copier, which makes about 70 copies per minute (Clausing 1994). The copier has four main sub-systems associated with: (1) positioning the original document to be copied, (2) positioning the blank sheet to receive the image, (3) transferring the image from the original sheet to the blank sheet, and (4) providing support for the first three sub-systems.

In QCAD, the copier was modeled two levels deep. The first level is shown with graphics in Figure 5.11. The dots under each heading of the tree on the right side of the figure indicate that that node has children. In the graphics viewport, each sub-system is represented by a box that contains all of the sub-systems and piece parts that compose the assembly. More detailed graphics are used at lower levels of abstraction. Figure 5.12 shows both the first and second level sub-systems for the copier. Due to the large tree size, the browser window was modified to show three columns of the tree. The first column shows the function tree; The second column shows the hardware tree; The third column shows the process tree. Due to its length, only part of the function tree is shown. There are no dots under the hardware and process tree nodes, because no further

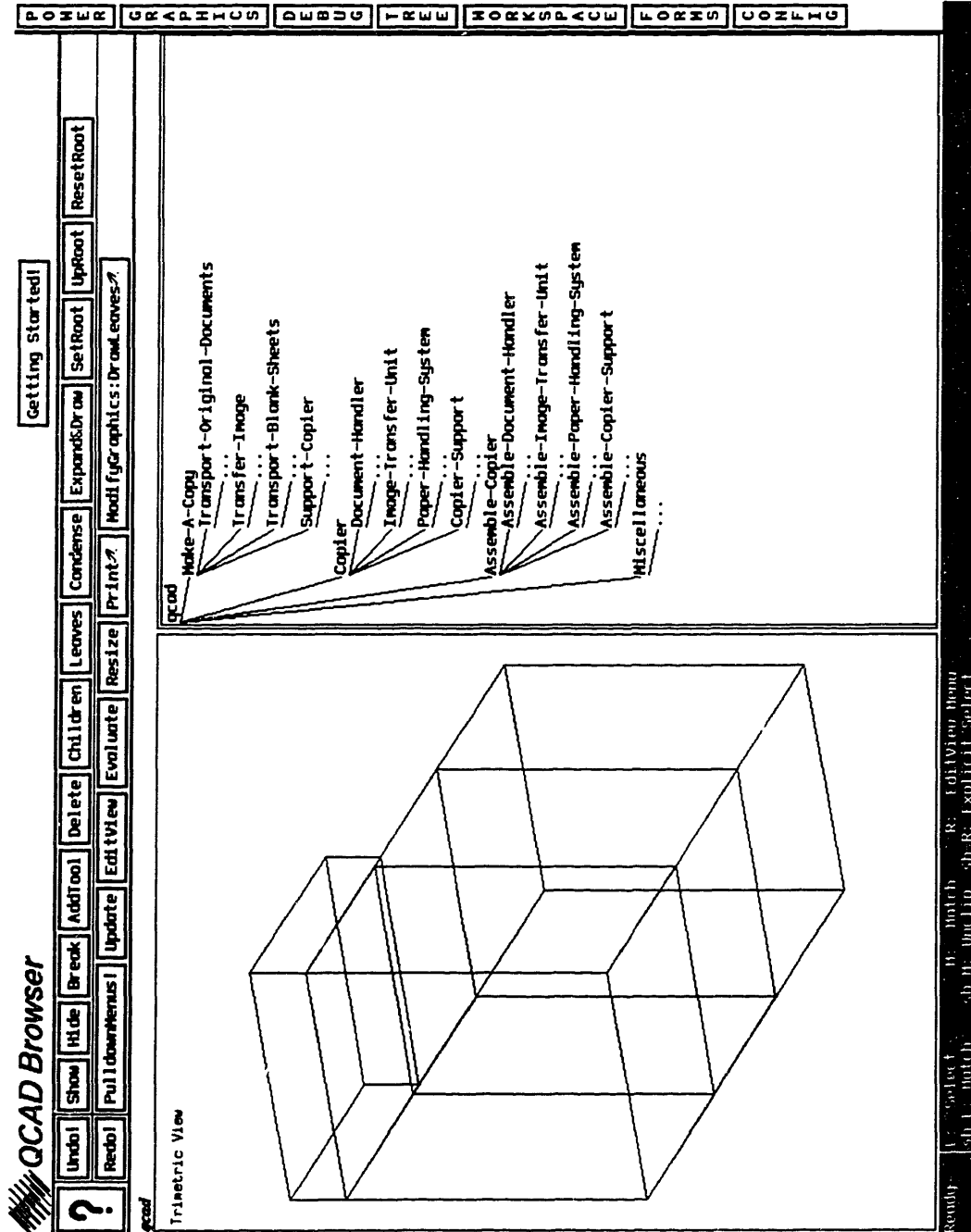


Figure 5.11 - QCAD Copier Product Model

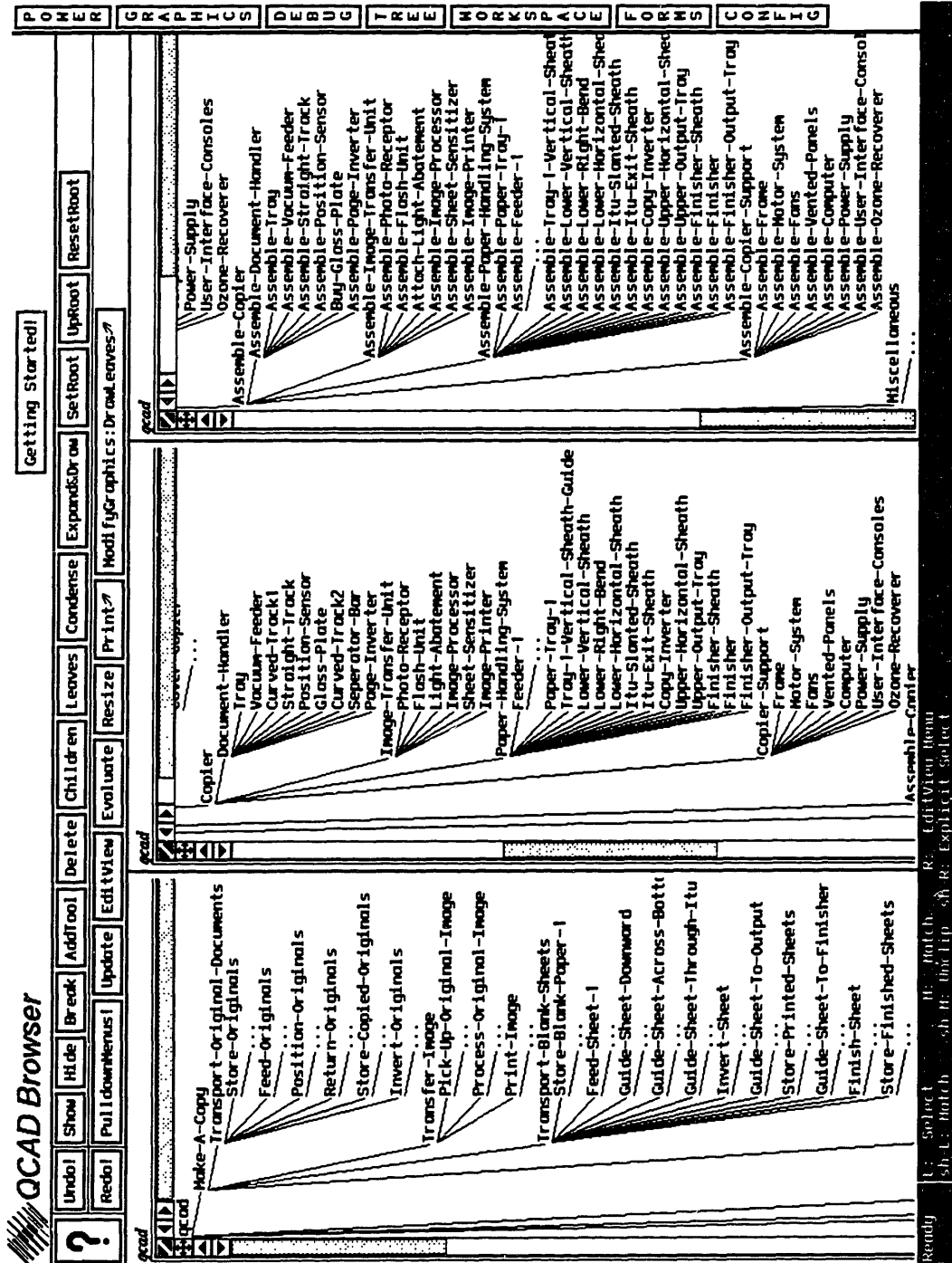


Figure 5.12 - Copier Product Structure Tree

hardware or processes are defined for those trees (except for the feeder). However, there are dots under the function tree nodes. Even though no further functions are defined for the function tree, the dots represent the ability for the user to add sub-functions to the tree and thus augment the product model.

In addition to modeling the high level sub-systems, the paper feeder sub-system was modeled in detail. Figure 2.2b shows the paper feeder for the Xerox 1075 copier. The QCAD model is shown in Figure 5.13. For ease in understanding the screen display, only the hardware tree is shown. Additionally, each of the top level paper feeder sub-systems are several more levels deep. However, for clarity, the multitudes of piece parts and part features are not shown in the tree. The piece parts and part features are shown in the graphics viewport, which has been changed to show four different views of the paper feeder.

In the hardware tree in Figure 5.13, only one paper feeder and one paper feeder tray is listed. However, the 1075 copier has two paper feeders, each with its own tray. To add these two sub-systems to the copier, the user only has to add two functions to the function tree. Of course, this is only a partially accurate statement, since the function defparts and their related hardware and process defparts must already exist in the system. In this case, the *feed sheet* and *store sheet* functions are already defined. The user only has to add them to the function list under the *transport blank sheet* function.

Figure 5.14 shows the result of adding the two functions—a copier with two paper feeders. The function tree displayed on the right side of the figure, shows two functions for storing blank paper and two functions for feeding sheets. To add the two additional functions, the user modified the :function-list attribute in the modifiable attributes inspector window, shown directly beneath the *transport blank sheets* function node. Note that both feeder sub-functions are identical. Even though the hardware tree is not shown,

