EVENT-DRIVEN KNOWLEDGE-BASED DESIGN

by

Gavin Alexander Finn

S.M., Civil Engineering
Massachusetts Institute of Technology, 1985

B.S., Civil Engineering
Oklahoma State University, 1983

Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy
in Computer-Aided Engineering

at the Massachusetts Institute of Technology
June 1995

Signature of Author: _____

Certified by: _____
Jerome J. Connor
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by: _____
Joseph M. Sussman
Chairman, Departmental Committee on Graduate Studies

EVENT-DRIVEN KNOWLEDGE-BASED DESIGN

by

Gavin A. Finn

Submitted to the Department of Civil and Environmental Engineering
On May 12, 1995, in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in Computer-Aided Engineering

ABSTRACT

In order to achieve competitive advantage and cost leadership in the industrial enterprise, it is necessary to have an extraordinary engineering design capability. Event-Driven Knowledge-Based Design addresses the needs of a distributed, diverse, and dynamic design environment by providing timely access to necessary information and knowledge across the multitudes of participants in a design organization. Event-Driven Knowledge-Based Design Systems are systems that are activated as a result of the occurrence of events within the design process. These knowledge-based systems are distributed across the organization, and play a coordination, control, communications, feedback, and knowledge access role. Such knowledge can relate to product or production cost, manufacturing issues, customer requirements, maintainability, and all other product realization issues. The technology and approach introduced here allows for the reduction in iterations in the design process because it focuses on giving engineers access to broader knowledge than the individual engineer (specialist) typically possesses about the spectrum of issues in product design and, thus, the ability to make better decisions without having to rely exclusively on time-consuming communications with other specialists. Such access is provided in an event-driven manner, so as to concentrate the knowledge supplied by the systems on the context of the state of the product and the design process. Consequently, higher performance levels can be achieved while decreasing the duration of the product development process. Issues addressed in this work include event characterization, representation, and architectures. Three different architectures have been developed: pre-dispositive, intelligent search, and event management. In order to establish organizational contexts for such system, simulation models of design organizations are presented, in the context of comparative design cycle times. The application of Event-Driven Knowledge-Based Design systems are described via an illustration of the three systems architectures in a pilot implementation.

Thesis Supervisor: Jerome J. Connor
Professor of Civil and Environmental Engineering

DEDICATED TO THE MEMORY OF MY FATHER,

ALFRED FINN

Acknowledgements

This work was accomplished with the assistance of many people. I gratefully acknowledge their contributions.

Dr. Kenneth F. Reinschmidt, President of Stone & Webster Advanced Systems Development Services, Inc., has provided a great deal of intellectual guidance, in the form of very specific help with individual tasks in this work, as well as advice on general directions and areas of focus. His advice has provided both a sense of the pragmatic and a vision which has guided me throughout the course of this effort. He spent many hours of his personal time providing comments, reviews, and suggestions. He has contributed in a very meaningful and significant way to all aspects of this work. He has a unique capability to synthesize and innovate, to generate new ideas, and to expand the horizons of technology and knowledge. I have been truly fortunate to work for Dr. Reinschmidt and to be the beneficiary of his help. He has shown a very personal interest in my efforts, and has demonstrated friendship through his tireless advocacy on my behalf. For all of this, and for his continuous support and encouragement, I will be forever grateful.

Professor Jerome J. Connor has provided me with support and guidance for many years, and played a major role in this work. Without his vision, encouragement, and guidance, this work would not have been possible.

I am grateful to Professor Leonard Albano, who has contributed substantially to this work, through his conscientious review of the research, and his honest and sincere critical analysis. He has shown a genuine interest in the success of this work, and has dedicated a great deal of personal time and effort to helping me accomplish this task.

My doctoral program was initiated at the suggestion of Mr. Bruce C. Coles, Chief Executive Officer of Stone & Webster, Incorporated. His encouragement was a key factor in enabling this work, and his personal support allowed me to spend the time necessary in an effort such as this. I am privileged to have received such personal attention and interest, and am very grateful to Mr. Coles for this opportunity, and for his leadership.

I have received support and encouragement from many friends and colleagues at Stone & Webster. I am particularly grateful for the enthusiastic support, and consistent confidence shown by Mr. William F. Allen, Jr., who is, as of this writing, preparing to retire as Chairman of the Board of Directors of Stone & Webster, Incorporated. He has served as a source of inspiration and motivation for me, and has demonstrated the meaning of executive leadership. I was honored by his personal interest in my welfare and my work, and I express my sincere gratitude to Mr. Allen.

Many people at Stone & Webster Advanced Systems Development Services, Inc. provided assistance during the course of the development of the pilot implementation system. I am grateful for the generous help provided by Thomas Mignosa, William Waid, Matthew Mason, and Fabian Donzé. I am also fortunate to have been assisted in a variety of ways by Barbara McMahon, and Elaine Piper.

My parents, Alfred and Lily Finn, always encouraged me to learn, and provided unconditional support for all of my educational experiences. They sacrificed in many ways because of their love for their sons and their desire to enable our education. I will be eternally grateful to them for all that they did, and for their many lessons, including the knowledge that education is a privilege to be cherished throughout one's life.

I am grateful to my daughter, Rachel, who has shown surprising maturity in her understanding and patience through my work. Without the periodic benefit of her loving little embrace, it would have been much more difficult for me to persist in this effort.

I am most grateful to and especially appreciative of one person in particular. Throughout this effort, I have been constantly supported and encouraged by my wife, Judy. She has endured, without objection and with a great deal of understanding, the extended periods during which I worked on the studies and research that constituted this program. My focus on this work placed an even greater burden on her in the management of our household and the raising of our daughter, Rachel - a burden that she has borne without complaint. She has supported me in innumerable ways, not the least of which was in providing constant encouragement and love throughout our marriage. Judy also spent many hours proofreading and editing this thesis, and providing valuable advice and help in a variety of other areas. In all of these ways, she has contributed significantly to this effort. For everything that she does, I thank her.

Table of Contents

List of Figures

List of Tables

1. Background and Introduction

Engineering design is a critical factor in the overall success of the industrial enterprise. Every industrial enterprise requires some degree of contribution by engineers and designers in order to produce goods and services. This contribution can take the form of product design, facilities design, process design, organizational design, or workflow design. While design has often been framed in the context of the engineering process (Dym, 1994; Simon, 1981), it is necessary to understand that engineering design is an important component in the more global missions of almost all industrial ventures. Indeed, engineering design has played a key role since the earliest recorded history of civilization.

> "Even with the earliest peoples at the dawn of history, the periods of economic and cultural prosperity are closely associated with, and indeed dependent on, the high standard of technical knowledge." (Straub, 1964)

In the determination of an industrial enterprise's competitive strategy, there are three traditional or generic mechanisms for formulating such a strategy (Porter, 1980), these being: (i) cost leadership, (ii) differentiation, and (iii) focus. The latter business strategy can be achieved as a combination of the first two. The first two strategies depend heavily on the industrial organization's ability to leverage its design engineering potential. In the realm of industrial products, *quality* is also a key competitive issue. This presents yet a third category of competition, one which is multi-dimensional (Garvin, 1987) depending directly on an organization's design and manufacturing capability.

Cost leadership can be achieved through tight controls on expenditures, or by improvements in the *process*. Thus, one of the key factors in this strategy is

the role of the engineering design in the product design *and* the product realization process, either in the process design itself, or in the people, plant, and equipment necessary to provide process breakthroughs. That is, in order to achieve cost leadership, it is necessary to have an extraordinary engineering design capability, beyond the capability of other players in the competitive field.

Differentiation can be achieved through the development of *unique* offerings, either in the product itself, in the delivery mechanism, or post-sales service. Product differentiation can take the form of unique features/functions, maintainability, reliability, etc. There are other, non-technical aspects to differentiation, such as service organizations, contractual terms and conditions, and others. However, a central issue in differentiation is the ability to provide uniqueness in the product; such uniqueness can be described as distinctive added value to the customer - a direct result of engineering design. A further differentiating factor can be the amount of time it takes to design and produce a product. This *time-to-market* factor is often a critical competitive issue in many industries and markets. It follows, therefore, that the potential for achieving competitive advantage on the dimensions of cost and differentiation (including quality and time-to-market) depends, to a large extent, on the organization's engineering design capability.

As the role of engineering design has become more and more important in the context of the ability of an industrial enterprise to succeed and prosper, the methods and techniques employed within the engineering design process have not seen many drastic changes since the evolution of the modern industrial organization. Traditional engineering design processes are based on many of the same fundamental processes that have existed for more than one hundred years. While some of the tasks in the engineering process have been adapted, in some focused and limited ways, to use new techniques and technologies,

the process itself has not been measurably affected in a positive manner by the use of these technologies. It is conceivable that progress in design has been impeded by the prevailing paradigm of the design process as a manufacturing assembly line, including division of labor among the designers analogous to industrial workers.

The concept of division of labor was first formally introduced by Adam Smith in 1880, when he described how, in the manufacture of pins, the process was divided into specialized tasks and people were assigned to these specialized tasks rather than to many tasks in the overall process. The productivity of the people in the pin-making team, it was argued by Smith, was almost five thousand pins per day per worker. The process included as many as seventeen specialists in drawing the wire, cutting the wire, etc. His argument was that if a single person was to be responsible for the pin in its entirety, then the productivity of the workers would be no more than twenty pins per day per worker! Thus, he reasoned, by creating specialists in particular fields, and by limiting people's roles to these specialties, the productivity of a few people greatly outweighs that of a large group of generalists. (Smith, 1880)

This model forms the basis of how engineering and manufacturing has been organized since the industrial revolution. There exists a division of labor between engineers and designers, and also between engineers and manufacturing personnel. The division between engineers and designers stems from the introduction of the mechanical drawing process at the start of the Industrial Revolution. (Reinschmidt, 1995n) Furthermore, within engineering itself, specialists perform geometric shape definition, materials selection, detailed performance analyses and routine design tasks, illustrating a further division of labor among engineering disciplines. The division of labor between engineers and designers is significant here, because this is one of the factors that inhibits the ability of the design organization to achieve significant cycle

time reductions. This occurs due to the specialist nature of the geometric tasks involved in drawing, or modeling of the designed artifact, as compared with the knowledge and expertise required to perform the engineering design and analysis. The use of drawings has become associated with the designer's activities, rather than the direct visualization mechanism of the artifact that was conceived by the engineer. (Ullman, *et al*, 1988) The distinction between the two classes of activities (i.e. engineering design and drafting, or drawing) in the engineering design process is a product of the system that engineers have devised and organizations have adopted, rather than an inherent feature of engineering design itself. By having to specify to a designer the design intent and other pertinent information (whether the designer uses a drafting table or a computer), the engineer must transfer a certain amount of information to the designer. In this transfer, some information is lost, some information is misinterpreted, and some designer errors are made. These are unnecessary costs that the design process bears. If it were possible to create a visualization and detailed geometric representation mechanism that was integral to the engineering design activity (not a separate drafting task) then many of these errors would be eliminated.

Most engineering design is performed according to processes that have not been designed or optimized for the specific design problem at hand; moreover, they have not been optimized for the group or teaming nature of design practice. Perhaps more importantly, optimization has not taken place with respect to the manufacturing process - keeping in perspective the objective of the design process in the context of the product realization process. Moreover, the **process** of performing the design, while critical in terms of design decision-making, and the ultimate outcome, is not analyzed in practice nor is it optimized.

This current condition in design is not based on the best approach, however.

An argument can be made that there is an inherent increased cost in moving towards a more generalist approach, because engineering specialists posses very important and rare knowledge. (Galbraith, 1973) If the notion that such knowledge cannot be distributed to non-experts prevails, then the specialist (craft) approach will not be altered. By examining why specialists are required in the design process, and developing technical systems that perform highly specialized tasks without requiring the user to be a technical specialist in the task domain, we can formulate a technology-based design approach that would allow generalists to perform in a highly productive manner.

That is, the *craft* nature of engineering is a limiting factor in accomplishing significant process breakthroughs. In the case of manufacturing (which started out as a highly craft-oriented field) dramatic process changes and the development of tools to support the new processes have resulted in significant improvements in productivity. By maintaining this craft model, the expansion of the role of each specialist to include decision-making or analysis of design decisions *vis-a-vis* other disciplines or areas is not possible. It is necessary to create mechanisms that allow for the distribution of specialized knowledge to non-specialists, in order to break out of the constraints of the master-engineer process model. In other areas, when the craft nature of the process has been changed, significant process improvements have been realized. For example, through the orientation away from craft production and towards more mechanized production, the level of effort required to assemble the major components into a complete vehicle was reduced by eighty-eight percent in the span of one year. (Womack, *et al,* 1990) Striking gains can be made by changing the prevailing paradigm of division-of-labor within engineering design. For example, when it has been necessary to develop secret aircraft for military purposes, engineers have functioned in multi-disciplinary roles on small design teams. These aircraft programs were characterized by highly innovative designs, and were often brought to production faster than traditional programs.

(Rich, 1994) Not only is there a requirement to change the paradigm, but it is also necessary to develop tools to support new ways of working.

Changes in business processes result from an understanding of the value of the processes in the overall business operation, and the desire to improve the operation through the transformation of the underlying processes. (Davenport, 1994) In order to generate an impetus to change the engineering design processes, the value of engineering design must be understood by the managers responsible for the overall organizational performance. A potential reason for the lack of evolution of the engineering design process stems from the perceived value of each element of the product realization process, and the relative lack of value-added that may be perceived by business managers of the technical steps in the process. (NRC, 1991) More importantly, engineers may not be able to elucidate the value of the design activity, and this may lead to a lack of appreciation of the potential value. For example, it may be possible to improve a product's design in some way through an iterative optimization of its shape. This iteration may take a significant amount of time, and this time would delay the product's availability in the marketplace. As such, it is necessary to understand the financial trade-off between having a superior product (and whether that would translate into higher sales figures - thus more revenue) and the negative impact of not being able to release the product to the marketplace for the duration of the additional design iteration.

Time-to-market has become an increasingly significant competitive factor, for both simple products, such as bearings or fasteners, as well as for complex products such as automobiles and aircraft. This scenario is further complicated by circumstances in which contractual obligations to deliver a product or service by a particular date limit the amount of time that can be spent in the engineering design process, and thus implicitly limit the degree to which engineers can spend in improvements or optimizations. This inertia is magnified

when the constraints of the traditional design process paradigm are not released, thus further limiting potential design improvements because *design process changes* are not even considered.

Much effort has been spent on the improvement in the manufacturing process, in part because this is a tangible entity - physical machinery producing physical products. This trend, however, in its limit, produces a highly efficient manufacturing process with little flexibility or variation allowed. (Henry Ford's Model T assembly line was an example of this process in the limit.) The modern competitive industrial enterprise cannot afford to be inflexible, rigid, and unable to adapt to new requirements. Again, this realization has lead to a focus of research on *flexible* manufacturing (the ability to modify a manufacturing process interchangeably within a set of pre-defined and known variations.) This concept has been extended into research in a new field called the *agile* manufacturing process, in which a manufacturing organization can adapt to *unforeseen* variations.

Note, however, that these efforts have been largely directed toward the production phase of the product realization cycle. In order to leverage the power of engineering design capabilities, it is necessary to create new design engineering processes that can accomplish the desired optimizations, modifications, enhancements, and improvements without incurring the costs that would result in the traditional process. It is essential, therefore, to evolve current design engineering methods, and in some cases perhaps to create entirely new processes.

The product realization process includes concept formulation, design, detailed engineering, component manufacturing, procurement, assembly/construction, and testing/evaluation. In recent years, some attention has focused on the reduction of the cost and duration of the product realization process. This has

often translated into the goal of reducing the cost and duration of every *element* of the process. It is obviously important to operate as efficiently as possible, at every stage of the product realization process. But while the global objective of cost minimization is important, that does not always directly translate into the reduction in cost or duration of each element of the process. Additionally, the assumption that cost and duration reductions in each step of the product realization process are necessary precludes the notion of non-linear, non-sequential processes, such as parallel steps. Many engineering managers argued against the use of three-dimensional computer-aided-design systems, insisting that the cost of using these systems was greater than the cost of a two-dimensional drafting system. Even if this argument were true, the downstream benefits of having the three-dimensional data can far outweigh the alleged increase in up-front engineering cost.

Although the engineering design component typically ranges only from five to ten percent of the overall product's initial cost; even less of the life-cycle cost, the importance of this part of the product realization process is paramount. The quality of a resulting design is, to a large degree, a function of the process that was used in designing the product. If, for example, engineers had time to perform numerous "what-if" studies, then a product might be of higher quality than if the engineering schedule allowed for minimal optimization of the overall product design. Moreover, the impact of engineering in the product realization process and the *life cycle* cost of the product is dominant. The effect of engineering on cumulative percent of life-cycle cost, as illustrated in Figure 1-1, can be as much as eighty percent (Nevins, 1989) and even higher if the development of manufacturing process plans is included in the definition of engineering. Other studies have shown the design process to be five percent (5%) of the direct cost of products, while the influence of the design process is seventy percent (70%) of the product cost. (Boothroyd, *et al*, 1994) Other influencing factors include the cost of materials, labor, capital, etc.

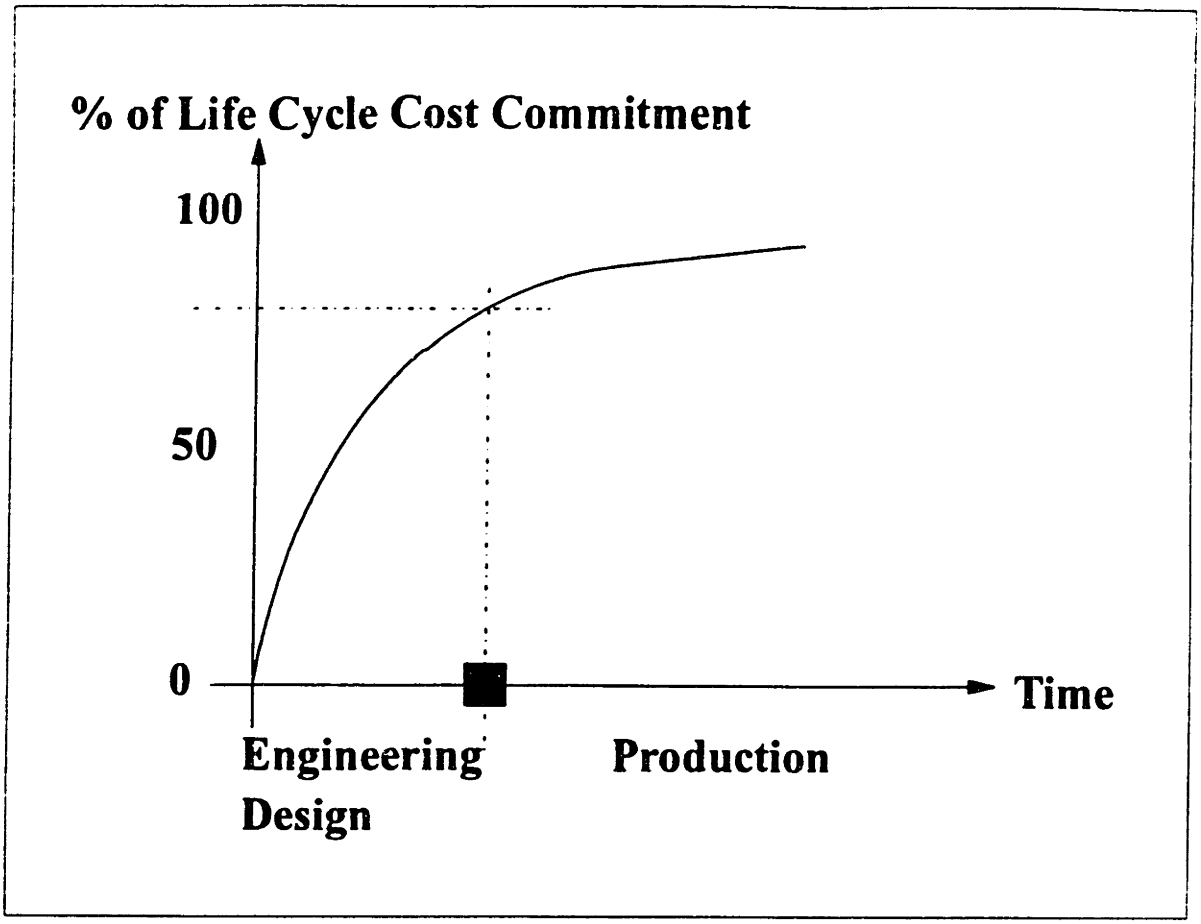


Figure 1-1: Cost Impact of Engineering (After Nevins and Whitney, 1989)

Following this knowledge of the value of engineering as a contributor to the product development cost, if we examine the total product development cost, and relate the cost of the engineering design process to the manufacturing cost, it can be postulated that an inverse relationship exists. That is, the more engineering that can be done, the lower the production (or manufacturing) costs would be. With respect to the total product cost, an optimum amount of engineering can be seen, such that the summation of the cost of design and the cost of manufacturing results in the lowest overall product realization cost. While many attempts are made to minimize the engineering cost of a project, if the cost of the engineering effort is increased by 20%, then the maximum overall product cost might increase by only 1% (if engineering is 5% of the

total product first cost.) It is more likely that an increase in the cost of engineering, if appropriately applied, would **reduce** the cost of the product because of the improvement in product design or manufacturing/ assembly/construction processes that would result. The argument made here is not for an increase in the cost or duration of the engineering effort, but rather that more attention should be focused on the design process itself, relative to the overall life-cycle improvements that can be achieved by better design of the design process. In many cases, a change in design process will reduce the cost of the engineering process, and will also result in overall product cost reductions. It is important to focus on addressing the engineering content of a project from the perspective of maximization of overall benefit, rather than simply minimization of the engineering cost. This concept is illustrates in Figure 1-2.

The actual amount of engineering is less than the optimal, and thus another effect must be accounted for: the effect of design delays on the overall product cost. As such, a further cost effect is introduced to the above analysis, that is, the imputed costs of time-to-market (Reinschmidt, 1995n.) In this analysis, depicted in Figure 1-3, the optimal amount of engineering is seen to be less than in the previous figure, as a result of the cost of delays in time-to-market. As can be seen from these curves, adding to the amount of engineering (by adding more personnel) can only decrease the product realization costs if additional time delays are not incurred. Therefore, it is necessary to find ways to do more (and better) engineering without associated time delays.

Competitive market conditions also contribute to the requirement for the adoption of different design practices. Such market conditions include an increase in the availability of low-cost engineering labor capacity in a variety of countries around the world, a world-wide reduction in the volume of large government funded military and civilian projects, a change in business practices

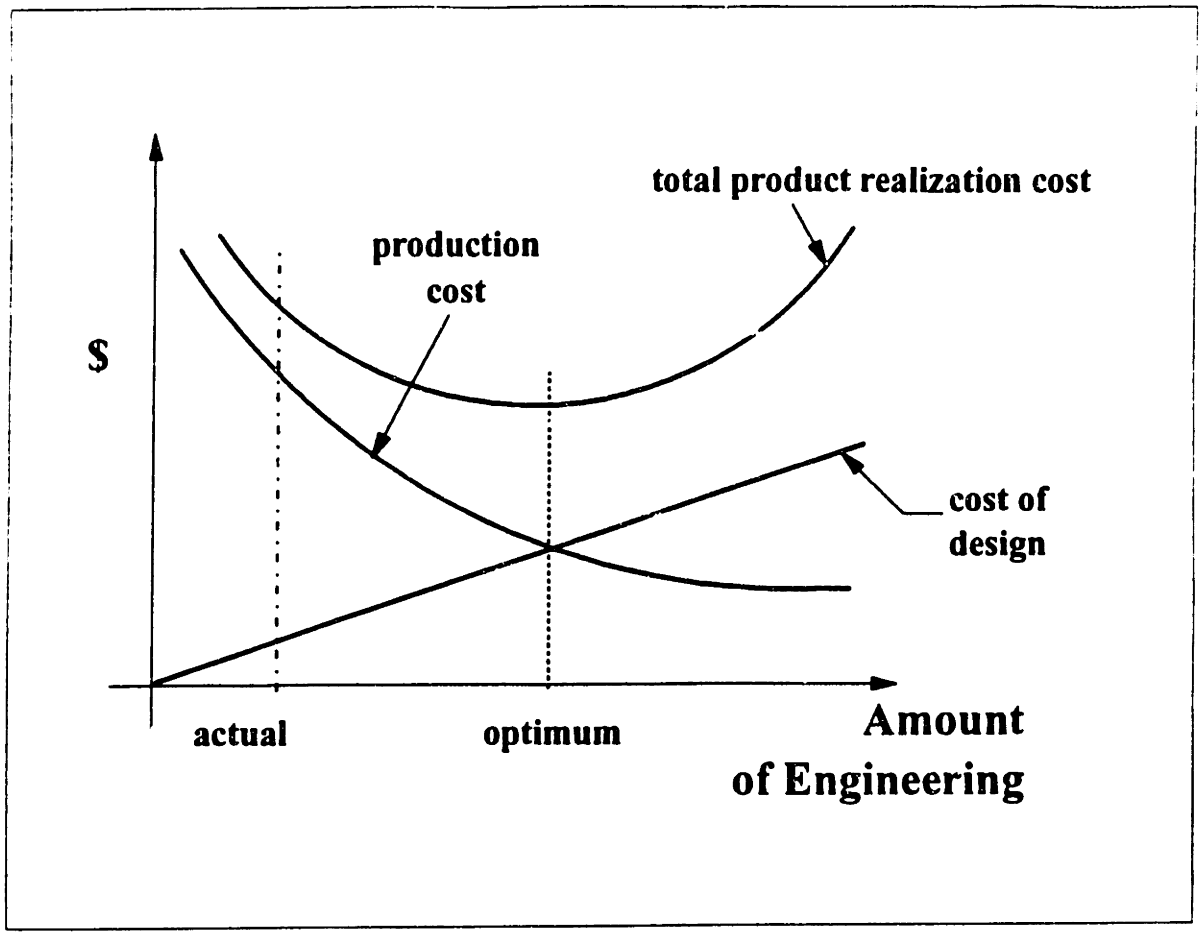


Figure 1-2: Product Cost as the Summation of Engineering Design and Other Production Costs

related to traditional engineering services (e.g. away from cost-plus contracts towards fixed-price, lump-sum contracts), and at the same time, greater focus on meeting consumer needs through product customizations, two potentially conflicting conditions. It is no longer practical to maintain the craft paradigm of engineering design, but it is also clear that a mechanized mass-production model of uniform product design will also not suffice. These conditions require new design strategies and design process tools that allow for the distribution of specialized knowledge, and the ability to reduce design cycle times.

The duration of the engineering and design development process has become a more significant competitive factor. Reduction of product lead times has become a principal competitive advantage in many industries. For example, it

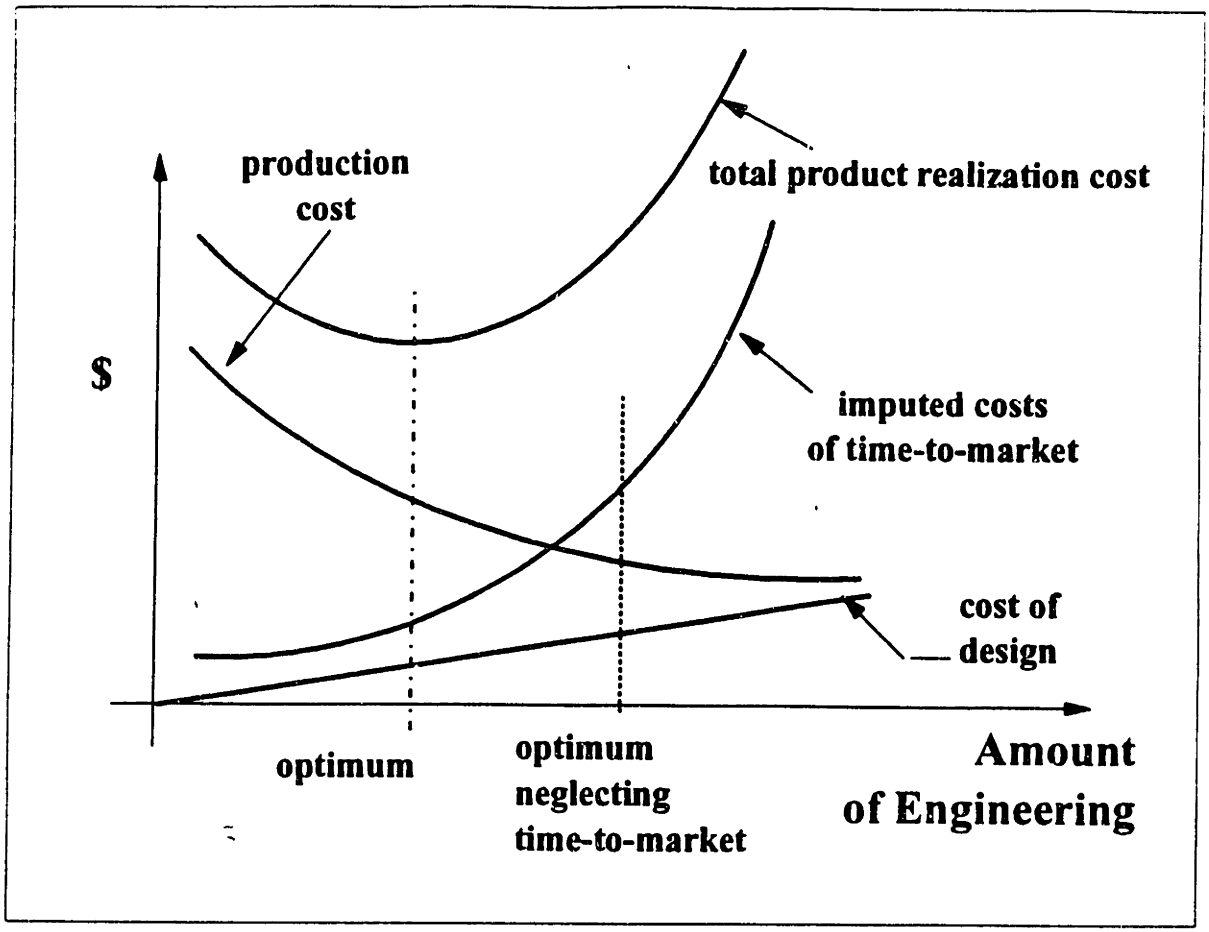


Figure 1-3: Production Costs Including the Imputed Costs of Time-to-Market

has been estimated that each additional day of lead time in the introduction of a new model automobile costs the manufacturer a great deal in lost profits (not merely lost revenues.) Therefore, an automobile manufacturer with a lead time to market of four to six months longer than the competition could lose tens or hundreds of millions of dollars in profits.

> Each day of delay for an average automobile has been estimated to cost a firm about $1 million in lost profits, thus amounting to hundreds of millions of dollars in potential additional profits for companies that are merely four or five months faster to market than competitors with comparable products. (Cusumano and Nobeoka, 1990)

This phenomenon also exists in the semiconductor industry, where the lead time for product development (including design, tooling, and manufacturing) form the basis for global competition. In the consumer electronics industry, it is common for a product to have a cycle time of three months from design through production and distribution. (Goldman, *et al*, 1995.)

The provision of an enhanced design engineering capability that would boost the current practice will have a leveraging effect. A lever is a mechanism that is used to overcome a certain resistance at one point by means of applying power at another point at a high multiplier. The effect that a more powerful and better design engineering process will have is to improve the overall effectiveness of the organization as a whole. This can be accomplished by making better use of information and knowledge, improved communications, and streamlined coordination in the process.

The concept introduced here, Event-Driven Knowledge-Based Design, is a means for capturing and distributing knowledge throughout the design organization in such a manner as to allow engineers to access and use *knowledge* (as opposed to information) about how other parts of the product and process are affected by their own decisions, and how they are affected by events in the design process. Event-Driven Knowledge-Based Design is not a theoretical design methodology. This is an attempt to build an infrastructure that embodies and distributes the knowledge of specialists and experts within the basic technical computing systems that *support* the design process. Having experts and specialists who are excellent at a specific task does not necessarily imply that this expertise is available to others in the design process in a manner that is accessible and available when it is needed. Similarly, having computing systems that provide individual specialists with the ability to perform their activities better or faster does not automatically provide the overall organization with the benefits of this improved capability. Indeed, "increases in the stock

of knowledge [affect] output only insofar as the increases are uniquely related to **embodied technical change** of physical capital." (Bahk, 1993) The technology and approach introduced here allows for the reduction in iterations in the design process because it focuses on giving engineers access to broader knowledge than the individual engineer (specialist) typically possesses about the spectrum of issues in product design and, thus, the ability to make better decisions without having to rely exclusively on time-consuming communications with other specialists. Such access is provided in an event-driven manner, so as to concentrate the knowledge supplied by the systems on the context of the state of the product and the design process. Consequently, a higher performance level can be achieved by the product realization organization as a whole without increasing the duration of the product development process.

In order to improve the overall effectiveness of the organization, it is necessary to consider the collaboration of the different people and functions involved in the product realization process. The existence of multi-party, cross-disciplinary design projects is not new. Facilities construction has traditionally required the close cooperation of architects, structural design companies, mechanical contractors, excavation contractors, and professional project managers. The new reality, however, is that design organizations may now be geographically dispersed, as in the case of multi-national companies or collaborative partnerships. The design organization may also be separated by language barriers, as in the case of design teams that span many countries. Additionally, design teams are also increasingly operating on a single design with temporal disparity. In the design of the Boeing 777 commercial aircraft, for example, Boeing and its suppliers together operated eight thousand CAD workstations across seventeen time zones. (Hughes, 1995) That is, while a multi-party design team, operating all over the world may operate on the same data, they do so in the context of the time differences between their respective locations. Hence a heightened need for coordination and communication.

One of the most significant reasons for attempting to provide some mechanisms for improving the design process is that *change* in the requirements, environment, constraints, and even context of the design process has become more of a competitive issue than it was previously. The ability to respond to changes in market conditions, customer requirements, and technologies is now a major competitive issue, relative to product design. That is, changes in requirements or other design parameters have always been a part of the product realization process, and these changes have traditionally resulted in significant increases in the cost of the design process, as well as significant expansions in the duration of the design process itself. Economic conditions and competition have resulted in a pressing need to create design infrastructures that respond well to change and adapt in an agile manner without incurring costs or delays that would result in a loss of competitiveness. In traditional design organizations, changes often result in partial or complete re-designs because the design methods used are rigid, rather than flexible.

The potential use of computer technology to facilitate the forging of new engineering design processes represents an opportunity to expand on a vast amount of practical applications of design systems already in place. While many clerical tasks are suitable for traditional computer applications, Engineering Design " ... is where the action is in the human-computer interface. " (Card, *et al*, 1983) The role of the computer as a facilitator of process change is not unique to engineering design. While data are not consistent on this subject, studies suggest that the investments in computer hardware and software through the 1980's, combined with restructuring of organizational topologies have had a direct impact on the overall productivity of United States industrial enterprises (Gleckman, 1993) despite a lack of corresponding general economic growth. Although some have argued that "... there is no clear evidence" that computer technology has improved productivity or profitability (Thurow, 1991), it has been proven using econometric techniques that computing technologies

have, indeed, had a positive effect on industrial productivity:

> "[the use of computer technologies is] correlated with significantly higher output at the firm level ..... [and that] computers were associated with more output growth in the sample period than all other types of capital combined ..." (Brynjolfsson, 1994)

For most new technologies, a characteristic phase lag in terms of consistent benefit versus initial implementation in industry has been observed. (Yates, 1989) Further detail is provided on this phenomenon *vis-a-vis* Computer-Aided Design in Chapter 3. Since computers have been in use for many years, and due to the maturity of the software and hardware technologies in the engineering context, there are few, if any, real impediments to the adaptation of engineering design processes to new configurations with integral participation of the computer systems. It is necessary to provide environments that *integrate* the use of the computer systems in the process so that engineers and computer systems interact on the basis of collaboration. Thus, the computer must simplify, not add to the complexity of the designer's task. Through both informal observation and formal study, it has been noted that

> "[W]hen computers stand between [the task and the engineer] they act as an intermediary, requiring that the user be proficient in both the task domain and with the intermediary (the computer). Inasmuch as operations required of the intermediary are in addition to and orthogonal to the task, they can add to the overall complexity" of the design problem. (Norman, 1985)

As in other industries, engineering design can be significantly improved through the development of computer-integrated processes.

Prior to the availability of computers, physical systems played a large role in the ability of an organization to compete and be effective. For example, the development of a manual filing system was a critical issue in business in the early 1900's, since the recording of business transactions and information was becoming a bottleneck for large organizations.

> " ... it is impossible to sever the problem of finding a good practicable filing system from the whole problem of business organization." (Cope, 1913)

In a variety of ways, many industries have changed the basis for competition through the use of computers. (McFarlan, 1984) This type of competitive revolution in the engineering design domain must be viewed from the perspective of a change in two dimensions: first, a design process transformation must take place, and second, the role of the computer in the design process must evolve. The fundamental technologies for achieving such change have been developed, and this must translate into new ways of applying these tools.

> ***As we change what computers can do, we must change what we do with computers.*** (Hopper, 1990)

The appropriate implementation of technology-based engineering design process change can have significant effects on the engineering market, as the influence of the technology can transcend cost advantage or differentiation derived directly from the technology, into changes in the other market drivers of cost or uniqueness. (Porter, 1985) For example, the delivery of electronic systems for maintenance and operations can provide further uniqueness in the competition for market-share in industries such as aircraft, machinery, and process plants. Thus, the cost of the product may have been reduced, and the

quality improved, but a further competitive factor available due to the innovative use of computers as part of the design process, can be the delivery of a system that can help the owner better operate or maintain the product. The use of innovative computer technology in design can have an impact across the entire product development organization and process, as described in terms of the Value-Chain. (Porter, 1985) The use of such technologies as Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) can "drastically change the manufacturing cost structure, capital intensity, and ability to deliver high quality products at low cost." (Hax, 1991)

While the basic building-blocks for integrated design systems exist, the current design processes do not lend themselves to adoption of new technologies. For example, the technology of knowledge-based design provides the ability to integrate engineering and downstream knowledge directly into the design process. This is extremely useful, since this integrated approach can significantly impact the effectiveness of the design process by providing automation, consistency, improved quality, and schedule reductions. If, however, there is no method for design engineers to effectively use this information, then the availability of such information about downstream processes may not be **effectively** integrated into the product design.

The dynamics of the engineering, manufacturing and general product realization processes as described above demand a better way to use knowledge-based systems in the design process. That is, now that technologies for the single user have been developed and, to a limited degree, implemented, new thinking must evolve relative to how to provide such capabilities to the engineering design organization as a whole.

Event-Driven Knowledge-Based Design systems address the needs of a distributed, diverse, and dynamic design environment by providing access to useful (necessary) information and knowledge across the multitudes of participants in a design organization. Event-Driven Knowledge-Based Design Systems are systems that are activated as a result of the occurrence of events within the design process. These knowledge-based systems are distributed across the organization, and can play a coordination, communications, and knowledge access role. Thus the capabilities of the knowledge systems are leveraged. It is this coordination, communications, and knowledge access capability that provides the context for event-driven knowledge-based systems as a technology extension beyond existing systems. The benefits of such integrated, distributed systems address both the technical needs of the design organization and the business needs of the enterprise in general, due to the breadth of scope of domain knowledge that can be accessed by any and all design engineers. Such knowledge can relate to product or production cost, manufacturing issues, customer requirements, maintainability, and all other related issues in the product realization process.

This work is intended to provide a basic framework for applying specific domain knowledge in a highly distributed fashion to the heterogeneous design engineering organization. The specific vehicles for this technology are those of Knowledge-based systems and Computer-Aided Design systems. In Chapter 2, a review of Engineering Design is provided, and several alternative design organizations and processes are described. A simulation of each of these processes is detailed, and comparative results of these simulations presented. On the basis of these results, a structure for applying Event-Driven Knowledge-Based Design systems is postulated. In Chapter 3, an overview of Knowledge-Based Design is provided, including a review of computer-aided design and knowledge-based technology. Chapter 4 describes the framework for Event-

Driven Knowledge-Based Design. A variety of alternative architectures are shown and detailed. In Chapter 5 a case study of Event-Driven Knowledge-Based Design is illustrated, based on an example problem from the automotive design environment. A summary and conclusions follows in Chapter 6.