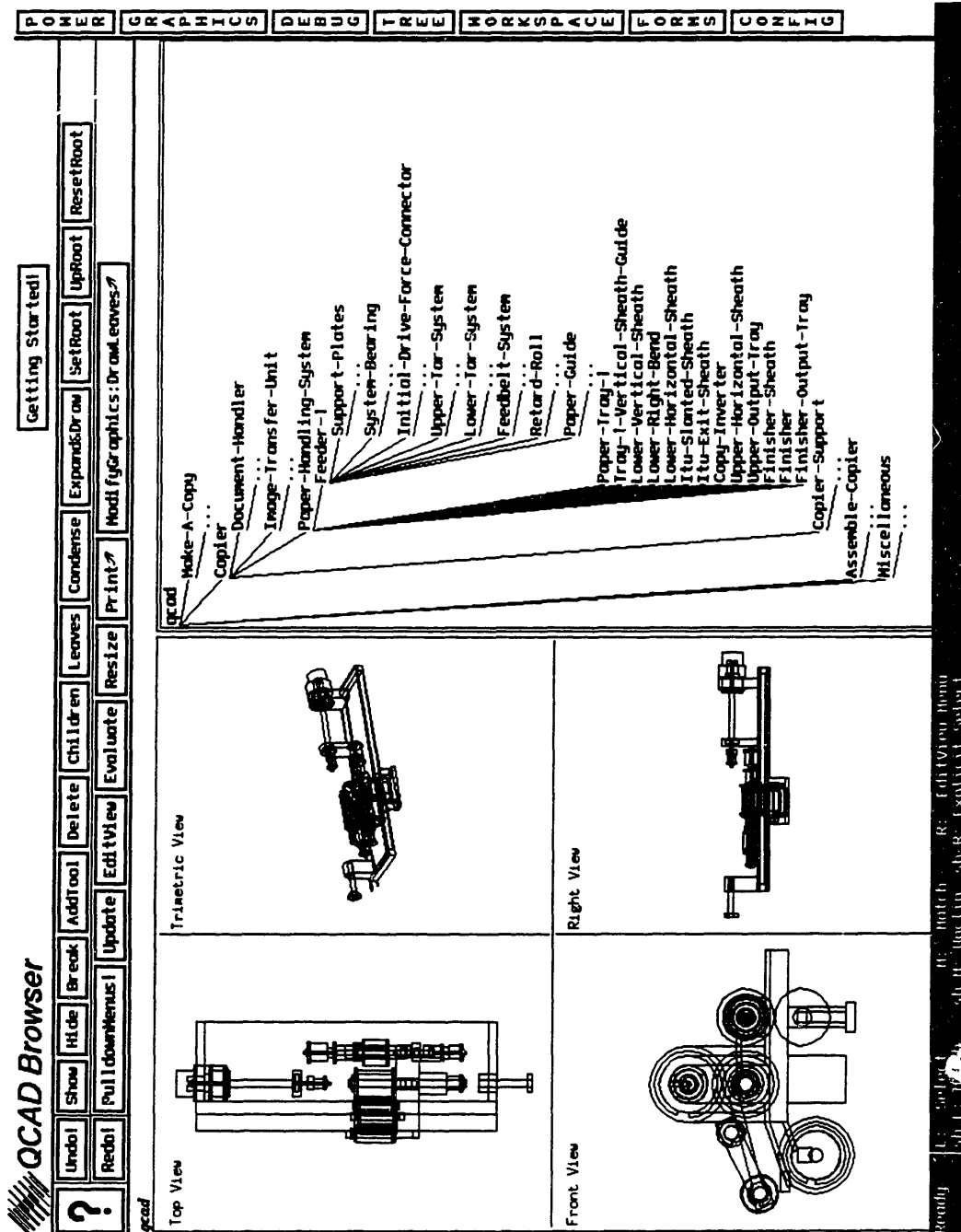


Figure 5.13 - Paper Feeder

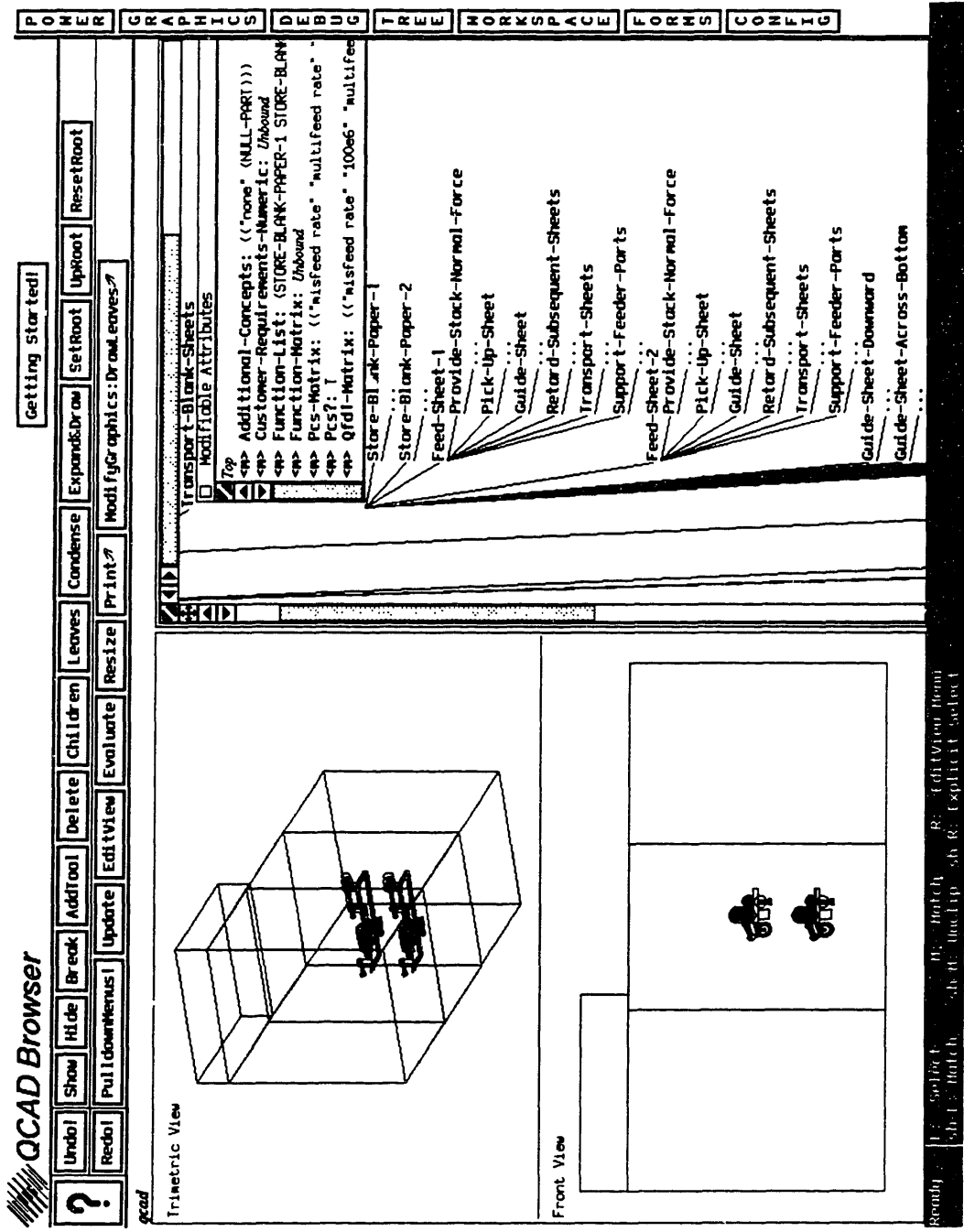


Figure 5.14 - Copier with two Paper Feeders

the graphics viewport on the left side of the figure shows identical paper feeders. The second feeder is positioned below the first feeder.

The hardest part of modeling the copier was creating and positioning the hardware components. The hardware and positioning for the clamp was relatively simple, since there were only six pieces and about two levels of abstraction. The copier on the other hand is about five levels deep, with positioning of parts occurring at different levels of abstraction. The hardware geometry and positioning is inherent of the ICAD system and does not affect the QCAD framework other than QCAD is written on top of IDL.

An interesting relationship between function and hardware can be seen by comparing the function net with the hardware definition for a particular function. The paper feeder connects to the paper tray by resting on two bearings. These bearings allow the feeder to swivel to facilitate jam clearance and to enhance paper delivery. The swivel mechanism can be easily seen in figure 5.13. In the lower right section of the graphics viewport, labeled "Right View," two bearings can be seen extending beyond the length of the feeder. These bearings connect to the top plate of the feeder and rest on the paper tray.

Figure 5.15 shows a close up view of the front bearing assembly. The graphic viewport is split into two sections: one which shows the full front view of the paper feeder support plates, and one which shows a close up view of the bearing assembly attached to the top plate. The front bearing assembly consists of a bearing and a shaft. The right side of Figure 5.15 shows the function net for the *connect to system bearing 1* function, which defines the front bearing assembly. The function net is read, "The nature of how the function *rotate freely* is satisfied depends on the nature of how the function *attach to feeder* is satisfied." In this case, a solid cylindrical shaft is used to attach to the feeder, and a bearing is press fit on the shaft to enable rotary motion.

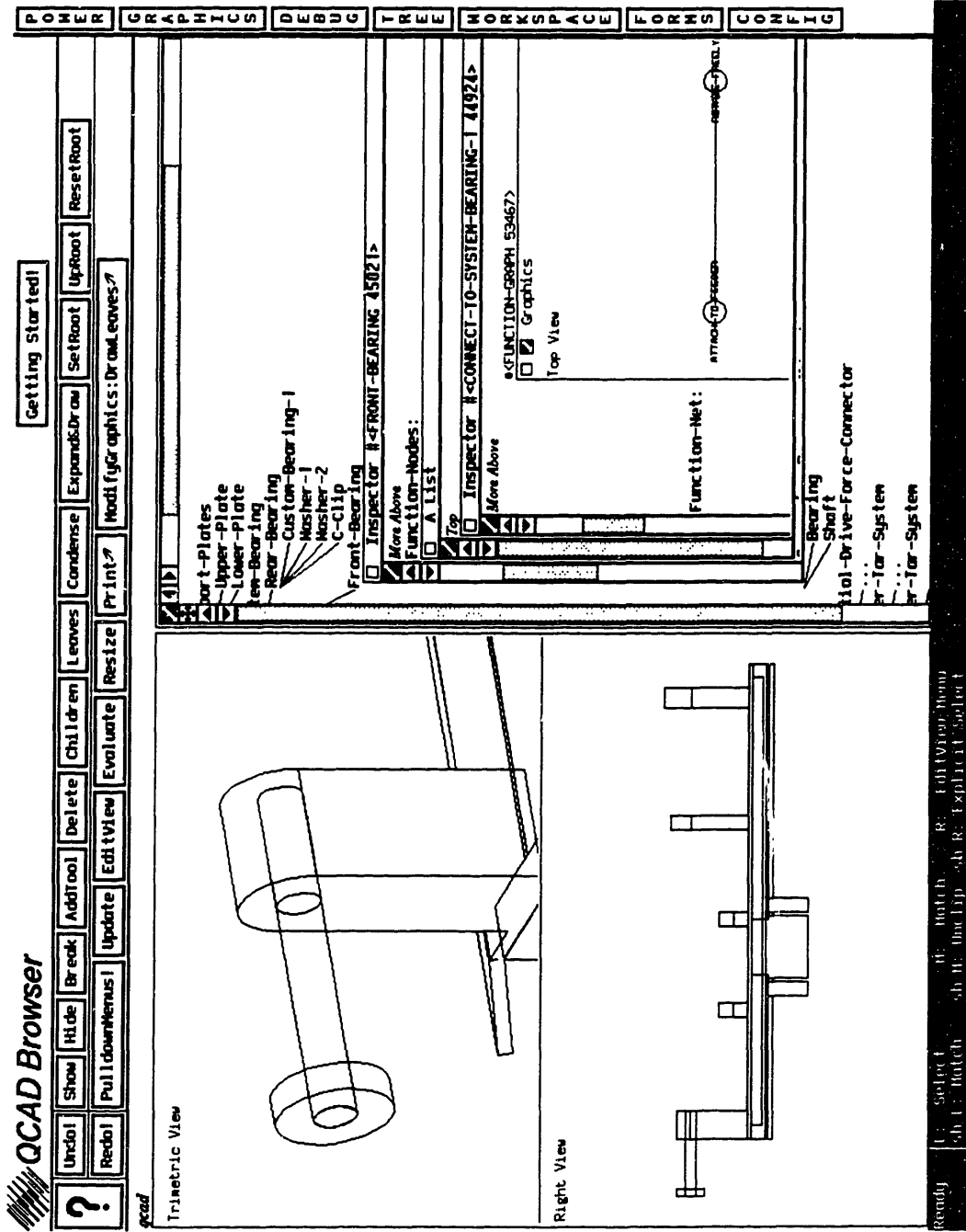


Figure 5.15 - Front System Bearing

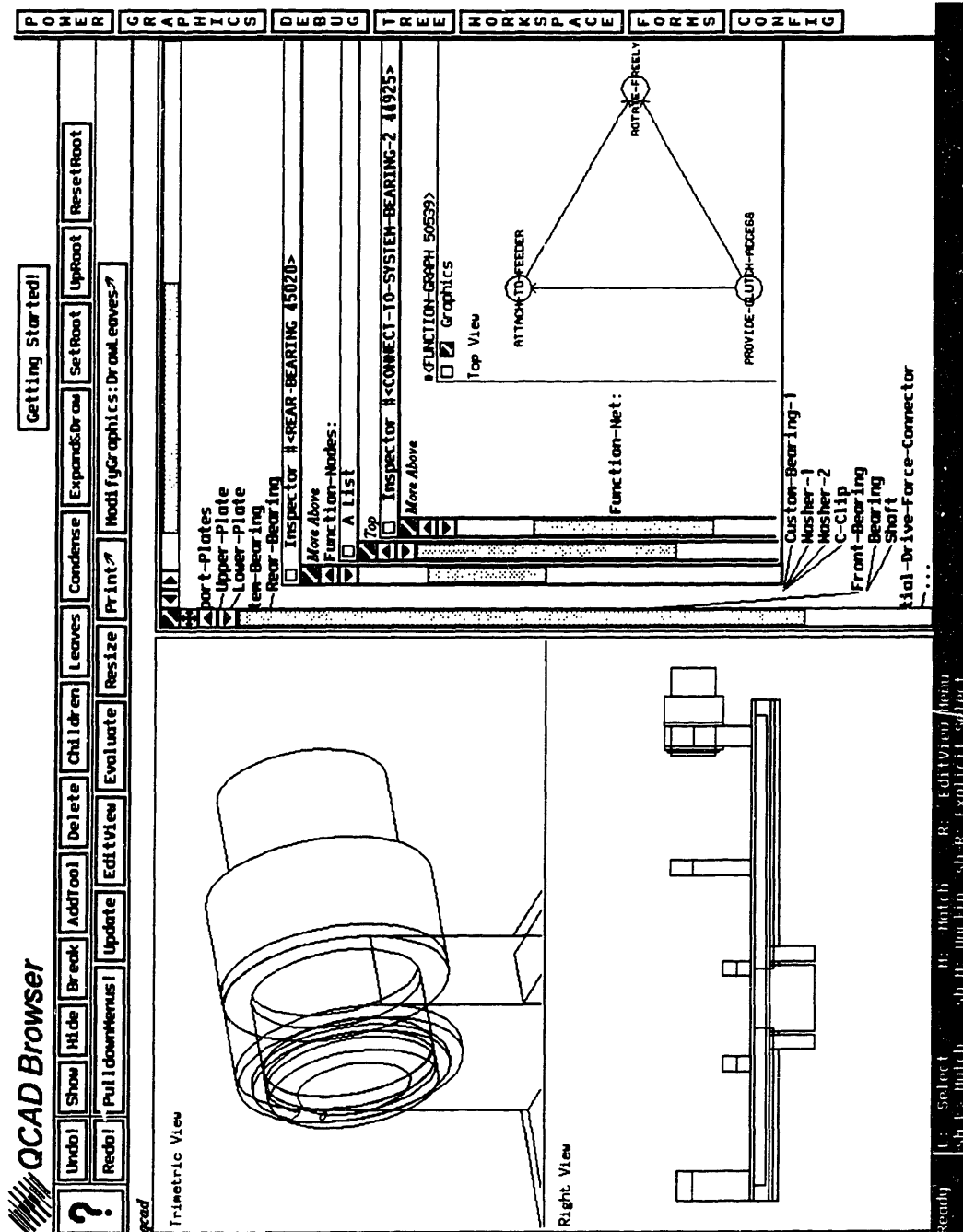


Figure 5.16 - Rear System Bearing

Figure 5.16 shows the rear bearing assembly, which looks completely different from the front bearing assembly. The function net for the rear bearing shows three interrelated functions. Two of the functions are the same, but the third function, *provide clutch access*, changes the nature of how the *attach feeder* and *rotate freely* functions are satisfied.

The belts and rollers on the paper feeder are driven by a central drive belt system, such that all of the paper motion inside the copier is synchronized. The paper feeder uses a clutch to engage and disengage from the centralized drive belt. Only two contact points can be used to enable the feeder to pivot. As such, the clutch mechanism goes inside the rear system bearing assembly. The bearing in this case sits loosely in the upper plate for free rotation; Several washers and a clip hold the bearing in its axial position; and A large hole inside the bearing provides room for the clutch mechanism.

5.2.4. Larger Product Models

The copier model was an exploration of the use of QCAD for the design of a complex product. This section describes some considerations for even larger and more complex product models.