2. Engineering Design

Engineering design activities consist of the application of knowledge about scientific facts and heuristic rules, practical problem-solving methodologies, and personal experience and skill. The fundamental purpose of an engineering design activity is to create a representation of an artifact that, when produced, will perform a specific function or series of functions. As part of the engineering design process, many activities are involved in order to use the available tools and information to arrive at a solution that meets the requirements. While there are many domains of design, such as architectural design, industrial design and urban design, engineering design has been characterized as a highly technical process. Engineering design has been described variously as:

> "the process of constructing a description of an artifact that satisfies a (possibly informal) functional specification, meets certain performance criteria and resource limitations, is realizable in a given target technology, and satisfies criteria such as simplicity, testability, manufacturability, reusability, etc.; the design process itself may be subject to certain restrictions such as time, human power, cost, etc." (Tong, 1992);
>
> "The purpose of design is to derive from a set of specifications a description of an artifact sufficient for its realization." (Mostow, 1985);
>
> "Engineering design is the systematic, intelligent generation and evaluation of specifications for artifacts whose form and function achieve stated objectives and satisfy specified constraints." (Dym, 1991)

While some of the descriptions of engineering design provide a somewhat dry, programmatic view, design clearly has both programmatic and creative

components:

> " . . intuitive and irrational aspects of thought have just as important roles to play in design as logical and systematic procedures." (Cross, 1984)

The utilization of creativity and innate talent in design sometimes allows people to conceive unique solutions to a given problem, and in other cases, allows people to conceive solutions where there is no problem statement.

The word "design" is derived from the Latin *designare*, which means to mark out, to trace out, or to draw in outline. Designare, is a derivative form of the noun *signare*, meaning a mark or sign. Its original application related to the planning and delineating of patterns by marking or sketching outlines (Oxford Latin Dictionary and Webster's Unabridged Dictionary).

Engineering design is important because it is a critical factor in the product realization process (NRC, 1991). It follows, therefore, that the manner in which the design is carried out has a significant impact on the ultimate product. While much attention is focused on the product itself, a central issue in design is the *process*. The design process can be thought of as the sequence of events and steps that forms the mechanism by which people and systems transform the problem statement into a specific solution. The engineering design process involves starting with the design objectives and constraints, applying all the tools, techniques, rules, experience, judgment, and whatever else is available to the designer, and moving to clearly defined representation of the product. While variations exist in the definition of the design process, attempts have been made to create formal, universally applicable design processes. In general, these models are prescriptive, and do not describe design as it is practiced. It is useful, nonetheless, to review some of the work in identifying formalized

design processes, for the purposes of process characterization. This examination is not an endorsement of any formalized approach, rather a review of some of the schools of thought on the matter. Some have postulated a more chaotic model of design:

> "Design is not, as some textbooks would have us believe, a formal, sequential process that can be summarized in a block diagram. . . it is clear that any orderly pattern is quite unlike the usual chaotic growth of a design." (Fergusen, 1992)

A variety of more formalized theoretical models of design have been postulated. These include (a) Bruce Archer (1965), who developed a six-phase design model that includes "programming", which is his representation of the problem formulation, "data c .ection", "analysis", "synthesis", "development", and "communication"; (b) Pahl and Beitz (1977), who describe a "study", "program", "synthesis", "analysis", "evaluation", "decision", and "implementation" process; (c) Mesarivic, (1970), who defined hierarchical organization and solution methods; (d) Stephanou and Spiegel (1992), who describe a systems engineering approach, including systems engineering, systems integration, manufacturing, and finance/administration; (e) Forrester (1962), who developed system dynamics, which is a simulation technique which allows for the modeling of a process to include feedback , and (f) Reinschmidt (1995n), who provides an overview of engineering design in terms of synthesis, analysis, evaluation, and iteration.

These processes collectively represent many views of design in most production applications. Other paradigms exist, such as "Concurrent Engineering", which allows for a less sequential, more simultaneous process. In the concurrent engineering mode, multiple disciplines operate concurrently on the same phase of the design process, reducing iterations and overall

durations. Nevins and Whitney (Nevins, 1989) describe one method for implementing concurrent product and process design in which the design of the artifact and the production processes to make the artifact are considered one activity, and produce, as the output of the activity, both a manufacturing system design and plan *and* a product design.

Two more formal models of the design process are described here for the purposes of illustrating the general nature of most design models, and some differences between models.

2.1 Systematic Design

Jones (1963) defines a "systematic" design methodology, and described this process as the three major stages of *analysis*, *synthesis*, and *evaluation*. Hansen (1965) described a systematic design procedure comprising the following steps:

(i) identify preliminary considerations and basic principles;
(ii) search for individual and combinations of elementary solution elements;
(iii) review shortcomings of each proposed solution; and
(iv) rational evaluation of each solution, selection of solution with the fewest shortcomings.

(Note that the above method, and several others described below, are linear processes - orderly and uncluttered in the theoretical design world. Jones' method was "tried out with some success in design teaching projects and in prototype developments in ergonomics and industrial design. In these cases it has seldom been possible to carry the method to a logical conclusion for want of a new kind of design organization that seems to be necessary to permit a complete change to systematic work." [Jones, 1963] In the real world of

practical design, the design process is non-linear and highly iterative.)

Further, Rodenacker (1970) defined a systematic design approach in terms of the following eight steps:

(i) clarify the task (identify functional requirements);
(ii) establish the function structure (logical relationships);
(iii) select the physical process (physical relationships);
(iv) determine the embodiment (structural relationships);
(v) use calculations to check the logical, physical, and structural relationships;
(vi) eliminate errors (optimization);
(vii) finalize the design, and
(viii) review the final design.

What is lacking in the above basic definition is a feedback cycle (what does one do if the final review fails?) and also a distinction between the "eliminate errors" phase and optimization, which is an improvement function not a correction of errors.

Based on the above two systems definitions, a source for repeatable approaches to the design process has been developed by the German technical society of professional engineers (the VDI), in their technical guideline for engineering design (VDI-2221.) They define the basic steps of the process as (Dym, 1994):

(i) clarify and define the task - produce design specifications;
(ii) determine functions and their structures - produce a function structure document;
(iii) search for solution principles and their combinations - produce a

principal solution;

(iv) divide the principal solution into realizable modules - produce a module structure;

(v) develop layouts of the key modules - produce preliminary layouts;

(vi) complete the overall layout - produce a definitive layout, and

(vii) prepare production and operating instructions - produce product documents.

One idea is that the central element of design is *representation* (Dym, 1994), incorporating the "image or likeness" of the artifact as well as the process of the artifact being designed. However, many different representations of objects are possible, and the design engineering problem includes the definition of appropriate characterizations, performance criteria, and, implicitly, declarative (or descriptive) representations.

2.2 Axiomatic Design

Suh (1990) defines a method for design based on two design *axioms*. While this method is not commonly practiced, it offers a useful perspective in terms of characterization and analysis. The **axiomatic design method** is based on five central principles:

(i) Domains

A fundamental tenant of this methodology is the recognition that the design process is structured into multiple domains. These domains are the ***customer domain***, the ***functional domain***, and the ***physical domain***. The customer domain is the statement of need, as expressed by the customer. The customer expresses need in terms of *customer attributes*, or CA's, which may or may not be specific definitions of the requirements of the design.

The first task in the design process is to determine a set of *functional requirements*, based on the stated customer attributes. These functional requirements, or FR's, represent the designer's perspective of the required properties that the design must satisfy.

Once an FR has ben established, it is possible to envision what possible feature or parameter might be used to describe or satisfy that FR. The physical domain is that domain in which FR's determine the *design parameters*, or DP's. The DP's are selected by the designer to satisfy FR's. This selection process may, in fact, yield many DP's that could each satisfy a given FR, and these lead to different design alternatives which must be compared with each other, and evaluated against performance or success measures.

Once a DP has been identified it is possible to define the means by which that DP method would be achieved. It is in the process domain that the identification of the process necessary to create the DP is identified. The resultant is a set of *process variables*, or PV's. Many PV's may be identified for a given DP, and, again, the designer must select from a set of process alternatives based on the appropriate performance measures.

(ii) Mapping Between Domains

In order to structure the relationship between domains, a syntax for creating relations is defined in the axiomatic design methodology. In this syntax, domains are related by means of design equations and design matrices. From the functional to the physical domain, the following equation applies:

$$\{FR\} = [A]\{DP\}$$

Where, $\{FR\}$ is the vector of functional requirements

$[A]$ is the design matrix relating FR's to DP's, and

$\{DP\}$ is the vector of design parameters.

In the mapping of the physical domain to the process domain, a similar equation is defined:

$$\{DP\} = [B]\{PV\}$$

Where, $\{DP\}$ is the vector of design parameters.

$[B]$ is the matrix relating DP's to PV's, and

$\{PV\}$ is the vector of process variables.

The nature and form of the [A] and [B] matrices relate to qualitative assessments of the specific design alternative in question. The axiomatic design approach places a great deal of positive value on *uncoupled* designs. As such, in this value system, an ideal design is one in which the [A] and [B] matrices are diagonal, so that one DP influences only one corresponding FR and one PV influences only one DP. A triangular matrix represents a *decoupled* design, in which the design or process sequence yields independence of FR's and DP's. In a fully or almost fully populated [A] or [B] matrix, a *coupled* design results. In this scenario, one DP may affect many FR's, and one PV may affect many DP's.

(iii) Hierarchical Decomposition

Within each domain, the FR's DP's, or PV's can be sub-divided into hierarchies which represent the underlying components of

that FR, DP, or PV. This implies modularity and logic in the design morphology.

(iv) Inter-Domain Correlation (Zig-Zagging)

In order to create the hierarchy of a given domain, it is necessary to first define the corresponding DP or PV in the associated following domain. Once the DP or PV has been defined, the necessary requirements of the DP or PV will determine the appropriate decomposition in the given domain. This is necessary because decomposition in one domain (e.g. functional) is dependent on the composite elements of the corresponding DP's physical characteristics. Thus, the physical characteristics of the DP will define the descendant FR's. In this sense there is *inter*-domain correlation, and *intra*-domain decomposition.

(v) Two design axioms

The axiomatic design methodology is anchored by two fundamental axioms, which drive the above process:

Axiom #1: *The Independence Axiom*

Functional Requirements must be independent

Designs that can be characterized as either decoupled or uncoupled satisfy this axiom. This axiom is used to drive the mapping process from FR's to DP's and to PV's. The designer is encouraged to work in a *solution neutral environment*, in which historical or prejudicial associations from one domain to another should not be constraining. That is, when FR's are selected, they must be (by definition) independent.

Axiom #2: *The Information Axiom*

Minimize the information content of a design

Every design can be characterized by the confidence or probability that the specific design will meet the specified design criteria. Using the concept of information (as defined in information theory), a mathematical definition can be stated:

$$I = \log_2(1/p)$$

where p is the probability of success.

In minimizing I, p is maximized due to their inverse relationship. Note that p is defined as the ratio *common range/system range*, which refers to the probability distribution function of the given specific design in relation to the probability distribution function of the design range.

2.3 General Characteristics of Design Processes

Without defining a specific process, issues in the general design process can be characterized by a number of factors, including (not limited to):

(a) Organization,
(b) Control,
(c) Coordination,
(d) Communication,
(e) Feedback, and
(f) Knowledge access.

The organization of the design effort includes human organization, infrastructure, and workflow. Control in the design effort includes procedural control of the process itself, as well as control over the quality and nature of

the specific outputs. Coordination involves the (harmonious) interplay between activities, people, and systems, in order to effectively progress toward the global design goal. Communication of information (explicit and implicit) is a key factor in the success of the effort, with respect to the accuracy of information, the timing of the activities, and the confidence in the result. Most design activities include some feedback of information, whether or not it is explicitly recognized or acknowledged. This feedback can include changes due to assumptions that are later invalidated, errors, external disturbances (changes in requirements, regulations, constraints, etc.) and even the discovery of new information during the course of the design process. Knowledge access is the mechanism by which design personnel avail themselves of necessary design knowledge, either in the form of interaction with other people, documents, or electronic data.

Traditional design processes are subject to well-defined phases or stages of design, similar to those defined by Hansen and Rodenacker. These are:

(a) Concept Planning,
(b) Preliminary Design, and
(c) Detailed Design.

The customary problem-solving trend (temporally) in design is from the general to the specific. (Dunker, 1935) This style is the basis for the common design approach in a variety of disciplines. For example, the automotive design process consists of the following general steps: (Nevins, 1989)

(a) Concept Development
(b) Major Dimension Development
(c) Exterior and Interior Basic Definition
(d) Detailed Engineering

(e) Manufacturing/Assembly Planning

(f) Design Release

Of course, within each phase, there are several detailed steps, but most design processes follow this basic progression. Sage describes a three-dimensional framework for basic design progression, which is applicable to most design efforts in practice today. This framework allows us to view the above issues in the context of the overall process, and provides a multi-dimensional survey. The three dimensions of this framework are *logic*, *time*, and *knowledge*. These are illustrated in Figure 2-1. The logic dimension relates to steps and progressive sequences of these steps in the design task. The time dimension relates to the chronology of the design process, and the succession of life-cycle phases of the product realization process. The knowledge dimension relates to the individual disciplines involved in the overall design. The *activity plane* relates to the time and logic axes, and assumes a specific discipline.

2.3.1 Organization of Design

Several concepts relative to how design is or should be organized have been developed over the last thirty years. Simon (1973) describes the entire process of design as "an organizational process," in which problems that may seem ill-structured, are broken down into smaller sub-problems, until a well-structured sub-problem is defined at the lowest level of this problem decomposition. Each task or sub-problem, is dealt with by a specific group of people. Mesarovic, *et al* (1970) describe a theory of hierarchical, multilevel systems in which the organization consists of a set of "decision-making units" incorporating a number of behavioral concepts. Each unit has a place within the hierarchy, as a subordinate to some superior decision-making unit, and perhaps as a superior to other subordinate units. They define the total organizational system as

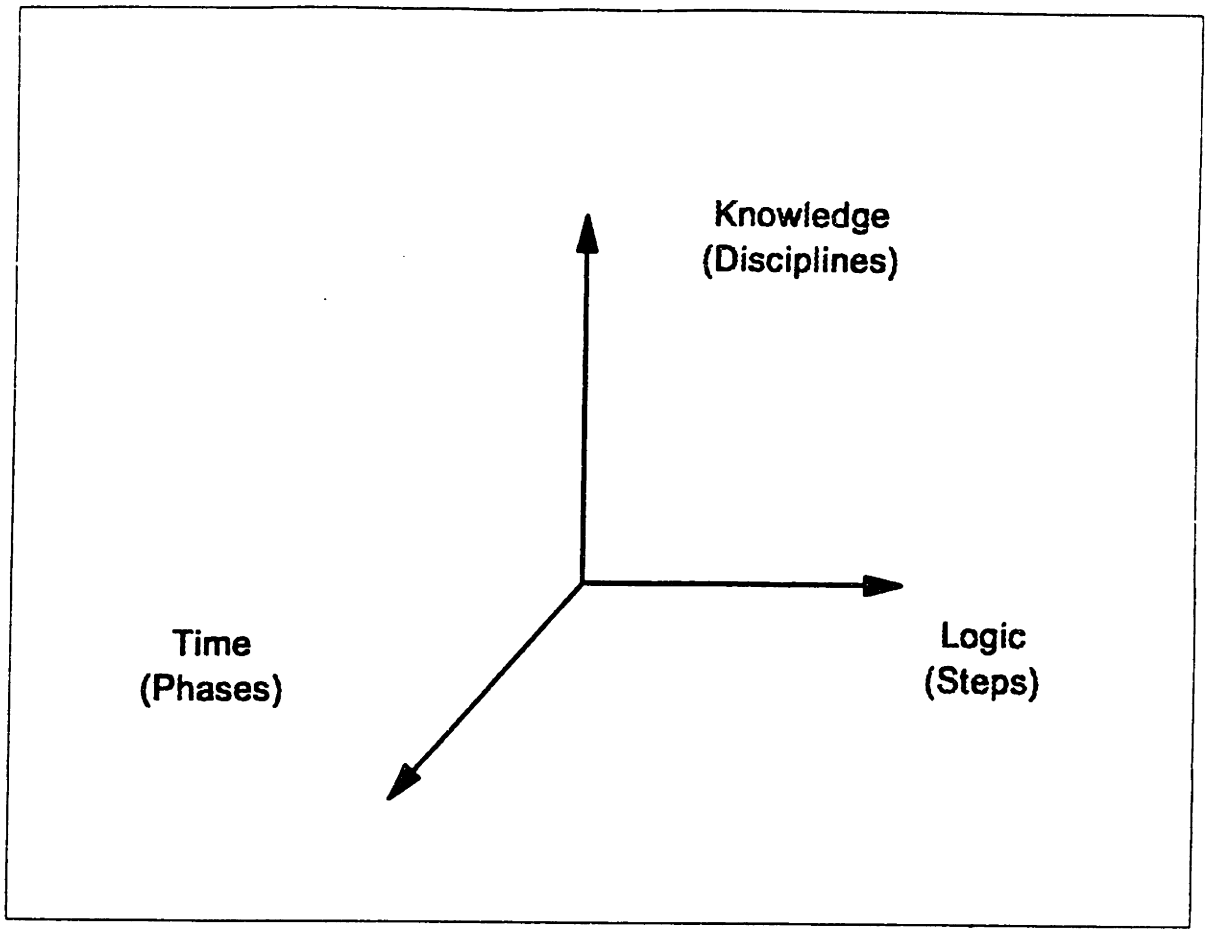


Figure 2-1: Morphological Design Dimensions (After Sage, 1976)

having *n* levels, and each unit within each level has an input function and an output function. Further, they define a variety of types of hierarchies, including *Strata*, in which levels of abstraction dictate a stratum's relevant variables (meaningful only to that level), *Layers*, in which levels are defined in terms of decision complexity, both in terms of timing and uncertainty, and *Multiechelon Systems* (See Figure 2-2), in which the system consists of families of hierarchies, self-contained and behaviorally analogous to individual units of the previous two hierarchy types. Their organizational models are defined by a rich mathematics.

The significance of the organizational structure can easily be underestimated. In fact, the organization can be so institutionalized as to "hide" certain problems and therefore limit or eliminate any possibility of effecting any

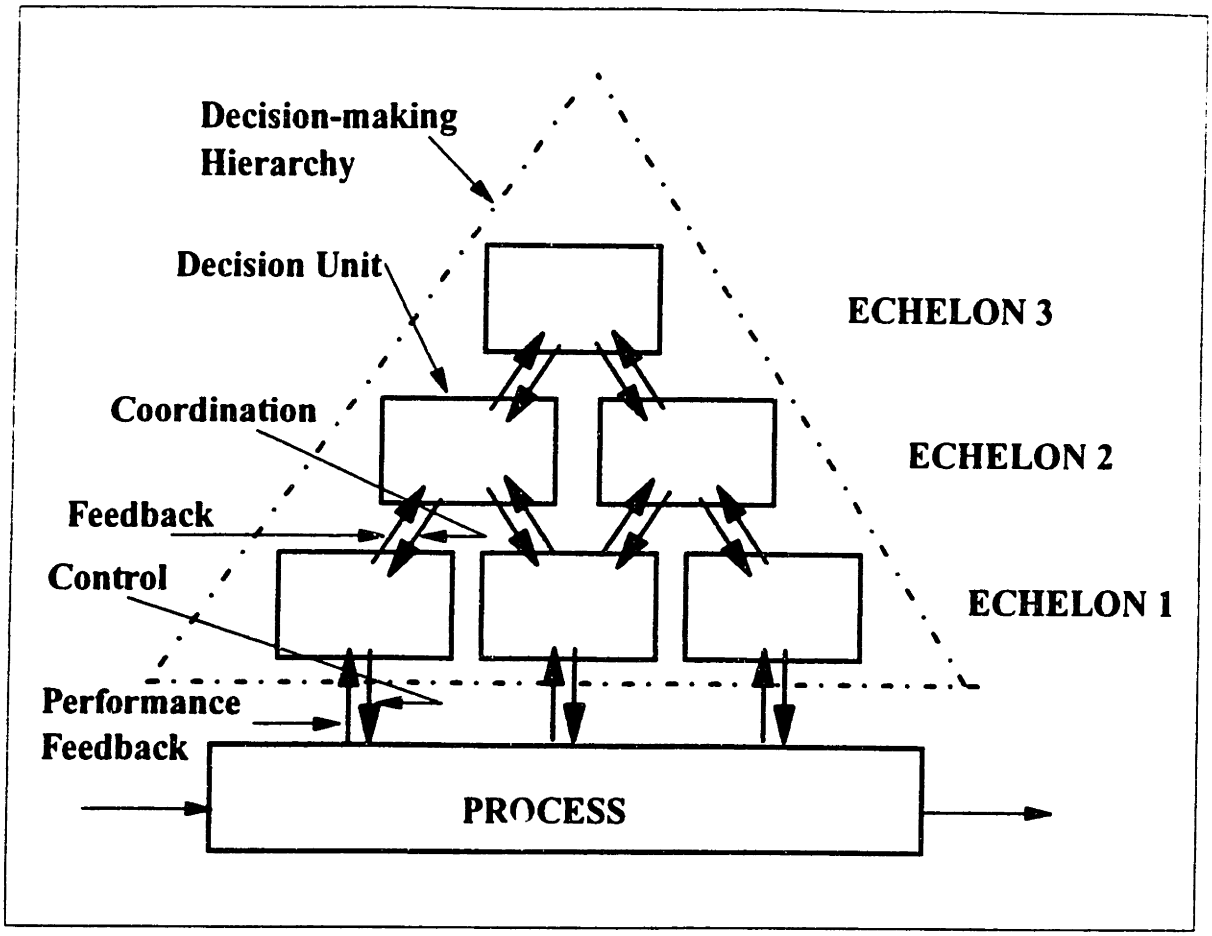


Figure 2-2: Multilevel Organizational Hierarchy (After Mesarovic, et. al.)

improvements. For example, in a study published in 1963, it was reported that the problem of job-shop scheduling was non-existent to certain factory schedulers, because "the organization which surrounds the schedulers reacts to protect them from strongly interdependent sequencing problems." (Pounds, 1963) The organization was so constructed that orders could not be placed with delivery times too short, so that a scheduling problem would arise. This framework serves to conceal the opportunity for improvement, however, because it is possible to apply job-shop scheduling techniques and technologies to be able to react to dynamic orders and production conditions, and perhaps offer customers better shipment delivery dates than this particular factory was in the habit of quoting. Similarly, in a design organization, certain improvements in the process may be impossible in a given organizational framework, because the definition of the organization precludes alteration of

the working mechanisms of the design process.

Malone (Malone, 1986) defines four organizational models based on Product Hierarchy, Functional Hierarchy, Centralized Market, and Decentralized Market principles. In the Product hierarchy, the organization is separated according to product boundaries, usually with product managers assuming responsibility for the overall product. Functional hierarchies are set up by "similar tasks" utilizing a shared resource theory to attempt to optimize the organization's performance, and these units are managed by functional managers who specialize in specific types of activities. The Decentralized Market organization is one in which the organization is free to choose any source to meet its needs, and may use an extensive network of external companies, organizations, or contractors to function. A Centralized Market organization use a centralized "broker" to coordinate the use of external resources (such as a central purchasing staff to work with vendors of components) regardless of whether the broker is part of the organization. In Malone's research, he identifies value metrics for each organization type in terms of costs of production, coordination, and vulnerability. This work determined that a Decentralized Market organization has the lowest relative composite cost, but the rates of change of the costs increases at the most rapid pace for this model when the size of the organization increases. Conversely, when the size of the organization increases, the lowest rate of change in cost results from the functional hierarchy. This may suggest that while an organization is growing, functional hierarchies are best while stable organizations operate best in a decentralized market structure.

While most design processes are organized in a hierarchical decomposition of some type, decentralization is increasing in its frequency of use. Once such model is described in terms of "concurrent engineering", in which design follows a parallel path, relative to different tasks or activities within the design process. In terms of the earlier model developed by Sage, this approach

requires the activities are performed concurrently, in terms of knowledge (functional) units as well as logic. As such, a general progression from abstract to specific is exhibited, but the sequence of activities is not serially defined in terms of specific input/output requirements for each activity. Tasks may be started with uncertain information, but basic concepts and ideas are established earlier than in more traditional, sequential processes. Nevins (1989) describes two classes of limitations, however, these being engineering ("some decisions simply cannot be made until others have been") and institutional. Institutional limitations include coordination, dealing with "ingrained habits and training" of people, and lack of experience in working as teams. However, they stress that it is " ... especially important that people whose decisions come *later* in the process are involved in the decisions that come *earlier*." In practice, concurrent engineering has been credited with many improvements in product realization, such as the reduction in cost when changes occur and improvements in design and manufacturing processes that result from inputs from plant floor personnel. (Woodruff, 1993)

2.3.2 Control In Design Processes

There are a variety of theories relating to how design processes are controlled. These theories relate directly to organizational structures. For example, in the Hierarchical Controller Theory, a 'super'-controller exists at the highest hierarchical level, optimizing over a certain function(s), and specifying constraints to lower design activities. No system control exists at the lower levels. The subsystems are internally controlled. In another theory, the Distributed Control Theory, each activity exerts a certain amount of control based on its own autonomous behavior knowledge. A global coordinator serves as a monitor, but not an autocratic controller. Feedback Control theory, while applied mainly to continuous and discrete manufacturing process control, also

provides some insight into how design processes can be monitored and controlled using quantitative measures of accuracy, and mathematical representations of error and adaptation.

Information is continually being generated and communicated during the course of a design process. Consider a specific design task, with inputs and outputs. The design activity can be represented as A_i, with inputs I_i and outputs O_i. This local activity information model is graphically represented in Figure 2-3.

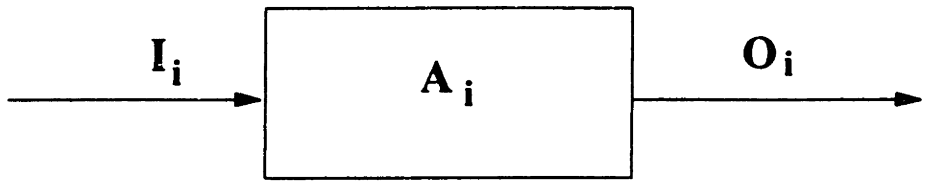


Figure 2-3: Information Associated With Design Activities

We note that the size of the total design information base increases with each activity because design activities always generate information. As such, for each activity, a subset of the available information is used as input, and the output of the activity is added to the design database. Note that this process is very much dependent on the timing and sequencing of design activities. At any point in time, the design database consists of a set of data, D_t. For each activity or task, some subset of the information is used as input, this being $I_{i,t}$ since the input vector is associated with the data at a particular point in time. Since each activity has a particular time constant (the duration of the activity), we can thus modify the above representation to include the output vector to include this time constant, as in Figure 2-4.

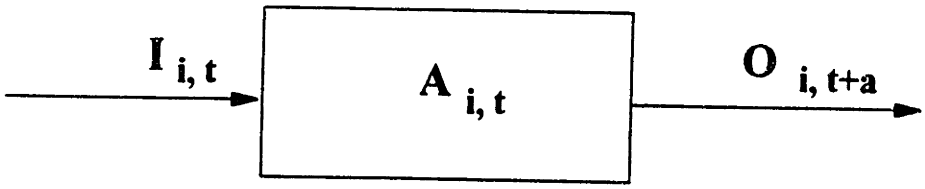


Figure 2-4: Information Associated with Design Activities with Durations

where: a = duration of activity A,
i = iteration number of activity A,
t = time at which activity A is performed.

We note that in reality, the duration of design activities often depends on the values in the input vector $I_{i,t}$ (in terms of content and certainty), and may also depend on other factors. As such, this becomes a non-linear function a(t,i). Clearly, the issues involved in controlling the activity to meet a specific output objective (as well as a duration objective) are complex in a real-world design scenario.

Design can indeed be formulated as a controls problem. In this view, the purpose of the design process is to produce an artifact (the product) and the process of producing this artifact is subject to analysis as a continuous process, with the people, computer systems, methods, procedures, documents, and information comprising the *system*. An important element of this design process system is that it incorporates *feedback*. While some modeling methods address the issue of feedback (such as system dynamics) no design infrastructures are specifically built to support this information feedback. Hence, they do not represent the real world.

In classical control theory, an elementary feedback control system would consist of an input (or reference) **R**(t), an output (or control) **C**(t), an error **E**(t), and a feedback **F**(t). These signals, or information flows are processed in two ways. The input and error signals are processed by a forward transfer function **KG**(t), and the feedback transfer function is **H**(t). These are represented diagrammatically in Figure 2-5.

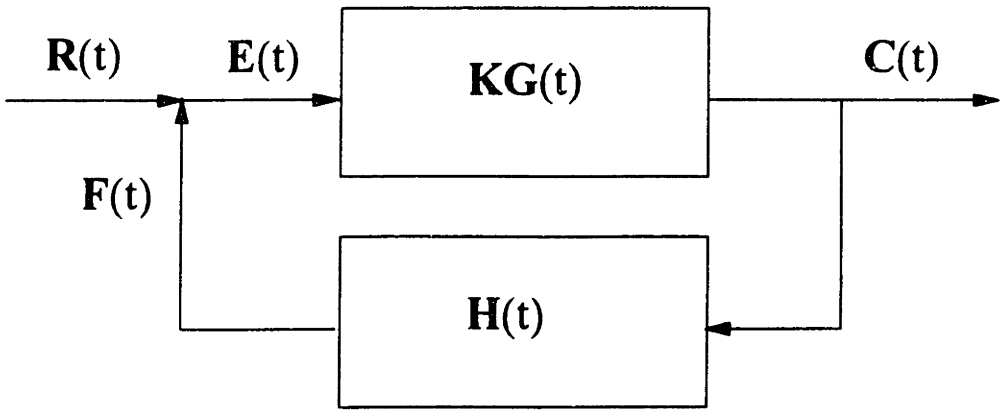


Figure 2-5: Control with Feedback Transfer Function

In examining the design process in the context of the above framework, we see that the design process as a whole can be viewed in this way. Here, **R**(t) are the functional requirements, the constraints, and all relevant information to the design process. Also, **C**(t) represents the design itself. Notice that the use of a temporal notation (t) is appropriate, because at any given point in time, the design will take on different forms, and may be significantly different at one point compared to the same design some time later.

Additionally, tasks within the overall process can be represented in the above manner. As such, the above representation may use the notational form described in Figure 2-6.

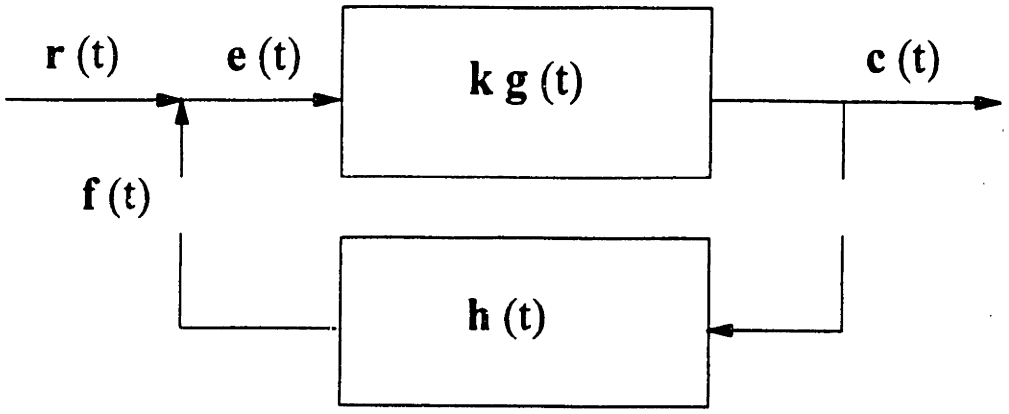


Figure 2-6: Localized Controls Model Notation

In this localized model, the design task has an input function r(t) that is some subset of the combination of R(t) and C(t). The reason that r is derived from the combination of overall design inputs and outputs is that some information is generated during the design process. This information is used as input (perhaps in conjunction with some basic design inputs) to tasks that follow.

The problem of control in the context of the design process is one of measuring c(t) and applying some feedback transfer function h(t) so that the design activity (or a portion thereof) can be repeated with some new or modified input in order to achieve a "better" result. In the context of engineering design, the feedback transfer function is often an analysis mechanism, such as stress, deformations, mechanical performance, aerodynamic performance, etc. Based on the results of the analysis, and error function is generated (usually the difference between desired or allowable outputs and as-designed outputs.)

In some systems, e(t) --> 0 asymptotically and monotonically. In others, e(t) --> 0 in an oscillatory manner, as the controller searches for an acceptable solution. Oscillations around e are not captured by most conventional design models of the design process, although these oscillations are frequently observed in practice. Note that the "controller" in the context of design is

typically a human function, not an automatic function. Optimization systems and design automation systems do play this role in some circumstances, but the majority of design control is exerted by people.

The combination of errors, inaccuracies, and sub-optimalities within the design process leads to overall design problems equivalent to the tolerance stackup problem in a manufacturing context. Tolerance stackup is the process whereby a part or assembly becomes non-manufacturable or out of acceptable limits of accuracy because of the combination of smaller elements of inaccuracy that build upon each other in a linear fashion.

This control perspective is based on the view of design as a continuous process. While discrete tasks are being performed, when observed from the perspective of an informational model, the design process can be thought of as analogous to a production facility. In this analogy, the machines and the product are constantly being measured relative to temperatures, pressures, flow rates, product quality, etc. Distributed control systems (DCS) monitor these measurements, and provide both a local control function (usually in the form of programmable logical controllers,or PLC's) for valves, machinery, and equipment, as well as a supervisory control capability. While local control may be performing its function, keeping measured values within acceptable limits, overall process behavior may not be acceptable. This supervisory control function, inherent in distributed control systems, provides the capability for the system to retain multiple levels of abstraction, and to utilize a variety of control strategies, depending on the appropriate level of abstraction. This model applies equally well to the design problem, where measurements are made relative to product design parameters and design process parameters (e.g.level of effort, cost, duration, and progress.) Control paradigms originally designed for process control thus may apply to design in an integrative fashion.

Relative to the above control system architectures, a high-level comparison may be made between a classical controls paradigm and the design problem, as illustrated in Table 2-1.

	Classical Controls	Design
r(t)	input/reference	functional requirements
e(t)	error	error
kg(t)	forward transfer function	design process or tasks
c(t)	control	design
h(t)	feedback transfer function	analysis activity
f(t)	feedback	feedback (results of analysis)

Table 2-1: Comparison of Classical Controls and Design Problems

Without controlling the output of each design task, sub-optimal and even unacceptable designs can result. The control of individual task outputs is a complex task, but this complexity is made dramatically worse when the task is viewed in the context of the overall design problem, and interactions between tasks are taken into account. This interaction requires coordination.

2.3.3 Coordination of Design Activities

The problem of coordination relates to the management of design activities in such a manner so as to influence the design activities and systems so that they act harmoniously, toward the common design purpose. This coordination is generally directly related to the control architecture, and the organizational structure of the design process. Coordination also involves the management and synchronization of information flow.

Coordination is the direction of activities such that the information required for each activity is available when it is needed, and that the sequence of each activity relates appropriately to the state of the design and the available information. For example, if a tool design activity is initiated prior to the availability of necessary part design information, then the tool would have to be designed based on some assumed values for the part design. A high degree of iteration within the tool design activity may result. If, conversely, the tool design activity was initiated after a minimal amount of necessary part information was released (comprising a partially complete part design), then the appropriate tool concept, layout, size, material, etc. can be selected and designed in detail without having to make major revisions based on actual part information that was substantially different from assumed data. Since the tool design task could be on the critical path relative to the product realization process, a delay in the initiation of this activity could result in costly product release delays.

Therefore, the coordination of the part design activity and the tool design activity is critical, both in terms of accuracy of the tool design (and hence accuracy of the manufactured part) and in terms of the overall cost and schedule. It is the delicate balance between operating with too few data and waiting too long for final, released data, that causes coordination problems in

the design process.

One method of coordinating interdependent tasks is by specifying behavior in advance of the task execution, in the form of *rules and programs*. (March and Simon, 1958) This theory states that the rules and programs will create a pre-defined coordination mechanism that reduces or eliminates the need for communication between tasks. It is not clear, however, that in practice a comprehensive set of rules and programs can be determined *a-priori*. This mechanism is a common framework in many design organizations, in the form of written manuals and procedures, despite its limitations.

Eppinger provides a matrix method for coordination via information flow definition and analysis, which is useful in understanding the basic information flow in a given design process. (Eppinger, 1989) Known as the Design Structure Matrix (DSM), this technique develops a two-dimensional square matrix with each design task numbered. Each task is then related to the other tasks in the process, and results in a row-by-row definition of all interacting activities. The DSM delineates interactions as *primary dependencies* (information from one task is required prior to initiating the other task); *secondary dependencies* (information from one task is required prior to the completion of another task); *tertiary dependencies* (information from one task is helpful but not essential for the other task); and *generational feedback* (information from one task is helpful in optimizing a further iteration of another task.) [Lauzun, 1992] Eppinger and McCord (1993) suggest a derivative of the DSM to be a natural generation of design teams, such that the teams are optimally designed for reductions in interactions between teams, and improved integration.

Matrix methods for identifying interactions have also been incorporated into other techniques, such as Quality Function Deployment, or QFD. (Hauser,

1988) Sage, in a similar fashion, uses a matrix method to identify which activities interact in a design problem. (Sage, 1977)

The hierarchical organizational structure relies on its decomposition of activities *and* the encapsulation of decision-making to coordinate activities. As each sub-system is controlled (in terms of inputs and outputs) by its superior system or sub-system, the superior systems coordinate the behavior of the sub-systems. One coordination method "imposes on each sub-problem additional constraints so that the optimal solution of one sub-problem is independent of the solutions of the other sub-problems. The additional constraints are selected so that the union of the solutions of the sub-problems will be optimal for the large scale problem." (Himmelblau,1973) Another coordination method, known as the "balance coordination method" attempts to balance level-wide interactions of activities in order to achieve optimal performance at each level, through iteration. (Haimes, 1973)

Coordination practices are influenced by a variety of characteristics of the problem, the organization and the people. Some research in this field indicates that there are three primary coordination layers (Keigler, 1992) , these being:

(i) the media layer (e.g. e-mail networks, offices, telephone systems, paper memoranda, etc.);
(ii) the process layer (e.g. Quality Function Deployment, software analyses, decision-support tools, checklists, etc.), and
(iii) the setting and attitude toward coordination (e.g. planned meetings, e-mail conversations, spontaneous meetings, discussions around a CAD workstation.)

In general, the problem of coordination is related to the complexity of the

design effort. As the various tasks within a design process become more interdependent, the coordination requirements multiply. (Galbraith, 1973) Coordination is one of the central problems of the organization, and organizational design must allow for activities to be coordinated across associated tasks. Galbraith later defined coordination methods as "rules and programs" when tasks are routine. If the rule set does not cover a particular circumstance (high degrees of uncertainty in the data) then the hierarchical supervisor-subordinate decision-making roles are invoked. (Galbraith, 1974) One of the issues in determining the effectiveness of the coordination capability within an organization is the "information processing load and capacity" (Christiansen, 1993.) The ability of the organization to effectively coordinate activities depends on resources, organization structure, the design data themselves, and the ability for the people and systems in the organization to communicate effectively.

2.3.4 Communication in Design

The mechanism for communication information transfer is a critical element of the design process, and may consist of ad hoc methods or a formalized process. Typically, the transfer of information from one unit process to another is directly related to the design organization and control structure. The effectiveness of the design effort is dependant on an accurate and efficient means for communicating information.

Both within a design organization and beyond the perimeter of the technical organization, communication is a key to successful product realization. Because there are physical and logical disparities in most large manufacturing or other product realization organizations (Hauser, 1988) it is not obvious that separate teams or groups will communicate effectively. Hauser and Clausing point out

that even if management could orchestrate meetings between different groups, "what should these people talk about?"