Based on the scale up of the clamp to the copier, it stands to reason that the QCAD methodology is capable of handling even larger products, such as a car or an airplane. Boeing currently uses the ICAD system to assist with airplanes design (Gregory and McIlroy 1990). However, some coordination of part names will be necessary, so as to not cause confusion with identical part names for different parts, especially if they are at different levels of abstraction.

Proper programming in IDL is based on creating generic defparts that can be called from anywhere, if given the correct inputs. If the system is created in such a generic manner, it is possible to build large products out of smaller sub-system models.

Another consideration for a larger product model is the amount of information that is stored and collected in the system. For instance, the fault tree is composed of attributes collected from the three product hierarchies. Suppose a model has over 1000 defparts, each with one fault attribute (usually each defpart would have more than one fault attribute). Compiling a full fault tree would result in about 1000 nodes on the tree. This would be crazy for an engineer to look through to find the solution to a problem. A better idea would be to only collect information from relevant parts of the system. The user could enter keywords that the system would search for in collecting fault data and only display the relevant data. This keyword search routine would also benefit the total cost and error location attributes.

5.3. Observations from QCAD Case Studies

The next several sections examine the effect of the integration of total quality development and knowledge based engineering through the use of the QCAD system. Observations about the advantages and disadvantages of the integration are made from the perspective of both the design process and knowledge based engineering.

5.3.1. QCAD and TQD

This section reflects on the use of a QCAD knowledge based engineering program with the total quality development design framework.

5.3.1.1. QCAD Improvements over TQD

At a high level of abstraction, QCAD improves over the existing design process in three areas:

1. Storing design information
2. Standardization
3. Exploring design concepts

The ability to store information is one of the strong points of the QCAD system. It can store and retrieve large quantities of many types of design information. Piles of papers and notebooks no longer need to be lost in file cabinets and storage sites. They can be stored on the computer. Of course there are limitations to unrestrained information storage, and these will be discussed in Section 5.3.1.2.

One of the by-products of storing the TQD design information in the computer is that it gives the reasoning behind certain design decisions. The TQD matrices show what the original designers thought about the relationships among different criteria, metrics, and design options. This information, along with the function specifications, hardware geometries, and process definitions, shows the original design intent. For future designers, this information will assist in streamlining the design and adapting the product to new requirements. The designer will be able to answer questions like “What does this gizmo do?” and “How are we supposed to make this?” Additionally, for new designers on a project, the system can help familiarize the designers with the product and with the design process used.

When designing a large project with hundreds of parts, a company will most likely break the product into sub-systems, assigning each sub-system to a product development team. We have seen that unless a design is truly modular, each sub-system will interact with other sub-systems (Wu and Azarm 1992, Ulrich 1992). These interactions can be coded into the QCAD product model for retrieval by other groups. That is, as a design progresses, PDTs can load information about the design into a central computer that processes the information, makes sure everything is compatible, propagates error messages to appropriate design teams, and disseminates information about constraints that PDTs need to look into.

The computer not only stores information, but requires the information to be stored in a certain way. The QCAD knowledge base is composed of engineering equations. These equations are only coded in once, standardizing the way products are

designed. For example, if a company has multiple formulas to calculate the length of a beam, one formula has to be selected for use in the computer. Alternatively, each formula is only used under certain situations. The resulting output is that all of the parts are designed using the same criteria.

Furthermore, once the design formulas are entered into the system, every time the program is used to calculate a parameter, the same formula is used. There are no human errors in running the calculations, other than when entering information or interpreting the results, as long as the formula is entered correctly. The computer also calculates the parameters quickly.

One of the ideas behind the QCAD system is to be able to store and select among many different design possibilities, including sub-functions, hardware, and processes. A small data base of these defparts is stored up for the first design. For the next generation design, the same defparts are used, but a few defparts are added. In this way, the design is based on the original design intent, but incremental changes are also possible. After some time, a large data base is collected and many variations on the design can be possible.

In addition to just entering the company's product into the QCAD structure, a competitor's product could be modeled in the system. This expansive modeling of products assists the company in completing and recording a detailed competitive benchmark, while expanding the number of design options.

Once an entire product is input to the system, sensitivity analyses can be performed at several levels of abstraction. Current computer design systems, through parametric design and variational geometry, change hardware components to see different design configurations. With the addition of more abstract design levels, more radical design variations can be seen. Changes in importance ratings or in perceived QFD interactions can change the nature of the design, radically, providing stimulus for future, inventive designs.

Furthermore, any sub-system that can be created to serve a function can be implemented in QCAD. When added, each sub-system must fit in with the rest of the product model, to ensure functionality and standardization. Each hardware piece is linked to a function through the concept selection matrix. After a while, many sub-systems will have been built up to provide even more flexibility in the design.

Another benefit can be seen when the design is in the prototype stage. After testing the prototype, engineering changes may be needed. QCAD is set up to call hardware based on the function it performs. If a change needs to be made to one sub-system, that part's interactions with other pieces can be found in two ways. (1) The hardware piece shows its calling function; that function can call all of the hardware pieces that satisfy it. Each hardware piece can then be changed. (2) The hardware piece also shows its immediate parent sub-system. That sub-system can call all of its children for modification.

Along these same lines, QCAD is object oriented. Identical hardware pieces can be created from the same defpart throughout the model. If an engineering change is more than just a parametric variation, the defpart can be modified, which will implement the change on all like pieces throughout the product model.

QCAD can also be useful farther down the product development process in production and customer support. If there is a problem with a product, the design history is stored in a computer for easy retrieval (this view is shared by Sriraman et. al. (1990) who have implemented a QFD system in an object oriented environment). Additionally, if fault information is coded into the product model, the computer can provide a structured view of how to solve a problem. Notes could also be added to the model showing where and why the sub-systems failed. This kind of knowledge can be very useful for next generation designs.

5.3.1.2. QCAD Difficulties with TQD

Despite the advantages described in the previous section, there are some significant difficulties when using QCAD with the design process:

1. Design rules must be known
2. Lots of programming
3. Mind numbing numerology

The first on the list states that the design rules used in QCAD must be known. It is hard to program a computer if you do not know what to program. Unfortunately, this is quite often the case in design. The general idea for the product exists, but the questions often asked include, “What hardware design satisfies that function?” and “How can it be made?”

There are a couple ways around this difficulty. One is to only program the parts of the design that have known design rules. However, this information is the same as that currently used with computers, limiting the usefulness of the QCAD system. A second alternative is to use the QCAD system to store generic, or text, information until design rules can be established for that particular hardware configuration. This second option is more like the actual design process, in that information is gathered and stored, and then, after sufficient information processing, portions of the design are solidified. The QCAD system allows for the initial information to be stored with room to add in more detailed information.

The second difficulty involves the amount of programming required to store a product model in the system. A lot of design information can be stored in the QCAD database, but that requires a sizable programming effort. Currently, most ICAD users only model the product hardware and some of those models require considerable programming effort. QCAD requests the user to include information about functionality, processes, and alternatives for each, in addition to the hardware programming.

This issue should be explored from three different perspectives. The first perspective is that the programming effort required is based on my subjective observations. People who are more experienced with IDL may consider the programming to require little additional effort. They may even see the QCAD system to require less effort, since the old product model can be used with new design concepts. The new product model can be stored with the old model.

A second perspective on the programming effort refers to how the information is stored without QCAD. Perhaps there is little effort in storing the additional information in QCAD. Currently, the information is stored in several different places: the QFD matrices are stored in a QFD program; the geometry is stored in a CAD system; and some calculations are done on spreadsheets. QCAD consolidates all of this information in one place, making the programming effort look more time consuming, but really just consolidating all of the effort in one place. Moreover, since the information is in one system, the designer always knows where to look for information. With the system, the information is available for the designer to use, which may not be the usual case. Additionally, the information can be linked together to provide further benefits.

The third perspective is that the programming effort is specific to the ICAD system, which is geared toward modeling hardware. QCAD provides a different product modeling philosophy. Using the same interface for both philosophies may be increasing the effort to model a product in QCAD. Additionally, while creating a user interface for QCAD, there is an opportunity to design an interface such that information can be stored without programming.

For now, the best way to address the programming concern is to start with a small product model and build up complexity as necessary. Once the information is in the system, it can stay there, and even more information can be added. One idea for where to start, is with the portion of the model that provides the most customer satisfaction.

Starting with this part of the model makes sure that that part of the design receives the proper attention with respect to the other sub-systems.

Furthermore, the QCAD system is only as good as the information stored in it. If the information is outdated, then the system will supply old information. Many times a system is started with the best intentions, but is not kept up for some reason or another and information is lost and the system falls out of use. This question of updating the system becomes even more of an issue, when dealing with design information. Most designs continually evolve into a final product, which means that at every step of the way, the information in the computer will have to be updated.

Another variation of the programming concern is that programming requires a different skill than the creativity required in design. Constantly switching from programming, to designing, to interfacing with the browser, breaks the designer's train of thought. For example, the designer could be using the system to get rough cost estimates for different hardware configurations. Somewhere in the middle of the task, the designer could discover that a hardware configuration is missing and need to program it into the system.

A different approach would be to use the hardware nodes to describe the indicated hardware. This solution uses no geometry statements and no positioning commands. It just describes the hardware with written statements, which permits flexibility in selecting hardware, without the complexity of detailing the hardware geometry and positioning. The nature of this kind of solution moves QCAD more towards the conceptual design domain than the detailed domain.

Lastly, there is an inherent danger in using a computer for design concept selection. Computers rely on numbers and make all decisions based on numbers. A potential solution may make sense according to the numbers, but not according to common sense. One area of the system where this problem could occur is in the sensitivity analyses, where the design changes, with the importance ratings. One design

may not even fulfill the design requirements, but could be selected anyway, because that is what the numbers indicated. An alert designer can watch out for this mind numbing numerology and override the selection process if necessary. However, even an alert designer could get all tied up in the numbers and lose track of the design.

5.3.2. QCAD and KBE

This section reflects on the use of a QCAD design process framework with knowledge based engineering programs.

5.3.2.1. QCAD Improvements over KBE

At a high levels of abstraction, knowledge based engineering design programs gain two benefits from the QCAD structure:

1. A framework for expanding the range of use of traditional KBE software
2. A structured method for planning a knowledge base

Figure 5.17 shows the current range of use of the ICAD software in design, superimposed over the QCAD structure. Most of the applications are in the hardware domain. There are some ICAD applications at the total system level, however most are at the sub-system, piece part, and part feature levels. This hierarchical distinction depends on what is meant by the total system level of the product. One person's total system can be another's piece part. Thus, the hierarchical distinction in this case is meaningless.

Of more importance are the relatively few applications in the function and process domains. A few systems have been developed using ICAD for high level functional specification (Oulette 1992). None have used ICAD for detailed functional specification. Additionally, a few systems have used ICAD for detailed process definition (Florusse and Clausing 1992). None have used ICAD for high level process definition. In these cases, the level of abstraction is defined relative to the abstraction level of the hardware tree, not

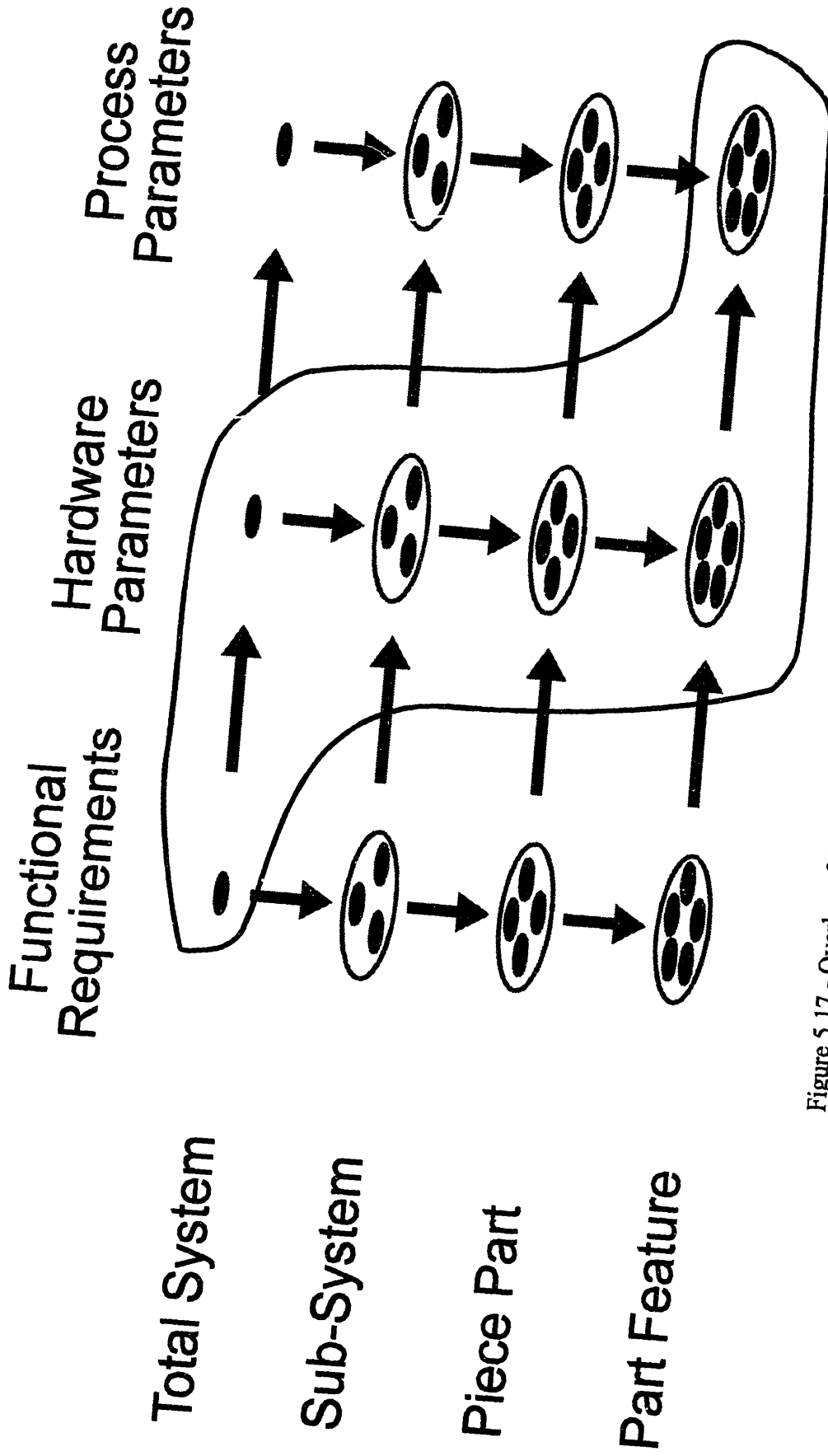


Figure 5.17 - Overlay of Current ICAD Applications on QCAD Structure

relative to their individual trees, which make the hierarchical distinction possible. Both the clamp and copier case studies define the product from high level functions to low level processes, traversing the entire range of the design.

The QCAD framework also provides a method of structured planning for knowledge based programs. By following the total quality development process, the knowledge based product model can incorporate a focus on customer satisfaction, an emphasis on robust design, and the ability to benchmark competitive products.

The TQD process follows EQFD, which is a systematic method to document relationships among parts of the design. By thinking of the relationships, thoughts are stimulated not only on how the part should be designed, but also on what information should be captured in the product model. This rigorous analysis of the design attempts to surface concerns about modeling the product while the PDT is meeting in a room together, instead of several months later, when one or two people are coding the product into the system.

5.3.2.2. QCAD Difficulties with KBE

The one main drawback of using QCAD with knowledge based engineering programs deals with the amount of ambiguity in a design. The design process does not fit precisely into categories. Sub-functions are specified after the hardware is selected. Some hardware pieces are defined with process selection. A design does not fit neatly into the objects on the computer as outlined in the QCAD structure. This creates some ambiguity in defining the product model. The QCAD system attempts to resolve this issue by straddling design information between boundaries, but the possibility for confusion does exist.

One of the main differences between the QCAD structure and its implementation in IDL deals with the hierarchical structure. The QCAD structure emphasizes layered hierarchies, where each part of a layer can talk with any other part of the layer. In the

layered structure, design components at one level of abstraction interact with components at the same level, at the level above, and at the level below. Parts of the design that perform different functions or that reside in different sub-systems also interact. Some design systems can operate in this way, however they are restricted to a few variables and quickly become cumbersome to use (Wu and Azarm 1992). Even though a layered structure is desirable, the QCAD system in IDL had to be arranged in a tree structure with referencing chains.

Additionally, in the QCAD structure, most objects get information from two different places: One type of information comes from a higher level of abstraction; The other type of information comes from the object immediately to the left of it, at the same level of abstraction. Although there are systems that can handle multiple parents, ICAD is set up to handle one parent. This may be easier for product models that are concerned only with the hardware. However, when functions and processes are considered, more than one parent is desirable.

When inheriting down the ladder of abstraction, a specific reference does not need to be made from the object to its parent to get information. The object just requests the information as an input. In systems with single parent capability, information that does not come from the object's parent or ancestors, must be specifically referenced. If the design is dynamic, the specific referencing method may run into difficulties, especially if the referenced objects do not exist.

5.3.3. Summary

The last two sections commented on the integration of the design process and computerized knowledge based engineering. The major comments are summarized in the table below. To interpret the table, read the column heading followed by the row heading. For example, the first box in the top left corner describes the QCAD benefits from knowledge based engineering.

	QCAD Benefits from	QCAD Drawbacks from
Knowledge Based Engineering	<ol style="list-style-type: none"> 1. design information storage 2. standardization 3. design concept exploration 	<ol style="list-style-type: none"> 1. design rules must be known 2. lots of programming 3. mind numbing numerology
Design Process	<ol style="list-style-type: none"> 1. framework for expanding the range of traditional KBE 2. structured method for planning a knowledge base 	<ol style="list-style-type: none"> 1. ambiguity of design.

An examination of the table above reveals some relationships among the benefits and drawbacks of integrating the design process with knowledge based engineering, as learned from the case studies.

Across the rows, it is clear that both knowledge based engineering and the design process have different emphases on design. Knowledge based engineering prefers using known, static design rules, that are easy to program. The programs are good at storing information that can be used in later stages of the design. The knowledge based programs store information in predetermined formats, for use in analyses, perhaps by people that didn't even enter the information into the computer originally. Alternatively, the total quality development process provides a framework and a set of structured tools to help guide the way through the ambiguity of design. Although not specifically mentioned in relation to QCAD, many of the TQD tools promote teamwork and alignment of individual and group goals, which help a group of people focus on a path to follow in product development.

Another observation about the rows is that some of the items in the benefits column are inextricably linked to items in the drawbacks column. For example, the knowledge based engineering program stores information, however, that information must be representable in a known format before it can be entered into the computer. The design

process, on the other hand, provides a framework for the design, but it is a loose framework that varies with different design situations.

The linkage of the benefits and drawbacks for the rows signifies that a middle ground is necessary for a beneficial integration of the two design methodologies. Designs that are constantly changing may not be the most suitable for use with the QCAD system, nor would designs that are conceptually static. In the former, the design changes would be so varied that the computer would not be able to handle the complex variations in the design. In the latter, the design has been made hundreds or thousands of times and the customer's requirements have evolved to the point where all of the design requirements are known, such that the up front design work is not necessary.

5.4. QCAD in Product Development

The first three sections of this chapter discussed the implementation of QCAD in IDL and show the concept's feasibility. The sections described two case studies, a clamp and a copier, and projected potential implementation concerns for larger product models. Based on the two case studies, the third section discussed observations about the integration of computers and quality methods in product development.

From that experience, this section describes several uses for QCAD in product development. The nature of the tasks performed change as the product flows through development. Although no explicit distinction is made in this section, the style of computer use is different in conceptual design, detailed design, and incremental product improvement. A further distinction could be made between initial designs with QCAD where no product model has been stored, and subsequent designs, where a product model is already in the system.

QCAD use is discussed in the next two sections: the first section describes a new product being developed in QCAD, and the second section describes additional ways to use QCAD when incrementally improving a product. The distinction is arbitrary, a design

team can use any of the methods described in either section whenever they think the method will help develop a product.

The overall theme, in the next two sections, is the human design process being supplemented by the computer. QCAD mixes people's abilities to be creative and to cope with uncertainty with the computer's affinity for structure and ability to store and retrieve information.

5.4.1. QCAD Uses in Product Development

This section describes how a product development team could use the QCAD methodology to develop a product. At the start of a product development effort, the design team uses the product development process to explore the design problem. Without the use of the computer, the team explores the voice of the customer, develops engineering metrics, and obtains some understanding of the problem. The computer can then be used in this process to reflect on models used in the past to understand the design. The computer uses a function net representation, described in Section 4.5.2.4, to display the interactions among the product functions.

Once the team has an understanding of the problem, the team generates hardware ideas. Some ideas may already have been generated during problem exploration and are documented in the computer for use in this stage of design. To generate solutions, the team looks both inside and outside the group for ideas. The team should go out and look at competitive products to stimulate ideas. The team can also solicit ideas from the computer, by choosing different functions, and looking at the associated hardware options. In their solution exploration process, the team may decide that they need to go back and reformulate the problem.

At some point, the team will have generated a sufficient number of ideas to perform Pugh concept selection. This matrix is done without the computer, such that the team can become absorbed with the problem and be free to develop new concepts

(Clausing 1994). When filling out the matrix, some additional information may be necessary to help evaluate ideas. Some of that information is stored in the computer and could help in completing the matrix.

After finishing the matrix, and selecting a concept, the team records the matrix in the computer. The final concept ideas are also stored. Everyone on the team can see what ideas were explored in concept selection and can use this information when working on the design.

From this point, the design team has several options of how to proceed. They can further explore the hardware and manufacturing process; they can explore more detailed hardware embodiments of the product; they can explore sub-functions; or they can explore a combination of all three. If the team chooses to continue exploring functions then they continue as already described, with finding the voice of the customer, creating an understanding of the functions, and selecting a concept. The computer is used to jog new ideas, store concepts, and help evaluate concepts. For sub-functions, the design team will also have to pay attention to the higher level hardware configuration.

A second option is to look at how the hardware will be manufactured. In much the same way as a hardware concept was chosen by looking at a set of functions, the manufacturing process is chosen by looking at the hardware to be fabricated. The design team starts the hardware exploration by completing a QFD design matrix. If the same house of quality engineering metrics were used with previous product designs, then the design team can use the product expectations already stored in the computer.

As the design team explores the hardware, possible manufacturing processes may come to mind. These are recorded in the computer for later use in material and process selection. The team can also get process ideas from comments that were stored in the computer from previous designs, and from the set of processes recorded in the process hierarchy.

For material and process selection, one set of criteria comes from the QFD matrix. Another set of criteria is devised by the PDT using the group's knowledge. Other criteria can be found in previously completed matrices, stored in the computer. The design team completes the selection matrix in the same manner as Pugh concept selection, using the computer to provide information to help fill in the matrix and to store the completed matrix. After selecting a process, the team can continue by further defining the selected process, by designing more detailed hardware, by exploring the next level functions, or by doing all three.

Lastly, in the detailed hardware design option, QCAD provides a rigid framework for defining and analyzing product components. A design team should first sketch out the design on paper. As the design freezes, bits and pieces of the hardware are modeled into the computer. In the best case scenario, many of the hardware pieces are already defined in the system and the design task is composed of assembling the right elements together. In a less developed system, hardware components will have to be defined before they are used. Note that the more information in the product model, the more flexible and powerful the model.

Since the computer looks for clearly defined hardware, as the team progressively freezes the design and enters the design into the computer, the QCAD system surfaces problems that would not normally be found until later. For example, in the computer, all of the interfaces among different sub-systems and piece parts must be clearly defined and consistent. The design team must resolve the inconsistencies before continuing with the design, preempting downstream problems.

Once a consistent, product model is defined, designers can use the computer to explore the design. As mentioned in Section 5.2.2, the designer can modify the design parametrically in two ways: (1) by just changing the dimensions, and (2) by changing the importance ratings and relationships in the matrices. These tools are used to help the designers further understand the design problem and to explore potential solutions for the

design. In addition to the parametric design capabilities, the product model, or parts of the model, can be sent to other programs, such as FEA, for analysis and optimization.

Other tools in QCAD that help the team explore the hardware design include the product structure net, the Taguchi system of quality engineering, and constraint propagation. The product structure net, as described in Section 4.5.3.1, is a method for representing interactions among the hardware parts. The Taguchi system, as described in Section 2.6, includes an experimental method to observe the interactions among hardware parts. Constraint propagation is done by the computer, which stores the constraints that designers should be aware of when designing, such as limitations on material properties and dimensions, and indications of analyses to be performed. The constraints were previously input by the design team.

After a design is defined, it goes through a prototyping cycle, to show that the design works and that the manufacturing processes are suitable for production. In this period of time, some problems will be found that require the design to be changed. The computer can assist here too. The design team can change the design of a hardware component where it was initially defined and be certain that the hardware definition will change everywhere that hardware piece is used in the model.

If in testing, the product does not perform a function as well as it should, the design team can look at the function definition in the system to find the root cause of the problem. The team can start with the function and look at all of the hardware that satisfies that function. Additionally, the team could look at the fault tree to find areas where the product could be failing.

Throughout the development process, QCAD helps the design team organize and use information. QFD matrices can be stored throughout the system. Notes can be left to remind people of work that needs to be done, and comments can be left with reasons why a design was defined in a specific manner. All of the information is in one place, in the

computer, making it easier for designers to find and use the information in product development.

5.4.2. Additional QCAD Uses for Incremental Product Improvements

This section discusses ways that a product development team could use QCAD for making small revisions to existing products, that were previously developed using QCAD. The ideas discussed in this section can be combined with the ideas discussed in the last section.

The first question asked when incrementally improving a design is, “Where to start?” The team will have many different ideas of what can be done to improve the design. One idea is to redesign the product to decrease the cost. Other ideas are to add features or to improve the technology.

These statements are at rather high levels of abstraction. In order to get more of an idea on what to change, the team should explore the computer model in at least one of several ways.

One way is to look at the function hierarchy and ask, “What functions are represented, but do not contribute directly to the primary function of the product?” That is, “What secondary functions are created when a certain hardware configuration is chosen over a different configuration?”

Another way to get ideas on how to change a design is to look at the hardware parts that perform a similar function, but look different. The product cost can be reduced by standardizing the different parts. A twist on this idea is to look at the hardware made by the same process and to standardize the hardware design to simplify the process operations.

Ideas for improvement can also be found in the design methods used in defining the design. For example, the roof of the house of quality shows tradeoffs among different corporate characteristics. Improving one tradeoff will increase customer satisfaction in

the product. Another place to look is in the concept selection matrices. Good ideas that weren't chosen may be feasible now, given advances in technology.

Changing ideas in the product model is relatively simple, since the system is modular. Instead of altering the currently defined model, a designer just adds new a design variation to the system and augments the appropriate selection matrix to incorporate the new design. If it turns out that the change is undesirable, then the system is easily restored to its prior state.

Furthermore, adding new designs, instead of replacing old designs increases the knowledge base in the model. In addition to hardware, both functions and processes can be added to the product model to increase design flexibility.

Chapter 6

Conclusions and Future Work

6.1. Summary and Conclusions

Conventional KBE connects engineers to engineers.

QCAD connects customers to the factory floor.

These two statements express the intent behind this thesis. That is, to provide a framework for using computerized knowledge based engineering design systems with a product development process that focuses on quality, cost, and delivery. Characteristics of the development process include emphases on customer wants and needs, competitive benchmarking, and the Taguchi system of quality engineering.

This thesis outlined a Quality Computer Aided Design (QCAD) methodology consisting of: (1) a design process framework and (2) a computer program for use with the design framework. It is intended that a multifunctional product development team use QCAD to assist in developing both new and variational products from function designation to hardware definition to process specification.