One form of communication is for management review purposes. "Frequent contact is necessary between technology managers and the personnel leading the work effort so that the managers are kept informed of how the work is progressing." (Stephanou, 1992) Obviously, communication needs extend well beyond the management review requirement to all aspects of the collaborative design process. Many factors may influence this communication, including the organizational structure, communications infrastructure (electronic or manual) and the communications culture of the design organization. In 1976, it was reported that physical distance between collaborators had a dramatic effect on the frequency of communication. (Allen, 1976) As the physical distance between people increased, frequency of communication dramatically declined. The precipitous distance was found to be between ten and twenty feet - beyond that, communication more than once a week became highly unlikely. This research was conducted before the proliferation of computer networks. Theoretically, the availability of these networks (and specifically electronic mail) increases the frequency of communication. (Christiansen, 1992)

There are, however, multiple dimensions to communication beyond simply increasing the frequency of communication. Shannon (Shannon, 1962) describes an "engineering model" of the communication process as having functional elements consisting of a Transmitter, Encoder, Decoder, Receiver, and a Noise Source. The communication itself starts out as a Message, becomes a Signal that is altered by Noise, and is decoded into a different Message. As such, we may derive the influences (or dimensions) of measuring the **effectiveness** of a communication process:

(i) Frequency

The frequency of communication is directly and proportionally related to the success of the collaborative effort. The more people and systems communicate, the better they will understand each other, and the lower the probability of incorrect assumptions pervading the design process.

(ii) Clarity of the message (noise/signal ratio)

As part of effective communication, the message being sent must contain relevant qualification or description, in terms of directness and clarity. The more "noise" the poorer the signal-to-noise ratio, and hence the less likely that the original message will be received.

(iii) Intention-Perception balance

Having a common protocol or language is essential for effective communication. Simply *connecting* is insufficient. "This goes beyond the need for basic connectivity. Using the human communication example, connectivity is like being able to pick up the phone and directly dial anyone in the world. That doesn't necessarily mean that you can speak to each other, because you may not know the same language. But you can make the connection." (Schay, 1991) In order to receive the message as it was intended, there must be a balance between intentions and perceptions (send/receive) achieved through common language and agreed upon protocols.

2.3.5 Feedback in Design

All design processes incorporate feedback, in some form. Feedback and iteration are inherent in design practice in part because: (i) design is a controls problem, and the approach that best manages such problems is the method that operates via the feedback in the system, i.e. feedback control theory; (ii) if the design problem can be characterized by simultaneous equations, modeling the design problem as a set of simultaneous equations must be iterative, since the equations are typically non-linear and some of the equations may be unknown, and the solution to simultaneous equations is an iterative process, and (iii) design optimization follows an iterative hill-climbing model, since global optima are difficult to know *a-priori*. (Reinschmidt, 1995n)

It is important to be able to model feedback, relative to the impact of some design results or decisions, and the impact of changes on the design itself. The feedback could include transfer of information to earlier processes as a result of completion of subsequent processes, or transfer of information to the same process because of a result being unacceptable for some reason.

Feedback in design is derived from two primary sources: analysis of local task outputs, and influences from design tasks other than the specific task in question. One definition of engineering postulates a tri-partite engineering philosophy, consisting of a theoretical set of principles, an operational methodology, and

> ". . . a critical feedback apparatus which measures the advantages, detects shortcomings, and illuminates the directions of improvement." (Asimow, 1962)

A general graphical definition of the context of feedback in design is illustrated

in Figure 2-7:

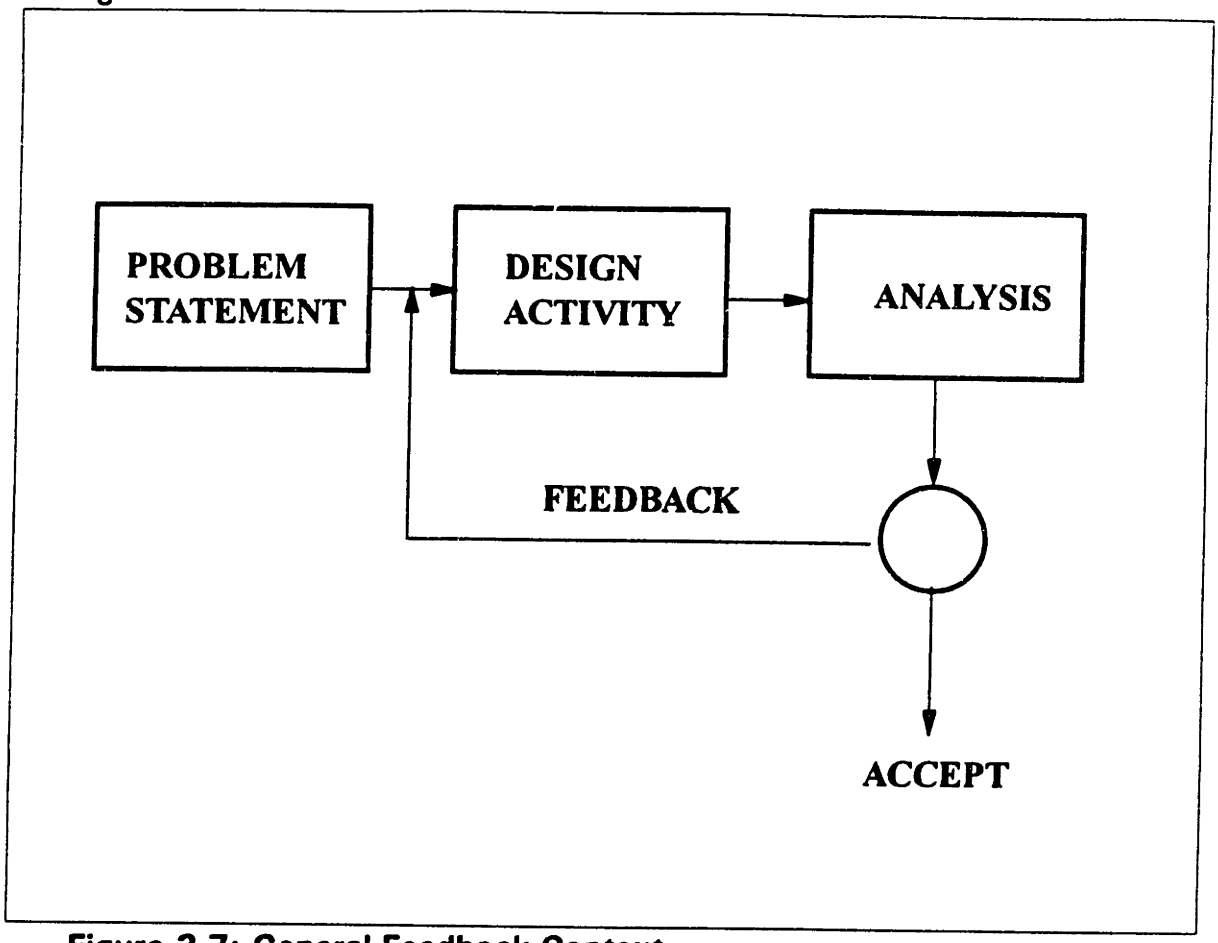


Figure 2-7: General Feedback Context

A more general controls formulation could be stated as follows:

(i) $C(t) = KG(t)R(t)/[1 + KG(t)H(t)]$ Classical controls formulation (Brogan, 1982)

(ii) $C(t) = KG(t)R'(t)/[1 + KG(t)H(t)]$ Modified controls formulation

Note that in *(ii) above, a new* $R'(t)$ *function is used. This function is described* as being a modified version of the requirements vector $R(t)$, to account for

feedback from other design tasks. This can be stated as:

(iii) $\mathbf{R'}(t) = \mathbf{R}(t).\mathbf{P}(t)$,

where $\mathbf{P}(t)$ is the perturbation (composite feedback) vector from all associated design tasks at time (t).

Note that it is also possible to determine the *sensitivity* of a system (or design task) to feedback, based on the following formulation:

(iv) $S = 1/[1 + \mathbf{KG}(t)\mathbf{H}(t)]$

where S is the sensitivity to the feedback function.

This sensitivity allows for the studying of the impact of variations in values for design parameters on the outcome of the design process. As such, this approach is important in that it allows for a prioritization of resources towards those issues which have the greatest effect on the design solution.

Beyond mathematical formulations, the importance of feedback in a design process is that design is often a highly iterative process, whether the iteration be on a local scale (inter-activity) or on a more integrated basis (intra-activity.) These iterations are embodiments of classical feedback loops, where initial results (designs) are modified based on analyses (feedback transfer functions) and information that may alter the initial functional requirements (input functions.) Note that this feedback process occurs in real-world designs both on a formal and an informal basis. When formal design reviews take place, or specific analyses performed, the results are then used to modify the proposed solution. When informal communications occur, information from these communications is used to improve or change the initial design.

Feedback can relate to local optima (i.e. the proposed solution does not meet the design requirements, or may not optimally meet design requirements) or to meet more global optima (i.e. the proposed design does not meet some system-wide optimum.)

2.3.6 Knowledge Access

In order to perform design tasks, people often need to access previous design information, obtain advice from specialists, and access information about the evolving design from other design personnel. The mechanisms for accomplishing these activities vary from oral discussions, to manual document retrieval, to electronic database access. It is important to recognize that this type of activity is an integral part of the design process, and may represent critical performance elements depending on the availability of personnel, the quality and accessibility of electronic data, and the availability and quality of paper documents.

Some studies have shown that a formal search for design knowledge in the Architectural/ Engineering/ Construction profession is often thought of as a waste of time by designers (Favela, 1993) Much of this time waste was perceived to be because the mechanisms for accessing this knowledge were not streamlined, and did not allow for a natural method for knowledge access. The information about previous designs was not perceived to be well organized, and was cited as incomplete. While this incompleteness and disorganization may be a significant issue in the AEC field, it is perhaps less true in other engineering design disciplines. For example, in the aerospace industry, design documentation is required to be of specific formats and in accessible locations. This availability and accessibility may produce an ability to access previous designs, but it is not clear that that translates into the

accessibility of *design knowledge.* Since design rationale and intent is not explicitly available on engineering drawings or even in detailed specifications, the issue of knowledge access is not directly addressed by documentation. The lack of such knowledge in the design documentation may result from the perceived vesting of the ownership of the design knowledge in design engineering organization (or individual) and, hence, perhaps, a dis-incentive to provide such documentation may exist.

As a general postulation of design itself, knowledge access can be the essence of the process. One design model uses as its "guiding postulate" the statement that design is the "acquisition, manipulation, and generation of information." (Becker, 1973)

Organization knowledge and experience are seldom documented effectively, if at all. "Every company recognizes its experienced people as its most valuable asset." (Lindgren, 1993) The ability to access more than data can be a critical issue in effective design. Traditional methods of accessing knowledge are confined to human-human interaction. The technologies of knowledge-based systems provide automated or computerized techniques to capture this knowledge and make it available in an on-line fashion to designers across the organization. (Reinschmidt, 1992) For a more detailed discussion of this subject, refer to the following Chapter 3, "Knowledge-Based Design."

2.4 Discussion

The issues involved in engineering design are complex. With respect to technical methodology to organization and coordination concerns, design processes are multi-disciplinary networks of activities, information, systems, and people. Many design problems and processes are so complex as to have

been labeled "wicked problems" (Cross, 1984) in which design problems "can never be completely described and therefore never susceptible to a complete analysis." Whether completeness is required can be argued, but the point that design problems and processes require the influence of qualitative methods (beyond closed form solutions) is clear.

2.5 Design Processes

There are a variety of methods that characterize how design is conducted in different design organizations. For example, sequential design processes are based on the concept of a sequencing of activities based on functional decompositions, and specific "hand-offs" of information between functional groups and, correspondingly, design phases. This approach is common because it follows a traditional view of the evolution of a design workflow: concept planning, preliminary design, detailed design, and manufacturing (or construction) engineering and planning. Of course, while this approach offers a sequence of distinct steps, iteration within each distinct step, and often between steps, is common.

Concurrent design is a commonly referenced alternate process, in which the design sequence (i.e. temporal progression of the design) does not follow functionally decomposed units. In this approach, functionally disparate groups work simultaneously using the same set of data to design different parts of the product at the same time. It is also used to perform simultaneous product design and manufacturing planning, in order to reduce the overall duration of the design process. In the context of complex product relation process, this design process often transcends not only functional boundaries, but also corporate entities. In the case of automotive design, for example, component suppliers to the prime manufacturer may co-locate designers with the

manufacturer, and component design and selection would occur while the requirements for the components are being developed. (Woodruff, 1993)

In this work, four basic process models of design are considered, some of which are based on theoretical research on the effect of organizational design structures on quality and lead time. (Gebala, 1991) The four basic process models are (i) sequential, (ii) parallel, (iii) coordinated, and (iv) concurrent.

(i) Sequential Design

In this process, design steps are organized in a linear, sequential fashion, so that at each stage of the design, a specific milestone or "hard-point" is reached prior to the initiation of the following step. As one activity completes, the results (output) of the task are provided as input to the following task. Progression of the design follows a sequential, linear process, as in Figure 2-8.

Note that each design unit (or step) contains a feedback loop, representing internal feedbacks (analogous to the process control feedback loop described earlier.) As such, the equation for production output is as follows:

$$O(t) = P(t) - E(t),$$

where O(t) is the output vector at time t, P(t) is the production vector at time t, and E(t) is the error (or feedback) vector at time t.

In addition to the internal feedback, the individual steps are subject to feedback from downstream steps' feedback, such that the equation for each step's

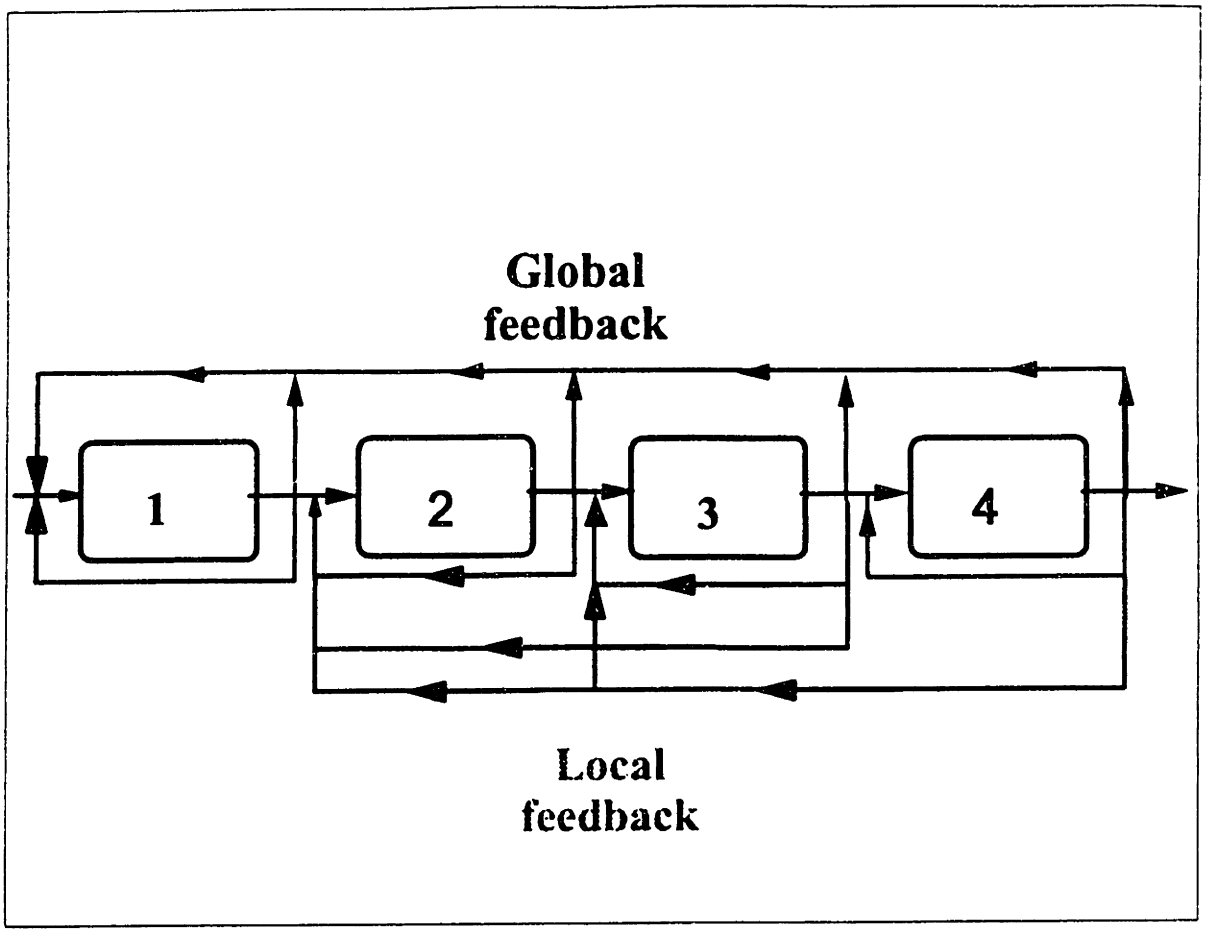


Figure 2-8: Sequential Design Process

feedback is as follows:

$$F_i = \sum F_j^p$$

where, F^p is the feedback returned to prior steps for downstream sequences.

(ii) Parallel design

In this model, illustrated in Figure 2-9, subsequent tasks begin before the

conclusion of prior tasks, and follow an execution sequence essentially equivalent to that in the sequential process. Design tasks begin, therefore, prior to the completion of all prerequisite tasks, and, potentially, prior to the availability of all required information. Here, feedback is provided internal to the design task, and, collectively, back to the initial steps.

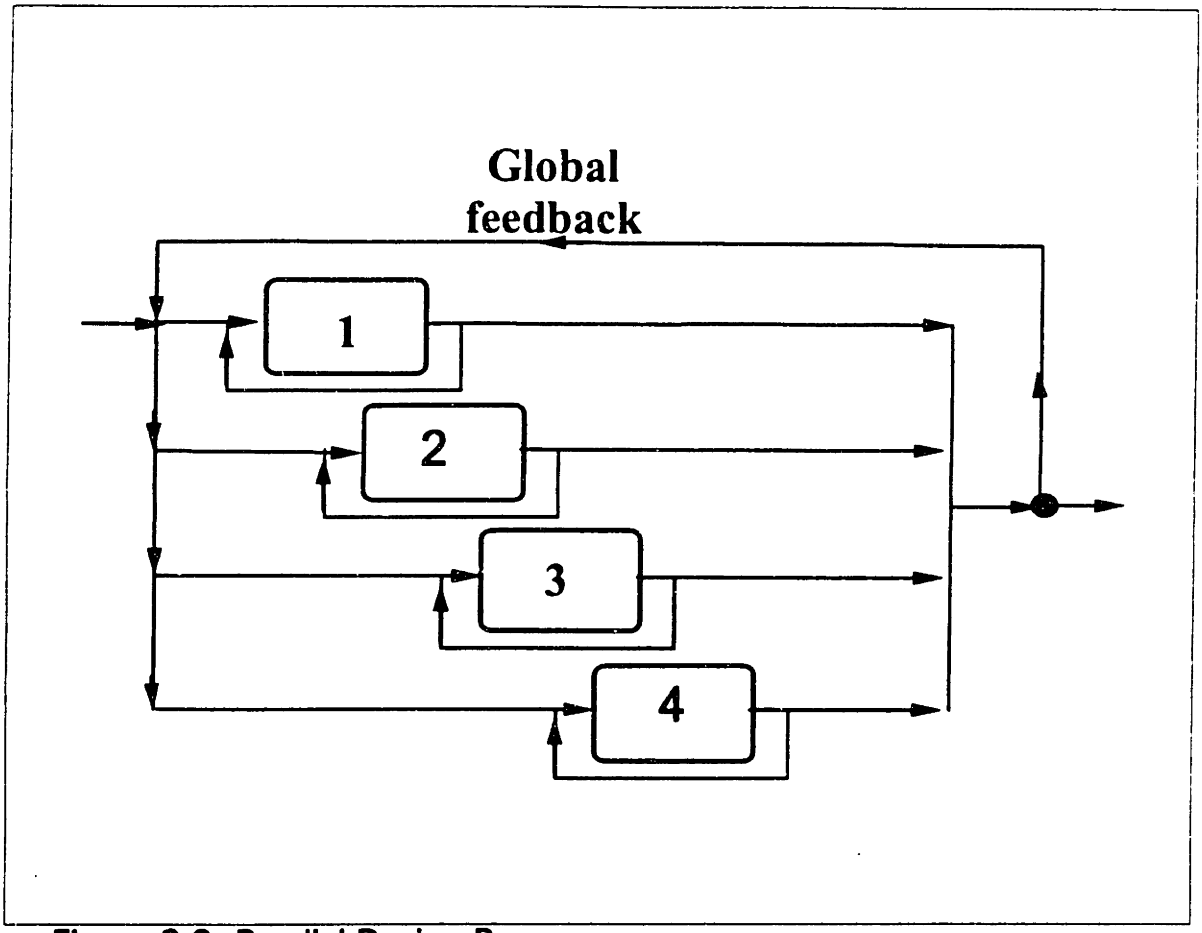


Figure 2-9: Parallel Design Process

In this model, therefore, it follows that the rate of internal and external feedback will be higher for the initial phase of all design tasks other than the first task, since the task is initiated with incomplete or uncertain information. As time goes on, the amount of information increases, and the uncertainty associated with input data decreases. Hence a feedback decay function results.

The feedback decay function would have the following form:

$$f(t) = \alpha\,(e^{\Phi \cdot t})$$

where the constants represent limits and rates of the decay function. That is:

α = asymptotic feedback constant

Φ = feedback decay rate

(iii) Coordinated design

In this model, a central coordinator performs the function of evaluating the output of each design task, and relays information to the appropriate following task. As can be seen in Figure 10, there is no linear sequence of information flow, and the process is highly dependent on the effectiveness and efficiency of the coordinator. In this model, each design task is not directly connected to any other design task, and they all perform based on their individual performance criteria and the information and instructions supplied by the coordinator. As such, the coordinator plays a key role in the effectiveness and success of these types of design processes. The feedback loops in this model are not distinct from the direct information transfer paths, in that the coordinator is responsible for managing the interface between tasks regardless of whether the information flow is direct or in a feedback mode.

(iv) Concurrent design

In concurrent design, each design task has direct knowledge of the other tasks, and works simultaneously on the problem together with the other design activities. As such, there is an overlap between activities, and no linearity in

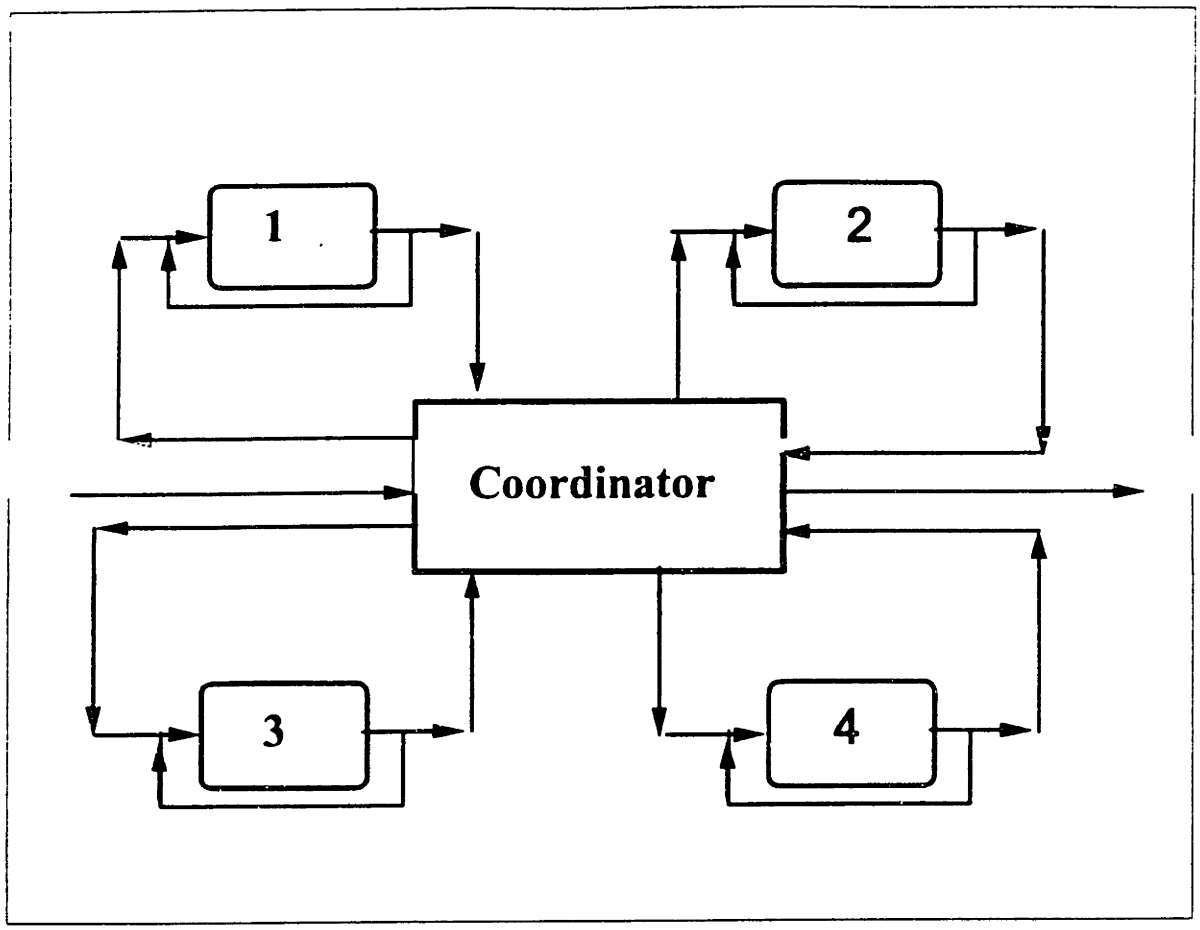


Figure 2-10: Coordinated Design Process

terms of design sequence. This is illustrated in Figure 2-11.

The ability to effectively accomplish a concurrent design process depends on the ability of the design activities to access and utilize knowledge of one another's disciplines, so that work that transpires relative to a particular task can be relevant to both the temporal and contextual state of the design.

While this process appears intuitively to provide optimal results, it is difficult to accomplish without the ability to effectively work in an environment of shared knowledge. As such, a unique infrastructure is required in order to facilitate this type of process. This infrastructure implies both a basic computing communications capability, and a design engineering infrastructure that allows for concurrent work from different disciplines. The process requirement has

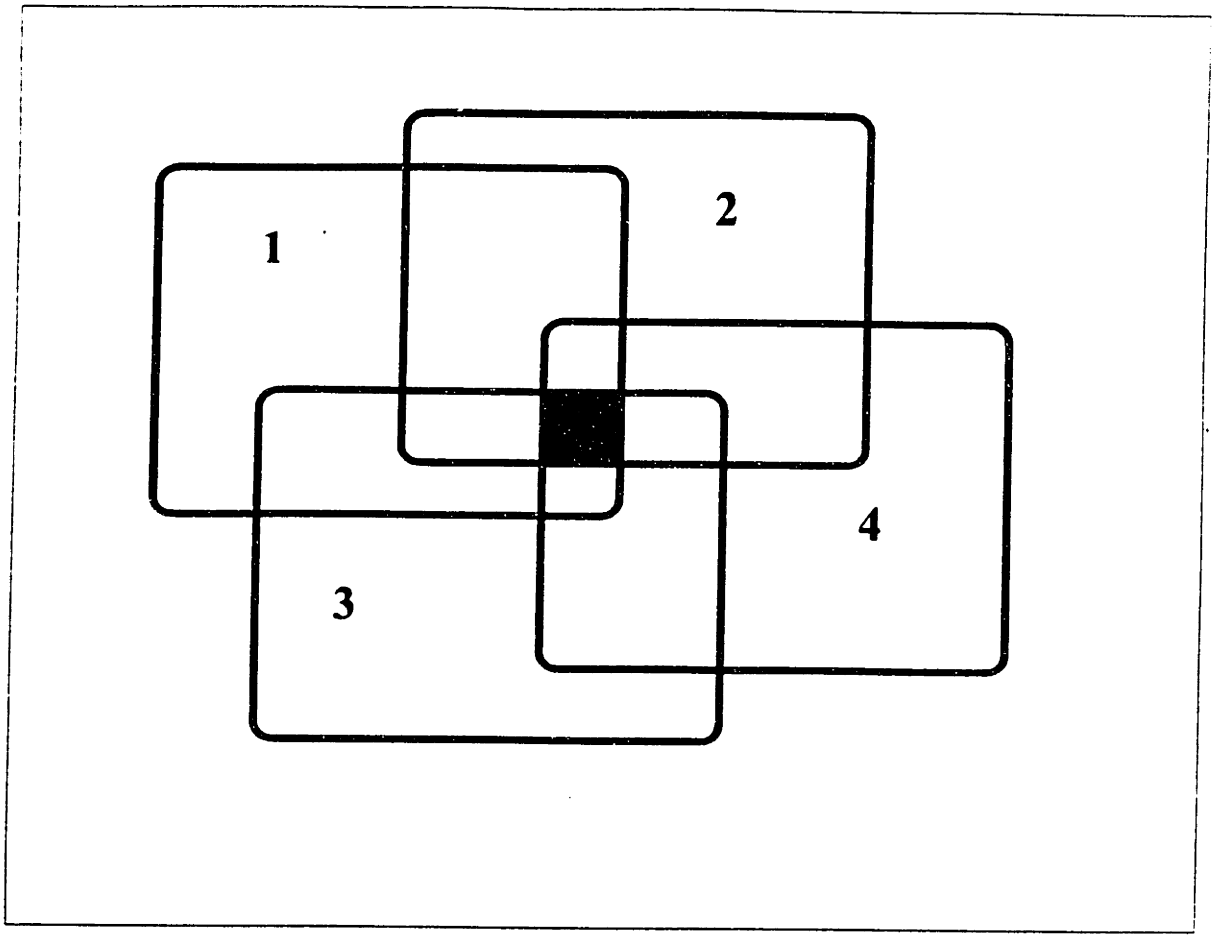


Figure 2-11: Concurrent Design Process

several implications:

(a) the sequential design paradigm must be abandoned, in view of the broader scope of each design activity's implications;

(b) design activities must be monitored, controlled, and optimized with respect to the global optima of the overall project, rather than the local optima of the individual activity;

(c) human and technical pathways for collaboration must exist, in order that excessive iteration between disciplines does not introduce phase lags between activities, and result in a parallel or sequential outcome, and

(d) designers within each activity cannot be limited in their capabilities (either by training, management, or technology) to a focus that is so specialized that they must turn over substantial parts of the design activity to designers in other activities. In the research described by Gebala and Eppinger (Gebala, 1991) they postulate that cross-disciplinary knowledge is required for this concurrent design process. That is, each discipline must be just as knowledgeable in the other disciplines as all of the other disciplines are. If this were to be realized in practice, one might question the need for disciplines. In fact, this is not possible in practice without the assistance of some technology that makes cross-disciplinary knowledge available to those who need it, at the time that it is needed. (This framework is the context for the introduction of the Event-Driven Knowledge-Based Design concept.)

2.6 Design Process Simulation

As described above, the structure and organization of the design process has an effect on the process and the product. As design organizations move away from sequential processes to concurrent processes, the way that the people in the organizations work changes. While it is sufficient to operate within the system boundaries of the design task in a sequential process, where the measurements and controls are explicit and measurable, the dynamics of the concurrent process require that the system boundaries be removed, to some extent, to allow for knowledge from other disciplines, functions, and activities to permeate the once-rigid design task perimeter.

The Event-Driven Knowledge-Based Design concept is based on the notion that individuals and groups in a design process are affected by the actions of other

individuals and groups, and that the progression of the design process is directly related both to the structure and organization of the design process and to the interaction among the people within the design organizations. Since engineers within a specific discipline are not necessarily as knowledgeable in other areas as specialists in those areas, it is not always possible for the effects of individual (or combinations of) decisions to be easily or quickly assessed. In a complex product design process, a variety of types of speciaiized knowledge are required in order to provide individuals and groups with the capability to accurately determine how the events and data in the design process affect them directly. The concurrent design concept assumes that an appreciation for such effects will allow for a shorter product development cycle due to a reduction in iteration and feedback in the process.

In order to illustrate the behavior of the design process as a whole, it is necessary to model the process, and to measure the performance of the model. Such modeling will provide for the understanding of the context for a new technology, such as Event-Driven Knowledge-Based Design, in the evolution of the design process. Simulation models of the above four design processes have been built. These simulations are based on the System Dynamics methodology (Forrester, 1962) and are framed in the context of normalized units and coefficients. This normalization exists in order to create an environment for comparison and evaluation.

The basic design unit is defined as a synthesis or production activity, which is fed tasks from an inventory of new work, and which feeds a second inventory of completed work or design tasks. We may define "design tasks" as the instantiation of a specific design variable with a specific value, or the completion of a task order (such as "generate drawing" for which no specific design variable exists.) The flow of work follows a path from the input inventory, through the process activity or mechanism, to the completed design

inventory. An intrinsic feedback loop exists within each design unit, representing re-work (or redesign) contained within the design activity itself. Rework can be considered to be re-instantiating the design variable with a new value, or re-doing the specific task order. In combination, the input to the design task may consist of input from the inventory as well as the sum of all feedback loops that provide rework inventory to the design task. The output of the task consists of completed designs that are acceptable (have passed through quality control) and some fraction of the output that must be sent back for re-work. The error, feedback, or re-work fraction consists of intrinsic feedback (contained within the design task) and extrinsic feedback that is sent to other design tasks for rework.

The production of completed designs is affected by the rate of work, which is a function of the number of people and the productivity of the individual people involved in the design task. The basic productivity rate per person is defined as the parameter **p**, such that **p** = number of work packages per person per unit time.

The duration of the design task (unit time) is defined as t_d, the unit of time that it takes for a discrete work package to flow through a design activity, where the "discrete work package" is the lowest discretization of a design task, in terms of a single designer. In this formulation,

$$t_d = 1/p$$

The duration of a design task, therefore, is dependent on the total number of discrete work packages and the number of designers involved. If:

N_i = number of designers involved in activity *i*,

then a simple relationship may be defined as follows:

$$T_i = W_i/(N_i \times p_i)$$

where T_i is the total duration of activity *i*, and W_i is the total number of work packages in activity *i*. We may refer to this as the *idealized productivity function,* illustrated in Figure 2-12.

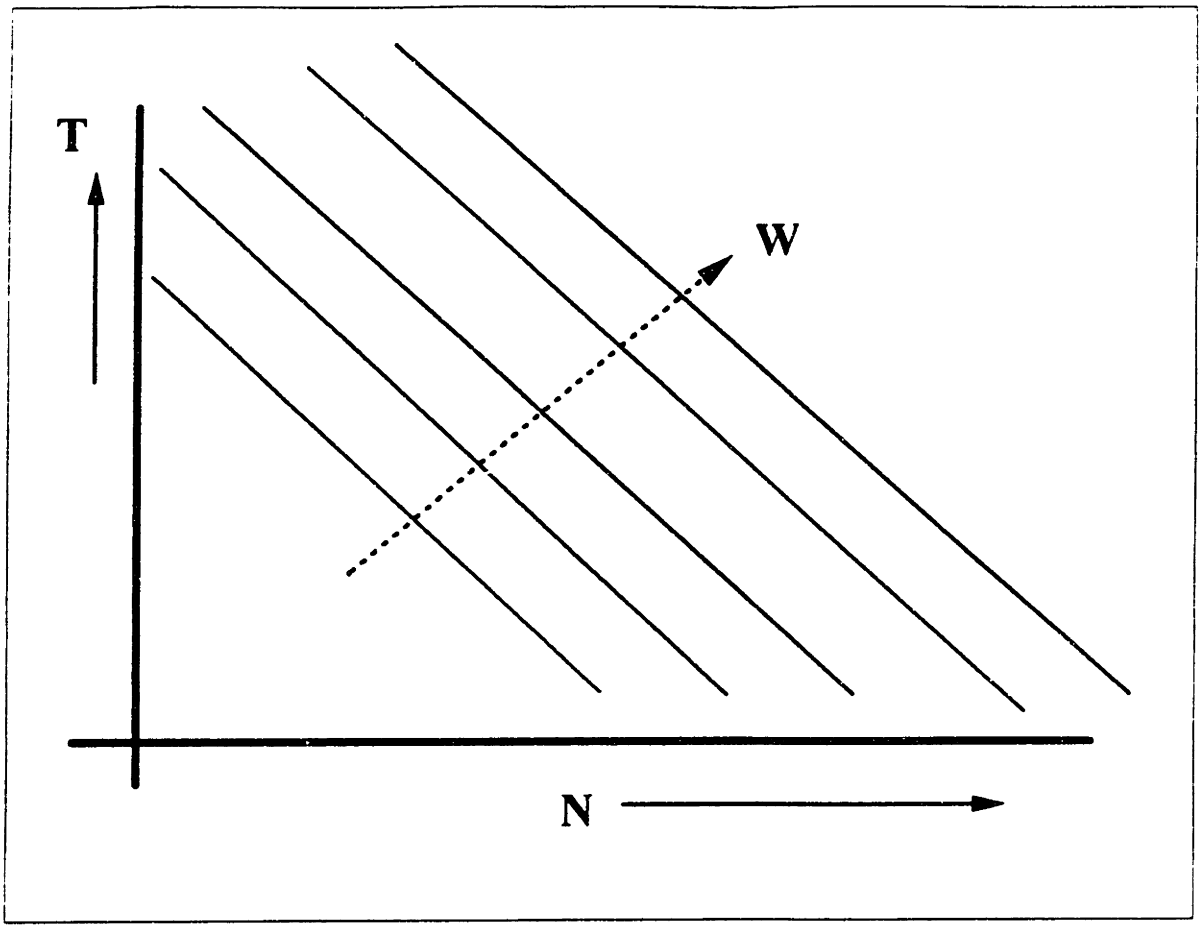


Figure 2-12: Idealized Productivity Function

Since in reality, there is a productivity decay phenomenon, the actual productivity function has the following form:

$$T_i = \left(\frac{W_i}{P_i}\right) \cdot (\alpha) \cdot (1 - e^{-\beta \cdot N_i})$$

where α and ß are productivity decay coefficients.

This function, illustrated in Figure 2-13, represents the reality that there is a non-linear relationship between the duration of an activity and the amount of resources applied to the activity.

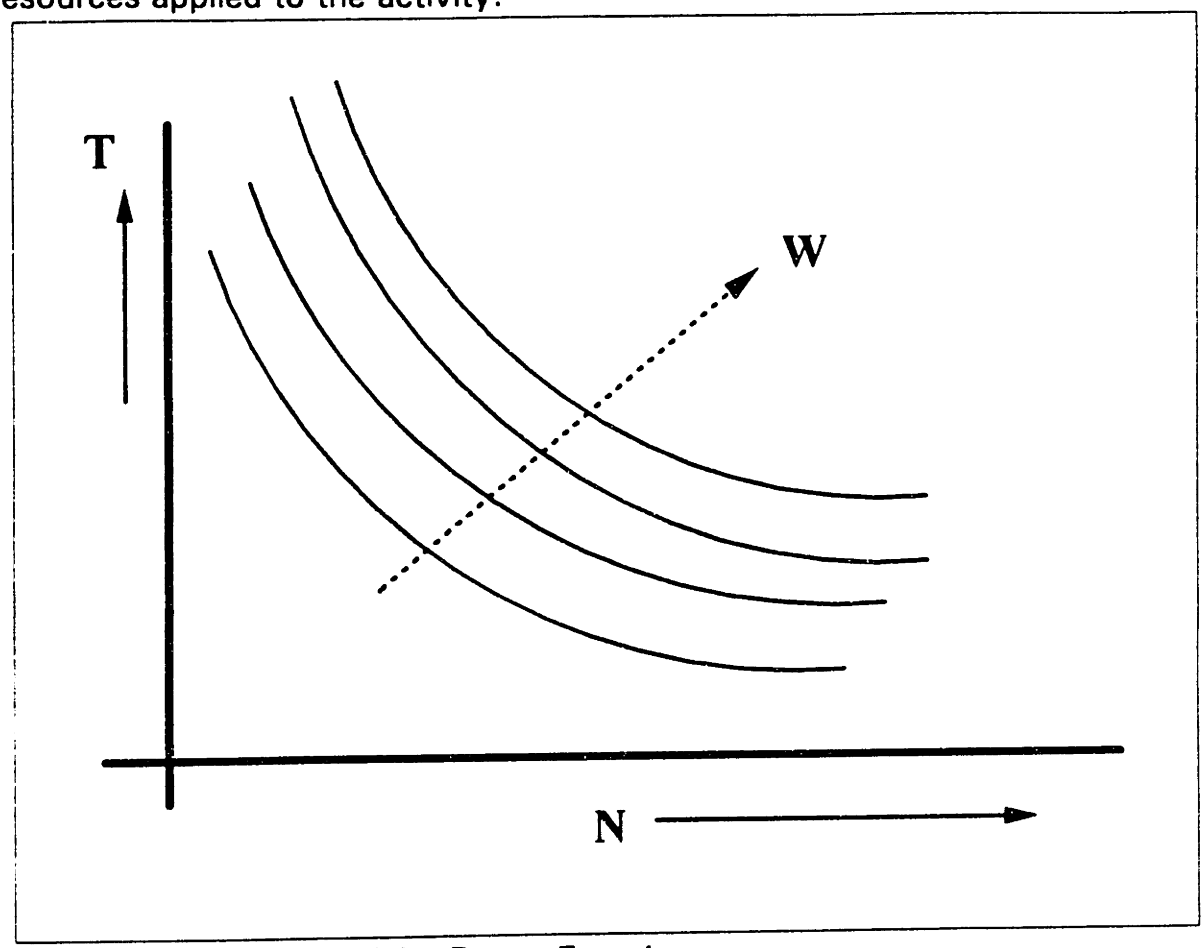


Figure 2-13: Productivity Decay Function

Notice that characteristic of these two functions are the following differences:

- The idealized function behaves in a linear fashion, such that the more designers (resources) applied to a particular task, the lower the duration of the task, ultimately approaching a duration of zero, as N approaches infinity. This holds for all constant values of W (amount of work) and p (productivity of each designer).

- Conversely, the decay function predicts that at a certain point, the benefit (in terms of duration) that is derived from increasing the amount of resources diminishes. That is, the effectiveness of the organization does not improve linearly with the amount of people, but reaches a saturation limit, after which the effect of applying more resources is a minimal, if any, improvement in duration.

We can also apply a cost function to these basic processes, such that the cost of the design activity is a function of the resources applied and the cost of the amount of time that the tasks take.

$$C_i = c_{ir} + c_{it}$$

where C_i is the total cost of activity *i*.

Note that c_{ir} is the cost component of activity *i* as a result of the cost of the resources applied to activity *i*, and c_{it} is the cost of the activity incurred as a result of the duration of the activity.

Depending on the nature of the product, the market, the type of design labor involved, and a host of other factors, these two cost components can vary in

relationship to each other. For example, the development cost of a new commercial aircraft can be as much as $3 billion, over a four-and-one-half (4 ½) year period, with approximately ten thousand people involved. Conversely, the cost of developing a new screwdriver may be $150,000 over a period of one year with a staff of three people. (Ulrich, 1995) Obviously, the cost of a day's worth of development in these two scenarios differs (1 day out of three person-years, versus one day out of forty-five thousand person years.) The effect of such a delay is significantly different, in terms of resource cost. If a day was totally wasted, then the cost would be ten thousand person-days versus three person-days for each process, respectively.

In general, the cost of a design activity may behave according to a declining cost (with respect to time) and an increasing cost (with respect to resources) as illustrated in Figure 2-14.

As can be seen by this curve, there is a point at which too many applied resources have a negative effect on the overall cost of the project, relative to the benefit of a shorter duration. Note that this relationship depends on the industry and the context. For example, the cost component due to duration may be much more severe for a manufacturing facility in the bio-technical industry, where time-to-market is a highly competitive issue. This may be less significant of an issue in a new layout of a wiring system for a new screwdriver design.

The resulting cost curve yields a non-symmetric parabolic function, with bias towards the dominant cost component. Figure 2-15 illustrates a potential resultant composite cost function for the general case.

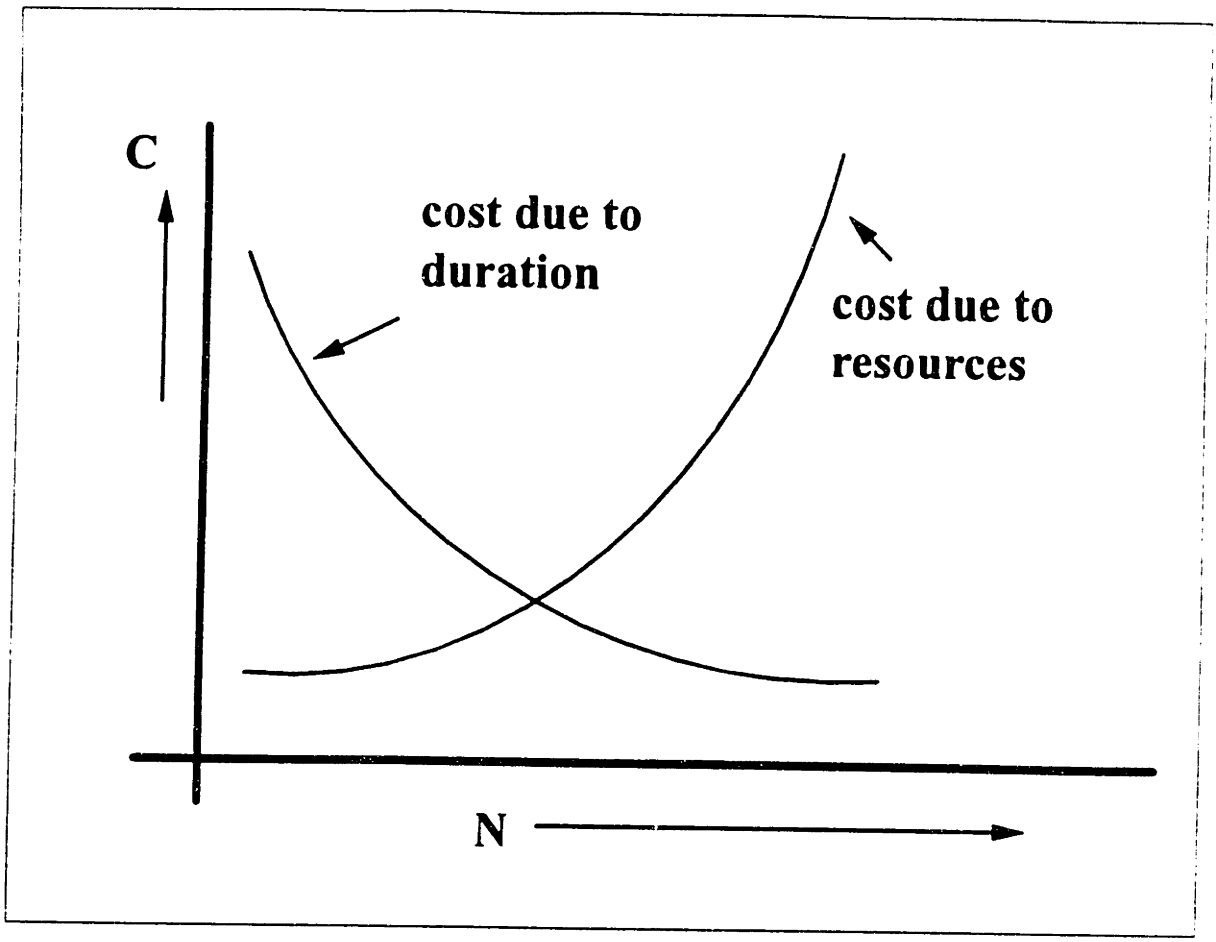


Figure 2-14: Cost Effect of Resources and Duration

2.6.1 Simulation Description

The discrete inputs to the simulation are, therefore, the productivity factors, the amount of labor (resources), the design quality (resulting in a discrete feedback or rework ratio,) the durations of design activities, and the basic process configuration. The quality ratios, number of people, and productivity curves were kept constant across the various models. The simulations were performed using the commercial software **ithink**® from High Performance Systems, Inc. (HPS, 1994)

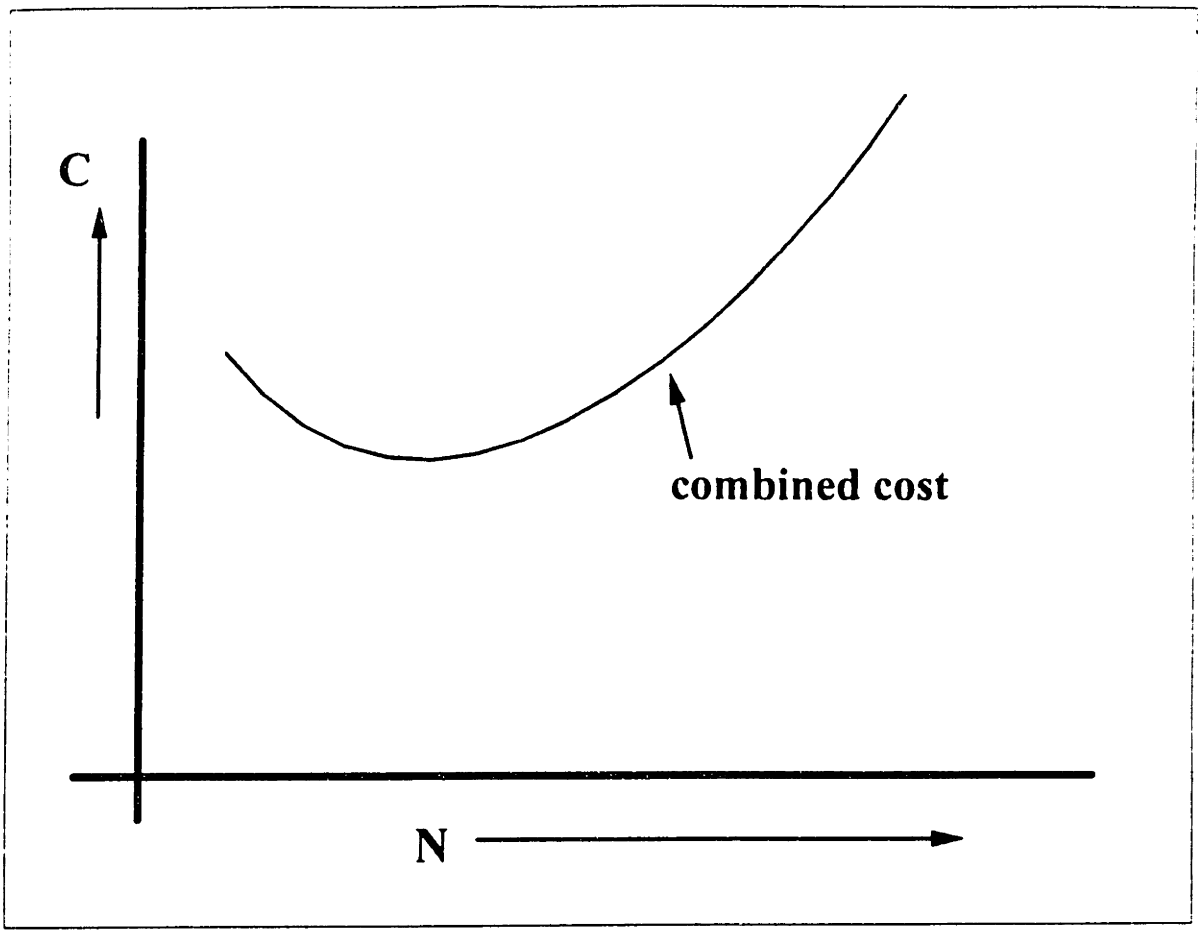


Figure 2-15: Resultant Composite Cost Function

2.6.2 Basic Design Unit

The basic design unit consists of a design input inventory, a design process mechanism, and a completed design inventory, as illustrated in Figure 2-16. There is an intrinsic feedback mechanism, modeled as a "leakage" in the design process. This feedback is controlled by a feedback ratio (the fraction of work that is returned for redesign) and a temporal factor controlling the time at which feedback occurs, and the span over which the feedback applies within the design activity.

As the work flows into the design unit, it is acted upon by the designers. This activity is driven by the productivity of the designers and the number of designers available, modified by the decay function described above. With no feedback, the performance of the unit is linear, as shown in Figure 2-17.

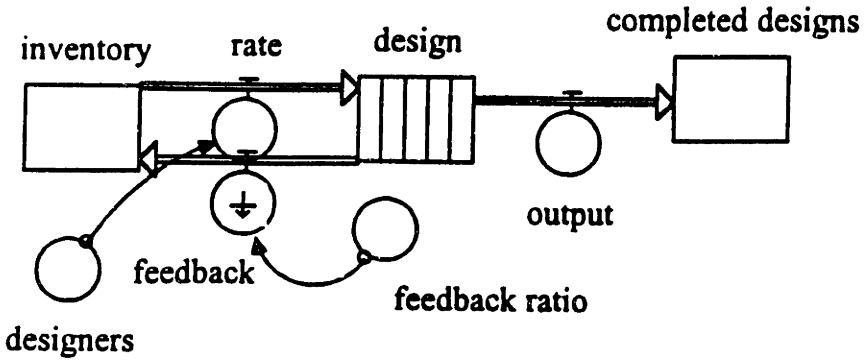


Figure 2-16: Basic Simulation Design Unit

As a quality factor is introduced, the linearity of the behavior changes, and an "S-curve" results. This affects the duration of the activity. Figure 2-18 illustrates the behavior of the basic unit with a quality ratio of 0.75 (or a feedback ratio of 0.25). This means that one quarter of the design output is rejected.

A feedback sensitivity analysis shows that as the quality of the process declines (increasing feedback) the duration of the process increases. Nota also that the process with no rejects (feedback = 0.0) has a linear behavior, while those processes that include feedback have the "s-curve" behavior. A comparative analysis yields the following behavior for increasing feedback ratios (or decreasing quality.)

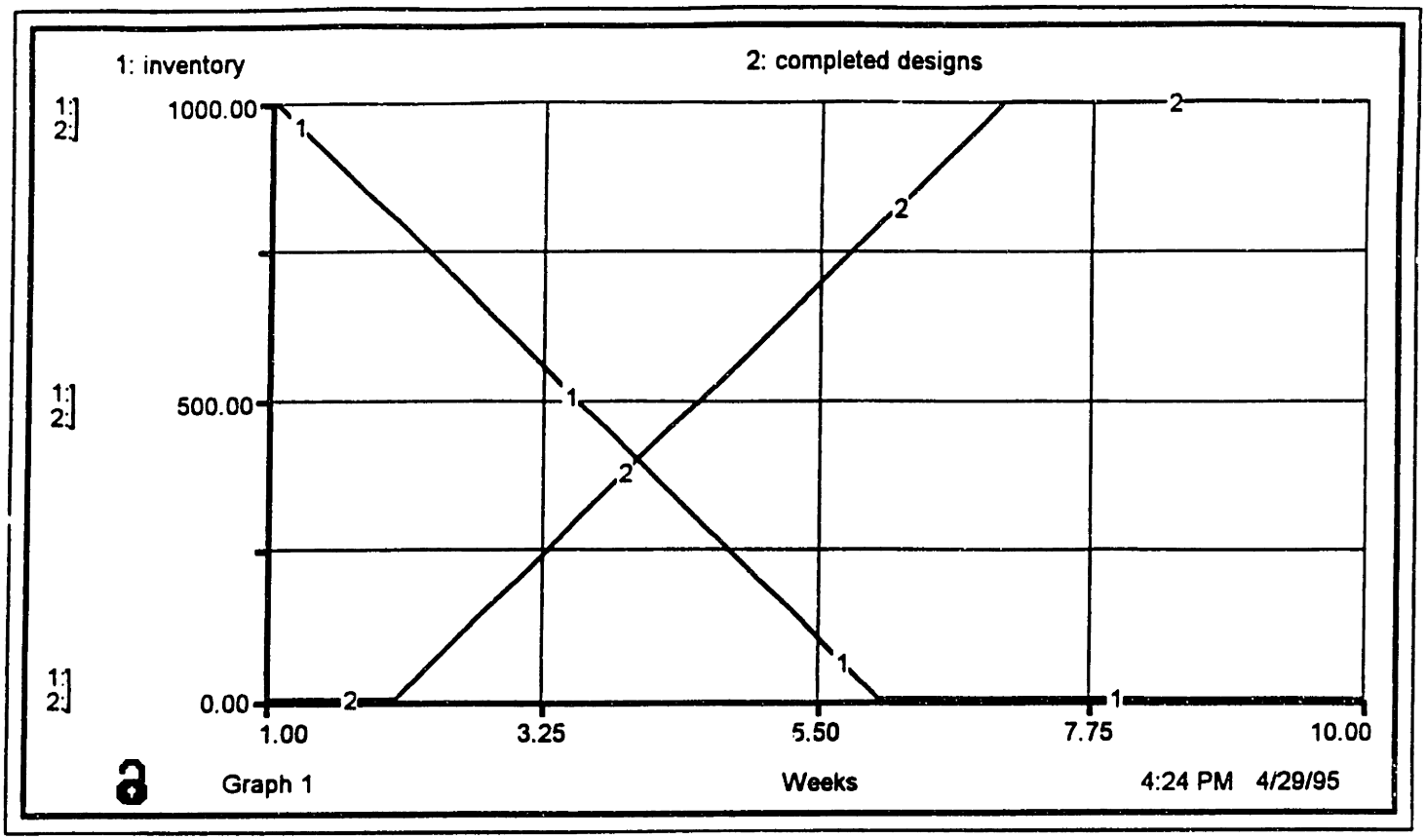


Figure 2-17: Behavior of the Design Unit with No Feedback

Figure 2-19 illustrates the increase in duration of the design activity as the quality declines.

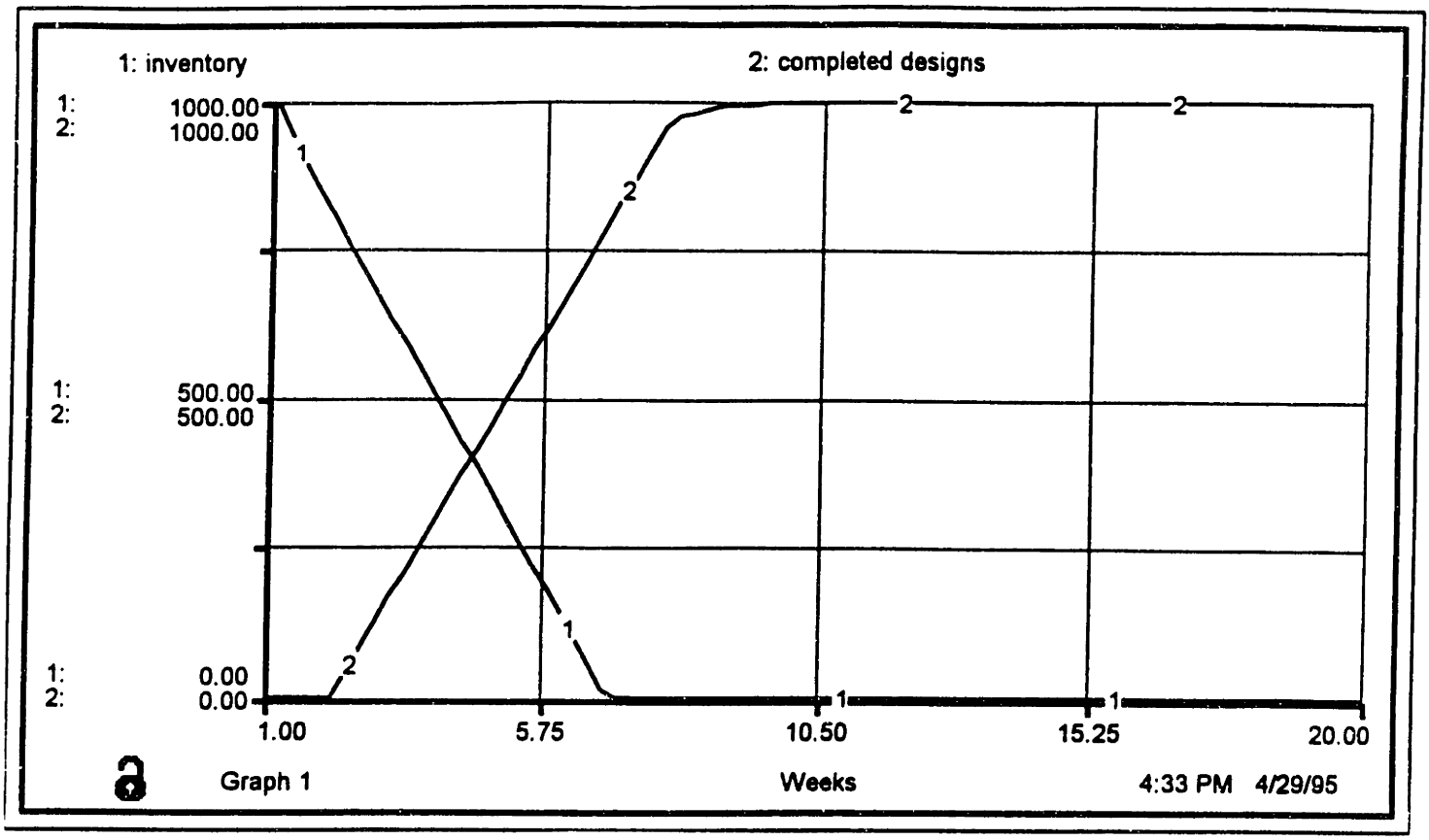


Figure 2-18: Basic Design Unit With Feedback Ratio = 0.25

The simulation is performed using a fourth order Runge-Kutta integration algorithm, with the model equations for the basic design unit as follows:

```
COMPLETED_DESIGNS(T) = COMPLETED_DESIGNS(T - DT) + (OUTPUT) * DT
INIT COMPLETED_DESIGNS = 0
OUTPUT = CONVEYOR OUTFLOW
DESIGN(T) = DESIGN(T - DT) + (RATE - OUTPUT - FEEDBACK) * DT
INIT DESIGN = 0
   TRANSIT TIME = 1
   INFLOW LIMIT = INF
```

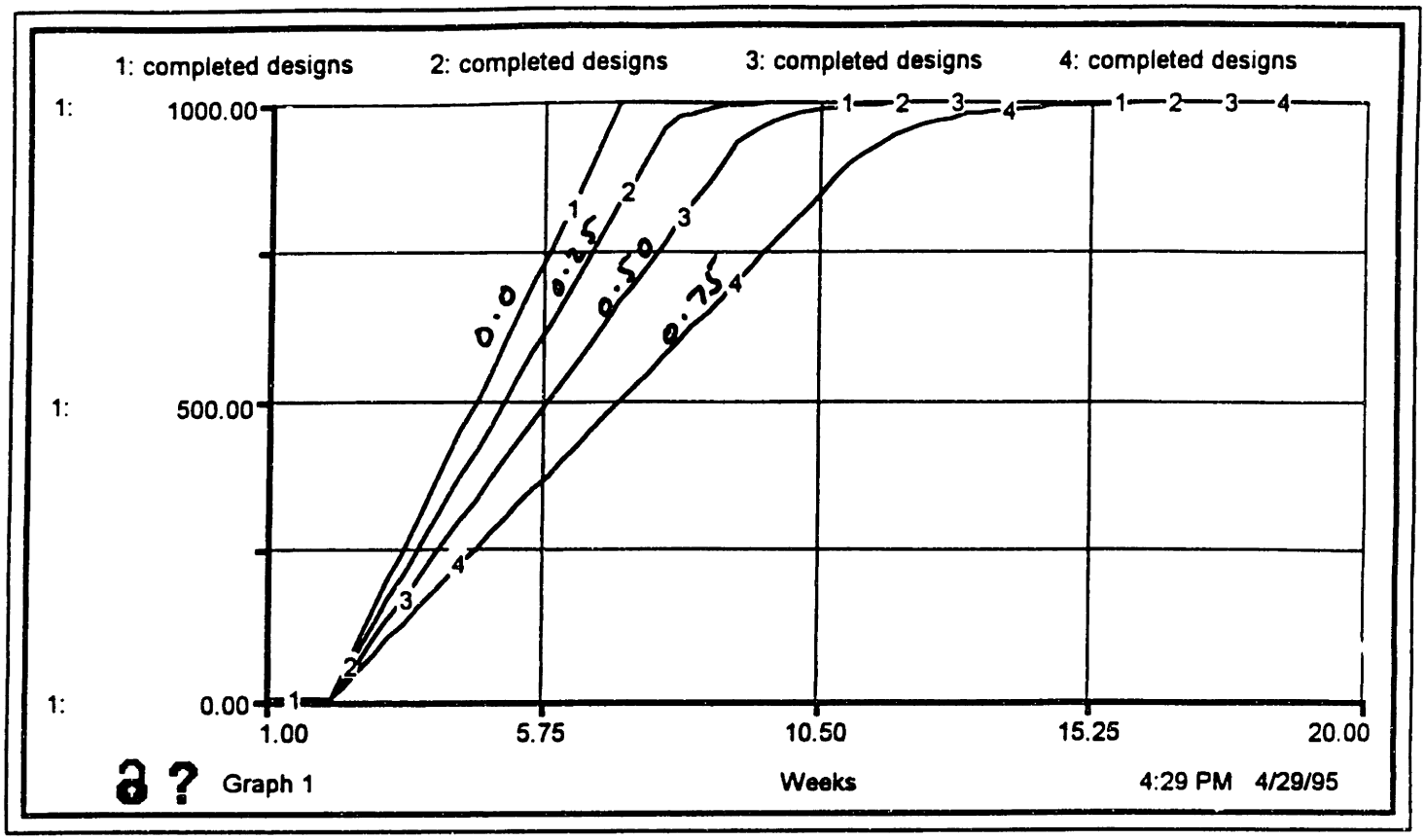


Figure 2-19: Basic Design Unit Feedback Sensitivity Study

CAPACITY = INF

RATE = DESIGNERS

OUTPUT = CONVEYOR OUTFLOW

FEEDBACK = LEAKAGE OUTFLOW

LEAKAGE FRACTION = FEEDBACK_RATIO

NO-LEAK ZONE = 0.5

INVENTORY(T) = INVENTORY(T - DT) + (FEEDBACK - RATE) * DT

INIT INVENTORY = 1000

FEEDBACK = LEAKAGE OUTFLOW

LEAKAGE FRACTION = FEEDBACK_RATIO

NO-LEAK ZONE = 0.5

RATE = DESIGNERS

DESIGNERS = 100

FEEDBACK_RATIO = .25

2.6.3 Sequential Design Model

The basic design unit is coupled with another basic design unit to form a chain of sequential units. This is repeated until a six-unit model is formed. Additionally, a global feedback function was created such that the downstream units could provide feedback to the prior design units in an extrinsic fashion. The quality function was assumed to be equal for each design unit, and was distributed to prior units in the following fashion, with the matrix values corresponding to the coefficient applied to each feedback loop on the (i,j) of the matrix:

	1	2	3	...	n
1	1	0	0	...	0
2	0.5	0.5	0	...	0
3	.33	.33	.33	...	0
.					
.					
n	1/n	1/n	1/n	...	1/n

Feedback Associativity Matrix for Sequential Design Process

where n = the number of design units in the sequence.

This means that for the (2,1) matrix value, the second design unit has a feedback loop to the first design unit with a fraction of 0.5 applied to its quality value, so one half of the feedback applies to the extrinsic rework to design unit #1, and one half applies to its intrinsic feedback, which has matrix value (2,2). The entire model is illustrated in Figure 2-20.

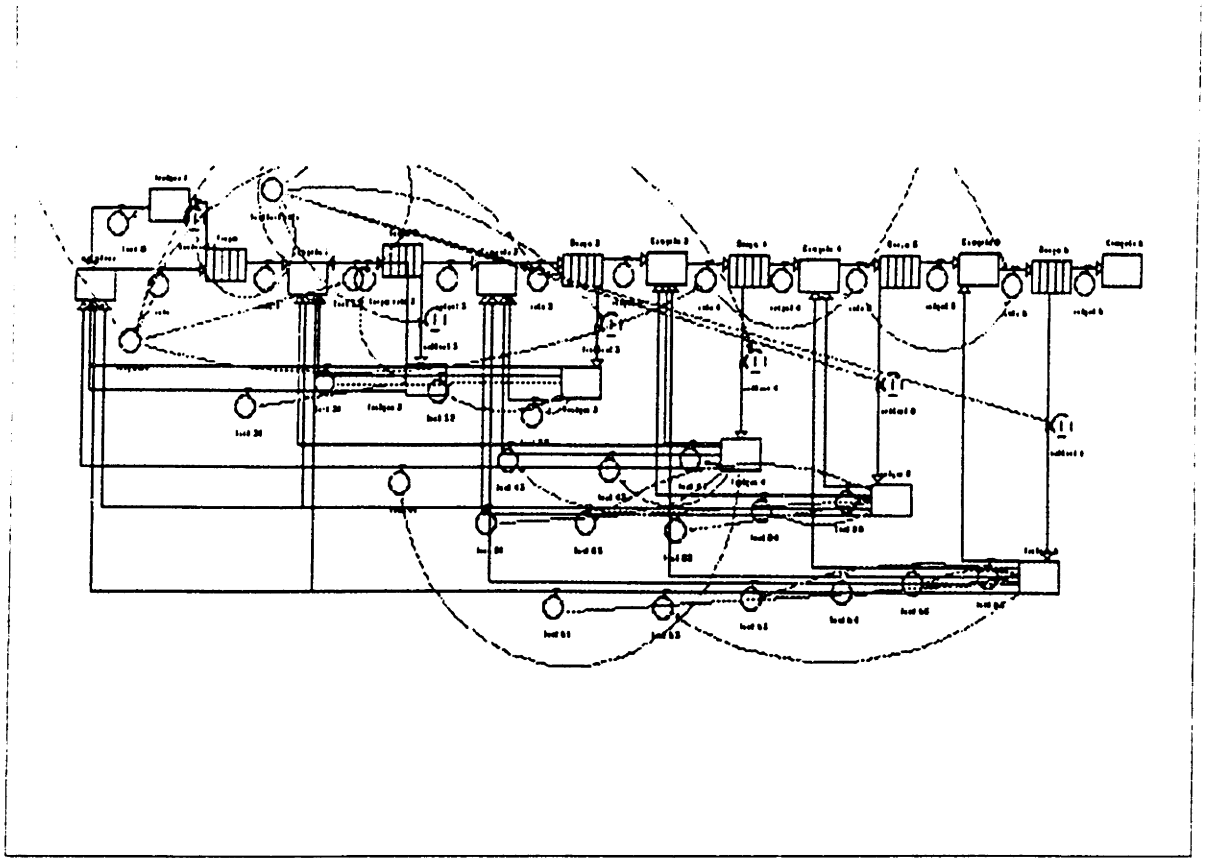


Figure 2-20: Sequential Design Model

The behavior of this sequential model is illustrated in Figure 2-21. Note that the basic process behavior exhibits similarities to that of the basic design unit, but the duration is, of course, much longer.

A feedback sensitivity analysis has also been performed, using the same sensitivity values as for the basic design unit, and the results are shown in Figure 2-22.

Note that for the normalized simulation values, the total design duration is 34 weeks. We can also assign a basic cost to the labor and to the duration itself. This translates into a total cost for the design activity based on the following formula:

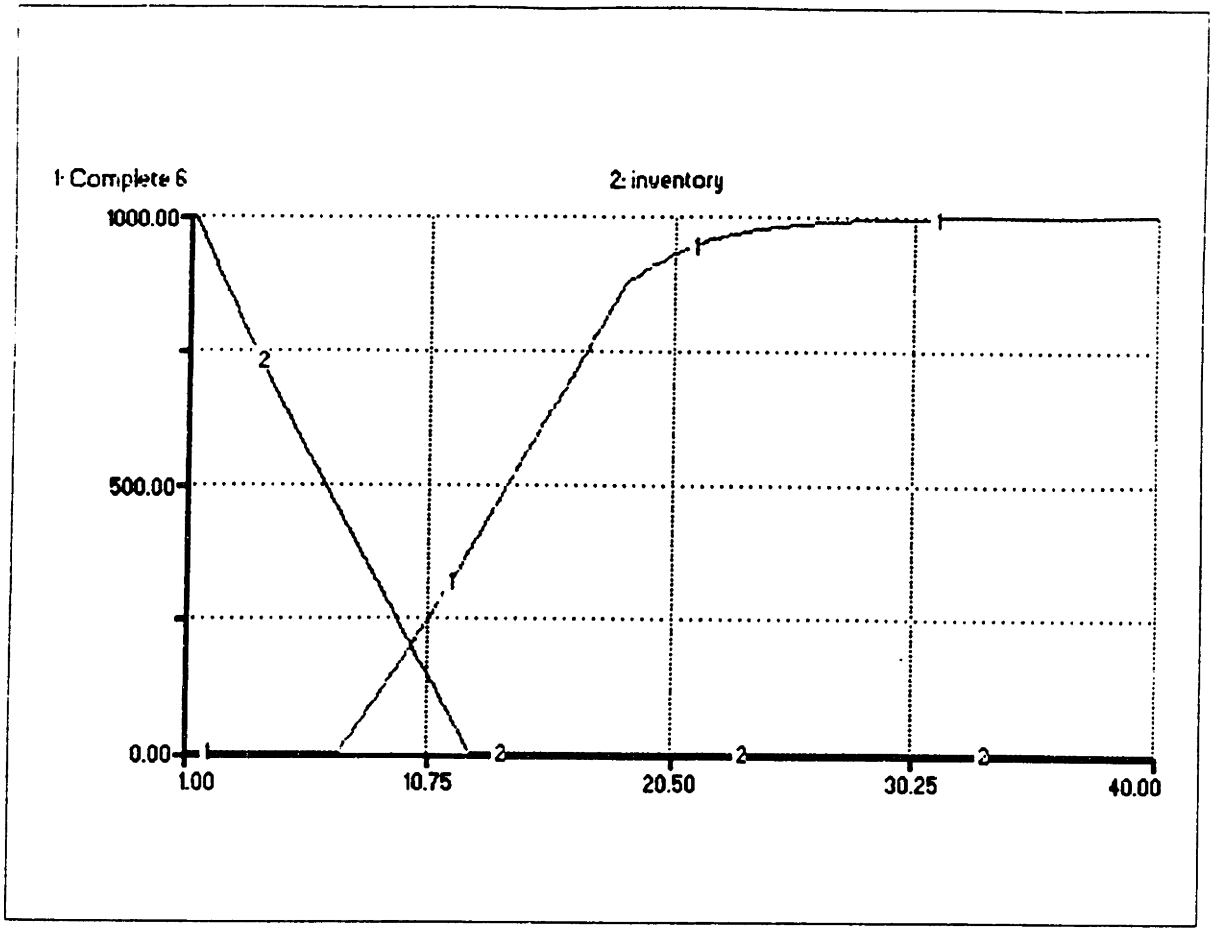


Figure 2-21: Behavior of Sequential Design Model

$$C = c_r + c_t$$

So, for this sequential process, the total cost is:

$$C_{sequential} = \$1000/wk \times 34\text{-wks} \times 100\text{-people} + \$3000/wk \times 34\text{-wks}$$

$$C_{sequential} = \$3{,}502{,}000$$

2.6.4 Parallel Design Model

The same six design units as in the sequential model are now organized in a parallel fashion, with a phase delay for each successive task's initiation. There

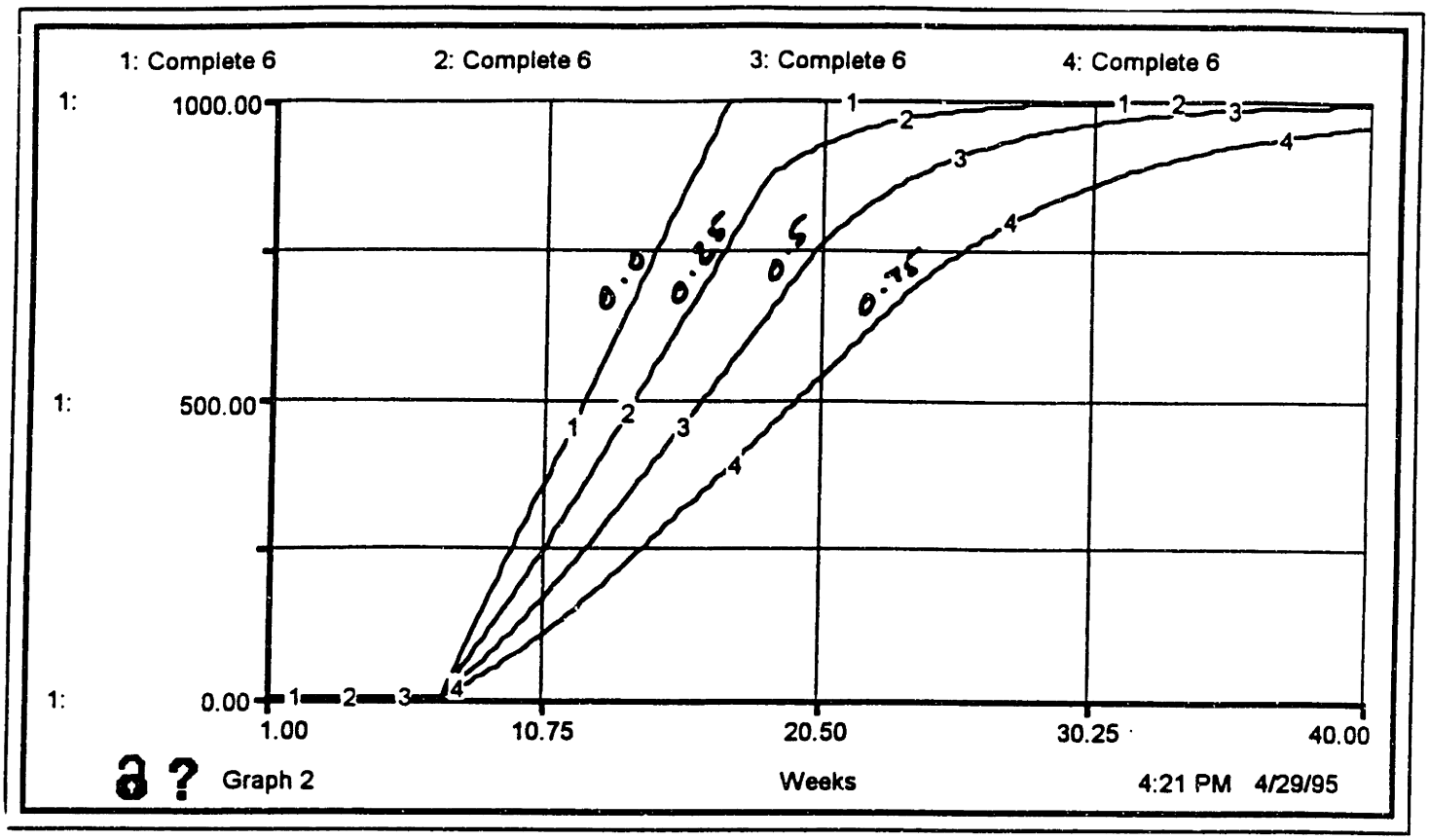


Figure 2-22: Feedback Sensitivity of Sequential Design Model

is a collector at the completion of the design task row, and a collective feedback is applied to the completed design work packages based on the accumulated designs of each parallel row. This feedback loop supplements the intrinsic rework that exists within each design unit. Based on the logic that the initial work is performed with incomplete or uncertain information from prior tasks, the parallel model has a non-linear decaying feedback function. This represents the increased feedback that results from uncertainty at the early stages of the overall process, but reduces to a comparable rework ratio to the sequential model at the later stages of the process. The decay function for this feedback is:

$$f(t) = \alpha (1 + e^{-\beta \times t})$$

where α and ß are stability coefficients and t is time. As such, the feedback decay function for the parallel design model has behavior illustrated in Figure 2-23.