The design process consists of a collection of tools that help a product development team understand the functional requirements, synthesize hardware and process concepts, and select feasible alternatives from a set of concepts. Some of the tools include Enhanced Quality Function Deployment (EQFD), Pugh Concept Selection, fault trees, and robust design of experiments. Many of the methods used are qualitative in nature and require human creativity. Thus, the design methods are first completed by a product development team and are then coded into the computer.

The computer both stores and manipulates information entered into the system to help a designer explore a design concept. Both the design methods and the information stored in the methods are assigned to a function, hardware, or process category. These

categories are broken down further into total system, sub-system, piece part, and part feature abstraction levels.

Every product is different. As such, it is at the discretion of the PDT to decide where to enter information into the system. The PDT may decide that the best use of QCAD is to optimize the geometry of a part using finite element analyses of the part shape, obviating the need for function and process hierarchies. Or, if a small sub-system is being modeled, total system information may not be necessary, and the product model contains information ranging from functions to processes and from sub-system to part features.

The QCAD system was implemented in the ICAD design language (IDL). ICAD is the leading supplier of commercial knowledge based engineering software worldwide. IDL is a super set of common lisp and is geared toward modeling products in the browser. The browser is the ICAD user interface, which has both a hierarchical and a geometric representation of the hardware, and many commands to manipulate the product model.

Both an easy to understand clamp model and a more complex copier model were developed in the methodology. These case studies not only tested and helped refine the methodology, but also provided insight about combining total quality development with computers.

There is a strong synergy between computers and quality design. KBE is a vast resource for storing information. It standardizes product information and provides the capability for accelerated what-if analysis. The design process provided a framework and procedures for coordinating design information in the computer and a structured method for organizing and planning a product model. Additionally, there is increased opportunity to use the information since it is all stored in one place.

Even though there is a strong synergy between computers and quality design, there are several drawbacks to the integration that have strong repercussions for designers. The ambiguity of the design process does not lend itself to being structured into any one

methodology. Design is ambiguous by nature. It takes both analysis and human creativity to develop feasible solutions. The computer is good at quantitative analyses, but to cope with ambiguity, the QCAD system requires programming effort beyond that required by geometry focused KBEs (The increased programming effort may be offset by increased availability of information and the decreased programming effort in other software).

Furthermore, any time computers are used in design, the designer must be careful of mind numbing numerology. It is easy to get caught up in running analyses and making all of the numbers work. However, a designer should never lose sight of the overall picture.

Overall, QCAD provides one leap forward to the integrated use of computers and quality design, by expanding the knowledge based product model to include functions, hardware, and processes.

6.2. Future Work

The QCAD system, including both the design methodology and the computer implementation, has several opportunities for future work, which range from design process improvements to computer specific improvements. Five areas for focus are presented.

1. Function tree development
2. Function matrix and function net
3. Information represented
4. User interface
5. Layered hierarchy implementation

The QCAD framework defines three intensely, interrelated trees: a function tree, a hardware tree, and a process tree. The hardware and process trees are relatively straight forward to create. ICAD usually tells its customers to set up the hardware tree in reverse of how the product is manufactured. Function trees on the other hand are less familiar

product structures. The function trees for both the clamp and the copier could take on several different forms depending on how the functions are grouped. Some exploration is needed to determine the best organization of the function tree for different types of product models.

To add to the confusion, the lower level functions and hardware parts are defined by iterating back and forth between the function and hardware trees (Clausing 1994). Different hardware embodiments that satisfy a given function will create different lower level functional requirements.

A nominal function tree structure was used in this thesis to investigate the QCAD methodology. However, variations on constructing a tree structure should be explored. The ideal QCAD methodology designated a layered structure, but tree structures were used to make the methodology compatible with IDL. A structured methodology for creating function trees would advance the QCAD methodology, as implemented in ICAD.

In QCAD, the functions in the function tree are directly connected to specific hardware that satisfy that function. The functions are diagrammed using a function net to help specify the hardware concept. An example of the relationship between the function net and the hardware is given in Section 5.2.3, describing the copier product model. Different functions and combinations of functions result in different hardware. It seems likely that different function diagrams would also generate different hardware ideas. Function structures (Pahl and Beitz 1984) and Bond Graphs (Paynter 1960) are two among several function diagramming systems that have been developed to chart out functions. Further exploration of the hierarchical function tree structure, used with IDL, may be necessary for experimentation with function diagramming methods.

QCAD is both a design assistant and a design history system. It can be used to explore designs, run analyses, and store product specific information. Initial customer concerns were addressed. The QCAD structure is flexible and can accommodate many types of information that a PDT would want to use in the system. However, little

customer feedback was solicited (ICAD did review the proposed system). What information do designers want in a quality design system, and how can that information be integrated into QCAD?

One reason for seeking little customer feedback on the QCAD concept is that its current implementation in IDL makes the concept hard to understand and system difficult to use. An improved user interface should be created to enable users who are unfamiliar with the product model to manipulate the model and to learn from the information stored in the computer. Appendix C presents several ideas for a user interface, at the conceptual level.

The QCAD system, as implemented in IDL, consists of three interconnected tree structures, which is a deviation from the originally specified layered hierarchy. A more accurate implementation of the methodology in a computer environment that can handle a layered structure would enhance the flexibility of the product model.

References

- Akao, Yoji. *Quality Function Deployment: Integrating Customer Requirements into Product Design*. Cambridge, Mass.: Productivity Press, 1990.
- Algor, Inc. *ViviCad Manual*. Version 11 Pittsburgh, Penn.: Algor, Inc., 1993.
- Asimow, Morris. *Introduction to Design*. New Jersey: Prentice-Hall, Inc., 1962.
- Bañares-Alcántara, R. "Representing the Engineering Design Process: Two Hypotheses." *Artificial Intelligence in Design '91* (1991): 3-22.
- Berg, John. "Computer Implementation of the Improved Product Development Process." Mechanical Engineering Master's Thesis, Massachusetts Institute of Technology, 1988.
- Cadence, *Cadence User Guide*. Version 4.2.1. San Jose, California: Cadence, 1993.
- Chen, Aihua, Brian McGinnis, David G. Ullman, and Thomas G. Dieterich. "Design History Knowledge Representation and its Basic Computer Implementation." *Design Theory and Methodology - DTM '90*, ASME DE-Vol. 27 (September 1990): 175-184.
- Chung, Jack C.H., and Martin D. Schussel. "Technical Evaluation of Variational and Parametric Design." *Proceedings of the 1990 ASME International Computers in Engineering Conference and Exposition 1* (1990): 289-298.
- CIME Staff Report "CAD Follows New Scripts." *Mechanical Engineering* 111, no. 11 (November 1989): 62-67.
- Clausing, Don P. "Concurrent Engineering." *Proceedings of the Design Productivity International Conference* (1991).
- Clausing, Don P., and Stuart Pugh. "Enhanced Quality Function Deployment." *Proceedings of the Design Productivity International Conference* (1991).
- Clausing, Don P. *Total Quality Development*. Fairfield, New Jersey: ASME Press, 1994.
- Cross, Nigel. *Engineering Design Methods*. New York: John Wiley & Sons, Inc., 1989.
- Dixon, J.R. "Designing with Features: Building Manufacturing Knowledge into More Intelligent CAD Systems." *Proceedings of the ASME Symposium on Product and Process Design 1* (1988): 51-57.
- Drake, Samuel and Samuel Sela. "A Foundation for Features." *Mechanical Engineering* 11 (November 1989): 66-73.

Eastman, C.M. "Recent Developments in Representation in the Science of Design." *Proceedings of the 18th Design Automation Conference, ACM, IEEE* (1981): 13-21.

Edwards, John S. *Building Knowledge-Based Systems: Toward a Methodology*. London: Pitman Publishing, 1991.

Eppinger, Steven D., Daniel E. Whitney, Robert P. Smith, and David A. Gebala. "Organizing the Tasks in Complex Design Projects." *Design Theory and Methodology* (September 16-19, 1990): 39-46.

Feigenbaum, Edward A., and Pamela McCorduck. *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*. Reading, Mass.: Addison-Wesley, 1983.

Finger, Susan, and John R. Dixon. "A Review of Research in Mechanical Engineering Design. Part II: Representations, Analysis, and Design for the Life Cycle." *Research in Engineering Design 1* (1989): 121-137.

Florusse, Leendert B. and Don P. Clausing. "A Detailed Framework for Quality Function Deployment Phase 3, Process Engineering." *LMP working paper*. Massachusetts Institute of Technology, December 28, 1992.

Ganeshan, R., S. Finger, and J. Garrett. "Representing and Reasoning with Design Intent." *Artificial Intelligence in Design '91* (1991): 737-755.

Garcia, A.C.B., and H.C. Howard. "Building a Model for Augmented Design Documentation." *Artificial Intelligence in Design '91* (1991): 723-736.

Gregory, Jack and Dan McIlroy. "Tool designers work to rule." *Engineering*. 230, no. 3 (March 1990): ACE15-ACE17.

Griffin, Abbie and John R. Hauser. "The Voice of the Customer." *Working Paper, Sloan School of Management*. Massachusetts Institute of Technology, 1991.

Hauser, John R. and Don Clausing. "The House of Quality." *Harvard Business Review* (May-June 1988): 63-73.

Head, George, Charles Pietra, and Kenneth Segal. *The AutoCAD 3D Book*. Chapel Hill, North Carolina: Ventana Press, 1989.

Heatley, Lynn C. and William J. Spear. "Knowledge-Based Engineering Design at Xerox." *Artificial Intelligence in Engineering Design 3* (1992), v3.

Hu, David. *Programmer's Reference Guide to Expert Systems*. Indiana: Howard W. Sams & Company, 1987.

Hunt, John E. and Mark H. Lee. "Towards a Knowledge-based Design Assistant." *Engineering Applications in Artificial Intelligence* 5, no. 4 (1992): 275-287.

ICAD, Inc. *The ICAD System User's Manual*. Version 4.0. Cambridge, Mass.: ICAD, Inc., 1993.

ICAD, Inc. *About Understanding Knowledge-Based Engineering*. Cambridge, Mass.: ICAD, Inc., September 1992.

International TechneGroup Incorporated. *QFD/Capture "Windows Version" User's Manual*. Milford, Ohio: International TechneGroup Incorporated, 1992.

Ishii, Kosuke and Krizan, Steven "Computer Aided Life-Cycle Design." *Aerospace Product/Process Design Interface* 912220 SAE International SP-886 (1991): 79-86.

Ishii, K. and L. Hornberger, "The Effective use and Implementation of Computer-aids for Life-Cycle Product Design." *Advances in Design Automation, DE-Vol. 44-1* (1992): 367-374.

Jackson, Greg. "Shareware for Engineering." *Mechanical Engineering* 114, no. 10 (1992): 80-81.

Jebb, Alan, S. Sivaloganathan, and N.F.O. Evbuomwan. *Design Function Deployment User Guide*. Engineering Design Center, City University, Northampton Square, London, 1993.

Jebb, A., S. Sivaloganathan, R.C. Edney, N.F.O. Evbuomwan, and H.P. Wynn. "Design Function Deployment." *ASME Engineering Systems Design and Analysis, PD-Vol. 47-1* (1992): 251-256.

Knight, Thomas P. "A Knowledge System for Functionally Integrated Product Development." Mechanical Engineering Bachelors Thesis, Massachusetts Institute of Technology, 1990.

Krick, Edward V. "An Introduction to Engineering and Engineering Design." New York: John Wiley & Sons, Inc., 1965.

Krinninger, Andreas and Don P. Clausing. "Quality Function Deployment for Production." *LMP working paper*, Massachusetts Institute of Technology, August 1991.

Kuffner, Tom A. and David G. Ullman. "The Information Requests of Mechanical Design Engineers." *ASME Design Theory and Methodology - DTM '90, DE-Vol. 27* (1990): 167-174.

Moses, Joel. *Organization and Ideology*. unpublished, 1990.

Oulette, Mark. "Form Verification for the Conceptual Design of Complex Mechanical Systems." Masters Thesis, Georgia Institute of Technology, 1992.

Pahl, Gerhard, and Wolfgang Beitz. *Engineering Design*. London: The Design Council, 1984.

Parametric Technology Corp. *Pro/Engineer User's Guide*. Version 12 Waltham, Massachusetts: Parametric Technology Corp., 1993.

Paynter, Henry M. *Analysis and Design of Engineering Systems*. Cambridge: The MIT Press, 1960.

Phadke, Madhav S. *Quality Engineering Using Robust Design*, New Jersey: Prentice Hall, 1989

Pugh, Stuart. "Concept Selection - A Method that Works." *Proceedings of the International Conference on Engineering Design ICED'81* (1981): 497-506.

Pugh, Stuart *Total Design*. New York: Addison Wesley Publishing Co., 1990.

Quality Engineering Services. *The Taguchi Expert Software User's Manual*. Webster, New York: Quality Engineering Services, 1992.

Ramamoorthy, C.V., and Benjamin W. Wah. "Knowledge and Data Engineering." *IEEE Transactions on Knowledge and Data Engineering* 1, no. 1 (1989): 9-16.

Reddy, Ramana. "Editorial: System Support for Concurrent Engineering." *International Journal of Systems Automation: Research and Applications* 1, no. 3 (1991): iii-iv.

Shiba, Shoji, Alan Graham, and David Walden. *A New American TQM*. Cambridge, Mass.: Productivity Press, 1993.

Siegel, Paul. *Expert Systems: A Non-Programmer's Guide to Development and Applications*. Pennsylvania: Tab Professional and Reference Books, 1986.

Sontow, Karsten and Don P. Clausing. "Integration of Quality Function Deployment with Further Methods of Quality Planning" *LMP working paper*, Massachusetts Institute of Technology, April 3, 1993.

Sriraman, Vedaraman, Phadhana Tosirisuk, and Hsing Wei Chu. "Object-Oriented Databases for Quality Function Deployment and Taguchi Methods." *Computers and Industrial Engineering* 19, no. 1-4 (1990): 285-289.