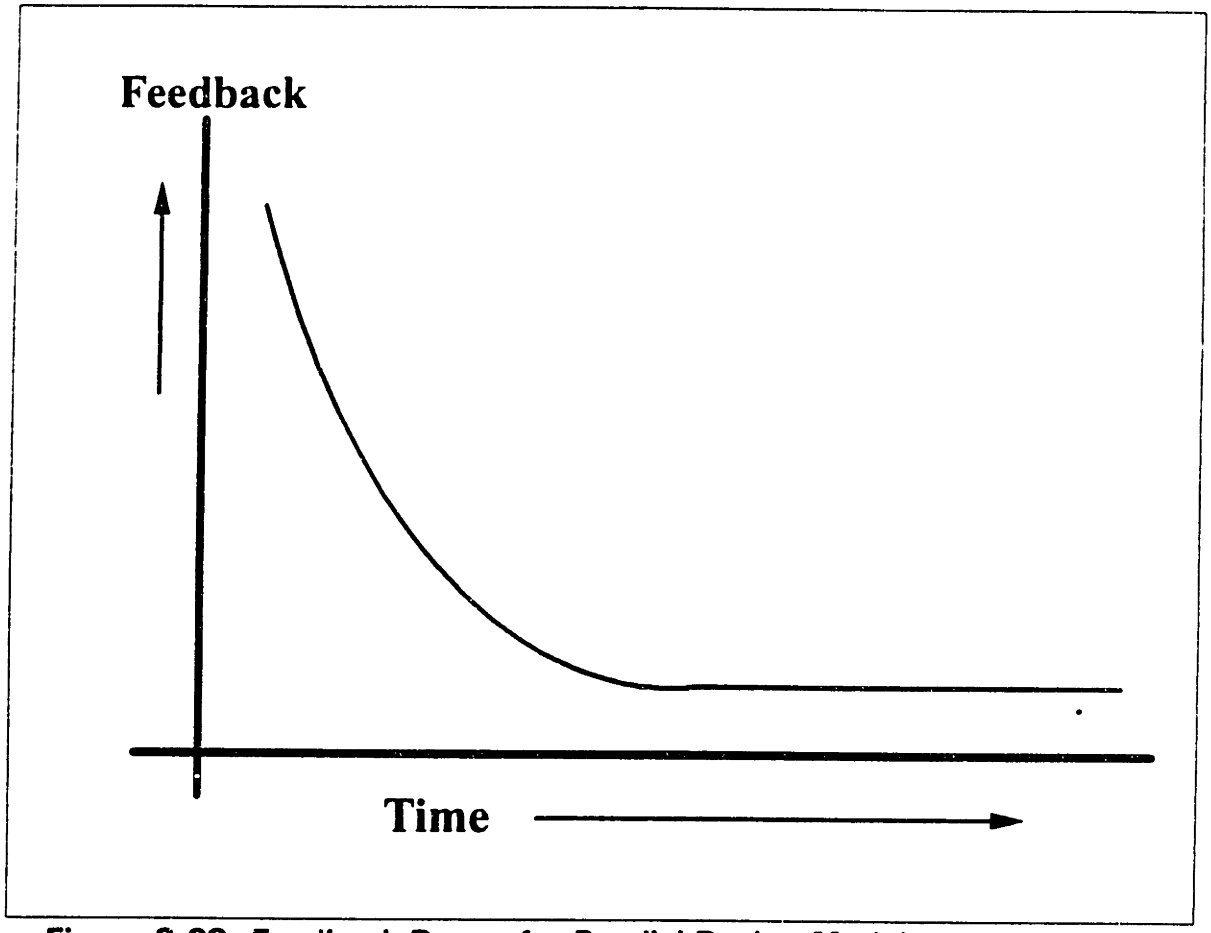


Figure 2-23: Feedback Decay for Parallel Design Model

The total process model for this parallel process is illustrated in Figure 2-24. The basic behavior of this design model is illustrated in Figure 2-25. As can be seen, the fundamental design process has similar behavioral characteristics as the basic design unit and the sequential model, but the duration is shorter. We calculate the total cost of the design effort as follows:

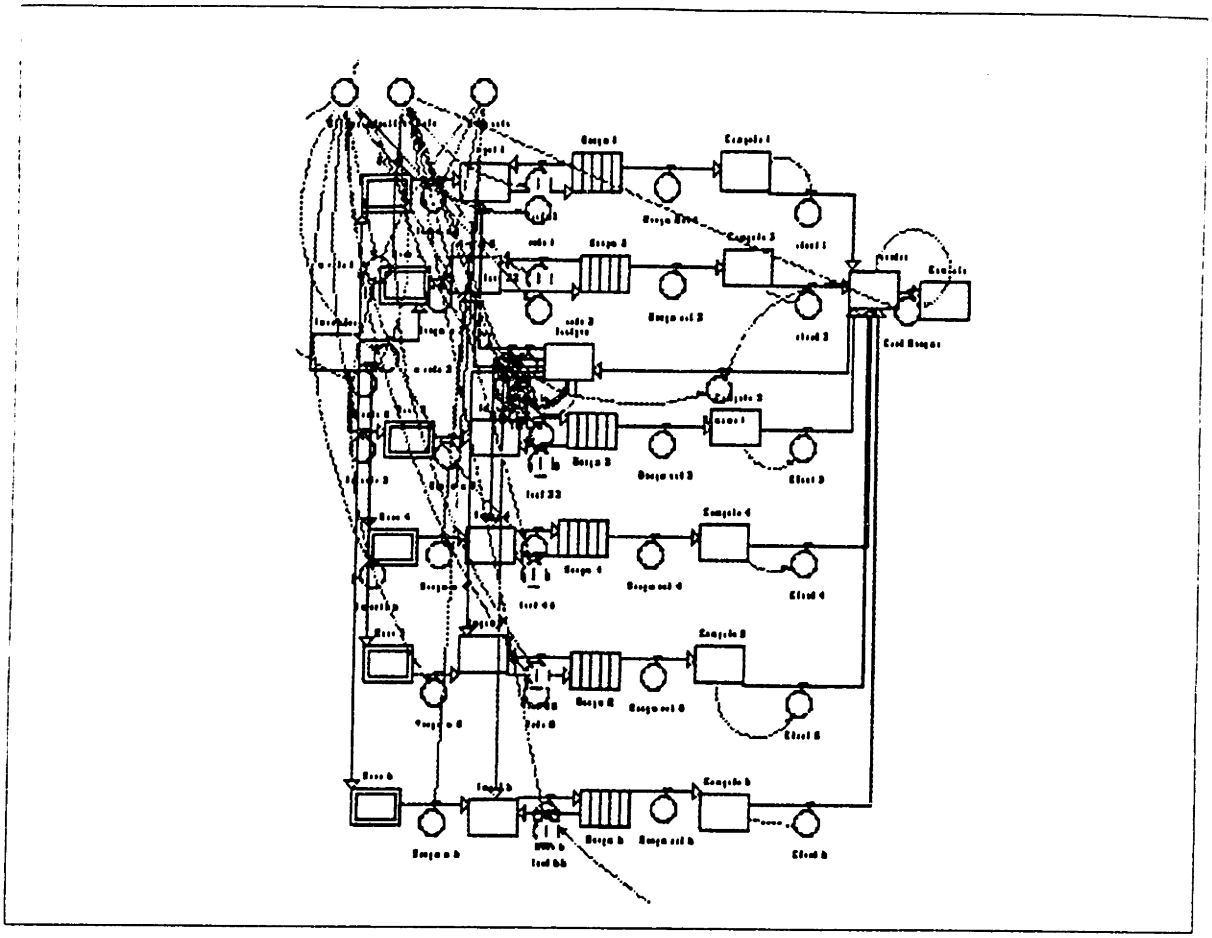


Figure 2-24: Parallel Design Model

$$C_{parallel} = \$1000/wk \times 25.6\text{-wks} \times 100\text{-people} + \$3000/wk \times 25.3\text{-wks}$$

$$C_{parallel} = \$2,605,900$$

2.6.5 Coordinated Design Model

In the Coordinated model, a central coordinator design unit collects the output from each individual design unit and provides direct feedback to the design units. The coordinator has the same structure as the production design units, and has a direct link from the output function of each design unit to its input function. The design unit feedback is generated as a fraction of the coordinator

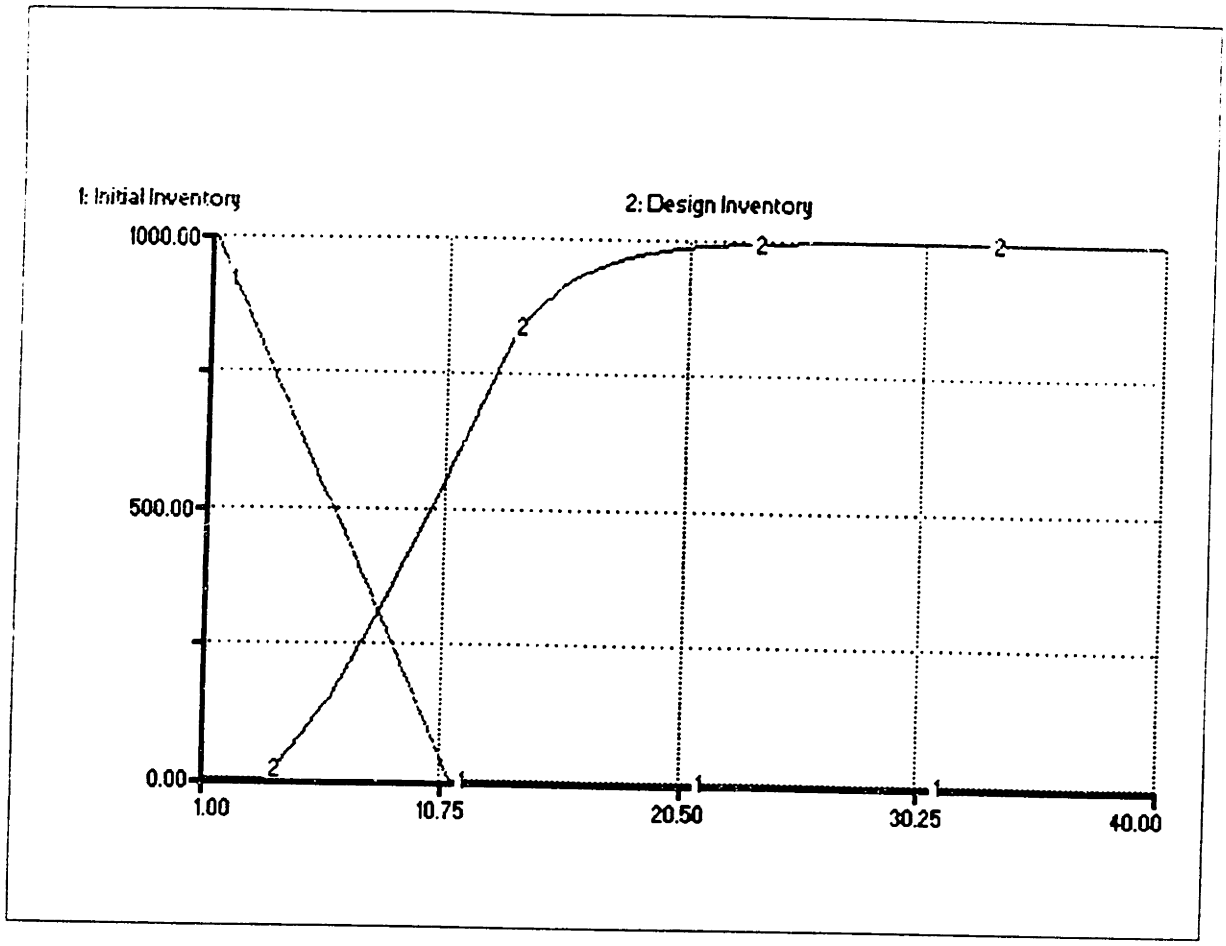


Figure 2-25: Behavior of Parallel Design Model

output, and is directed back to the individual design units as a summation to the input function.

The structure of the coordinated design model is illustrated in Figure 2-26. Note that the coordinator does not perform any design activity, but acts to coordinate the flow of information and design tasks among the design units, and determines the adequacy of the completed designs for release. The coordinated process exhibits behavior illustrated in Figure 2-27.

The Cost of the coordinated design process is calculated as follows:

$$C_{coord} = \$1000/wk \times 21.7\text{-wks} \times 100\text{-people} + \$3000/wk \times 21.7\text{-wks}$$

$$C_{coord} = \$2,235,100$$

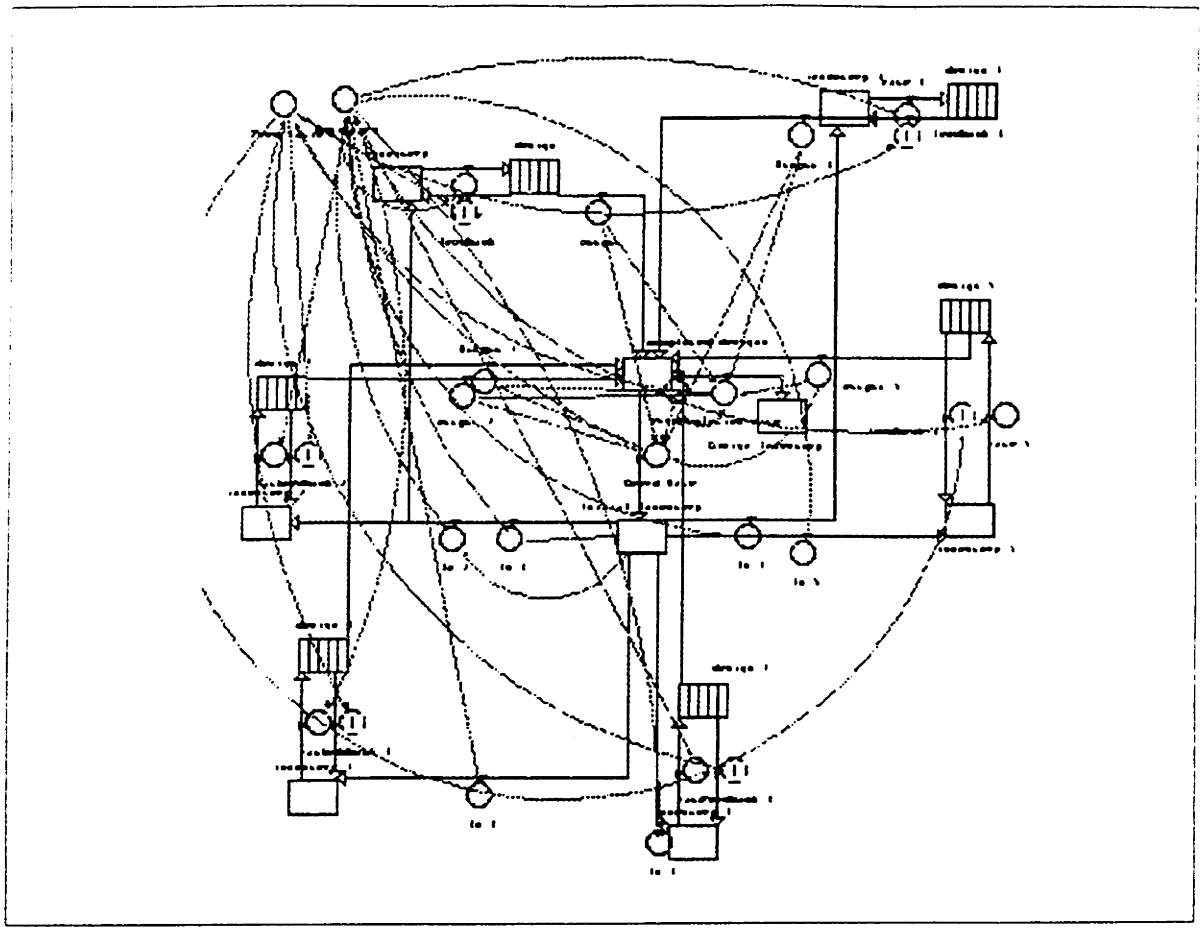


Figure 2-26: Coordinated Design Model

2.6.6 Concurrent Design Model

In the concurrent model, each design unit has a direct connection to each other design unit, such that the performance of each individual design unit depends on collaboration with every unit in the design organization. This collaboration involves not only communication of appropriate information, but ***knowledge*** of the other design units' functions and capabilities. The equations for the concurrent model simulation are provided in Appendix A.

This collaboration implies a great deal of communication between design units. Without some transfer of knowledge from one unit to another, some large

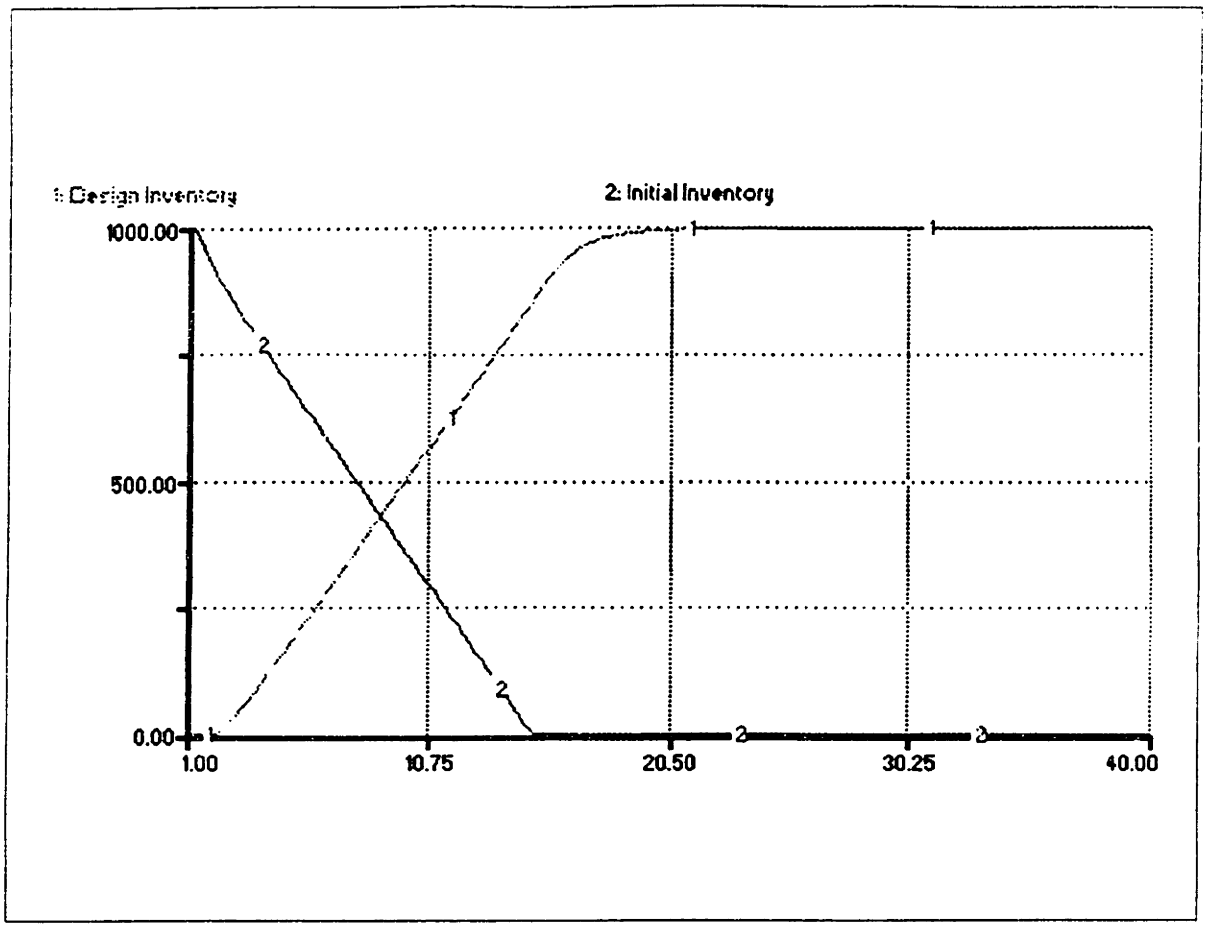


Figure 2-27: Behavior of Coordinated Design Process

proportion of designers' time is spent in communication with other design units. The basic concurrent collaborative model is illustrated in Figure 2-28.

We observe a behavior of this model that is highly feedback intensive, as a consequence of the increased amount of communication. In absolute terms, the communication between design units is positive, and results in a performance pattern as illustrated in Figure 2-29.

Communication is the transfer of information from one party to another. In this communication, losses are incurred as a result of many factors, including noisy signals, translations from one language to another, clerical errors, and many others. As such, one solution to this problem is to provide access to the source knowledge that leads to the information, rather than simply providing access

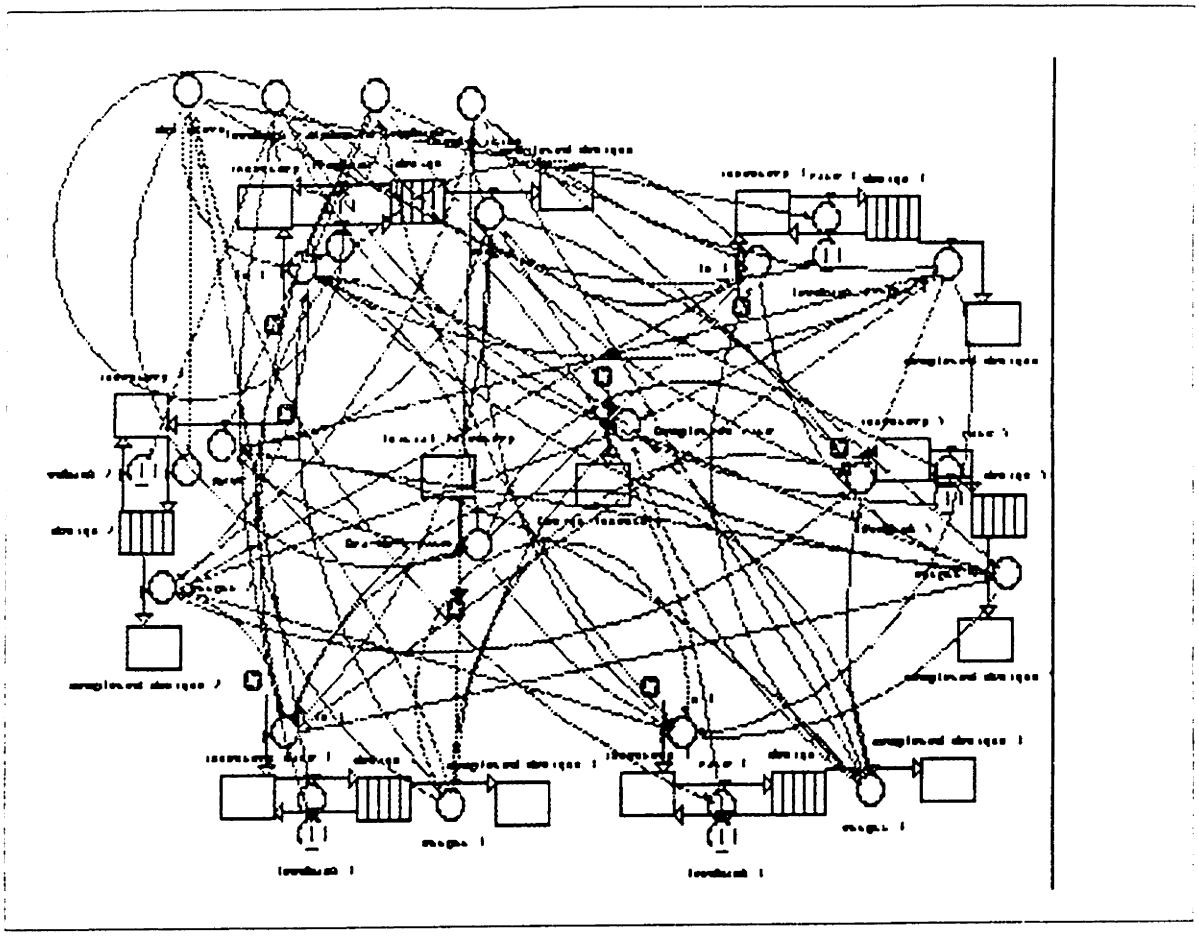


Figure 2-28: Concurrent Design Model

to the information. Since accomplishing this knowledge transfer task manually would be time-consuming and repetitive, and may not even be possible given inadequate representations and languages, it would be preferable to provide such a knowledge access mechanism using a computer technology. It is in this context that Event-Driven Knowledge-Based Design is posited.

In this improved knowledge access mode, we observe a behavior that is illustrated by Figure 2-30. As can be noted in this figure, an improvement in the knowledge access has reduced the amount of communication and a corresponding reduction in the amount of communication traffic between design units follows. This has a significantly positive benefit, in that the duration of the design activity is decreased.

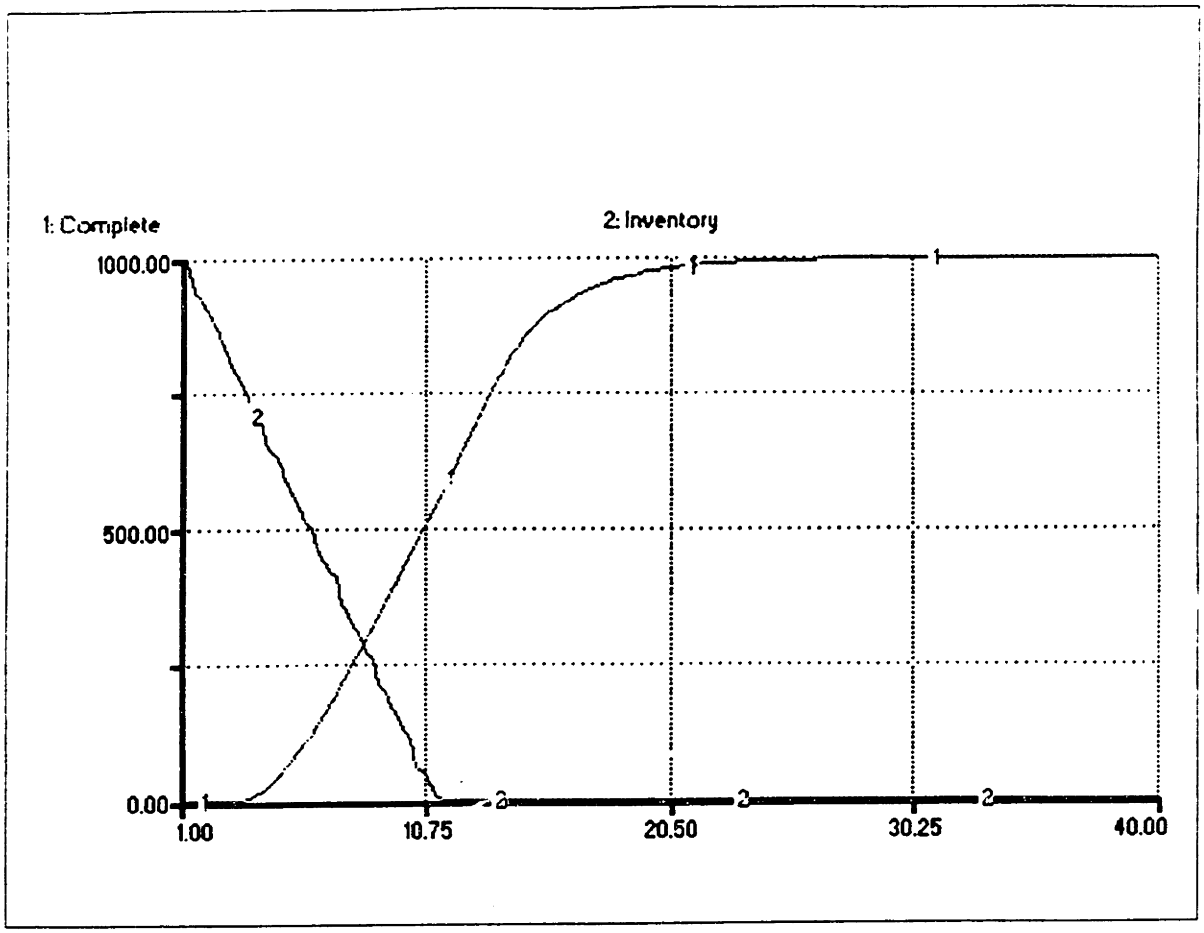


Figure 2-29: Behavior of Concurrent Design Model Without Automated Knowledge Access

The mechanism for improving such communication is to transfer knowledge about other design units across the design organization, in a manner that not only provides knowledge about other areas of specialty, but also takes into account the context of the design process, both from the perspective of the design state and the impact of design decisions on other actors in the design process, using Event-Driven Knowledge-Based Design.

The Cost of the concurrent design process without automated knowledge access is calculated as follows:

$$C_{concur\text{-}1} = \$1000/wk \times 22.4\text{-}wks \times 100\text{-}people + \$3000/wk \times 22.4\text{-}wks$$

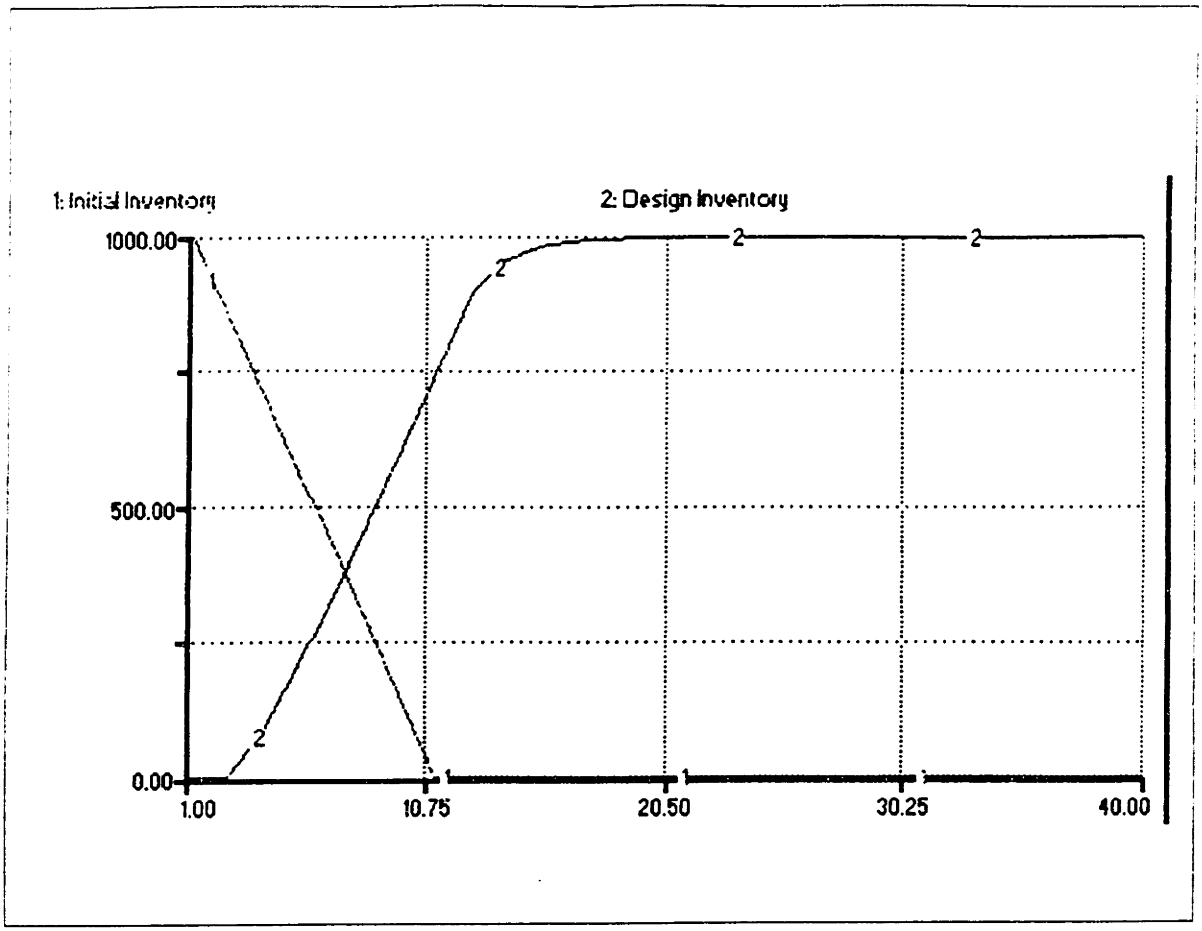


Figure 2-30: Behavior of Concurrent Model with Automated Knowledge Access

$$C_{concur\text{-}1} = \$2{,}307{,}200$$

The Cost of the concurrent design process with automated knowledge access is calculated as follows:

$$C_{concur\text{-}2} = \$1000/wk \times 18\text{-}wks \times 100\text{-}people + \$3000/wk \times 18\text{-}wks$$

$$C_{concur\text{-}2} = \$1{,}854{,}000$$

2.6.7 Conclusions of the Simulations

As can be seen from the duration and cost results, the most cost-effective and productive organization is that with the concurrent process and the on-line

knowledge access system. A comparison of the results of the simulations are tabulated in Table 2-2.

Based on the general process models postulated and executed, it is clear that the structure of the organization affects the design activity from the perspective of cycle time and cost. Using traditional methods of workflow, information management, and communication, the process changes can have a positive impact on the lead time and development cost. In order to accomplish a significant breakthrough in terms of dramatic cycle time reductions, however, a mechanism is required to improve the knowledge access of designers in the individual design units and amongst all of the design units.

Organization	Duration	Cost	% Savings
Sequential	34	$3,502,000	0
Parallel	25.6	$2,605,900	25.6
Coordinated	21.7	$2,235,100	36.2
Concurrent without knowledge access	22.4	$2,307,200	34.1
Concurrent with knowledge access	18.0	$1,854,000	47.1

Table 2-2: Comparative Analysis of Simulation Results

The effectiveness of the design organization is a product of both the structure (infrastructure) and the ability of the units in the design organization to provide timely information to the rest of the design organization, access information when it is needed, and minimize the amount of time spent in administrative information management activities that do not add value to the design itself. These process issues can only be addressed by systems that act in a highly distributed fashion, that respond to information and events, and that act transparently as a support infrastructure for the substantive design activities themselves. This requirement for distributed systems transcends the basic information management technology common in the networked environment. The need to access *knowledge* and to be able to apply knowledge in the context of the temporal and functional state of the design is critical to the ability to effectively reduce cycle times while maintaining quality.

The result of the simulations indicate that *if* such an infrastructure existed in the context of the concurrent engineering process paradigm, the results would be dramatic and profound. We observe that the difference in cycle times between the concurrent process without the distributed knowledge access and infusion mechanism and the performance of the same organizational model with such an infrastructure is significant. Such an environment, were it to exist in the design organization of a commercial enterprise, would provide dramatic competitive power.

This conclusion is consistent with the earlier observations regarding the requirements for *knowledge access* in the design process. With a re-designed process, the critical issue is the availability of knowledge whenever it is required across the distributed organization. Basic information availability is provided by distributed databases, product data management systems, and network infrastructures.

The encapsulation and distribution of domain knowledge, and the provision of ready access to such domain knowledge in a timely fashion are the pivotal issues in the success of the concurrent design process. Without ready access to knowledge from disparate fields and functions, the collaborative effort becomes bogged down in basic communication. The expectation is that individuals focused on particular aspects of the design problem will be able to utilize knowledge from other sources to supplement their activities, and that they will spend their creative and discipline-focused intellectual efforts on their tasks without having to perform non-essential knowledge retrieval functions (such as many long meetings aimed at providing knowledge perspectives other than their own.) This can only be accomplished via a technology that captures knowledge, distributes it across the organization, and participates in a dynamic fashion in the design process.

3. Knowledge-Based Design

In an increasing number of ways, the practice of engineering is becoming unalterably defined as computer-based engineering. As part of this progression, the technology of knowledge-based design has further accelerated and integrated the use of computers in the engineering design method. Knowledge-based design systems are software systems that combine knowledge, such as rules based on experience and intelligence, with other, more conventional design systems, such as computer-aided design systems, databases, engineering calculations, and often optimization methods. The objective of knowledge-based design systems is to achieve significant improvements in the engineering, design, and manufacturing process by providing ready access to specialized knowledge in an integrative fashion. This serves to improve the capabilities of the engineer or designer, by expanding the realm of available capabilities to include qualitative as well as quantitative reasoning.

The most commonly computerized activity in engineering design is the drafting activity. This circumstance has arisen due to the proliferation of Computer-Aided Design (CAD) systems throughout the engineering community. In engineering design, the application of CAD software now spans many engineering design functions, ranging from drafting to three-dimensional geometric modeling and design. Many CAD systems allow for the inclusion of non-geometric attribute information in the CAD database, assigning physical properties in association with the graphical representation of components, subsystems, and systems. The primary role of the CAD system, however, remains that of a geometric definition of parts or assemblies. In addition to advances in software architectures, hardware developments, such as the development of high-performance Reduced Instruction Set Computing (RISC) chips, have made fully functional CAD more accessible to the majority of

engineers. A great deal of engineering design work is now being accomplished with the help of CAD systems.

As described in Chapters 1 and 2, competitive factors are now influencing the practice of engineering due to greater market demand for cost minimization, flexibility in product features, reduced lead times, and higher product quality. One of the required responses to the competitive conditions is the implementation of different organizational structures, such as concurrent engineering and design.

As was demonstrated by the simulation models in the previous chapter, concurrent engineering requires integration of all aspects of the design process together with the manufacturing engineering and assembly process.

> To succeed with Concurrent Engineering requires a high degree of organizational integration within the engineering function and between engineering and other functions. (Davenport, 1993)

As lead times and product quality become more significant competitive issues in global manufacturing industries, CAD becomes too important to the organization's competitive strategy and survival to be left to the traditional CAD processes. Those organizations that do not adapt their methods, organizational structures, and operating technologies could face competitive difficulties, whereas those that do adapt will find their CAD strategy and their competitive market strategy must be integrated. This alignment of a highly specialized technical discipline with overall organizational business strategies has not been common practice in the engineering community. In non-engineering operations, business computing systems are becoming aligned with the strategy of the organization as a whole. (Scott Morton, 1991) Such a framework is presented in Figure 3-1:

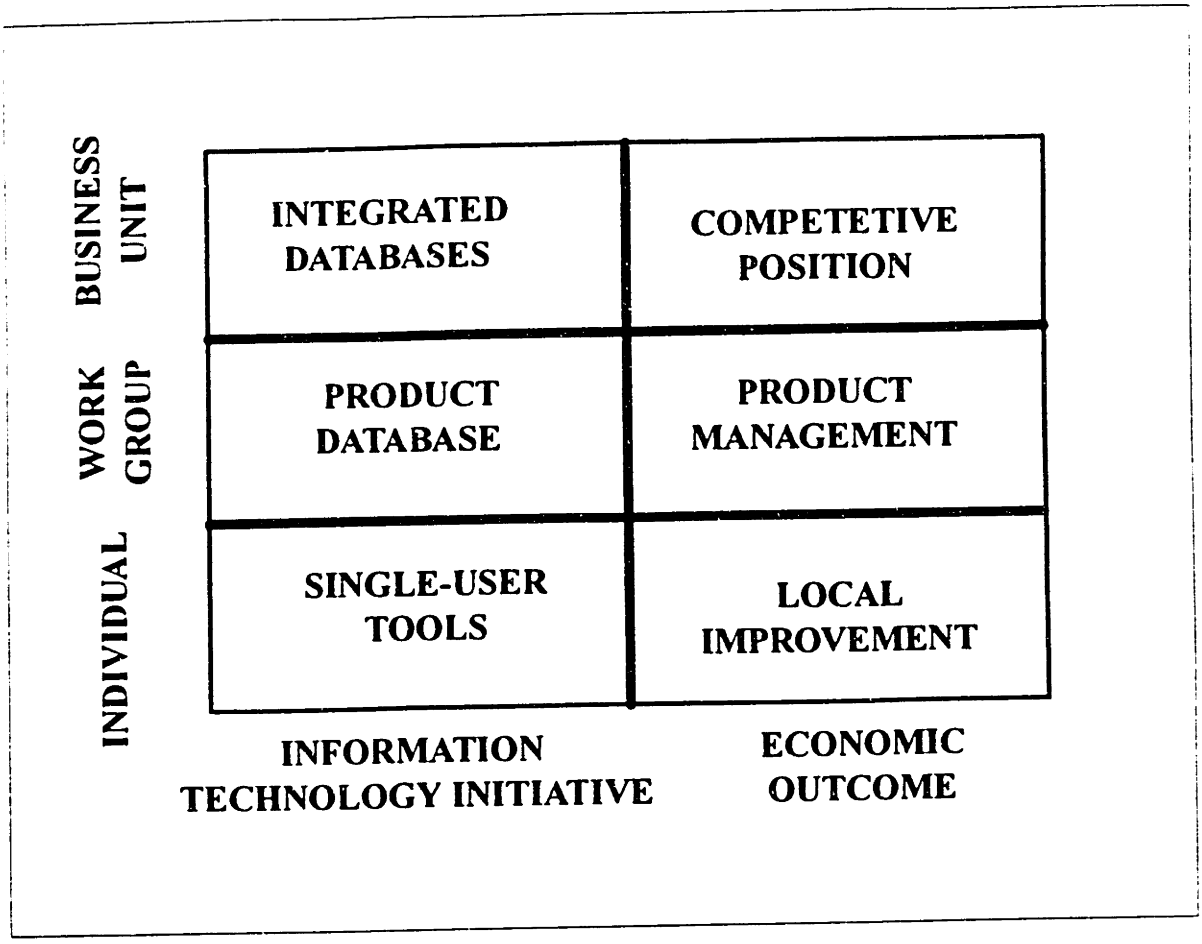


Figure 3-1: Framework for Technical Information Systems Contexts

The transformation of this business systems framework to the engineering and CAD world allows for the understanding of the strategic role that these technical systems can and must play in the organization's operations as a whole. The traditional fragmented, specialized design process leads inevitably to sub-optimization - the optimal design of subsystems rather than the optimization of the complete system or product. Such sub-optimized products will not be competitive with products optimized for manufacturability, and the present global marketplace for manufactured products will tend to eliminate inefficient producers.

Competitive factors have led to the demand for CAD systems specifically oriented toward providing a more comprehensive engineering platform for

improving local (single) user's productivity, and for facilitating concurrent engineering. The ability to make the design process more intelligent by including multi-disciplinary knowledge and by considering manufacturing and other considerations during design, are essential for the realization of concurrent engineering.

It has long been the desire of engineers to include design expertise within the CAD software, but conventional software technologies have not been well suited to this task. Using the technology of knowledge-based design systems, it is now possible to incorporate subjective, experiential design knowledge within the CAD environment. This ability allows for the distribution of relatively rare expert design and manufacturing knowledge to a wide range of designers and engineers, thus improving overall design performance and reducing the design time.

Integrated knowledge-based design systems are practical tools for accomplishing *design automation* tasks. The integration of computer-aided design systems and knowledge-based systems allows for an expansion of the functionality of CAD systems to provide *on-line advice* regarding the feasibility and desirability of proposed designs. In the *design review* mode, this architecture allows designers to propose configurations, have the knowledge-based system review the design and provide modification advice, and then modify the geometric and non-geometric aspects of the design. Knowledge-based design systems integrated with computer-aided design have provided practical, convenient, and cost-effective solutions to a number of common problems associated with engineering and manufacturing.

3.1 Early Implementations of Computer Aided Design: Drafting

The use of CAD systems has evolved significantly throughout the period since the introduction of computer-based geometric representation systems by Sutherland in the early 1960's. (Sutherland, 1963) The use of CAD has largely followed two major phases, and is now in a third evolutionary age. (Reinschmidt, 1995n) This evolution is illustrated in Figure 3-2.

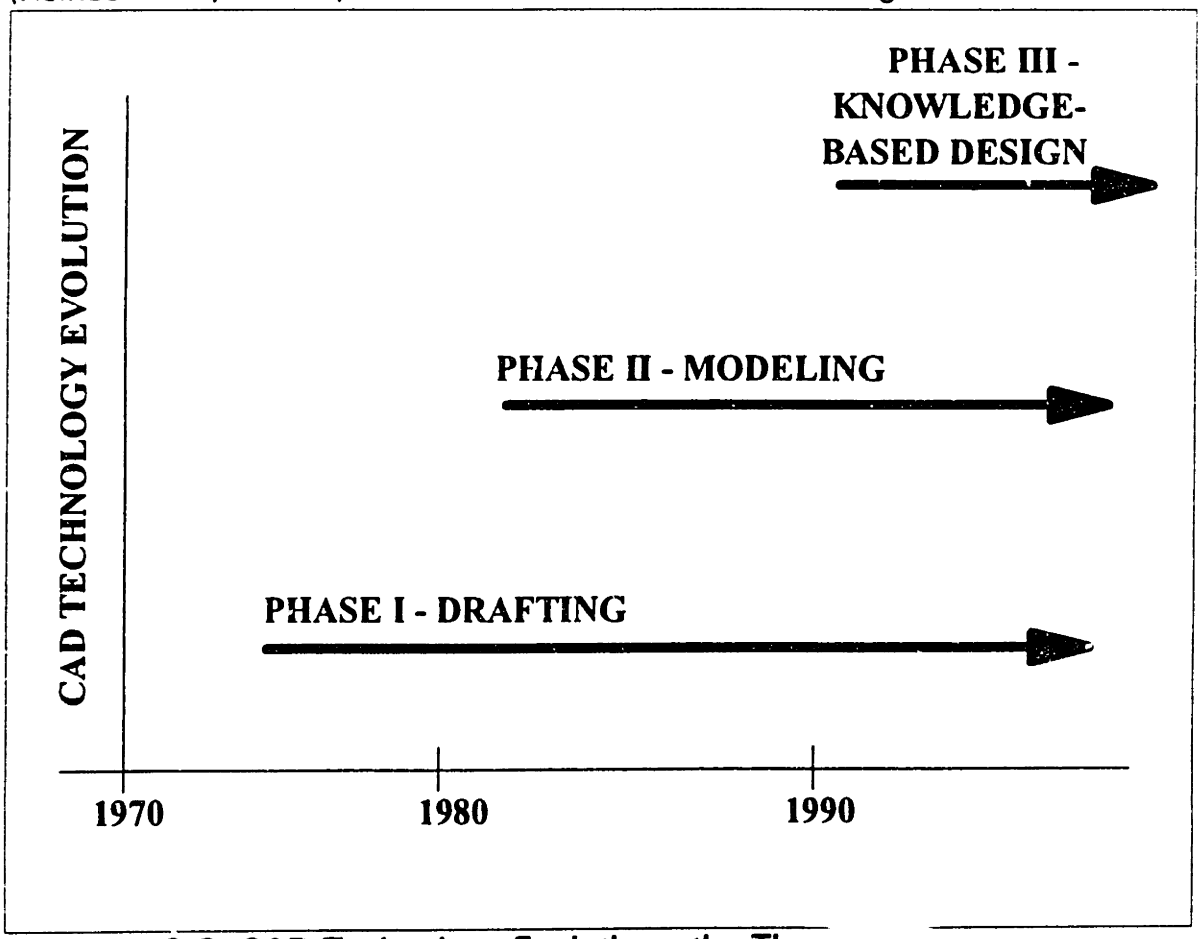


Figure 3-2: CAD Technology Evolution - the Thre es

The first phase of computer-aided design was driven by advances in technology and automation of existing methods and processes, whereas the second phase, as will be seen, is driven by larger organizational, competitive, and market forces that are shaping the nature of engineering and manufacturing.

The first phase in the introduction of computer-aided design views the introduction of computer technology into an engineering and design organization similarly to the introduction of a machine into a factory, in that there are capital investments, mechanization, and divisions of labor.

In this stage, the traditional engineering and design functions are mechanized: calculators are replaced by computers and drafting boards are replaced by CAD equipment, without fundamentally altering the product or the process.

> Historically, CAD systems have utilized the metaphor of the drafting board: drafters create part designs by manipulating lines and arcs in much the same way as they are manipulated with a pencil on a drafting board. (Robertson, 1991b)

The fundamental principle behind this approach is that productive efficiency is obtained by specialization and the division of labor. Whether in manufacturing or in engineering, productivity is maximized by dividing the process into many distinct tasks and by developing resources that are specialized to the performance of each task. In the engineering process, these specialized human resources are largely professional engineers, designers, and draftsmen. While advances in CAD systems had been advertized by the vendors of such systems for many years, in the implementation of such systems fewer of these advanced features had been utilized.

> Although the promises of CAD are technically feasible and although they are being realized in isolated instances..., in most cases CAD is merely being used as an expensive replacement for the drawing board. (Liker, 1992)

CAD systems have contributed to and perpetuated the division-of-labor between engineers and designers because the use of the CAD systems has required specialized skill that most engineers do not possess. This has resulted in very little use of CAD systems by engineers.

Because the division-of-labor philosophy had dictated the structure of the design organization for many years, the collaboration and cooperation amongst engineers and designers had been limited, and as a consequence, the CAD systems served as single-user tools. They were designed specifically to help a single individual perform a set of drafting tasks.

> Designers typically function as islands, developing their own designs from scratch and borrowing little from others or from their own previous designs. (Liker, 1992)

The consequences of this specialization or organization around function can be seen in a number of ways. Large firms that manufacture products have large engineering staffs organized by function, with specialized groups oriented toward particular types of engineering, design, or analysis. In many firms, the engineering function and the design function have been divided into separate organizations or at least assigned to different professionals. Engineers perform calculations (i.e., work with mathematical models) and designers make drawings (i.e., work with graphical models). The introduction of computerized engineering analysis capabilities has contributed to the perpetuation of this division of functions between engineers and designers, as the differing technical requirements of the mathematical formulations (finite elements, etc.) and of the computer-aided drafting systems have increased the learning time and inhibited communications between different operations. This is the

embodiment, in effect, of the computer standing between the engineers and the objectives (Donald A. Norman, referenced earlier), adding overall complexity and requiring specialization in the expertise required to operate different types of computer systems.

> Many companies have traditionally had a group of drafters serving a group of engineers; the engineer would sketch or otherwise specify the design geometry for the drafter, who would then produce a detailed drawing. But the continued use of CAD systems primarily by drafters is also due to the fact that CAD systems are difficult to learn and to use. Engineers use CAD at most 25% of their time. (Robertson, 1991b)

The division of labor between engineering and design, whether based on traditional (drafting boards) or modern (CAD) technology, has pursued efficiency in each operation at the expense of coordination and optimization of the whole. The engineering and design staff is one component of a system organized traditionally on the basis of the multi-echelon hierarchical model.

> When there is an elaborate functional division of responsibility and the people who are assigned those responsibilities are narrowly specialized, they lose sight of the larger picture and they no longer possess the knowledge required to coordinate their activities with those of other parts of the organizational structure. In order to make sure, then, that the parts fit back together to form the whole, the separate elements of the productive process, or the separate divisions of the organizational structure, must be strictly subordinated to some higher authority which does possess the knowledge and the information required to coordinate the separate activities. It is this requirement that explains the tightly

integrated, hierarchical structure of the old corporate organization. (Piore, 1989)

Consistent with the industrial, hierarchical model, the first stage of computer-aided design involves the mechanization of the design process by the introduction of capital equipment. By the introduction of computer-aided drafting equipment, some organizations achieved economic benefits through increased productivity, ranging from 50% to 200%, depending on the repetitiveness of the operation. Many engineering organizations, however, did not achieve the productivity improvements that they had been led to expect.

> While productivity gains of up to 5,000 percent were once predicted (and vendor claims of 1,000 percent are still common), 100 percent to 200 percent gains are the best that most firms can realistically attain. Many firms gain no productivity at all, and some claim that their staffs are less productive using CAD. (Liker, 1992)

While the utilization of computers to create and store drawings is an essential step in the evolution of the engineering and design process, there is a need to achieve greater productivity. The need to go beyond the drafting board metaphor has led some organizations to implement the technologies developed in the second phase of computer-aided design.

3.2 Advanced Implementations of Computer Aided Design: Modeling

The second phase of computer-aided design is characterized by the introduction of advanced functionality within the CAD systems. This increased functionality is based both on the requirement to broaden the scope of the CAD systems' applicability, and also to increase the range of functions available to the drafter user. Included in the advances in this phase of CAD systems technology evolution are three-dimensional design using surface definitions (as opposed to wire-frame models,) solid models that include the notion of volumes, mass properties, etc., as additional functions attached to conventional two-dimensional drafting systems. This use of three-dimensional representations has ultimately lead to the ability to use the CAD systems as geometric modeling environments, rather than simply electronic drawing systems.

Because products exist and are manufactured in a three-dimensional world, the use of three-dimensional computer-aided design is essential to advanced design processes, such as concurrent engineering. The use of three-dimensional solids modeling for the design of entire products, previously impractical, has been made practical by the rapid advances in computer graphics hardware. Reduced Instruction Set Computer (RISC) workstations now permit the display of full three-dimensional CAD models with hidden lines removed or in image (shaded) mode, and the improvements in the price/ performance of computer memory permit very large three-dimensional solid models to be stored at commercially acceptable costs.

The introduction of three-dimensional CAD reflects an advance in understanding of the use of CAD beyond the view of increasing the efficiency of drafting production to a larger view of providing a better representation of reality as the

foundation for improvements in downstream usability of the model:

> While designing in three dimensions is a more difficult and more time-consuming process, it provides significant benefits. Design in three dimensions requires greater mental involvement with the part to be designed - it requires that the entire part be considered simultaneously and completely. Design in two dimensions allows some cheating - some important design details can be ignored and left for a downstream process to determine.... The result of designing in three dimensions is that (1) the time to produce the first set of engineering drawings is lengthened and (2) the downstream processes (e.g., the production and testing of prototypes) are shortened. (Robertson, 1990)

Not only does three-dimensional design reduce "cheating" (it is much easier to design a product that cannot be assembled using two-dimensional CAD than with three- dimensional CAD), there are a number of other advantages as well. Three-dimensional design leads to an object orientation: instead of lines comprising projections on a viewing plane, the designer can deal with the objects themselves in a virtual space. These virtual solids have area, volume, mass, and other engineering properties that can be quickly computed by the CAD system; they are objects, not merely lines on paper or on a computer screen.

Associated with three-dimensional design is engineering analysis in three-dimensional worlds. Three-dimensional mathematical models can be developed from three-dimensional geometric models for finite element analysis, computational fluid dynamics, vibration analysis, structural analysis, thermal analysis, hydrodynamic analysis, manufacturing planning, tool design, cutter path planning, and other engineering purposes. Moreover, the results of these

engineering analyses can themselves be visualized in three-dimensional space. As a consequence, the barriers between the mathematical representation and the visual representation, between the algebraic model and the geometric model, between engineering and design, and between design and manufacturing, are reduced.

With three-dimensional design, dimensions (lengths, areas, volumes) are maintained in true one-to-one scale, without the distortions characteristic of orthographic or isometric projections. With solid modeling, the computer can automatically compute true surface area, volume, weight, center of gravity, moments of inertia, and other properties directly from the three-dimensional representation. That is, these three-dimensional models are not merely pictures, but are databases of the engineering properties of the design.

> Existing computer aided design systems (CAD) are most useful for modeling a design, that is, once a design concept has been chosen, the CAD system is used to make an accurate record of the idea. This record, or "model," can often be used to analyze the design and generate realistic graphical renderings. The systems do little (if anything), however, to help the designer conceive a better design idea; they only help to create a useful record of the designer's solution....The designer must still generate the original design and it is for this process that we seek to provide assistance. (Jakiela, 1989)

The benefits of using three-dimensional design transcend industries and disciplines. These systems play a major role in improving the use of the computer, to be sure, but they also improve the overall process of engineering as it integrates with the project as a whole.

> Specifically, its [three-dimensional design's] role in better linking the

> activities among project management, design, engineering, construction, and operation can be used to promote more effective coordination of information at both the overall project management level as well as among the often "isolated" activities of design, engineering, construction, and operation ..... The full potential of 3D modeling can best be realized when it is applied all the way through a project. (Wilson, 1994)

In the context of construction, 3-D design systems have been used for developing construction sequence models and simulations, providing a tool for management of the construction process hitherto impossible without physical models. (Coles, 1994) The use of three-dimensional design systems linked with scheduling systems can also be used to perform what-if analyses for construction activities, prior to the detailed development of construction plans and schedules. (Hapgood, 1995)

In other industries, the use of three-dimensional solids-based design has been credited with significant lead-time reductions and cost savings, in addition to product quality improvements. For example, interference problems with hydraulics on the Boeing 777 landing gear were detected early in the design. Without the three-dimensional CAD technology, this problem would have only surfaced once the landing gear was being tested (cycled) for the first time - after it had been manufactured and installed in the aircraft. (Gottschalk, 1994) Boeing reports a dramatic improvement in how the parts of the aircraft fit together, compared with previous aircraft which were designed using two-dimensional drawings and physical mock-ups. (Procter, 1994) Note that due to the organizational, cultural, and technical issues involved in transforming the design process from a paper-oriented, two-dimensional process to a full three-dimensional design, there was no reduction in lead time (time-to-market) for the first 777 aircraft design program, but expectations are that derivative models

will experience a fifty percent reduction in engineering development time. (Moeller, 1994)

Similar results have been reported in the automotive industry. The Chrysler Neon was designed essentially in three-dimensions, and this process lead to significant improvements in the overall product realization process. A full twenty weeks was eliminated from the development schedule due to the electronic modeling capabilities versus traditional wood body panel model techniques. This technical backbone of electronic modeling allowed for the implementation of concurrent engineering, because of the ability of disparate functional groups to work on the same basic design model without having to wait for data hand-offs. (Haase, 1994)

Beyond the use of three-dimensional representations, other significant features of CAD have evolved into practice. Included in these advanced features are explicit modeling, parametric modeling, variational modeling, and feature-based modeling. These modeling environments all approach the development of geometric models from different perspectives, allowing for the engineering approach to be dictated by the user of the CAD system, rather than the system itself. (Kurland, 1992) Other CAD advances include the inclusion of Automated Finite-Element Analysis (FEA) mesh capabilities, associativity between relations, dimensions, parts, and assemblies, and bi-directional associativity.

The advanced CAD functionality described above provides for a platform that is well suited for transforming the CAD environment into a fully functional engineering setting. However, none of these functions contain contextual knowledge, nor do they provide the ability to capture reasoning and decision-making capability essential for elevating the performance of CAD users beyond

the geometric definition of their respective designed products. Even with these advanced functions, CAD systems still lack sufficient capabilities to break existing lead time barriers. Existing barriers include functionality that is driven by mathematical and geometric representations, skills and expertise that are specific to the use of CAD technologies, and the single-user orientation of the modeling environments. Some automotive companies are examining additional technologies, such as holographs and virtual reality to further reduce cycle times by eliminating physical clay models. (Christian, 1995)

3.3 Knowledge-Based Computer-Aided Design

The use of computer-aided design technologies largely followed existing traditional organizational and functional structures through the second phase of CAD technology evolution. Conversely, the third phase in the implementation of computer-aided engineering and design is represented by re-design of work methods and adoption of technologies and tools that enhance the functions of engineers and designers. Unlike Phase I, in which the introduction of computerization is intended to mechanize the production of drawings and calculations without changing work methods, this phase is characterized by an intent to use computer-aided engineering and design to effect change in the product and in the process.

The use of Artificial Intelligence (AI) techniques, of which knowledge-based technology is a derivative, have been in use to a limited degree in engineering for many years. An AI approach for applying symbolic reasoning to highly analytical problems was proposed in the field of Continuum Mechanics in 1970, in order to:

> "... (1) delegate to the computer the recognition (by their pattern) of formulas and expressions, and their logical manipulation, complemented by a capacity to step directly, when desired from symbolic expression to the performance of numerical evaluation; and (2) as a consequence of 1, use the computer as a tool for retrieving and associating knowledge .... performing algebraic manipulation at various levels ... proving theorems and generating new formulas and equations." (Wong, 1970)

Knowledge-based systems have been used in a variety of engineering fields, such as equipment diagnostics, welding, and project management (Finn, 1986), real-time plant operations and control, (Finn, 1988) planning and scheduling

in finite capacity continuous production operations (Hoffman, 1993) and many other technical applications. (Feigenbaum, 1988)

The integration of knowledge-based systems with traditional CAD constitutes the third phase of computer-aided engineering and design. Knowledge-based systems are tools for improving productivity through knowledge. The demonstrated benefits of knowledge-based design include making expertise more widely available in the organization, documentation of design intent, improved decision-making, and enhanced productivity.

Knowledge-based design systems can help engineers and designers to achieve better designs by providing interactive design suggestions.

> Besides making CAD software easier to use, AI technology can also be implemented in such a way as to help CAD tools play a larger role as a design assistant - actually helping the engineer make design decisions. (Robinson, 1990)

Conventional CAD systems are appropriate for precision drafting but not for conceptual design or free-hand sketching. Conceptual design may be characterized as the working-out of alternate designs in geometry, and the exploration of alternate realizations of design ideas on a what-if basis. Thus CAD systems for conceptual design may not involve freehand sketching, but rather preliminary designs generated by knowledge bases from performance parameters, known by the system itself, and for which values have been input by the designer or through access to design databases.

> Computer-Aided Design (CAD) systems should support and enhance the

> product development process.... CAD systems should help [to] minimize the cognitive complexity facing the engineer.... CAD systems should 1) assist in managing design-related complexity and 2) be easy to use.... Design can be thought of as a multiple objective, constraint satisfaction problem. Typically, successful designs must achieve many diverse performance targets and satisfy competing constraints. For example, designing a gas turbine engine may involve meeting customer-specified performance criteria such as thrust, fuel consumption, cost, and weight.... [A] way of reducing task complexity is to embed design constraints in the procedures of a CAD system.... Rule-based parametric design systems allow the automatic generation of part geometry from a series of design rules and basic parameters. These design rules can include rules relating to material properties, weight and cost minimization, production process limitations, and other constraints. (Robertson, 1991b)

The parameters in a knowledge-based design approach are not just relations between geometric variables, but factors describing engineering performance, calculations, stress analyses, system behavior, manufacturing processes, and other design criteria. Knowledge-based design systems incorporate intelligence or knowledge, typically but not necessarily in the form of rules, inside a CAD system. Thus, knowledge-based design systems enhance engineering productivity by making the CAD system smarter. This is accomplished by the systems having the capability to solve more problems, to find solutions to sets of performance constraints, and to provide a higher level of assistance to the designer.

The complexities of the knowledge in the design environment, coupled with the requirement for technical and business strategy alignment, are documented in both a general manner (refer to the general technical/business alignment

discussion above) and also in a specific manner *vis-a-vis* knowledge-based systems.

> "To be considered an 'expert,' one needs a large amount of knowledge of only a relatively few varieties. In contrast, ordinary people's 'common sense' involves a much larger variety of different types of knowledge - and this requires more complicated management systems." (Minsky, 1985)

A 1991 study concluded that highly complex knowledge environments coupled with highly complex technology provide strategic business impact. (Meyer, 1991)

From the research perspective, intelligent CAD systems have several requirements. They must:

> model complex objects,
> enable consistent modifications easily,
> allow for dynamic model building and manipulation,
> incorporate deductive inference,
> reason with a large amount of data,
> provide automated consistency checking, and
> automatically convert external expressions (languages) to internal expressions.
> (Ohsuga, 1989)

Intelligence in CAD systems can come in many forms. Features such as associative dimensioning, line "snapping" in sketch mode, on-line dynamic context-sensitive help, and parametric associativity can all be thought of as intelligence. (Haase, 1992)

A rule base is one kind of knowledge base, in which knowledge is encapsulated in the form of rules expressed in a natural manner. Because the rules are in an English-like form (or that of any other spoken language), they can be examined, improved, and added to by many knowledgeable persons. Because the knowledge base is not a procedural computer programming language, the order in which the rules are generated or added is immaterial to the performance of the system.

Knowledge-based systems also include explanatory facilities, which explain to the user why or how the knowledge-based system arrives at a particular conclusion. That is, if the user asks the knowledge-based system to explain its results, the explanatory facility determines the specific knowledge (from the knowledge base), combined with the user's input data, that led to this conclusion, and displays the resulting logic or reasoning to the user. The existence of an explanatory facility distinguishes a knowledge-based system from other computer programs.

Design and manufacturing are fertile fields for the application of knowledge-based systems because many expert practitioners have reduced much of the scientific theory into simpler, heuristic representations; that is, many problems are solved on the basis of experience and judgment (which may have been built upon a strong theoretical foundation) rather than by using the fundamental, more theoretical approach. The distilled embodiment of this knowledge is, therefore, generally suitable for representation by knowledge-based systems. The natural language-like nature of the rules lends itself to knowledge base development by practicing engineers and designers, as shown by much experience in the development of rule-based systems for engineering, design, and manufacturing.

By integrating the reasoning system with the CAD system, the knowledge-based system can automatically interrogate the geometric database for any data required to satisfy its rules, and its results can be automatically inserted into the CAD database in the form of geometry, attributes, or other information.

With this integration, designers can execute knowledge-based systems interactively while in the process of design using the CAD system, retrieving any necessary information from the CAD models or from the non-geometric database. Advice provided by the knowledge-based system is displayed on the workstation screen during the computer-aided engineering and design session. The output of the knowledge-based system can also be geometric information, which is inserted directly into the computer-aided engineering and design models. In this way, knowledge-based systems can check existing designs and drawings, can change the dimensions of objects in the design model, or can produce entire drawings automatically. These interactive knowledge-based systems are virtually transparent to the CAD user.

> The system must be very easy to use so the user can actually design with it. The user should spend as little time as possible thinking about the system and as much time as possible thinking about the design problem. (Jakiela, 1989)

Knowledge-based systems have been developed to assist engineers in formulating analysis problems, developing specifications, selecting design concepts, selecting materials, selecting equipment, checking drawings, planning tool paths, and planning production schedules. Knowledge bases for interpretation and evaluation of analytical results are becoming integral parts of engineering analysis programs. Manufacturing personnel use knowledge-based

systems for manufacturability review, for manufacturing planning, for selection of procedures, for diagnosis of defects, and for many other purposes. Entire engineering design processes have been expressed in the form of rules, so that certain design steps can be made completely automatic, and the results of the knowledge-based system output through the CAD system as detail drawings ready for fabrication. Knowledge-based systems increase engineering productivity by eliminating design errors and by eliminating the engineering hours in detailed, tedious, and repetitive design steps.

Such knowledge-based systems perform specific functions under the control of responsible engineers and designers. They assist but do not replace engineers and designers in truly creative functions. Like the more familiar engineering analysis programs, knowledge-based systems can free engineers from tedious details in order to give them more time to deal with the significant and interesting issues of engineering design. Knowledge-based systems combine the advantages of standard designs (increased designer productivity and reduced design time) with the advantages of custom designs (performance optimization and accommodation of individual customer requirements).

3.3.1 Benefits of Knowledge-Based Design

The implementation of knowledge-based engineering systems provides significant advantages when a design organization is faced with any combination of the following situations:

> Experienced engineers and designers have been lost or are about to be lost due to retirements, promotions, transfers, relocations, reductions in force, or other reasons. Knowledge-based design systems can be used to capture this experience before it is lost, and integrate it with the experience of many other engineers and designers; thus making it

available to the entire engineering and design work force.

Available engineers and designers are unfamiliar with the requirements of the other engineering or downstream manufacturing processes. Knowledge-based design can capture and apply manufacturing knowledge (such as manufacturability review, tooling requirements, feasibility of manufacturing, etc.) in such a way as to make such insights available to engineers and designers throughout the design process, through the medium of the CAD system normally used for design.

Product design requires highly skilled specialists with specific and scarce know-how. Knowledge-based design systems can multiply the know-how of skilled specialists by making it available to all engineers and designers through the CAD system network.

Designs are complex, with tight manufacturing tolerances, special finishes, complex surface geometry, or complex interactions of components, subassemblies, and assemblies. Knowledge-based systems provide on-line assistance to engineers and designers, providing procedures that encapsulate the knowledge needed to achieve complex designs.

Product design requires a large, multi-disciplinary team of engineers and designers. Knowledge-based design systems make standard procedures, rules, and knowledge of all disciplines available to all design team members, wherever they are located, through the CAD systems network.

Corporate or project standards for product design are very specific and detailed. Knowledge-based systems provide automatic procedures to generate designs according to corporate standards, and can check

designs against corporate or program (project) standards for drawing release, design content, and manufacturing.

Competitive factors make it necessary to increase the productivity of engineers and designers by automating part of the design process, such as the production of details or the development of specific customer modifications to basic generic designs. Knowledge-based systems can provide substantial productivity improvements by automating design functions, based on the expertise of the most experienced designers and engineers. Knowledge-based systems can drive parametric CAD functionality, providing the ability to generate new designs automatically from changed input parameters with little or no manual intervention.

It is necessary to respond quickly to dynamic customer orders to keep business. Knowledge-based systems can generate preliminary designs from customer specifications, create drawings, calculate material and labor costs, and produce bid estimates and proposals, often in a matter of hours.

Manufacturing, machining, assembly, or other production requirements have a significant impact on the design. Knowledge-based systems can automatically check designs against manufacturing requirements and suggest designs that meet production constraints, early in the design process.

There is considerable iteration in the design process due to multi-discipline engineering or manufacturing engineering review and comment on designs. Knowledge-based design systems can eliminate or reduce this iteration by incorporating a spectrum of diverse engineering and manufacturing rules as an integral part of the design process.

Manufacturing, machining, or assembly is to be performed by automated machinery, such as numerically-controlled (NC) machines, tools or robots. Knowledge-based systems can automatically generate NC cutter paths and inputs to other numerically-controlled tools directly from the CAD models.

Production scheduling, procurement of materials or components, group technology, or reduction of work-in-process inventories are important considerations. Knowledge-based engineering systems can search databases for standard parts or components, and can generate or modify production schedules to meet changed requirements. These capabilities can incorporate group technology requirements for standard part family procedures, raw materials, etc.

Competitive factors make it necessary to reduce the engineering and production lead time in the development and introduction of new or modified products. Knowledge-based design systems can contribute significantly to reduction of lead times through efficient automated design, elimination of the need for physical mockups, and reduction of design iterations due to improved first-pass design quality.

The results of the use of knowledge-based technology to enrich the computer-based design engineering environment are visible in three dimensions, these being **lead time, product quality,** and **cost.** The general benefits of this technology are illustrated in Figure 3-3.

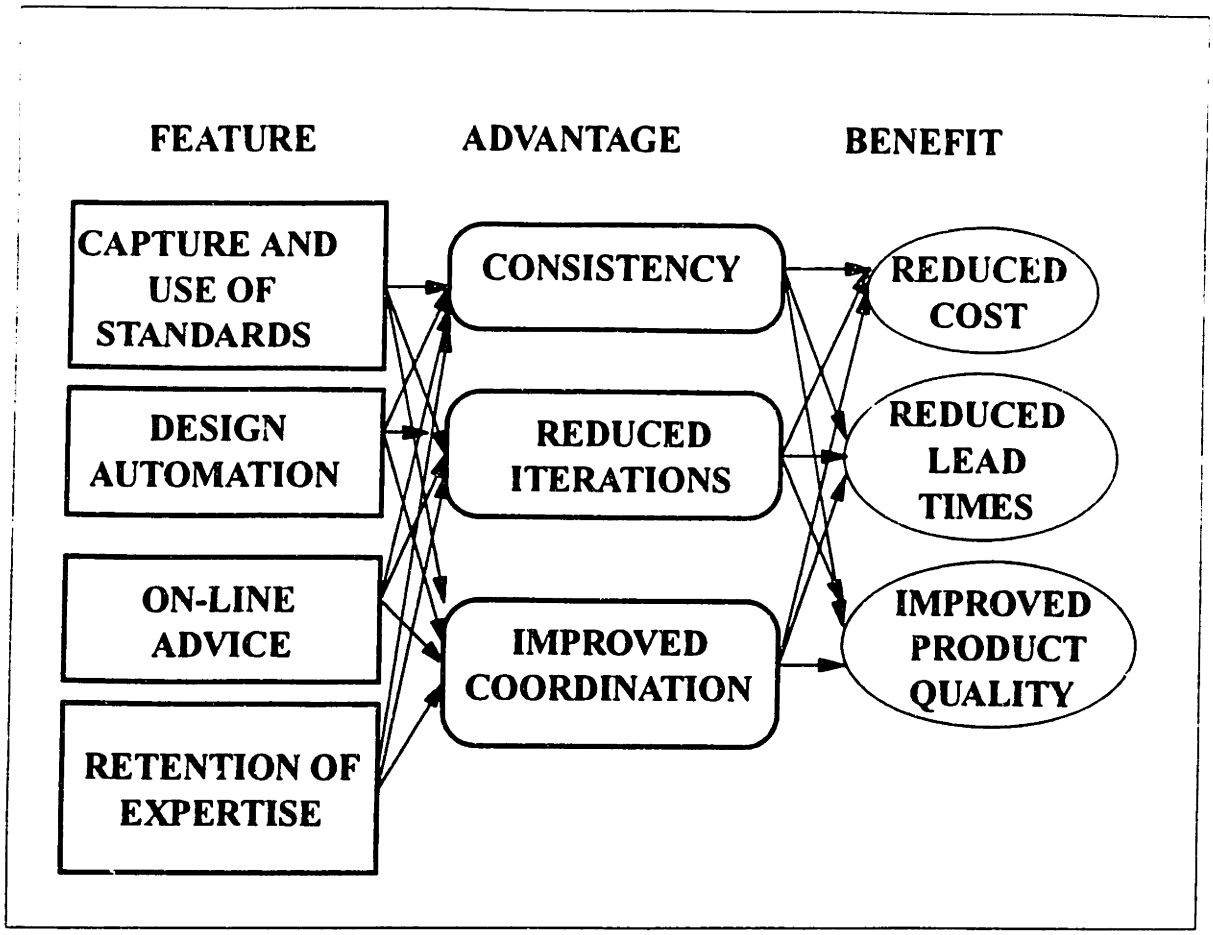


Figure 3-3: Benefits of Knowledge-Based Design on the Product Realization Process

3.3.1.1 Organizational Aspects of Knowledge-Based CAD

The third stage of computer-aided engineering and design is based on the integration of the engineering process and of the engineering product, and on the recognition that engineering organizations must be changed to remain competitive. The introduction of information technology does not by itself change organizations and processes, but enables these organizations to be changed. The strategic use of information technology, enlarged to encompass communications between designers and the coordination of the engineering process, is consistent with the trends toward shallower organizational structures and matrix management among engineering organizations.

> The effect of matrix management is to increase lateral communications within the organization as subordinates try to forestall conflicts among their several supervisors by resolving problems at lower levels of the organizations themselves. Matrix structures also give more independence and authority to subordinate levels of the organization and increase their direct responsibilities. A second reform involves the increased use of managerial teams and of parallel, as opposed to sequential or iterative, decision-making processes. This is most important in engineering where the classic procedure is to pass a decision down an engineering hierarchy, starting with product engineering, passing to process engineering where the equipment is designed, and finally to industrial engineers who design the plant layout and the initial manning tables.... The new procedures involve groups of engineers working simultaneously on all aspects of the project and interacting with each other continuously so that process and industrial engineers have a say very early, long before the product design is complete.... These new engineering procedures have been facilitated by technological developments, particularly computer aided design, which makes it possible to keep the product design fluid until relatively late in the engineering process and to simulate alternative manufacturing approaches without actually creating and testing a physical model. (Piore, 1989)

Organizational changes and the introduction of engineering technology go hand in hand, as each supports and sustains the other. If the future of computer-aided engineering and design is to enable organizational change, then it follows that organizations will be designed to exploit the advantages of the technology, and the technology must be matched to the organization.

Organizational integration, supported by information system integration, is a feature of the third stage of computer-aided engineering and design. Only a few firms have reached this stage, and those that can reach it can achieve a competitive advantage over those that have not.

> Despite all of the research on the relationship between technology implementation and organization, none of the companies [interviewed in the study] had seriously changed their organizational structure when they implemented CAD.... The high degree of specialization and fragmentation of the design process is at the heart of the difficulties U.S. companies are having exploiting the potential of CAD.... Managers often assume that CAD/CAM technology will, by itself, automatically integrate design and manufacturing. In fact, the lack of social integration is a major cause of the failure to utilize CAD/CAM's potential. The "brick wall" between design and manufacturing does not automatically come down, nor is it traversed, when computer technology is introduced. (Liker, 1992 [emphasis in the original].)

Phase 3 implementation of computer-aided engineering and design stresses integration of the multi-disciplinary engineering groups as well as the integration of design and manufacturing process, and appropriate integration of data and information systems. As systems become more integrated, the distance between engineers and designers is reduced. The distinction between engineering analysis and geometric design development is diminished as integrated databases support both functions on the same workstations and inside the same CAD/CAE system. Thus, engineers can focus on the design of products and systems, instead of concentrating on specialized functions within a hierarchical organization.

Competitive factors lead to the demand for computer information systems specifically oriented toward team design. The traditional fragmented, specialized design process leads inevitably to sub-optimization - the optimal design of subsystems rather than of the complete system. Such sub-optimized designs are not competitive with those optimized for total performance, and the global marketplace will tend to eliminate inefficient producers.

> Traditional benchmarking studies, which test the speed with which drawings can be generated and changed, have consistently found that users of CAD systems are more productive than users of drafting boards.... These individual gains did not lead to organizational gains. While drawings were produced more quickly, this merely added to the buffer of drawings between groups. Electronic transfer of design data did occur between engineering and manufacturing, but drawings were still being "thrown over the wall" and the only effect of CAD systems was to "perfect the throw." (Robertson, 1991a)

Of perhaps more importance to organizational productivity than the number of engineering hours is the length of the engineering and design process. Reduction of product lead times can be a significant competitive advantage. This is the goal of combining the technology of integrated knowledge-based CAD systems with the organizational shift to concurrent engineering.

> The design management response to the challenge of reducing product development lead time has typically been to encourage engineers to develop the product and its associated manufacturing process concurrently. This policy has two beneficial effects. First, it emphasizes the need for design engineers to be aware of production issues, and this is the focus of the popular 'design for manufacturing' approach. Second, we find that designers are sharing or transferring information to

their counterparts in manufacturing engineering much sooner than they had done previously. (Eppinger, 1990)

Mathematical models coupled with knowledge-based systems can be used to simulate manufacturing processes at the design stage; for example, for studying injection molding processes, for planning tool paths for metal cutting, and for designing dies for sheet metal forming. Automotive companies use mathematical models of plastic deformation, strain distribution, and shape retention to design metal forming dies for automobile panels without physical models or prototypes, thereby eliminating the lead time and expense of building dies that do not function properly, and eliminating excessive constraints on product design caused by conservatism in die design.

Even further, life-cycle design, in which the requirements for maintenance, repair, and modification are also simultaneously considered in the design process, is increasing in importance as competitive factors force engineers to seek higher quality products.

The integrated CAD database represents a computer model of all stages of product development, from conceptual design through engineering analysis, detailed design, material requirements planning, purchasing, and manufacturing. Whether the database resides on a single server or is distributed over a number of computers, all participants have access to the necessary data through compatible knowledge-based engineering and database management systems. The database management system also provides facilities for maintaining the security and integrity of all data in the system.

The implementation of knowledge-based engineering systems requires a reduction in paper drawings as the master design documents. Paper drawings are useful as media for human communications, but not for computer

communications. Even drawings scanned into computer imaging systems are useless for this purpose, as these raster images are not intelligent and cannot be understood by knowledge-based engineering systems or other concurrent engineering applications. The CAD database must become the reference design and the basis for materials procurement, manufacturing resource planning, and manufacturing process design.

Computer-aided design systems preserve the design electronically, but knowledge-based engineering systems also preserve design intent. That is, the knowledge-based engineering system can retain the data and the rules that were used to generate a design, in case the intent behind the design must be examined later, and can regenerate a design using these rules with the same or different data.

To create drawings when needed, the three-dimensional computer models can be sliced and compressed into two-dimensional projections for final drafting. This extraction and projection onto two-dimensional drawings can in many cases be accomplished or assisted by knowledge-based engineering systems, which contain the knowledge needed to create standard two-dimensional views from the three-dimensional database. Design changes must of course be made only on the three-dimensional model, rather than on the two-dimensional drawings, to preserve the integrity of the database; new drawings can be regenerated as necessary.

In addition, graphical design information can be distributed to engineers, managers, and manufacturing personnel who are not familiar with the use of CAD systems, in order to view and mark up models and drawings. This is accomplished with software systems that have reduced geometric functionality and may also be linked to the database of design and manufacturing information. These developments increase the value of the design data and

extend the usability of the CAD models, by making design data accessible throughout the engineering organization and at designated locations on the plant floor.

Three-dimensional design models can also be closely linked to engineering application programs for stress analysis, vibration analysis, etc. By integration of the data, the design basis and the engineering analyses are kept consistent. Designs developed in this way are of higher quality, less information is lost in the interfaces between specialized engineering functions, and engineers have greater job satisfaction because tedious transcription of data is eliminated.

3.3.2 The technology of knowledge-based design

A knowledge-based design system is a computer system that integrates reasoning, heuristics, calculations, and geometry to solve problems that would typically require an expert engineer, designer, or manager to play a substantial role in the problem-solving process. This section provides a brief overview of some of the basic technologies involved in knowledge-based design systems.