- Suh, Nam P. *The Principles of Design*, New York: Oxford University Press: 1990.
- Swanson Analysis Systems, Inc. *ANSYS User's Manual for Revision 5.0*. Houston, Pennsylvania: Swanson Analysis Systems, Inc. 1992.
- Taguchi, Genichi. *Introduction to Quality Engineering: Designing Quality into Products and Processes*. White Plains, New York: UNIPUB/Kraus International Publications, 1986.
- Taguchi, Genichi. *System of Experimental Design*. 2 Vols. White Plains, New York: UNIPUB/Kraus International Publications, 1987.
- Takeuchi, Hiroataka, and Ikujiro Nonaka. "The new new product development game." *Harvard Business Review* 64, no. 1 (1986): 137-146.
- Traughber, Thomas J. "Expert System for Tool Selection." *SAE International Congress & Exposition on Artificial Intelligence* 860338, SP-664 (1986): 37-41.
- Vora, Lakshmi S., Philip C. Jackson, and Robert E. Veres. "Technical Information Engineering System (TIES)." *SME Technical Paper*, MS89-736 (1989): 38-45.
- Ulrich, Karl. *The Role of Product Architecture in the Manufacturing Firm*, Course Notes for 2.739J, Product Development, Massachusetts Institute of Technology, October 1992.
- United States Congress. "Integrated Computer-Aided Manufacturing (ICAM) Architecture Part II Volume IV - Function Modeling Manual (IDEF₀)." AFWAL-TR-81-4023, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, 1981.
- Ward, Allen C. "A Recursive Model for Managing the Design Process." *ASME Design Theory and Methodology - DTM '90*, DE-Vol. 27 (1990): 47-52.
- Wu, Bi-Chu and Shapour Azarm. "A Reduction Method for Nonhierarchical Optimization-Based Design." *Advances in Design Automation* 1 (Sept 13-16, 1992): 203-212.

Appendix A

Starting QCAD Product Models in IDL

This appendix gives step by step instructions for starting the clamp and copier product models in the ICAD Design Language (IDL). Both product models are in ascii format on the DOS disk in the pocket on the inside back cover. An introductory course in IDL is helpful for using a product model once it is running in the Browser. The instructions assume that the user is familiar with the Unix operating system and with emacs.

The product models were created using:

<u>Workstation</u>	<u>ICAD Software</u>
<ul style="list-style-type: none">• Sun SparcStation 1+• 64 MB RAM• 1.05 GB hard drive• 200 MB swap space• SunOS 4.1.3• Open Windows 3.0	<ul style="list-style-type: none">• ICAD, Version 4.0• Allegro Common LISP, Version 4.1• Unigraphics II, parasolids, Version 4.2.81

The instructions below are written generically for any ICAD supported platform. When appropriate, an example of the command is given, as it would be used on the above described workstation. For reference, the machine name is "winston". As a typographic convention, commands that the user types are printed in bold font and computer output is printed in normal font.

1. Log in to an account with access to the ICAD system.

Login: **qcad**

2. Start the window system, if not already active. This should pop up several windows, including a console and a cmdtool.

winston% **openwin** (the qcad account automatically starts Open Windows)

3. Load the product model files from the DOS disk into a new directory using **dosread**, or a similar command. On winston, the clamp product model is in a directory called clamp and the copier model is in a directory called copier.

The files with the .cla extension are part of the clamp product model. The files with the .cop extension are part of the copier product model. All of the files, except for the readme file should be changed to end in .lisp (i.e., function.cla.lisp).

The system.?.lisp file may need to be changed depending on the configuration of the account running ICAD. (The ? represents cla or cop, referring to clamp or copier, respectively). The system.?.lisp file has a line defining where to find the clamp and copier files. Follow the instructions in the system.?.lisp file to change the directory path that references the clamp and copier files.

4. Open an emacs window, if one is not already active. Position the mouse in an active unix window and start emacs. The text cursor in the window may need to be activated by clicking the left mouse button once, while the mouse is positioned in the desired window.

winston% **emacs &** (the qcad account automatically starts emacs)

Emacs is used for bringing up the Browser and for interacting with IDL. Emacs buffers are used for editing and compiling product models, debugging product models, and issuing commands directly into IDL.

5. Start ICAD. With the mouse positioned in the emacs window, type **<meta-x> icad**. On the Sun SparcStations, meta is the diamond key next to the space bar. Meta works just like the shift key. Keep the meta key depressed while typing x. Type icad when the cursor appears on the last line of the window. (The qcad account automatically starts ICAD).

This command loads the IDL world (including Allegro Common LISP) and pops up the Browser when it is done. The Browser accepts both keyboard and mouse commands. In general, the mouse is used to select and run commands, while the keyboard is used for supplemental information. The three mouse buttons have the following assignments:

Mouse-left:	Select
Mouse-middle:	Match
Mouse-right:	Pop up a menu

There are two actions involved with issuing a command. The first is to select a command to run or an object to manipulate. The second is to match the command to an object or to match the object to a command. The border around the selected item will blink to signify that it is selected. Pressing ESC will de-select the item. Pressing mouse-right on an item will pop up a menu of additional commands. The next instruction gives a chance to practice issuing a command.

6. Start solid modeler.
 - a. **Mouse-left** on “Workspace” on the right side of the Browser to open the workspace menu.
 - b. **Mouse-left** on “Start Solid Modeler” and a window will pop up.
 - c. Under “host”, type the name of the machine licensed to run the solid modeler. (winston is licensed to run the solid modeler). Ignore the space about the journal file name.
 - d. **Mouse-left** on “Accept” to start the solid modeler. The pop up window will disappear and the menu button will change to “Stop Solid Modeler!” On winston, the Lisp buffer in emacs should say “Started Parasolids Version 4.2.81.”

7. Compile and load product models. The clamp and copier product models are stored in several different files. The file, system.?.lisp, contains a script of files to compile and load in order to run model ?, where ? stands for clamp or copier. In order to run the script program, type **<meta-x> open-lisp-listener** and wait for the emacs buffer to change. At the command prompt in the lisp listener type the series of commands listed below. Some commands may take a long time to execute.
 - a. **:cd ~/directory-with-product-model-files** This command changes the active directory to the one with the product model files. Replace the “directory-with-product-model-files” with the name of the directory with the files that was created in step 3.

(on winston) for the clamp type: **:cd ~/clamp**
 for the copier type: **:cd ~/copier**
 - b. **:ld ~/system.?.lisp** This command creates the functions that compile and load the desired product model. Replace the ? with either cla or cop, for the clamp or copier models.
 - c. **(compile-qcad)** This command compiles the files that comprise the product model and stores several binary files with the “.sparc” extension in the directory specified in part a. An error may occur if there is not enough disk space in that partition to create the binary files.
 - d. **(load-qcad)** This command loads the compiled code into memory.

8. Instantiate QCAD. Now that the product model is loaded into memory, it can be instantiated in the Browser.
 - a. **Mouse-left** on “Tree” on the right side of the Browser to open the tree menu.
 - b. **Mouse-left** on “Make Part” and a window will pop up.
 - c. Type **qcad** under “part name” in the window. This tells IDL to instantiate the qcad product model. Ignore the “more” button on the bottom of the window.
 - d. **Mouse-left** to select the make part window. The border around the window should blink.
 - e. **Mouse-middle** on the workspace rectangle on the left side of the screen. After a few seconds, a top level inputs window will pop up on the screen.
 - f. In the tree menu, **mouse-left** on the “close” button directly above the menu title in order to close the menu.
 - g. In the top level inputs window, **mouse-left** on “accept” to use the values for the top level inputs. One of the inputs is titled “strings-for-display”. It contains the name of the root node, which in this case is qcad. The other input is titled “display-controls” and contains values for how the part will be seen in the workspace. A few seconds after choosing accept, the top level inputs window will be replaced with a graphics viewport and the start of a product structure tree.

9. Congratulations, you have just instantiated a QCAD product model. Several useful commands for exploring the product model are listed below. The commands are activated by the mouse, as described in step 5. You may also wish to try the mouse-right command, which pops up menus. For further information on commands, see the ICAD user's manual (1993). Have fun.

children:	Add one layer to the selected node on the tree.
leaves:	Expand the entire selected region of the tree.
condense:	Shrink the selected region of the tree to the parent node.
expand and draw:	Expand the entire selected region of the tree and draw the associated hardware (QCAD only associates hardware with the hardware tree. Drawing nodes on the other trees will display points in the graphics viewport.

References

ICAD, Inc. *The ICAD System User's Manual*. Version 4.0. vols. 1, 2, 3, and 6. Cambridge, Mass.: ICAD, Inc., 1993.

Appendix B

Overview of ICAD Concepts

This appendix describes the ICAD software, including the ICAD Design Language (IDL) and the browser user interface. Only the major ICAD concepts are presented here. More complete documentation is available in the ICAD User's Manual (ICAD 1993).

B.1. ICAD Design Language (IDL)

ICAD product models are created using the ICAD Design Language (IDL). IDL is a super set of common lisp and is geared toward modeling products in the ICAD browser interface. IDL extends the capabilities of common lisp, while keeping the functionality of common lisp intact. IDL is an object oriented language. It is demand driven, such that a variable is only evaluated if it is needed. In a conventional programming language, like Fortran or C, all of the variables are evaluated in sequence, whether they are needed or not (Harmon and Sawyer 1991; ICAD 1993).

Some features of object oriented programming languages are (1) described generically in the first section below and (2) described in the context of IDL, in the second section below.

B.1.1. Objects and Information Sharing

Object oriented languages have sets of building blocks, or objects, that are linked together to create a program. An object is a structure that can store both information and procedures. A structure is a generic term for a pattern of organization. In conventional programming languages, a variable is a structure that stores information. The information is usually of type integer, floating point, or string. In the C programming language, there are structures that combine multiple types of information into one structure (this is done with the struct command). One structure can have string, integer, and floating point information (Harmon and Sawyer 1991, 2).

An object is an even more general structure that combines both information and procedures. Whereas information just sits around waiting to be looked at, a procedure is a routine that is executed when it is called. One reason to use objects is to separate the program into categories such that one object only contains information and procedures representative of a certain class of problems (Harmon and Sawyer 1991, 4).

Objects are linked together in three different ways to create an entire program. One type of linking is called inheritance. Inheritance occurs when one object assumes all of the information and procedures from another object. In Figure B.1, Object A is said to define a class of objects. The horizontal and vertical lines in Figure B.1 represent lines of inheritance. Object A has a base set of rules and procedures which objects B, C, and D augment or modify in order to create a more specialized object (Taylor 1990, 29).

For example, imagine that the objects in Figure B.1 represent a set of living animal cells. Object A defines the part of the parts of the cell, i.e., the membrane, the nucleus, the mitochondria, etc. Objects B, C, and D represent specialized cells, like a *muscle cell*, a *blood cell*, and a *nerve cell*, each with the same parts defined in object A. A *blood cell* is a type of *living cell* (Taylor 1990, 29). In ICAD terminology, objects B, C, and D mix in the attributes of object A (ICAD 1993). This type of object linking will not be discussed in the rest of this appendix.

The second type of linking, in an object oriented system, takes place when one object needs information from another object. For example, if one object is calculating the stress in a beam and doesn't have the beam's cross-sectional area, the object will send a *message* to the appropriate object to request the cross-sectional area. Figure B.2 shows five different objects, with different data and procedures. Object A is sending a message to object B. Object B is responding to object A with the requested information. In an object oriented language, one object can send a message to any other object, requesting information (Taylor 1990, 41).

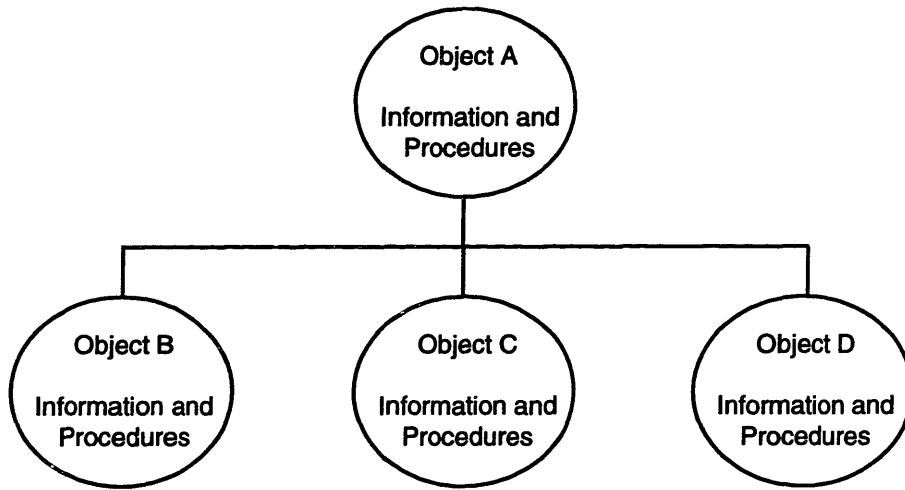


Figure B.1 - Object Inheritance Structure

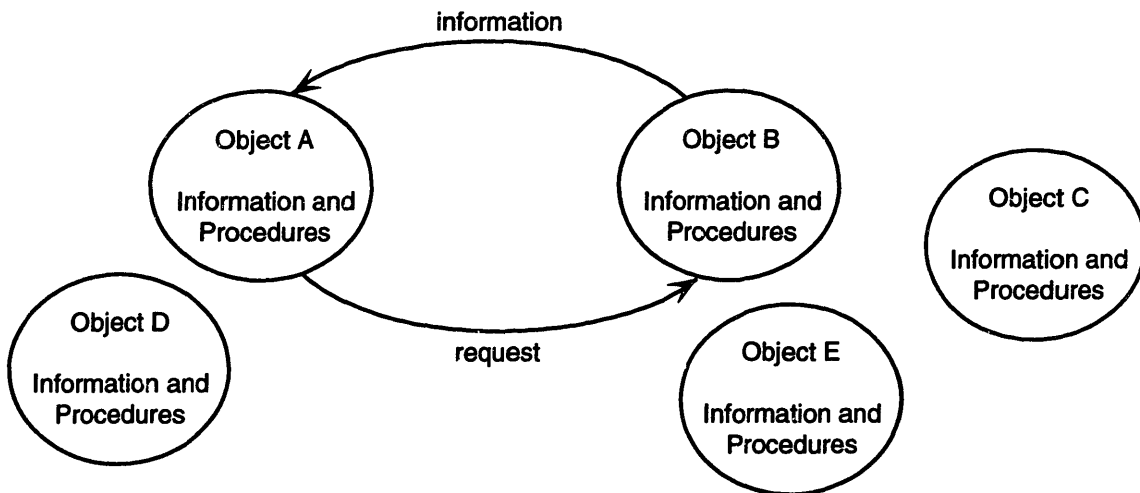


Figure B.2 - Sending and Responding to a Message

The third type of linking occurs when more sophisticated structures need to be represented. This type of linking is actually an assembly of other objects and is called a *composite object*. Figure B.3 shows a composite object, where object A is a composite object with references to objects B and C. The lines in Figure B.3 are diagonal and indicate references from object A to the other objects. This reference is different from the inheritance concept described earlier, in that inheritance describes *a type of relationship*, where the composite structure describes *a part of relationship*. For example, in Figure B.3, a relationship is read as, “Object B is *a part of* object A.” Whenever the information and procedures in object A are desired, the links to objects B and C also come along (Taylor 1990, 39).

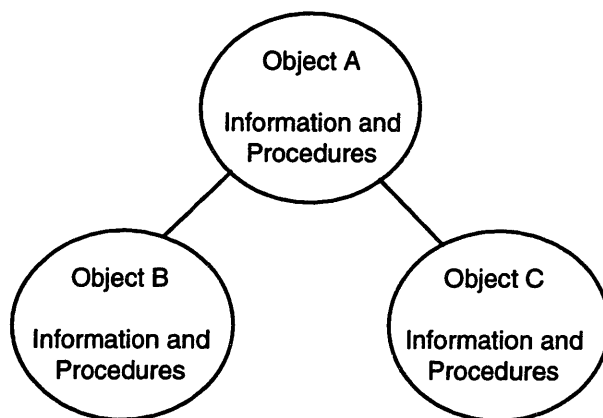


Figure B.3 - Composite Object

Objects B and C, in Figure B.3, could also be composite objects. When one or more composite objects reference one another, the group of objects form a tree hierarchy, as shown in Figure B.4 (Taylor 1990, 39). In Figure B.4, objects A and C are composite objects because they are composed of other objects. Table B.1 explains some terminology associated with a tree structure (ICAD 1993).

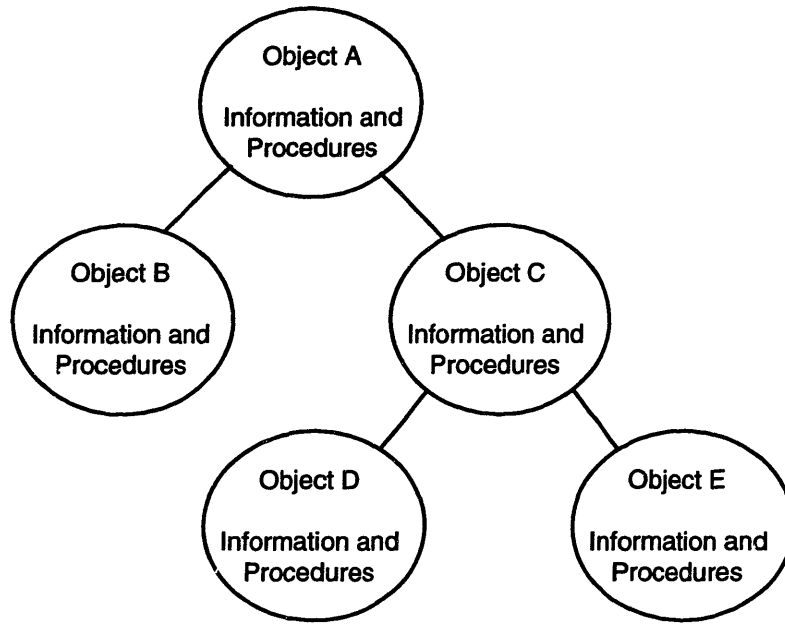


Figure B.4 - Nested⁵ Composite Objects

<u>Term</u>	<u>Example</u> (See Figure B.1)
node	Each object is a node.
root	A is the root node, because it is at the top of the tree.
parent	A is a parent to B and C.
child	D and E are children of C.
siblings	B and C are siblings.
leaf	B, D, and E are leaf nodes, because they have no children.
ancestor	A is an ancestor of all of the objects.
descendants	All of the objects are descendants of A.

Table B.1 - Tree Structure Terminology (ICAD 1993)

⁵ In this case, nested means that one composite object is part of the definition of another composite object.

B.1.2. IDL Description

This section relates the generic concepts described in the last section to IDL specific terms. Some new concepts associated with IDL objects are also introduced. When reading this section, remember that IDL is used to model physical products, such as compressors, engines, and vending machines. As such, objects, information, and procedures are geared for geometric applications.

In IDL, an object is called a defpart, for definition of a part. Just like objects, each defpart can contain both information and procedures. In IDL, the information and procedures are organized into several sections. Each section is titled with a keyword. The ICAD manual describes many different types of keywords, all of which divide the defpart into specialized regions. At a high level of abstraction, the most common keywords categorize information and procedures according to inputs, attributes, and parts. Figure B.5 pictorially shows the sections of a defpart. The arrows show the direction of information flow, both inside and outside the defpart (ICAD 1993).

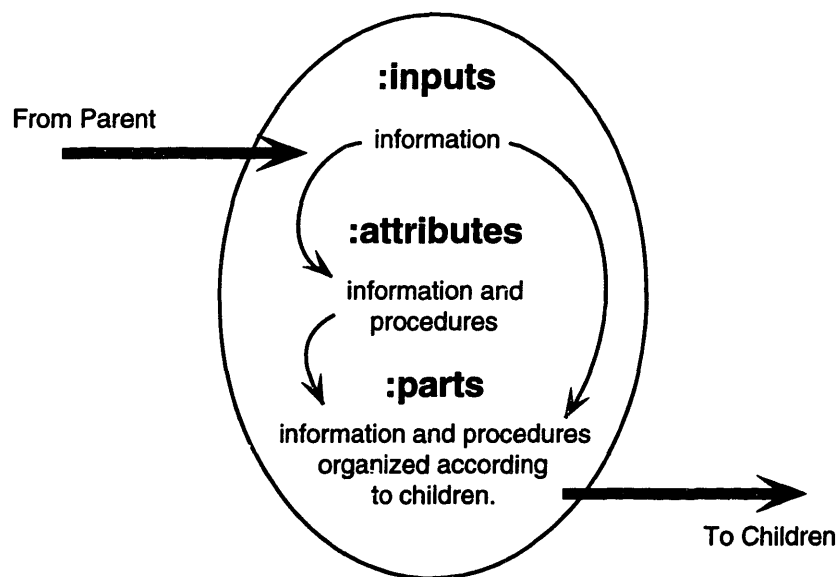


Figure B.5 - Graphic Description of an ICAD Defpart

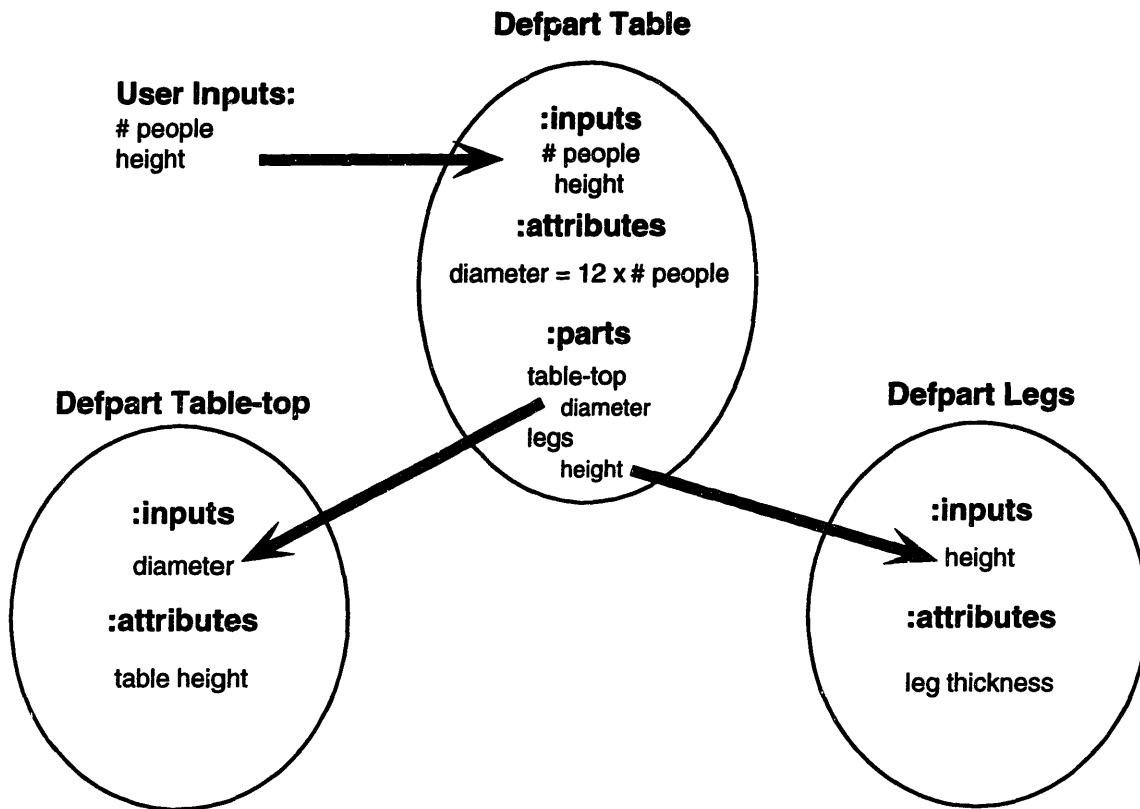


Figure B.6 - Tree Structure for a Table

The inputs section gets information from the parent defpart. If the defpart is the root node, the information is requested from the user. In generic terminology, the defpart sends a message to its parent or to the user requesting information. The attributes section lists the information and procedures that characterize the defpart. For example, a defpart representing a table, might have an input requesting the number of people sitting around a table and have an attribute to calculate the size of the table to accommodate the number of people. The parts section identifies more detailed parts of the table, that will become the defpart's children. In generic terminology, the parts section references other defparts that altogether create a composite object.

In IDL, most defparts are related to each other as composite objects. In a sense, the entire product, modeled in IDL is one big composite object, which forms a tree

hierarchy. All messages are passed up and down the tree hierarchy to the appropriate nodes (ICAD 1993).

Figure B.6 shows a tree structure for a table, along with representative information flows (ICAD 1993). The table is made of a table top and legs. Note that both the table top and legs are listed in the parts section of the table defpart. To create the table, both the legs and the table top need more information, which they get from the table defpart. In order to satisfy this request, the table defpart asks the user for inputs.

<pre>(defpart table (box) :inputs (:height :number-of-people) :attributes (:table-top-height (inch 3) :width (* 12 (the :number-of-people)) :length (* 12 (the :number-of-people))) :parts ((table-top :type cylinder :height (twice (the :width)) :length (the :table-top-height) :radius (the :width) :position (:top 0) :orientation (:rotate :right)) (legs :type box :quantify (:matrix :lateral 2 :longitudinal 2) :height (- (the :height) (the :table-top-height)) :width (the :table-top-height) :length (the :table-top-height) :position (:bottom 0))))</pre>	<p>This defpart defines a table.</p> <p>This is the inputs section. The program will prompt the user for the height and number-of-people when this part is called.</p> <p>This is the attributes section. Set the table-top-height to 3 inches. Set the width and length to be 12 times the number of people.</p> <p>This is the parts section. This section defines the table-top. The table-top is round. Set these values to previously defined values from the inputs and attributes sections. These two lines position and orient the table-top to the rest of the table.</p> <p>This section defines the table legs. The table legs will be square. This expression creates four legs to hold up the table. This is an equation to subtract the table thickness from the leg height. The leg thicknesses (width and length) are set equal to the table thickness. The legs are positioned below the table-top.</p>
--	--

Figure B.7 - ICAD Defpart for a Table

For reference, Figure B.7 shows the IDL code for the table defpart. In the IDL code, all of the words that start with a “:”, but are not keywords are attributes, that contain information. All of the parenthetical expressions to the right of the attributes are procedures to calculate attribute values.

In ICAD, the composite object tree structure, as shown in Figure B.6 is called a product structure tree. Figure B.8 is a generic representation of a product structure tree, since no specific product is shown. The arrows in the figure show the direction of information flow in the system. The attributes in the :inputs section of the lower level defparts ask for information from the higher level defparts(ICAD 1993).

Another type of information flow is represented, in Figure B.8, by the dashed arrow between defparts D and F. In IDL, this type of information request is called a referencing chain. Defpart F needs information from defpart D, but needs to specifically send a message to defpart D requesting information. In order for the information to go from defpart D to defpart F, the information must travel up and then back down the tree, along the path D-B-A-E-F. ICAD does not recommend this cross-branch information sharing, because the product model becomes confusing to implement and hard to decipher. The referencing chain also makes the product structure tree non-generic, because defpart F can only be used in its current position in the tree. A generic defpart can stand alone, or can be called from anywhere in the tree (ICAD 1993).

Up until now, only the structure of a product model has been described. The collection of defparts in a product structure tree only organizes information for use in a product model. An actual design is created when the product model is *instantiated* with specific values. One instance of a table is a tall, wide table, five feet high, with twenty people seated around it. A different instance of a table is a small, squatty table, one foot high, with four people seated around it. The process of creating one specific product in a product model is called instantiation (ICAD 1993).