A knowledge-based design system consists of two primary parts: domain knowledge and a mechanism for reasoning with and interpreting this knowledge. (Maus, 1991) The knowledge base consists of two parts: methods, functions, heuristics, or rules that represent the knowledge to solve problems in a particular domain, and facts about the specific problem to be solved. The objective of knowledge-based design systems development is to capture the knowledge and expertise in a particular area, to represent this knowledge in an electronic format, and to transfer and distribute it to others.

A knowledge-based system contains all of the following elements, the basic structure of which is depicted in Figure 3-4:

a reasoning engine,
a knowledge base separate from the reasoning engine,
a knowledge capture and development environment,
a context database, and
an end-user interface.

The reasoning engine draws logical conclusions, or inferences, from the expert knowledge contained in the knowledge base and the specific problem conditions contained in the context database. One of the discoveries that made

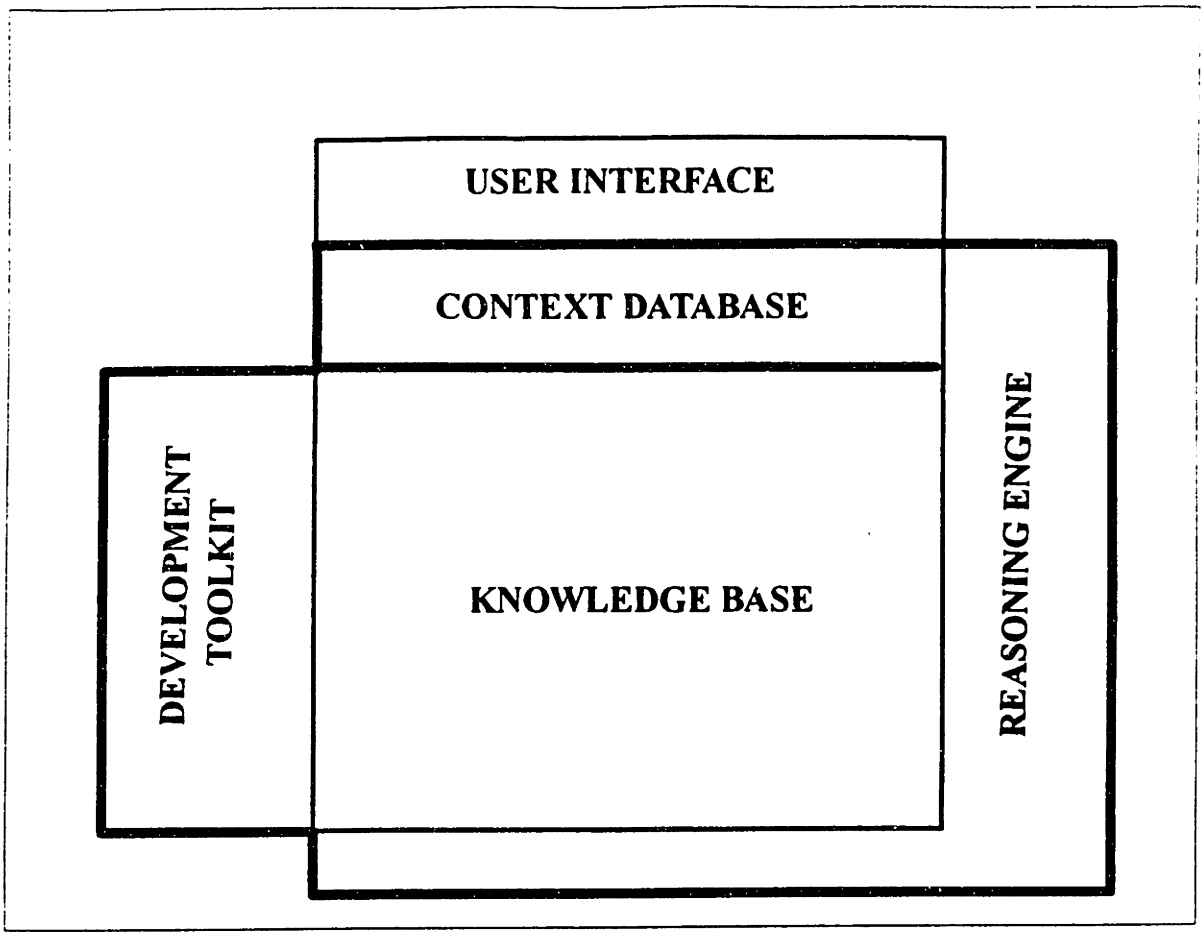


Figure 3-4: Elements of a Knowledge-Based System

knowledge-based design systems useful was the realization that, by separating the reasoning process from the knowledge base, the reasoning engine may become a generic tool, used in any number of knowledge-based design systems.

The knowledge base is a computer representation of the knowledge of the domain expertise, expressed in one of several forms, for example, as rules. A knowledge base is specific to each expert system application.

The context database is a set of data that define a specific scenario or case, pertinent to a particular problem, that is used by the reasoning engine to reach a conclusion. These data may be obtained from the user of the expert system,

from a computer database, from a CAD system, or from the output of another computer program.

The user interface is the means by which the end-users communicate with the system. In the case of integrated knowledge-based design systems, the most effective user end-user interface is one which is seamlessly integrated into the CAD system's environment, allowing designers and engineers to access the knowledge-based system in the same manner as they would any other CAD system function. Knowledge-based design systems also include explanatory facilities, which explain to the user why or how the expert system has arrived at a particular conclusion. That is, if the user asks the knowledge-based design system to explain its results, the explanatory facility determines the specific knowledge (from the knowledge base), combined with the user's input data, that led to this conclusion, and displays these to the user. The explanatory facility makes the knowledge-based design system easier to use, easier to understand, and easier to accept than conventional computer programs. The existence of an explanatory facility is one of the features that distinguishes a knowledge-based system from any other computer program.

The knowledge capture and development environment is an intelligent knowledge representation and editing system, which provides extensive utilities for creating and editing knowledge-bases in an efficient manner. This would typically include context checking, consistency evaluation, detection of circular reasoning, and tools for mapping objects together.

A "rule base" is the most common kind of knowledge base. In a rule base, the domain expertise is expressed in the form of production rules, each of which consists of a set of conditions and a set of consequents. These production rules are commonly expressed in a form as close as possible to natural

language (e.g., English). A typical rule is of the form:

IF <Set of Conditions is true>
THEN <Set of Conclusions can be drawn, including actions>
ELSE <Alternate Set of Conclusions can be drawn including actions>

If the conditional statements are all evaluated to be true, then the rule is said to fire, and the consequent statements are asserted to be true as well. The consequents of one rule may be used as the conditions of other rules. This produces a chain reaction effect, as the firing of one rule leads to the firing of others. This chain reaction is carried out by the reasoning engine.

The chain reaction behavior of rule-based systems is of three types: forward chaining, backward chaining, and mixed. In forward chaining, or data-driven, systems, the reasoning engine uses the given data (from the user or from the database) and determines whether any rules can fire. If so, the chaining process continues, until one or more conclusions, or terminal consequents, are reached, or no more rules can fire. If rule-firing stops before a conclusion is reached, the inference engine asks the user for more information, that may cause new rules to fire. This approach is generally applicable for design problems.

In backward chaining, or goal-driven, systems, the inference engine posits a possible conclusion, and then chains backward through the rules from this conclusion to the data. If the conditions required to sustain the hypothesis are incompatible with the actual data, then the conclusion is rejected and a new conclusion is hypothesized. This continues until one or more conclusions have been validated by the data or all possible hypotheses have been rejected. This approach is appropriate for configuration problems (from a known set of

possible configurations, pick one that best meets the input data) or diagnostic problems (hypothesize a cause for the symptoms, and attempt to prove or disprove the hypothesis). Mixed chaining is a composite of forward chaining and backward chaining.

Rules can be stated in the rather simple format illustrated above, or they can be more complex. For example, a rule may have several conditional tests, each nested within the context of the previous condition's evaluation. One such case may have the following form:

```
IF          <Condition A is true>
      OR    <Condition B is true AND Condition C is true>
THEN        <Conclusion X can be drawn>
ELSEIF      <Condition A is false AND Condition B is true>
      AND   <Condition D is true>
      THEN  <Conclusion Y can be drawn>
ELSE        <Conclusion Z can be drawn>
```

Not all knowledge-based design systems are based on rules. Advanced systems support rule and object-oriented reasoning. In an object-oriented system, data (which many include CAD data) and procedures (or methods) are grouped together into logical units or chunks. Thus objects include not only data, but the procedures that can be performed on these data.

A class is an abstract data type for a generic class of objects, which may be standard parts, components, or assemblies. An example of a class might be a fastener, which includes a bolt, a nut, and washers. Subclasses of this class would be the specific types of fasteners, such as bolt, rivet, screw, etc., as illustrated in Figure 3-5.

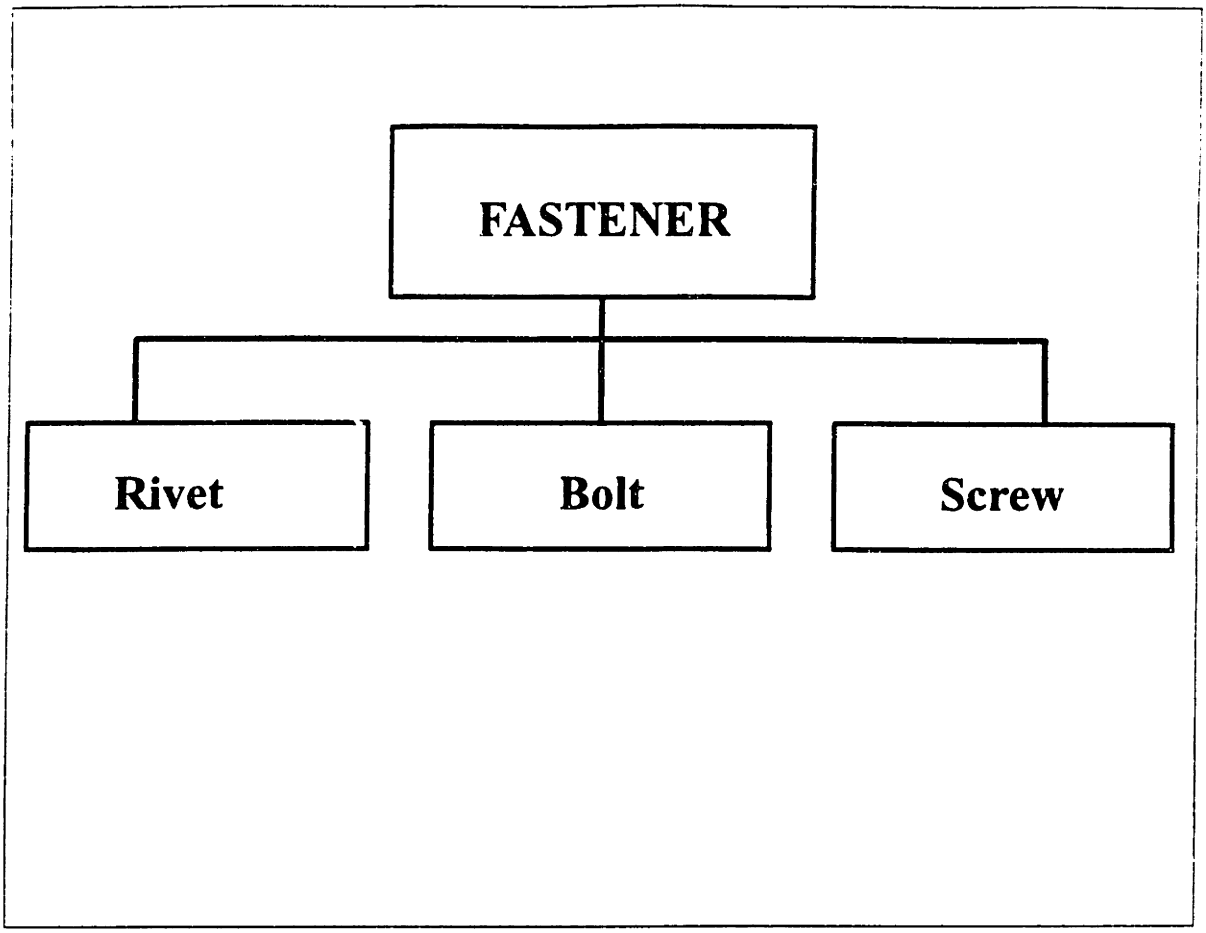


Figure 3-5: Class Hierarchy for Fasteners

Each class object contains properties defined by slots of the class. Properties of the bolt class might include material, diameter, length, and type name.

Characteristic of object-oriented reasoning is the concept of inheritance: each child object can inherit properties from its parent class. Thus the subclasses rivet, bolt, and screw would inherit all of the properties of their parent classes, such as material, diameter, length, etc. Conversely, the length of the thread of the bolt might be a property of the bolt that is not inherited from the parent class, since not all fasteners have threads. This inheritance is illustrated in Figure 3-6.

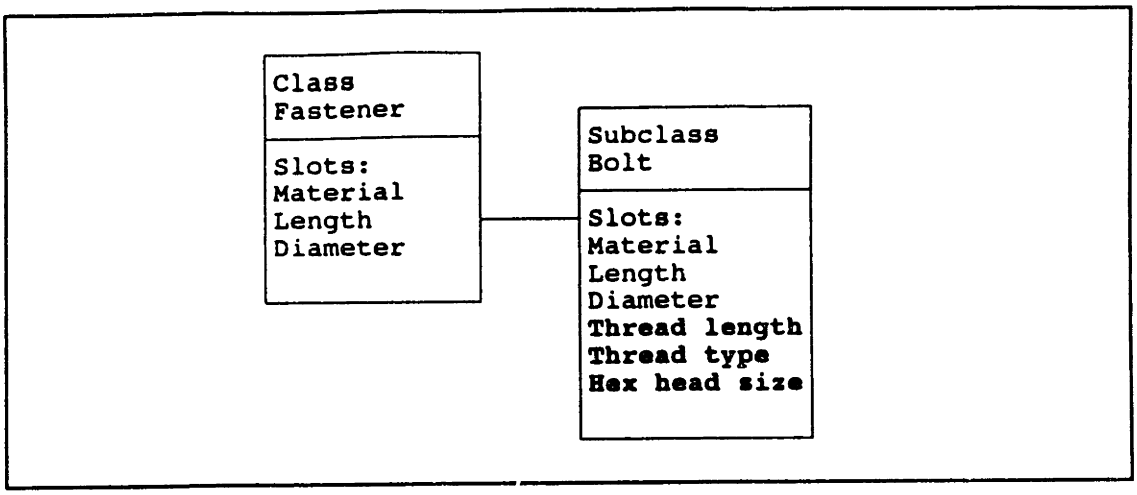


Figure 3-6: Class Inheritance

Values may be set by default in a parent class, and then would be assigned by inheritance to the sub-class. These values may be over-ridden by other values assigned by a rule, by the object's methods or by an external source, such as the user. Value assignments are illustrated in Figure 3-7.

Since a class is a *generic* description of a type of object, and a sub-class is defined as a *type-of* its parent class (e.g. a bolt is a type of fastener), the system thus has the capability of describing general representations of physical entities or other types of objects. In order to describe a *specific* object, a concept known as **instantiation** is used. When a specific element is created within the knowledge-base, as opposed to a type of element, this specific object is known as an instance of a class or sub-class. Instances can be viewed as specific references to the template described by the class, with values assigned to the slots. For example, a specific bolt mat have a part number, a specific length, a specific thread type, etc.

The relationship between the **specific bolt** and the subclass "**bolt**" is not the same as the relationship between a subclass and a class. As described above, the relationship between a subclass and a class is a *type-of*, or a *kind-of*

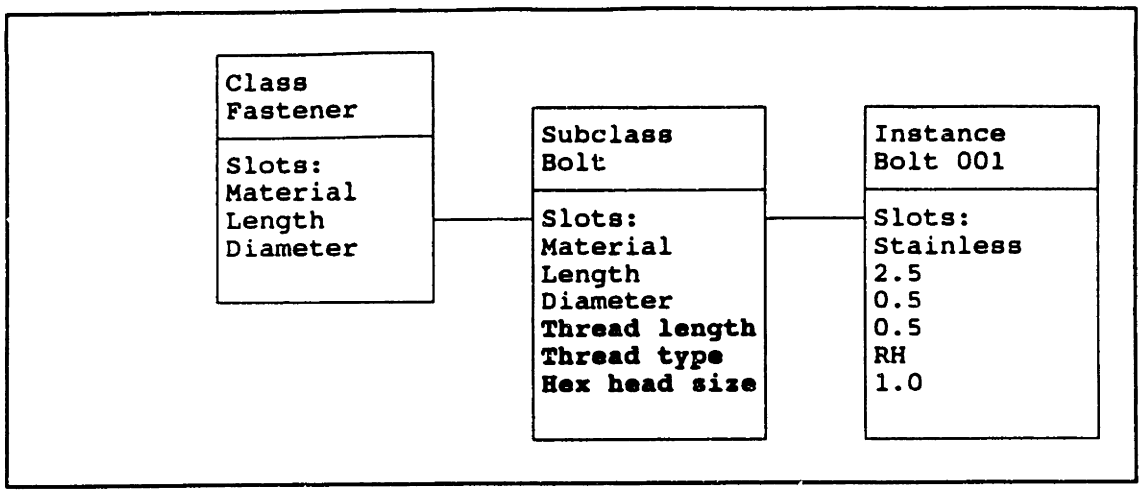


Figure 3-7: Illustration of Value Assignment in Instances

relationship. (A bolt is a *kind-of* fastener.) The relationship between an instance and its template class is known as an *is-a* relationship. (Bolt 001 *is a* bolt, which is a *kind of* fastener.)

Another powerful object-oriented technique is to represent complex objects as collections of simple objects. For example, an object such as a clamp might contain several distinct components, including a fastener. In this instance, one of the slots within the "clamp" class would be a fastener slot. This would be defined as a *reference* to an instance of the class "fastener". So, an instance of the "clamp" class, Clamp 189 may, in fact, refer to Bolt 001 as its fastener. This type of relationship is known as a *part-of* relationship, because the referenced instance is a part of the assembly, or complex part, "clamp." Such a relationship is illustrated in Figure 3-8.

Object-oriented data structures differ from other data structures (relational, etc.) in that they can include *procedures or methods* for operating on the data pertaining to the object. For example, the diameter of the threaded fastener could be determined by a method that uses the material strength, the desired safety factor, the thickness of the fastened structure or object, and the applied

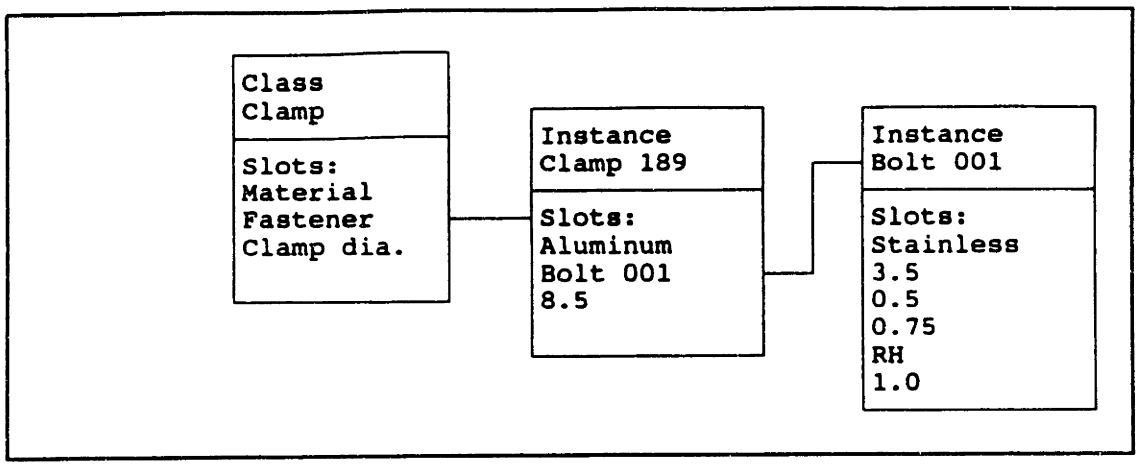


Figure 3-8: Relationships Across Classes - the *Part-Of* Relationship

load to determine the standard bolt diameter. In an object-oriented system, information is transmitted between instances or classes by message passing; in this example a message would pass the applied load to the threaded fastener instance in question.

The object-oriented methodology for representing knowledge about things directly within the object itself (intrinsic knowledge representation) differs philosophically from the representation of knowledge extrinsically, as in the case of rules. Rules act as independent representation mechanisms, operating on a variety of entities (even across objects.) These two paradigms are not mutually exclusive; they are often used in concert within a knowledge-based system) but the representation schemes do differ significantly.

By encapsulating data and methods pertaining to an object, other objects are isolated from changes that might occur with that object. That is, whenever any changes occur to a property of an object, such as by passing a message giving a new value for the applied load, the object's methods recalculate the properties of that object. No other objects are recalculated, unless a message is passed to them. Also, any changes that are made to the procedures themselves, such as a different method for computing bolt diameter, are

confined solely to that object. Thus changes are localized and the maintenance costs of the computer system are greatly reduced.

Object classes are useful in creating part libraries, such as threaded fasteners in standard sizes and combinations. Parts generated by the computer-aided design system can be cataloged into libraries and subsequently selected by the knowledge-based system, thus reducing the proliferation of components and unique component designs.

As higher level knowledge-based systems permit the rules in the knowledge base to be expressed in English, new rules may be added over a period of time by the original knowledge base developer or by others. This evolutionary improvement is characteristic of knowledge-based design systems, as compared to other types of computer programs. If many experts contribute their expertise to a knowledge base, the system over time may become more knowledgeable and capable of problem-solving than any one of the experts who contributed to its development. This is particularly true of knowledge-based design systems that make manufacturing knowledge available to designers through CAD systems: the rule base may become a synthesis of manufacturing know-how beyond the capabilities of any single person.

3.3.3 Integration Architecture for CAD and Knowledge-Based Systems

Many commercial CAD systems provide a programmatic means to access the geometry and model attribute data through an Application Programming Interface (API). This API functionality is the mechanism through which knowledge-based systems integrate with the CAD systems. There are two basic integration paths, these being (i) the user interface and (ii) the geometric database.

With respect to the user interface, the most optimal integration architecture is for the knowledge-based system to be called from within the CAD system just as any native CAD function would be called, using the same protocols, user dialogues, and function key menus. This is possible with many commercially available CAD systems. Whenever a user dialogue is requested by the external application (in this case, the knowledge-based system) a panel or screen is generated in the CAD system's vernacular and look-and-feel, and the appropriate user interaction is generated.

In regards to the geometric model access, a direct integration with the CAD geometry is optimal, in which a single geometry engine (the CAD system) is used to represent and manipulate all geometric data. Such an architecture is illustrated in Figure 3-9. The use of more than one geometry engine poses potential translation problems, leading to accuracy and tolerance errors that could invalidate the entire geometric database. As such, a direct integration with the CAD geometry eliminates this potential source of errors. In order to accomplish the most direct integration possible, the CAD system's API functions are embedded within the knowledge-based system. One mechanism is to use integration software that provides transparency between the applications (CAD and knowledge-based system.) [Reinschmidt, 1994] Such integration software provides transparent access to the callable functions of the CAD system that create and analyze geometry, in order to relieve the knowledge base developer from the necessity of knowing all these functions (there may be hundreds of such callable functions in advanced CAD systems). This integration software consists of a set of class libraries that embed the CAD system's API calls within the knowledge-base objects. To create or read geometry, the knowledge base developer simply refers to a library of geometric elements available within the Knowledge-Based Systems engine. This library contains primitive parametric and associative geometric elements (spheres, cylinders, cones, cubes, etc.) and complex parametric associative geometric

objects made up of multiple primitives. This library can be continually expanded. When a library object is instantiated by the Knowledge-Based Systems engine (i.e., a parametric object is selected and all its parameters defined), the integration software relates the object in the application to an object in the CAD model through inherited methods that utilize the callable geometric functions of the CAD system. Instances can be created and reasoned with in the Knowledge-Based Systems application, and upon initiation of the appropriate method, the link to the CAD model is made. This link can be bi-directional, since the Knowledge-Based Systems application has the ability to record information in application elements in the CAD model which refer to Knowledge-Based Systems application instances.

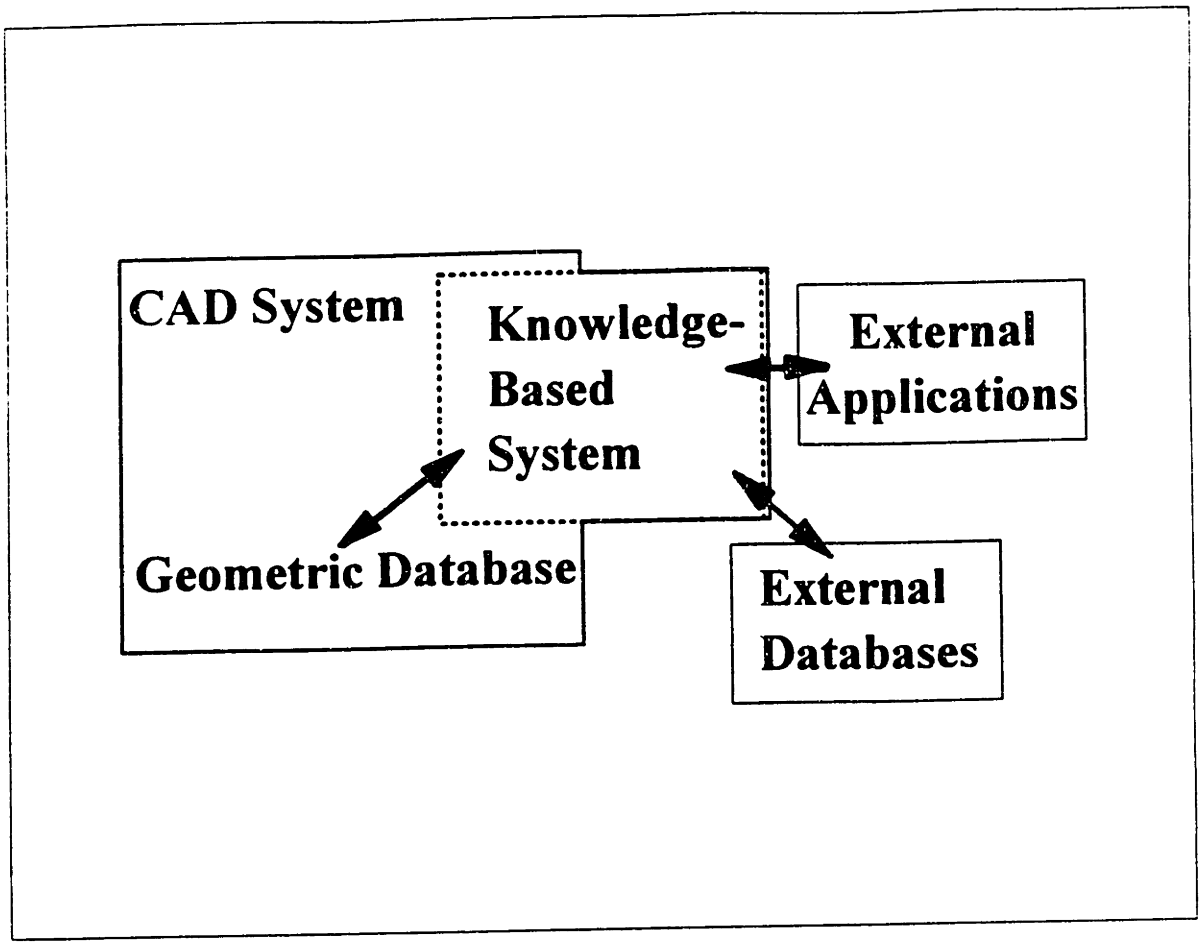


Figure 3-9: Integration Architecture for Knowledge-Based Systems and CAD Systems (After Reinschmidt, 1994)

Prior to the mid-1980's, it was generally believed that the development and use of knowledge-based design systems required specialized computer languages, such as LISP or PROLOG, and specialized computer hardware, such as LISP machines. (Keyes, 1990) With the development of commercial expert system shells written in the industry standard C language, this position changed. LISP-based systems consume larger amounts of memory, are inferior in performance, require much more skill and computer expertise to develop, and are generally obsolete.

In an open, integrated system, it must be possible for the user to temporarily exit from the CAD system to execute an application program, a query to the database, or a knowledge base, and then to return to the CAD system at the point from which he left. The user wishes to see an intelligent CAD system, not the interface between an CAD system and an external program; therefore, the execution of the knowledge-based system from within the CAD session must be transparent and seamless.

If pertinent data are stored in a database management system, then the interface to this system must be intimate and interactive. That is, the designer at a CAD/CAM workstation must be able to interactively insert and retrieve data from the database without delays, and a user of the database must have immediate access to the geometric information. Conversely, the inclusion of attribute information in a proprietary CAD database does not support the integrated database concept, as such information is available only through the CAD system. In the integrated database, any user, not only CAD users, must have the ability to access all data to which he is entitled.

One of the most commonly available database models is the relational data model. The relational data model provides flexibility, as this data structure is easily modified and expanded, for all participants in the project to develop new uses for the data. A more modern approach is that of the object-oriented database structure. Object-oriented database systems allow for encapsulation of both the static and dynamic properties of information, in that they allow for methods and functions to be associated with static data, and they incorporate properties such as inheritance and persistency.

References to the database must be processed very rapidly, in order to provide acceptable response time to the users. Large memory capacity and high processing speed are essential to the operation of the database.

In order to find the data necessary to satisfy the conditions of the rule base, the knowledge-based system must have access to more than graphical design data. Specifically, some CAD systems' databases store graphical elements in the form of lines and points, relating the geometry of the design rather than the engineering information. In order to accommodate engineering data relating to the geometrical elements, it is most convenient to use a relational database structure. By assigning attributes to the graphical objects, the relational database can contain almost any type of information. In this scheme, database management functions (such as searches) can be applied to the database, allowing the system user (or knowledge base) to obtain necessary information or to make changes easily.

In the interactive mode, a run-time module of the inference engine and knowledge base is executed from within the CAD system. The designer, using the CAD system, may decide to invoke a knowledge base, or the knowledge base may be automatically invoked as a consequence of some action or decision of the designer. The run-time inference engine executes the knowledge base; forward or backward chaining may be used in these applications. To fulfill the requirements (conditions) of the rules, functions, or methods, the inference process requires data. These data are obtained in the following default sourcing sequence:

If the required data can be inferred or determined by the inference process, then this will be the method for satisfying the conditions;

If the required data are contained in the CAD model (or another database) then these data are used to satisfy the conditions;

If all the required data cannot be determined by the inference process, and they are not contained in the CAD or other databases, then the

system asks the user for the required data.

If all the data needed to satisfy the rule base and to reach a conclusion are available to the inference engine from either the CAD model or the database, then no questions will be asked of the designer, who simply sees the resulting advice or suggestions displayed on the screen. If all the data necessary to reach a conclusion are not available from the CAD model or the relational database, then the user sees questions displayed on the CAD screen. This process continues until a conclusion is reached.

3.3.4 Applications of Knowledge-based Design

A variety of knowledge-based design applications have been developed both in the research community and in the practical industrial environment. These applications typically fall into the following categories:

Automated Part, Tool, Assembly, or Product Design
Design Advisors
Design Reviewers
Manufacturing

3.3.4.1 Automated Design

By extension of the interactive use of knowledge-based design systems discussed earlier, entire engineering design processes for specific domains of part, sub-assemblies, tools, and other applications can be expressed in the form of well-specified method, procedures, or rules, so that certain design steps can be made completely automatic. In this way, knowledge-based design systems

can be used for automated design, or the automatic generation of complete designs from a set of parameters input by the designer or obtained from available data sources.

This type of knowledge-based system, for automatic design, increases engineering productivity by eliminating design errors and by eliminating the hours required for this design step completely.

Examples of this type of systems include:

Plastic Injection Mold Design. The process of designing molds for plastic parts involved the definition of two halves of a mold insert, which are machined to the inverse geometry shape of the part itself. The molten plastic is injected into the void, and the part made. The mold design process is typically performed by different individuals from the part designer, and involved significant consultation with the part designer in order to determine appropriate gate locations, ejector pin locations, aesthetic requirements of the part, etc. (Gammons, 1992) This process also involved a great deal of routine CAD operations, and has been a fertile area for application of automation technologies through the implementation of knowledge-based systems. This has resulted in lead time reductions from eighteen weeks to twelve weeks, or approximately 33% reductions. (Cinquegrana, 1990)

Tool Design. The design of tooling for machined parts in aircraft can present significant problems in the generation of tools that adequately meet the process and structural requirements of the part. As such, knowledge-based design systems have been developed for automating the generation of tool designs based on part geometry and manufacturing process knowledge. (Hood, 1994) Such systems can

operate on the basis of Feature Recognition (Dong, 1993) or on the basis of generative tool design knowledge. Other examples include tools for complex components, such as aircraft engine stator vanes. Knowledge-based design systems for automating such tool designs use the surface shape defined in the part geometric model, and knowledge of tooling processes and material behavior. (Field, 1992)

Hydraulic and Electrical Systems Design. The detailed design of hydraulic and electrical components involves knowledge of the functionality of the components and systems, the geometric constraints, the manufacturing processes, and the functional requirements. The use of knowledge-based systems for automating detailed component design and integrated systems design has produced lead-time reductions of up to three months on aircraft delivery schedules, and labor savings of between fifty and seventy percent. (von Hardenberg, 1995) The use of knowledge-based systems for electrical systems design and integration has resulted in savings of seventy-seven percent in the engineering design cycle for the electrical wiring portion of the V-22 Tilt-Rotor Aircraft. (Springer, 1995)

Automatic Drawing Production. Knowledge-based design systems can be used for the generation of two-dimensional engineering drawings from three-dimensional design models. In this case, the knowledge base represents the rules for generation of standard drawings, such as cross-sections or isometrics, from three-dimensional models. Given these rules, the two-dimensional projections or cross-sections can be created by the CAD system, and the appropriate notes and other annotation added. In one application, fire wall drawings and details have ben automated through the implementation of a knowledge-based design system for architectural applications. This application reduced the amount of time required for detailed drawing development from one

hundred and sixty hours to four-and-one-half hours, a significant savings. (Fisher, 1986) In other applications, these types of systems have been used to increase accuracy and decrease costs of producing drawings for parts designed in three-dimensional CAD systems. (von Hardenberg, 1995)

3.3.4.2 Design Advisors

By providing the capability to interact with the CAD user and the electronic systems themselves, knowledge-based systems can be used in a consultative manner. These consultations can provide advice about part selection, configuration, integration effects, or downstream design or manufacturing process implications of design decisions.

Design For Manufacturing and Assembly. Knowledge-based systems that provide design engineers with the capability to understand how the parts that they are designing will be made and assembled can have significant positive effects on the lead times of engineering projects.

> " ... drawings are then passed to manufacturing and assembly engineers whose job it is to optimize the processes used to produce the final product. Frequently, it is at this stage that manufacturing and assembly problems are encountered and requests are made for design changes. Sometimes these design changes are large and result in considerable delays in the final product release. In addition, the later in the product design and development cycle the changes occur, the more expensive they become. Therefore, not only is it important to take manufacture and assembly into account during product design but also, these

considerations must occur as early as possible in the design cycle." (Boothroyd, 1994)

Such systems have been developed in a variety of industries, including the automotive industry. One example (transcending the assembly issue and entering the realm of operations and service) is the development of a system for analyzing the slide path of door glass in automobiles. This problem is a complex dynamics problem due to the compound curvature of the glass and the door, the numerous parts in the door (including regulation-required side impact panels and electronics components), and variable part-to-part interference requirements. The development of an acceptable glass drop path is complex, and often problems do not arise (or become visible) until the manufacturing and assembly stage. Knowledge-based systems for advising design engineers on an appropriate path have significantly reduced assembly- and service-induced design changes, and have provided first-pass design solutions for vehicle designers, eliminating weeks from the automobile development cycle. (Clark, 1994)

Standard Parts Selection. The use of standard parts and tools can be a significant cost factor with respect to high-frequency parts in complex products (such as aircraft, power plants, and automobiles.) By reducing the number of parts, and standardizing on a pre-defined list of acceptable parts, dramatic cost reductions can be realized. For example, in the automotive development process, reductions in the number of fasteners used can result in savings of millions of dollars. Chrysler reports that it uses over four billion fasteners in a given calendar year. Since these parts are priced according to volume of procurement, the more of a given part that is ordered, the lower its cost will be. Since each automobile contains thousands of fasteners, fastener standardization can lead to

significant reductions in redundant fasteners, and hence cost savings. (Clark, 1994)

3.3.4.3 Design Review

The process of ensuring that designs meet standards, are manufacturable, and meet the functional requirements involves a review of the design by a person or system other than the designers themselves. This review process is intended to eliminate errors, but the checking process itself can be fraught with errors. This is due to the tedious nature of the process, the degree of detail that must be addressed if errors are to be eliminated, and the degree of difficulty in detecting errors.

Knowledge-based systems for providing automated design review can reduce the amount of time that is taken in checking, and can improve the quality and accuracy of the checking process. A general framework for such systems includes an automated design checking systems which either feeds information back to the design engineer, corrects the problems automatically, or releases the acceptable designs to the following design steps. (Reinschmidt, 1992)

Examples of such systems include:

Parts Lists Checking. A major source of errors in the design process is the incorrect transcription of parts numbers and data from CAD systems to other electronic systems and manual systems. This type of error can account for up to seventy percent of all non-contextual design changes in a given engineering operation. Knowledge-based systems to check the accuracy of such cross-system parts number indexes have been

found to eliminate such errors. (von Hardenberg, 1995)

CAD Model Standards Checking. Although a non-value-added process, it is necessary in large design organizations to check the consistency of CAD models and drawings with respect to organizational standards, since release control and downstream applications depend on the enforcement of such standards. While this process is not intellectually challenging, it can be significant in terms of costs and cycle times. An automotive company reports a sixty-to-one reduction in time allocated for this activity due to the implementation of a knowledge-based design system for model checking, and a direct cost savings to the vehicle development program. (Schultz, 1993) This type of system can provide cycle time reductions and direct cost savings in a variety of industries and settings. (Helfner, 1995)

3.3.3.4 Manufacturing

In addition to contributing to the implementation of concurrent engineering, knowledge-based design systems can be directly significant in the manufacturing planning and execution phases.

Manufacturing Process Planning. The development of process plans for part manufacture typically involves the application of manufacturing knowledge to the part design and the development of manufacturing sequences, parameters, machine selection, and quality control development. This process depends on experience and knowledge of manufacturing method applicability to different part types, as well as experience in determining limits or boundaries of manufacturing process

applicability. Knowledge-based design systems that automatically generate process plans provide the advantage of standardized process developments, as well as reductions in time and planning errors. (Marefat, 1992) Other systems illustrate how feature extraction and modular knowledge base development can provide solutions to the dynamics of the process planning problem. (Hummel, 1988)

Cost Estimation. Cost estimation for parts is typically a highly specialized task, performed by domain experts of the classical variety. This task is typically accomplished late in the development cycle, and has, therefore, not traditionally played a major role in the design decision methodology. The development of knowledge-based design systems specifically oriented toward the early development of cost estimates, and for the purpose of providing decision-support to engineering designers has proved to have a significant effect on the overall product development effort. (Crowfoot, 1992)

Quality Control. The development of methods for classifying parts according to quality control criteria has proved to be a major positive contributing element in the integration of design and manufacturing systems. By providing automated mechanisms for examining part geometry, materials, and performance characteristics and requirements, it is possible to improve the quality control process by an *a priori* understanding of significant quality control parameters, and a subsequent higher degree of accuracy in quality control. (Halgamuge, 1993)

3.4 Discussion

Contrary to previous periods in which technical functionality drove developments in computer-aided design, competitive forces are proving to be the critical factor in defining the role of information systems in the process of engineering, design, and manufacturing. It is necessary to align technical and business strategies in such a manner as to obscure the difference, so that competitive position is driven by technical capability, and technical capability is driven by competitive demands.

Engineering job functions are being enriched, as the specialization of functions and the division of labor will decline in the face of improved computer-aided engineering and design technology. Instead of performing one engineering analysis or design operation over and over again, engineers use computer technology to effectively design entire systems, through the integration of three-dimensional computer-aided engineering and design, databases, engineering analysis applications, and knowledge-based systems.

As detailed specialization declines, knowledge-based systems will take up many of the routine functions of design detailing, reviewing designs, checking designs, and providing on-line design advice and assistance to engineers at their workstations.