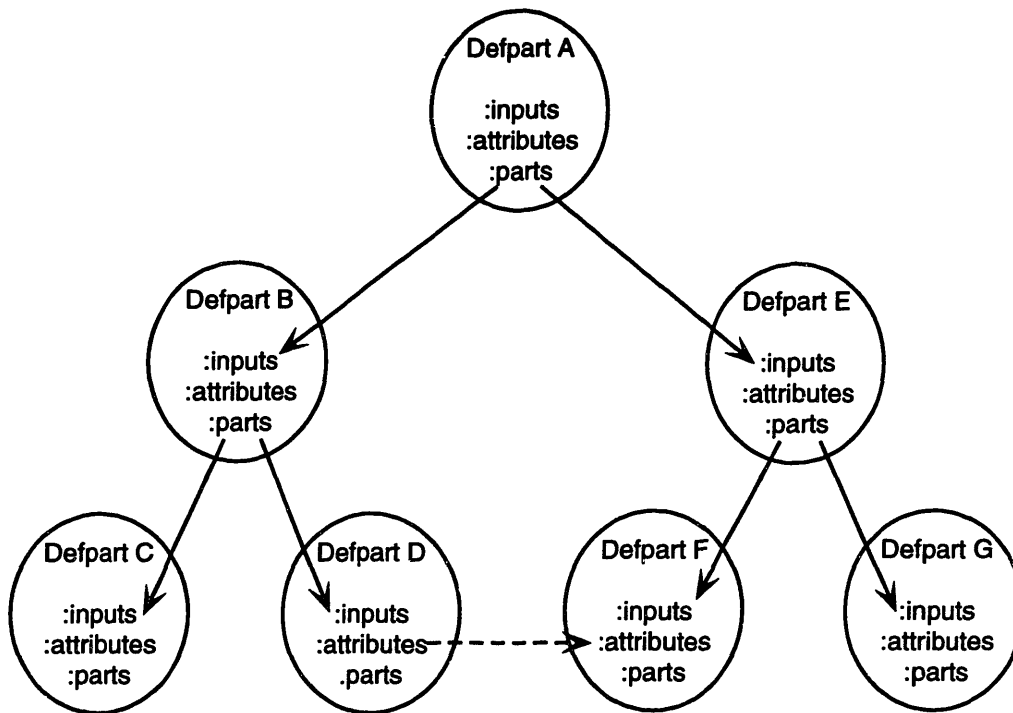


Figure B.8 - Generic ICAD Product Structure Tree

As shown in this example, the ICAD defparts and product structure tree enable the ICAD parametric capability (the ability to create several instances from one product model). In the table example, the table defpart uses equations to relate table size to input values, such that different table instances are created just by changing the input values. The ICAD system also allows changes in part geometry with different inputs. For example, an equation could be written in the table defpart to make a round or square table top, depending on the number of people sitting at the table.

There are many more concepts about object oriented programming and IDL which are not explained here. For more information, see a general reference book on object oriented programming (Taylor 1990), the ICAD user's manual (ICAD 1993), or a common lisp reference book (Steele 1990).

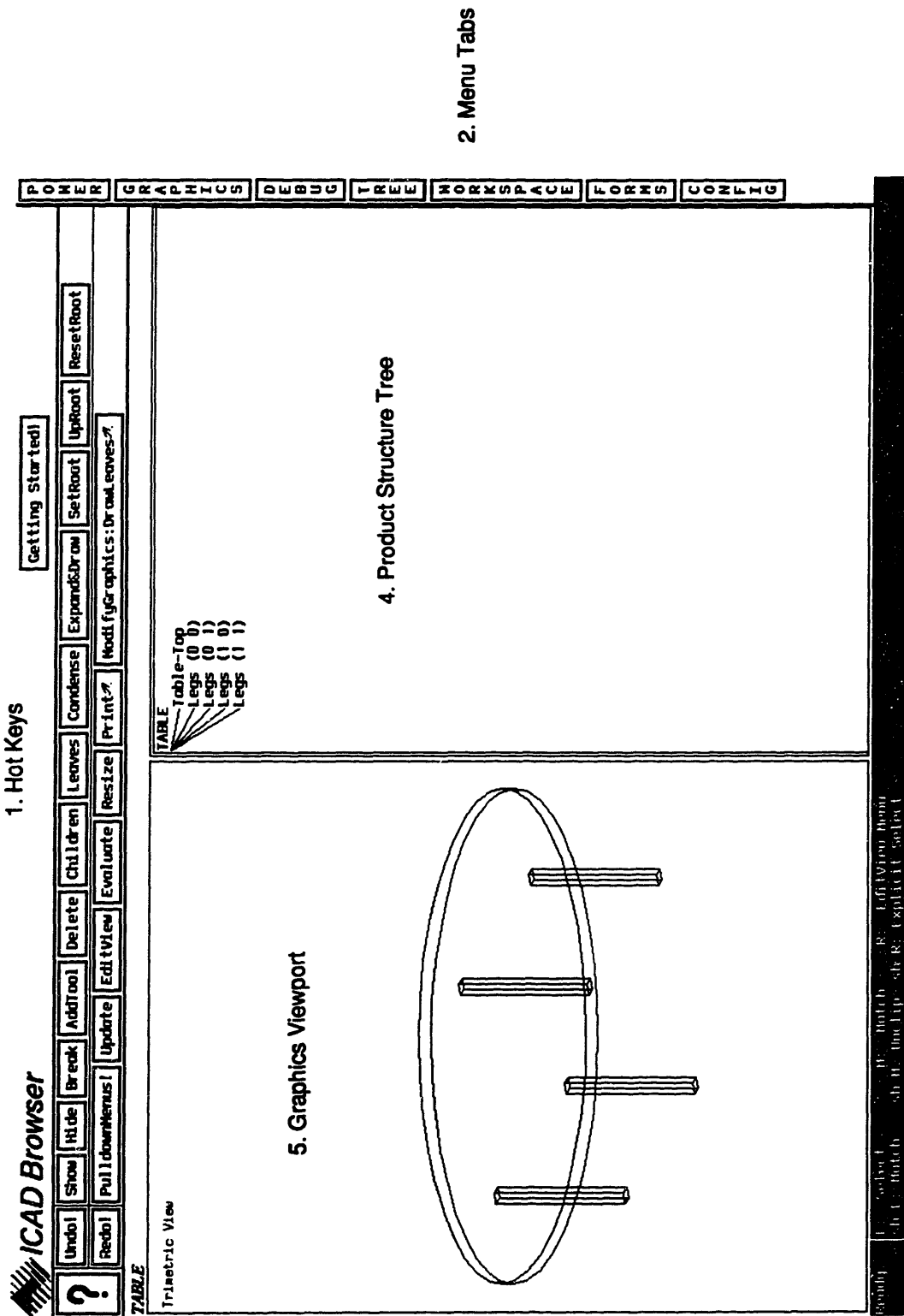
B.2. Browser

The browser is a user interface environment that reads the product models defined in the ICAD Design Language. The browser interface is used to help develop and debug product models and to provide an environment for the production user to view a product model.

Figure B.9 shows the browser, with an instantiated table defpart, as described in Section B.1.2 on the ICAD Design Language. There are five parts to the screen, numbered 1 through 5 in the figure. Across the top and right side of the screen are menu options (1 and 2). The bottom line is a status bar (3). The two parts in the middle of picture comprise the workspace area (4 and 5).

The workspace has two parts: the product structure tree on the right (4) and the graphics viewport on the left (5). Both the product structure tree and the graphics are specific instances of the product model. The product structure tree was defined in IDL, using defparts. Sometimes, when modeling large products in ICAD, it is hard to keep track of all of the relationships among all the defparts. The product structure tree facilitates the creation and interpretation of the product model. Since each name on the tree corresponds to a user defined defpart, it is possible to open an inspector window to look into a defpart on the tree to examine its attribute values.

The geometry displayed in the graphics window matches the same instance represented by the product structure tree. However, the defpart name does not have to be visible in the product structure tree for the geometry to be displayed in the graphics viewport. The graphics window displays the three dimensional part geometry. Depending on the settings, the view can be changed to show orthogonal graphic views, such as top and front, or a custom user view. It is also possible to zoom in on the graphics image. The geometry of a specific part is viewed by drawing the appropriate defpart in the product structure tree.



2. Menu Tabs

Figure B.9 - ICAD Browser

Along the right side of the screen is a set of menu tabs (2). The menu tabs pull to the left to display a menu of commands. The commands are grouped according to those that change the graphics, manipulate the product structure tree, alter the workspace, debug the product model, and configure the browser.

Quick menu commands are provided along the top of the screen (1). The hot keys across the top are a customizable set of buttons that facilitate exploration of the product model. Scrolling through levels of menus is time consuming and cumbersome. Some of the most frequently used commands are initially assigned to the hot keys. However, if a different set of commands is desired, new commands can be pasted into place. The user can even define commands that are connected to the product model.

The bottom status bar (3), indicates whether the system is working or waiting and gives helpful messages on how to use selected commands.

For more information on the browser, see volume 3 of the ICAD User's Manual.

References

ICAD, Inc. *The ICAD System User's Manual*. Version 4.0. vols. 1, 2, 3, and 6. Cambridge, Mass.: ICAD, Inc., 1993.

Harmon, Paul and Brian Sawyer. *Object Craft: A Graphical Programming Tool for Object-Oriented Applications*. Reading, Mass.: Addison-Wesley, 1991.

Steele, Guy. *Common Lisp: the language*. 2nd ed. Bedford, Mass.: Digital Press, 1990.

Taylor, David A. *Object-Oriented Technology: A Manager's Guide*. Alameda, Calif.: Servio, 1990.

Appendix C

User Interface

C.1. Introduction

This appendix describes a conceptual user interface for QCAD at a high level of abstraction. The goal is to convey an idea of what the user interface should be able to do, without discussing any concrete details about what the screens should look like, or how they should be implemented. This appendix is meant to be a starting point for the development of an user interface.

The user interface should provide easy access to QCAD's capabilities for both the initial programmer and the end user. There should be two modes: input mode and design mode. The input mode is used to store permanent information in the system, such as part geometries, parametric relationships, concept review matrices, and process definitions. The design mode is used to explore designs and to change information stored in variables. It should be possible to save the state of the system when in design mode.

In input mode, the user adds to the knowledge base of the system. However, in design mode, the user *plays* around with the knowledge already in the system, making non-permanent changes to the stored information. In design mode, the user should also be able to select a product model to explore.

As discussed in Section 4.3, there are multiple views of a product. The user interface should be able to display all of the possible views, from functions to hardware to processes and from total system to part features.

Since QCAD is specified with three hierarchies, the user interface should support three primary screen sections: function, hardware, and process. Other screen sections, such as menu bars and status bars, should also be available. In each of the primary screen sections, the user should be able to move up and down in the abstraction levels. The discussion here refers to sections of one display screen, but the user interface can be

different (i.e., composed of separate screen windows). Each section must be linked, such that changes in one section are reflected in the other sections. The user should also be able to look at any screen section at will.

The rest of this appendix discusses elements of the user interface.

C.2. Function Interface

The initial screen for the function interface is the total system function net. At the total system level, there is only one function. Thus, the function net is just a circle, with the function written inside. The user should be able to change the level of abstraction to see function nets at different levels in the hierarchy. It should be possible to view all of the functions on that level at the same time, even though some functions may be part of a different function net.

The user should be able to modify the function net, by adding and removing functions, and by changing the links between the functions. A different function net arrangement may result in a different hardware configuration.

Furthermore, the user should be able to look at two different sets of information: (1) information related to the layer, and (2) information related to an individual function. Some of the information at the layer level includes the house of quality, concept selection matrix, and concept review matrix. Each function also has related information, including constraints and related sub-functions. Additionally, each function should be able to look at its associated hardware.

C.3. Hardware Interface

The hardware screen has two parts. One is a graphics viewport. The other is a hierarchical representation of the product structure. The viewport is used to display and analyze the product geometry. The user should be able to look at the geometry for interferences and should be able to send a part geometry to an analysis program.

In the viewport, the whole product can be displayed, or just pieces of the product, depending on what the user selects. The hierarchical representation helps the user to select hardware to be displayed.

The hardware is represented by a product structure net. Just like the function net, the user can change the level of abstraction to see different product structure nets. The user can select different hardware nodes for viewing, or can select the whole structure to draw the entire product.

Additionally, the user should be able to look into each hardware node. For information about sub-hardware, hardware parameters, the QFD design matrix, and the manufacturing process. The hardware node should also be able to display all of the functional requirements that it satisfies.

C.4. Process Interface

Just like the other interfaces, the process interface shows the process flow at several levels of abstraction. The higher abstraction levels show diagrams of how the parts fit into an assembly. The lower levels show the process steps to manufacture a piece part. Each element on the flow diagram should be set up such that a user can look at the parameters, QFD matrix, and design of experiments for that process. The user should also be able to display the hardware made by that process.

In addition to displaying a process flow chart, the user should also be able to look at a specific process type, which should give information about the process, the company's capabilities with the process, and the parts made with the process. It would be useful to display all of the hardware made by one process simultaneously in order to both refine the hardware design and to plan the manufacturing process.

C.5. Miscellaneous

The user interface should also have features that affect the whole product model.

One such feature is to display the fault tree. Fault information is dispersed throughout the model. There should be a command that collects all of the faults and displays them in a fault tree. This command should also be selective such that a user can define keywords that only find faults related to a specific area. This fault command could also keep track of fault information, i.e., the number of times a fault occurs.

Furthermore, there should be a command that finds inconsistencies in the product model. The user should be able to define places where problems might occur. If these problems develop, the command should display a list of the inconsistencies, from which the user can select one to explore. Upon selection, the display changes to show the area in question, with the inconsistency highlighted.

Another important capability for the user interface is the display of generic information about the product, such as the total cost and bill of materials. The user should also be able to define other information to be displayed.