Three-dimensional computer-aided engineering and design is being linked to computer simulation of manufacturing operations, using dynamic simulations or network scheduling methods, so that an assessment of the cost and time to manufacture is fed back into the design process. Engineers are learning more about the process of manufacturing.

The iterative engineering process is being rationalized, through the study of design prerequisites, so that the design process can proceed in the most efficient manner, with the minimum of iteration to resolve design problems.

Computer-aided design software is increasingly oriented toward engineering and design by teams, rather than only by individuals. Communications, integration, design release, design change control, and the consistency and integrity of design databases assume more importance in the development of engineering software.

With developments in knowledge-based design occurring at a ever-increasing rate, these systems will have an impact on the engineering profession at least as profound as the impact of the advances in analytical methods and computer-aided drafting of previous years. New technologies for reasoning are providing insights into future knowledge-based capabilities, such as Genetic Algorithm-Based reasoning and design mechanisms for structural design. (Chapman, 1994) Whereas previous computer innovations in engineering stressed stand-alone applications, current trends in information technology stress integration and communication between the various design professionals. The communication and integration of product engineering data, experience, and heuristics are likely to be major factors in the efforts to improve productivity and to gain a competitive advantage in the world engineering market. Improved knowledge-based design methods, by increasing the productivity of design professionals, by improving the quality of designs, and by improving the manufacturability of designs, may well strengthen the firms that can take advantage of these trends.

Much of what is considered to be expertise is actually experience distilled in the form of heuristics. Experts are those people who have accumulated more knowledge, in certain specific areas of interest, than have non-experts. These

heuristics can be coded into knowledge-based design systems that embody the superior problem-solving capabilities of the human experts.

The value of knowledge-based design systems lies not in the replacement of experts by computer programs. The value of knowledge-based design systems lies in their ability to replicate the performance of experts on judgmental tasks in situations in which there are not enough human experts, because they are retired, sick, on vacation, in another location, or otherwise not available.

In the proper situations, knowledge-based design systems can improve productivity, reduce errors, and improve quality. Many knowledge-based system applications in manufacturing have proven to be valuable, but these have barely scratched the surface of the possibilities in this field.

In summary, this engineering software technology enables increasing opportunities for engineers who understand knowledge-based design technology to expand their roles in the overall organizational mission, and allows the organization as a whole to profit from the increased capabilities of the knowledge-based computer-aided design infrastructure.

4. Event-Driven Knowledge-Based Design

Design information systems consist of programs and data that play a variety of roles in the design process. In general, these systems are referred to as Computer-Aided-Design or Computer-Aided-Engineering programs, because this type of system is activated and executed as a result of specific, conscious initiation by people. It has been shown in previous chapters that regardless of the type of system, the software systems play the part of an aide, in the sense that the task that they perform is encapsulated in a well-defined scope and bounded by the actions of the human designer or engineer. This concept is depicted in Figure 4-1.

Rather than maintain the premise that all design systems must be activated in response to a person's commands, the *event-driven knowledge-based design* concept introduces the idea of integrated dynamic knowledge-based design systems built into the design process. Fundamental to the concept of event-driven knowledge-based design is the notion of automated initiation of knowledge-based systems based on the occurrence of certain events, either internal or external to the design process. Rather than being executed by the initiation of a conscious act of a person, event-driven knowledge-based systems are stimulated by computer programs and data that cause action independent of direct human control. The knowledge-based design concept, described in the previous chapter, is thus enhanced and expanded, because of the availability of systems that are an integral part of the dynamics of the design process.

Why is it necessary to implement an event-driven architecture? Is it not sufficient to electronically store data, and to build applications that use those data in a prescribed fashion? It is suggested here that the design environment is fluid - that it is entirely dynamic, both from a data perspective and from the perspective of the boundaries and assumptions that form the external

environment. Traditional systems depend on the static nature of data and processes - they are constructed based on the assumption that the progression of the design process will follow a prescribed path.

> "The design process is difficult to support, because it is not a simple sequence of steps, but a dynamically changing, multi-dimensional, self-regulating system which must coordinate the individual and common goals of the electronic product enterprise. Everything about the process is constantly changing; the managers and engineers come and go, the tools change, the design team is re-structured, the computing environment ages and evolves. In spite of the constant change, the design itself must somehow continue to grow in functionality and consonance.
>
> In the face of such dynamics, a CAD framework must be able to monitor and react to activities in the design environment. Not all activities can be monitored, but those activities conducted by software can generate communications which signal a change in the state of the design or design process." (Allen, 1992)

Event-driven knowledge-based design provides an architecture for the control and execution of design processes. This architecture is based on the active participation of the design systems in the design process itself. As such, it is different from other design information systems - this type of integrated system is an active participant in the design process. Further, this architecture does not preclude other information architectures and systems within the context of the design infrastructure. It is complimentary to other systems, pertaining specifically to the temporal and dynamic nature of the design process itself.

Active participation by the knowledge-based systems can take many forms: monitoring design progress, reporting on the condition (health) of the project, smart communication of design information among participants and to management, diagnosis of design problems, and contextual automated design.

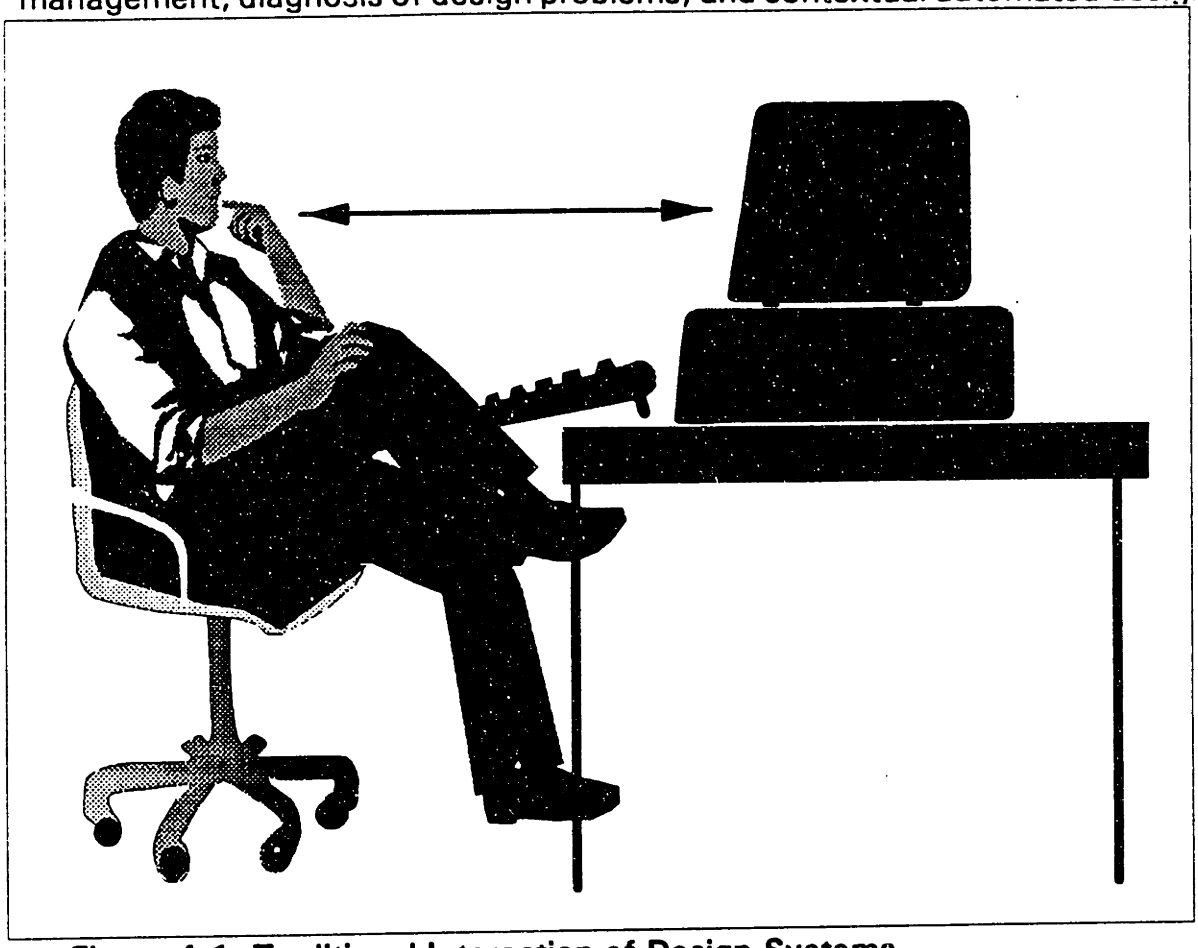


Figure 4-1: Traditional Interaction of Design Systems

Knowledge-based systems have been used for design automation, design advice, and design review. These kinds of systems are targeted, focused, and bounded in terms of the context of the problem, and the manner in which they are used by the design personnel. They are designed to reduce design cycle times, increase accuracy and quality, and lower costs. They have several system characteristics: they are typically focused on a particular part of the design process; they are highly specific; they are initiated by a design engineer

or designer (or they are programmed to execute at pre-determined times.)

In contrast, event-driven knowledge-based systems are not focused on a narrow part of the design context - they focus on the interfaces and boundaries between individuals and disciplines within a design organization. These systems are initiated not at the instructions of a person, but upon the occurrence of certain design events, as illustrated in Figure 4-2. As such, they are initiated and executed regardless of whether the design personnel involved are aware of their presence in the process.

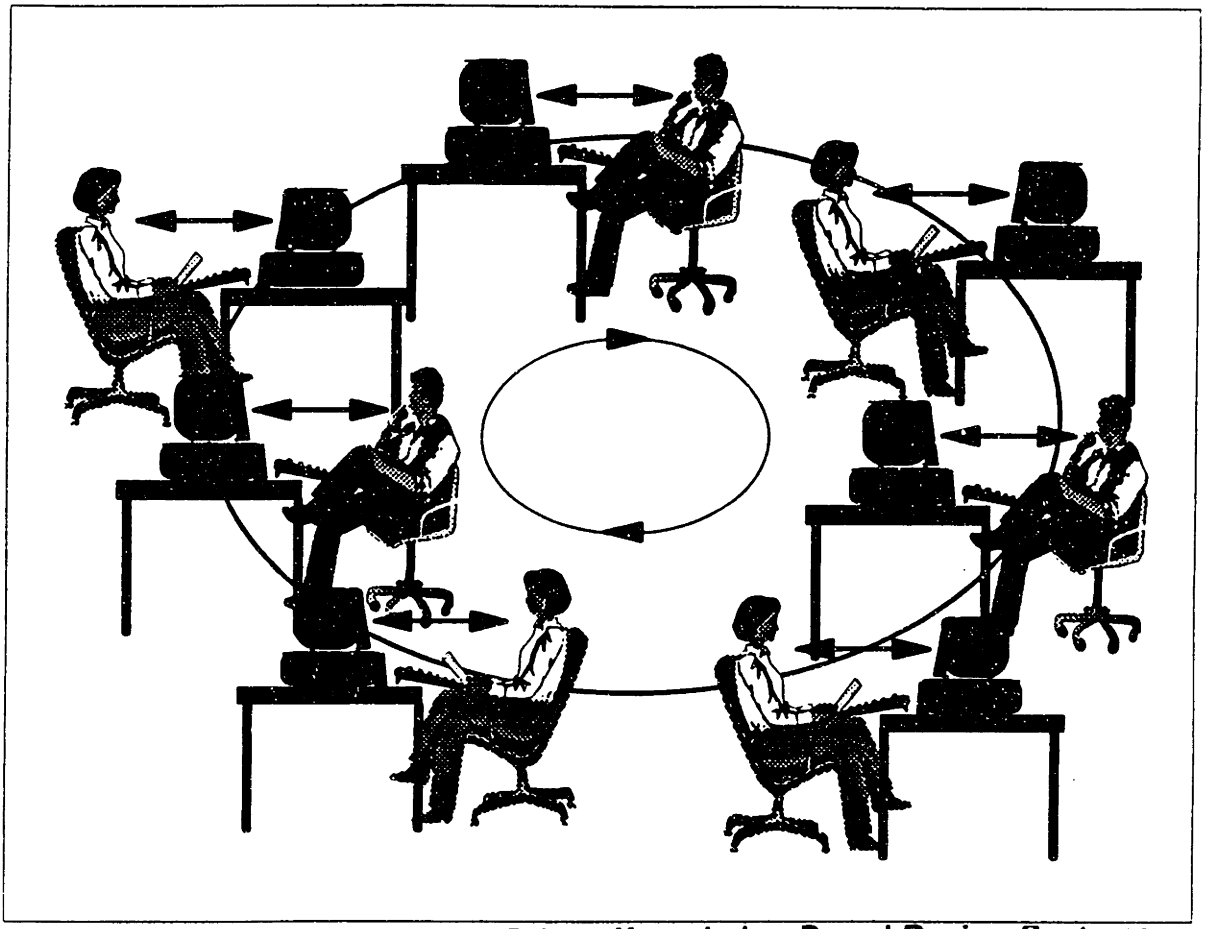


Figure 4-2: Context of Event-Driven Knowledge-Based Design Systems

The context of this work is the process of design, and a technology designed to support the design process. It is not intended here to suggest a generalized design methodology - rather, an architecture for supporting a variety of design

methodologies.

Many different design methodologies exist, both in theory and in practice. It is at once true that no single methodology applies to or is optimal for all design problems, and also that no one technology will support all design methodologies. However, Event-Driven Knowledge-Based Design provides a framework and a context for any viable methodology in that the underlying architecture is not process-dependent, rather it is process-driven.

4.1 Definitions of events:

Within the design process, there are two major classes of entities - these being data and events. People and systems perform activities using data. The action involved in performing each significant design activity constitutes an event (e.g. performance of a structural analysis calculation) and the production of new data constitutes an event. An event can be thought of as *some occurrence that may cause the state of the design to change*. (Booch, 1994) Note that while a generalized event description may apply to every design action, only those activities that are of interest to another party (or system) need be dealt with in the context of event definition. A tree that falls in the forest may make no sound if no one hears it, and an occurrence in which no one (or thing) expresses an interest is not an event. (Reinschmidt, 1995n)

In conventional design methodologies, "milestone" events are categorized as the completion of major portions of the design process. Examples of these events include completion of functional design work packages, release of basic data and release of the final design.

Events can be defined in two dimensions, these being the event characterization (or type) and the event scope.

4.1.1 Event Characterization (Event Types):

a. Binary:

A binary event is that type of event that can be described in a single bit of information, and is analogous to a boolean entity in computer logic. The event either occurred or it did not. An example of this is an event that describes the communication of design information from one discipline to another. The information is either sent to the appropriate recipient, or it is not. This can be measured simply and in a highly defined manner.

b. Synchronous:

Synchronous, or periodic events occur at certain regular and repeating time intervals. These events are exemplified by the execution of a computer program that is timed to run each week at a pre-determined time, in order to perform material commitment analyses for the design in its current state. For example, a system that might execute at the end of each week would count the instances of standard parts committed in the design that week, and would perform a bill-of-materials and cost analysis. These synchronous events can also be performed by people. For example, a design review can occur at regular intervals, in which the quality assurance manager inspects the design data released since the previous review. The performance of such analyses or quality reviews constitute synchronous events.

c. Asynchronous:

A conditional asynchronous, or non-periodic event occurs under specific compound circumstances, whenever the conditions of the event have

been satisfied. These events are analogous to "rules" in a rule-based, or knowledge-based system, in which a set of actions are taken if certain conditions are met. In the context of events, these actions are taken "whenever" the conditions are met.

c.1. Threshold:
As a subset of asynchronous events, threshold events require that certain threshold values within the conditions of the events be reached, in order to activate the actions associated with the events. These conditions can be single or multiple (simple or compound) statements, and include some quantitative measure of the threshold value (e.g. a numeric quantity.)

The other dimension of the definition of events is the scope and context within each event applies. An event can be viewed at a local level, or at a more system-wide level of abstraction.

4.1.2 Event scope categories:

a. Local:
A local event is specific to a designer or engineer, and has no meaning beyond the context of the individual's working state. An example of a local event is the initiation of an automatic warning whenever an engineer or designer exceeds the allowable limits of a design program or standard. Local events apply to many people throughout the design organization, but are not communicated beyond the framework of the local view of the individual.

b. Regional or System:
A group of individuals and systems, working together on a functional or

structural system or sub-system communicate amongst each other within the boundaries of their system. For such issues, confined to the system or sub-system, but broader in scope than a single individual, system events exist. These events are actions that may be relevant to a variety of individuals and/or computer systems. When one person makes a decision that affects others within the system boundary, that constitutes a system event. For example, in a hydraulic system design process, the determination of an operating pressure affects all of the component design processes within that system, but may have limited (if any) impact on other components or systems. As such, when that design parameter is released, that constitutes a system event.

c. Global:

In many cases, individuals performing local activities can have a significant effect on many people and activities across the design organization. Any event that has more than a local or system implication can be characterized as a global event. A global event is any event that transcends beyond the boundaries of any single system or sub-system. For example, if a structural designer places a structural element in space, and that structural element has an effect on the location of other equipment and systems (such as hydraulic lines) than the placement of the structural element constitutes a global event. Another type of global event is one which is has a deliberate impact on all aspects of the design process, such as the alteration of an external specification or regulation that affects the overall product, and may impact each element of the design process in a unique way.

The above two dimensions provide a framework for classification of events, as depicted in Figure 4-3.

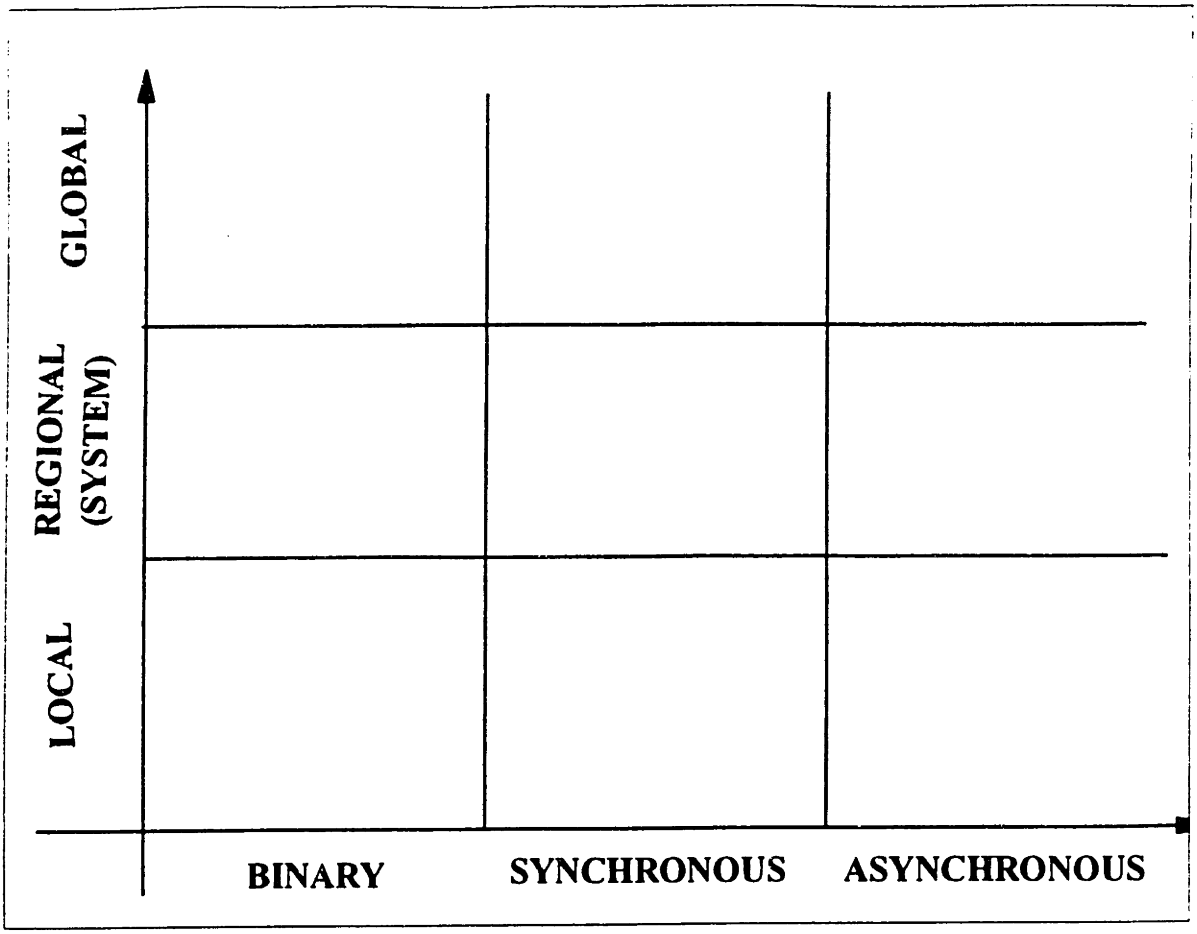


Figure 4-3: Scope and Type Contexts for Event Classification

In the context of an event-driven environment, one can think of events as being defined from the perspective of the those affected by the events - it is these parties for who the event has significance. This may sometimes include the systems or people who contributed data or actions to the events, but is most importantly inclusive of others in the design or manufacturing organization impacted by the events. Specifically, an event can constitute the collective contribution of several individuals, and the event exists only because of the sequential or concurrent participation of all of the contributors. Thus, there exists a set of recipients of the impact of the event, such that the event is described in terms of their interests - not the interests of the generators of the data or actions. To be sure, there is no certainty of linearity in the information chain. In fact, there may even be circularity, in which recipients of information or actions as a consequence of the occurrence of certain events may

themselves generate information and actions that are of interest to the contributors of the events.

Note that, in general, recipients of information (including all those affected by the event) have more information about *and interest in* how the event impacts them than the contributors do. As such, the concept of *asymmetry* in information and interest is introduced here. This refers to the degree of interest in a particular event that any set of designers may have in contrast to another set of designers who are not impacted (or impacted to a lesser degree) than the former set.

It is not necessarily true that event-driven knowledge-based systems are limited to or concerned only with situations in which some designers (the contributors to certain events) have no motivation to communicate with other designers (the recipients of information or event actions.) There are three conceivable scenarios:

(a) contributors are knowledgeable of the impact of their individual or collective actions, but require consistency and automation in the communication of the information and the impact thereof;

(b) contributors are not aware of the potential impact of their actions, and as such, event-driven knowledge-based systems use information in a manner unbeknownst to the contributors; and

(c) contributors to the event are unaware that their actions constituted an event because the event was defined as a **combination of actions or data** derived from multiple sources, not simply the communication of information from one contributor to one or several recipients, as is the case in a message-passing paradigm.

Note that within this event context, the design organization consists of a set of designers for which each event has a specific connotation. That is, for each event, the event regards the designers as contributors (participants in the causal initiation of the events) or recipients (parties affected by the event.)

4.2 Representation

Events are entities, and as such, have (within the context of the event-driven design framework) a representation that characterizes them. This representation must, by definition, allow for the *description* of the event as well as a description of the *behavior* of the event. Rather than use a one-dimensional representation (such as parameters, or variables) a more complex and rich representation is desired.

4.2.1 Event Classes

Events can be represented as class hierarchies, with object-oriented characteristics and attributes. This allows for a flexible implementation, with powerful constructs. Flexibility is achieved because the classes are independent of the implementation, and can be contained in programming systems (such as those developed in object-oriented programming languages,) in databases (such as those incorporating object-oriented database representations,) or in knowledge-based (inference-based) systems, using object-oriented class representations.

By structuring event representations as classes, the power of the active aspect of classes (through methods) combined with the declarative representation of the hierarchical inheritance-based attribute structure can be achieved. This allows for the separation of mechanisms for dealing with events from the events themselves, a critical design issue, as will be described in the

architecture section below.

An example event class could be a release_event. A sub-class of the release_event is the working_to_preliminary event, in which data are released from a working state to a preliminary release state. Another sub-class of the release_event is the preliminary_to_final event, in which data are released to their final condition. As another example of an event class, consider a sub-class of the working_to_preliminary example above, in the automotive industry. When surface data for the exterior body surface of the automobile are released, these data are used for a variety of internal detailed design purposes. A class_A_release class would refer to critical "class A" surfaces, and would contain pertinent information about the location of the surface, the critical boundaries, curvatures, tangencies, and other design information that must necessarily accompany the surface itself. Body designers, window glass designers, door designers, engineering analysts, and a host of other focused design engineering groups depend on these data in the context of the state of the design in order to perform their tasks. Once this class was instantiated with the appropriate specific values, the event system would transmit the relevant knowledge (based on the source information and the impact thereof) to the appropriate individuals for use in downstream engineering design activities.

Note that event classes represent the information *and* the actions associated with the event. As such, these event classes exhibit dynamic behavior, either in performing operations, triggering other events, initiating some activity, ending some activity, or communicating information from sources to recipients. (Booch) Note also that the classes are the generic descriptions of the events, and that whenever an event is detected, initiated or otherwise activated, an *instance* of the appropriate class is dynamically created, populating the slots or attributes with specific values, and performing specific actions associated with the generic methods defined therein.

One example event class may have the following representation:

EVENT NAME:
EVENT TYPE:
CONDITIONS:
ACTIONS:

For each event type, there may be sub-classes, that represent patterns of events within a given design environment or framework. For example, a synchronous event type may have a sub-class that includes synchronous events of a particular frequency. This attribute is unique to the synchronous event type. The threshold event sub-class might have an attribute (or set of attributes) specific to the threshold levels. Again, this is a unique feature of this type of event. A generalized framework for this class - sub-class hierarchy is represented in Figure 4-4.

Note that the concept of *inheritance* applies to these event hierarchies. (Booch) In this object-oriented methodology, parent classes (e.g. *event*) have attributes that become inherited attributes of the children, (e.g. *binary, threshold, Synchronous, Asynchronous.*) As such, the basic properties of the event class are transferred to the sub-classes, and each sub-class may have its own unique additional attributes. These sub-classes can also take the role of parent classes, and in this manner multi-level hierarchies result. At the lowest level, the leaves of the hierarchies can trace their inheritance vertically through the object or class hierarchy, and can thus establish a grouping or family relationship to all related classes.

An example of one possible detailed sub-class description for the above event classes follows:

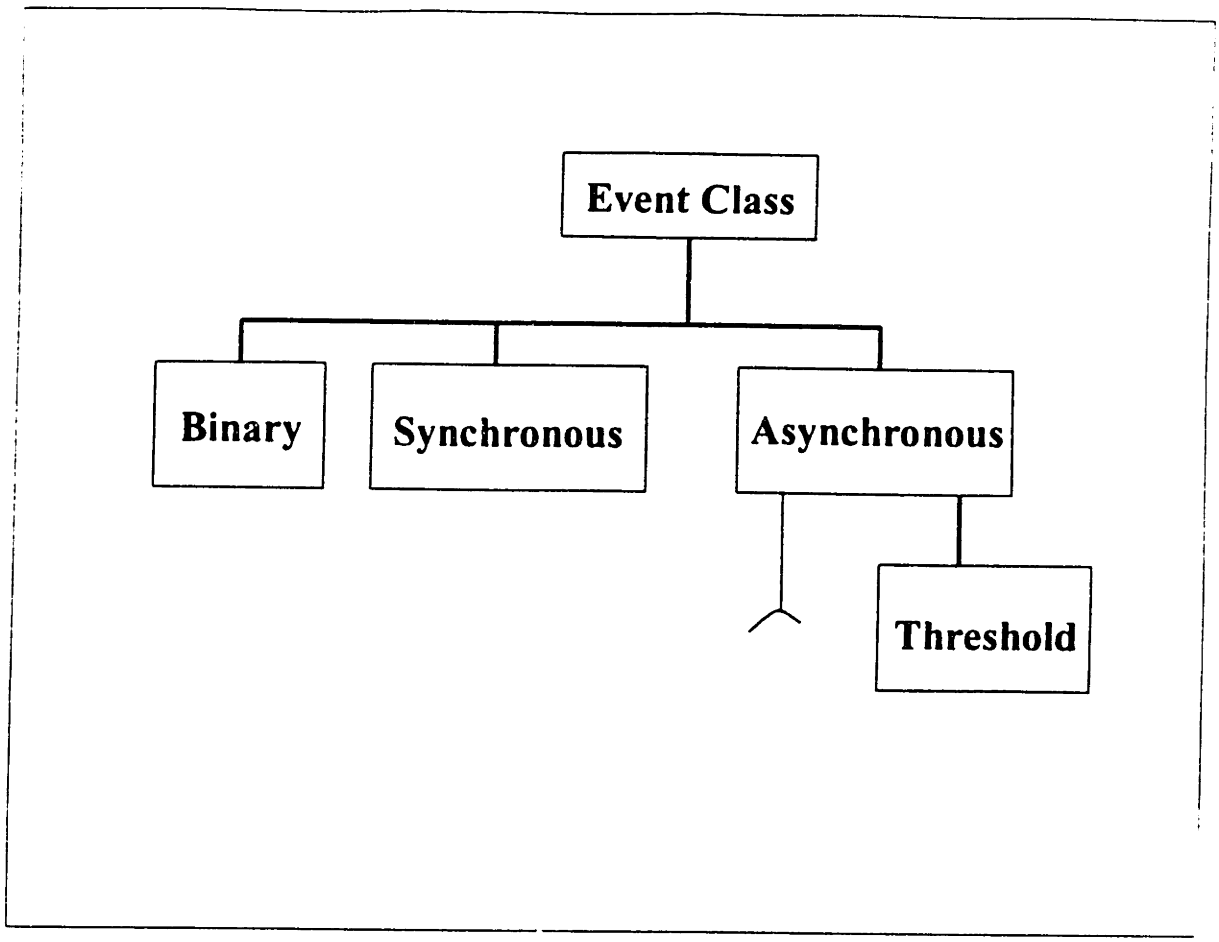


Figure 4-4: Class Hierarchy of Event Types

EVENT NAME:

EVENT TYPE: BINARY

CONDITIONS:

ACTIONS:

EVENT NAME:

EVENT TYPE: THRESHOLD

CONDITIONS:

THRESHOLD SET:

ACTIONS.

EVENT NAME:

EVENT TYPE: SYNCHRONOUS

FREQUENCY:

CONDITIONS:

ACTIONS:

EVENT NAME:

EVENT TYPE: ASYNCHRONOUS

CONDITIONS:

ACTIONS:

Note that *instances* of these event classes would represent actual events in the context of a specific problem or design framework. That is, the above represent the basic class descriptions, and actual descriptions of meaningful events would include values for the Name, Conditions, and all other attributes.

4.2.2 Event Currency

The "Currency" of an event is defined here as the period during which the event has meaning, with respect to the recipients and the configuration management of the source data. That is, when an event is generated, it may not apply in perpetuity; rather, it has some limited applicable span of time during which the event has meaning.

An event can lose its meaning if the source data change, global characteristics of the design environment change, or the recipients change their focus of attention or scope of operation. An event can also be determined to be no longer current if all of its recipients have been made aware of the event and have responded appropriately. In any of these cases, the event is no longer

valid, and has no further bearing on the design situation. The event may be superseded by another event, or it may become irrelevant for other reasons.

The currency of an event can be determined by the contributors, or by some event management system. For example, it may be appropriate to place time limits on events of certain types, and the events (having object-oriented representations) can have built-in self-terminating mechanisms. The currency management issue can be addressed either extrinsically (by the event management system) or intrinsically (by the events themselves.)

4.3 Architecture

The internal architecture of the event-driven knowledge-based design system is a critical element of the system's ability to perform its function. As has been described, the notion of an event as an entity is central to this hypothesis, and as such, the event entity must be dealt with separately from the systems that work with the design data.

Several architectures are possible within the framework of an event-driven design system. Fundamental to the infrastructure of such a system are certain common elements. Included in these common elements are:

- designers (people or systems)
- design systems (CAD/CAE/CAM systems)
- design databases (CAD files, independent databases, etc.)
- knowledge-based systems
- an event system (including event representation and a detection/reaction mechanism)

One feature that is not common to all architectures is a separation between the event system and the design databases. Where this separation exists, it constitutes an independent software environment, operating separately from the other components of the overall system.

When the event system is integral to the other components, it is not the primary function or focus of these traditional systems. As such, in the latter case, the event system must be specifically added on to the native functionality of the other components, and thus constitutes an inherently less desirable architecture. We can classify these two basic types of event-driven design systems as *explicit* and *implicit* systems, depicted in Figures 4-5 and 4-6, respectively.

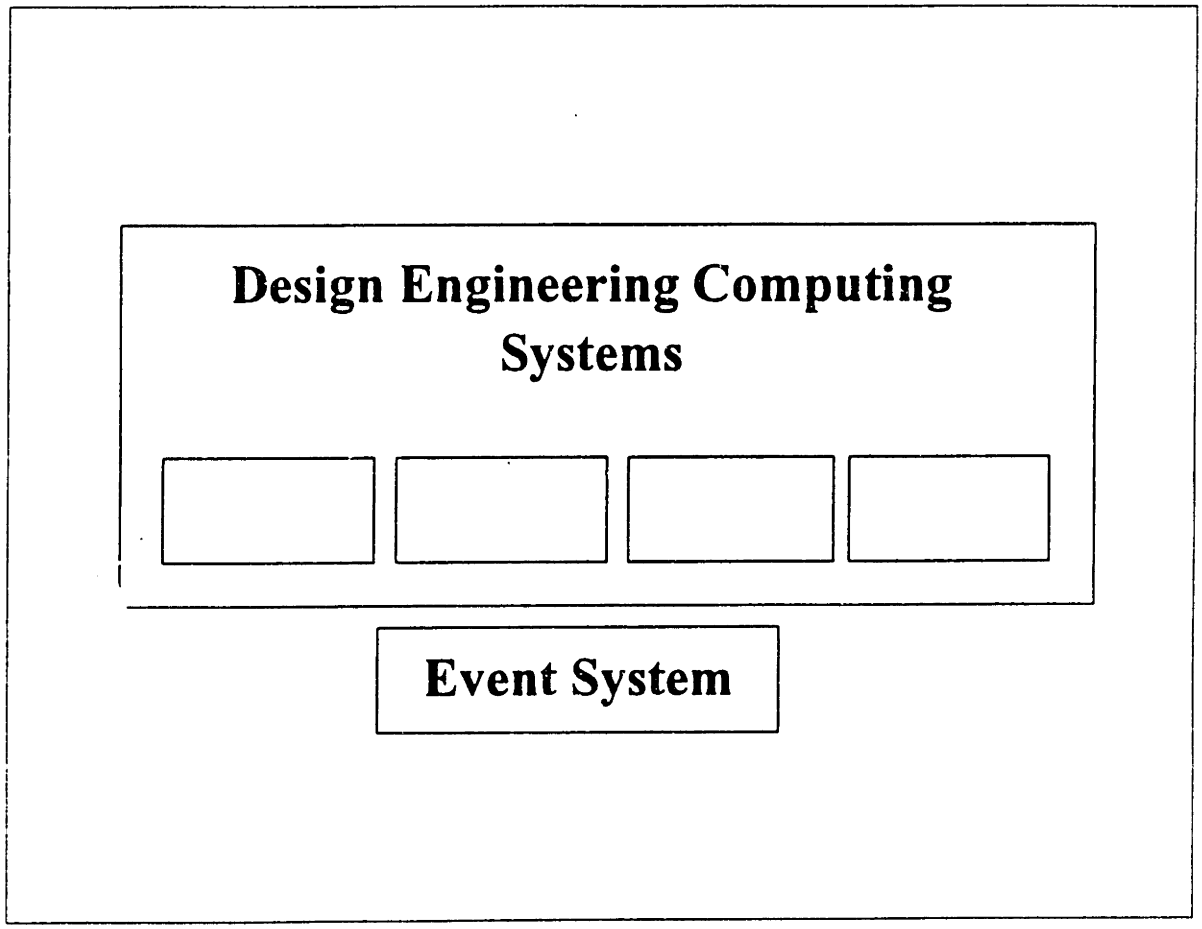


Figure 4-5: Explicit Event System Architecture

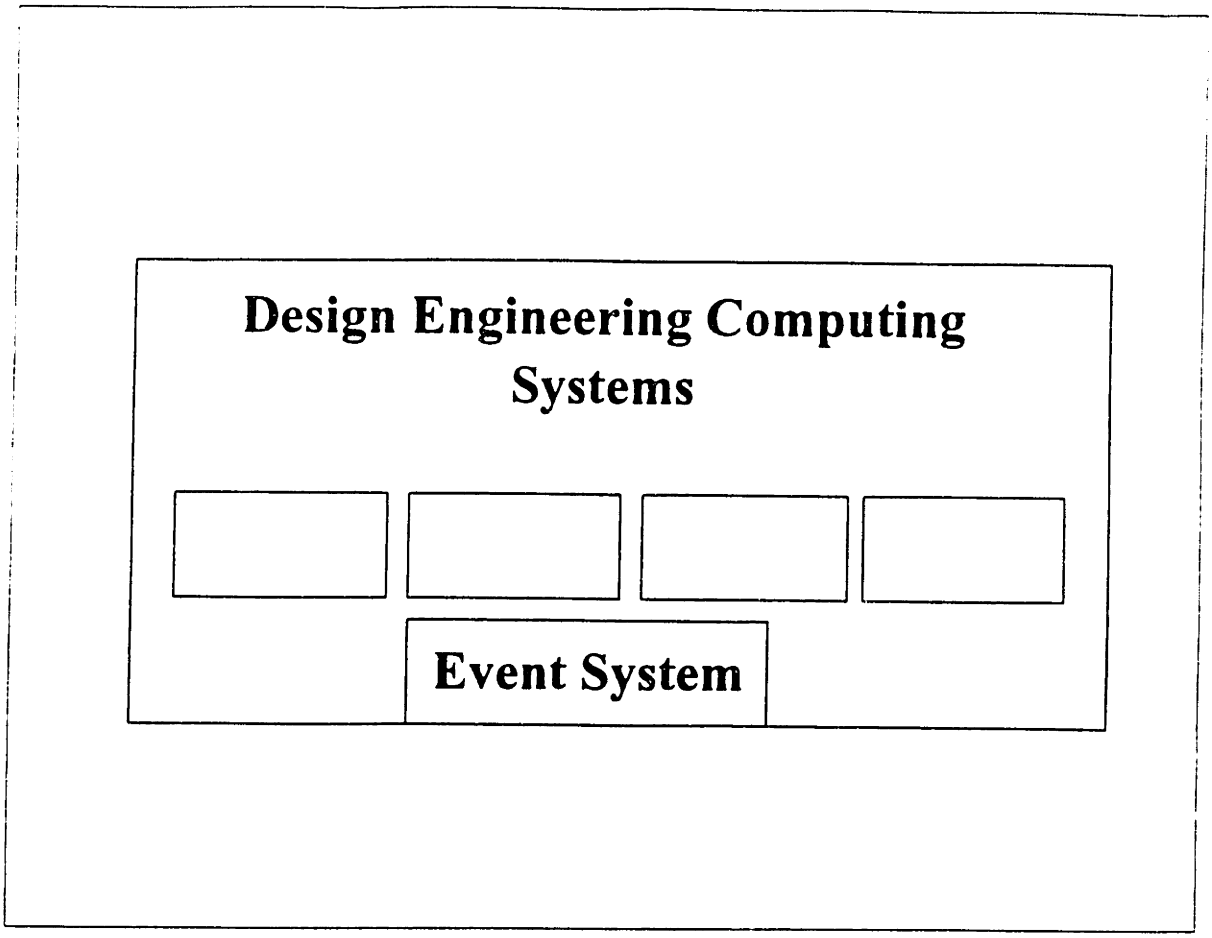


Figure 4-6: Implicit Event System Architecture

In any computer-based design process, the design data are stored in a collection of databases, either proprietary to the CAD software components or more generally accessible, such as those that are hierarchical, relational or object-oriented in nature.

The event system acts on, and is driven by these design data, but is not an inherent part of the design database itself. An examination of the two primary architectures is provided, for comparison purposes:

4.3.1 Implicit Event Architecture: Integrated event/data system

In this architecture, each single element of the design data also has associated with it methods and rules describing the events that are associated with the data themselves. In this manner, the representation of the design data includes not only a description of what the data are, but how they are used, and by whom. This is analogous to the "event callback" model (Booch, 1994) in which widgets (or objects in the database or in memory) know how to respond to specific events.

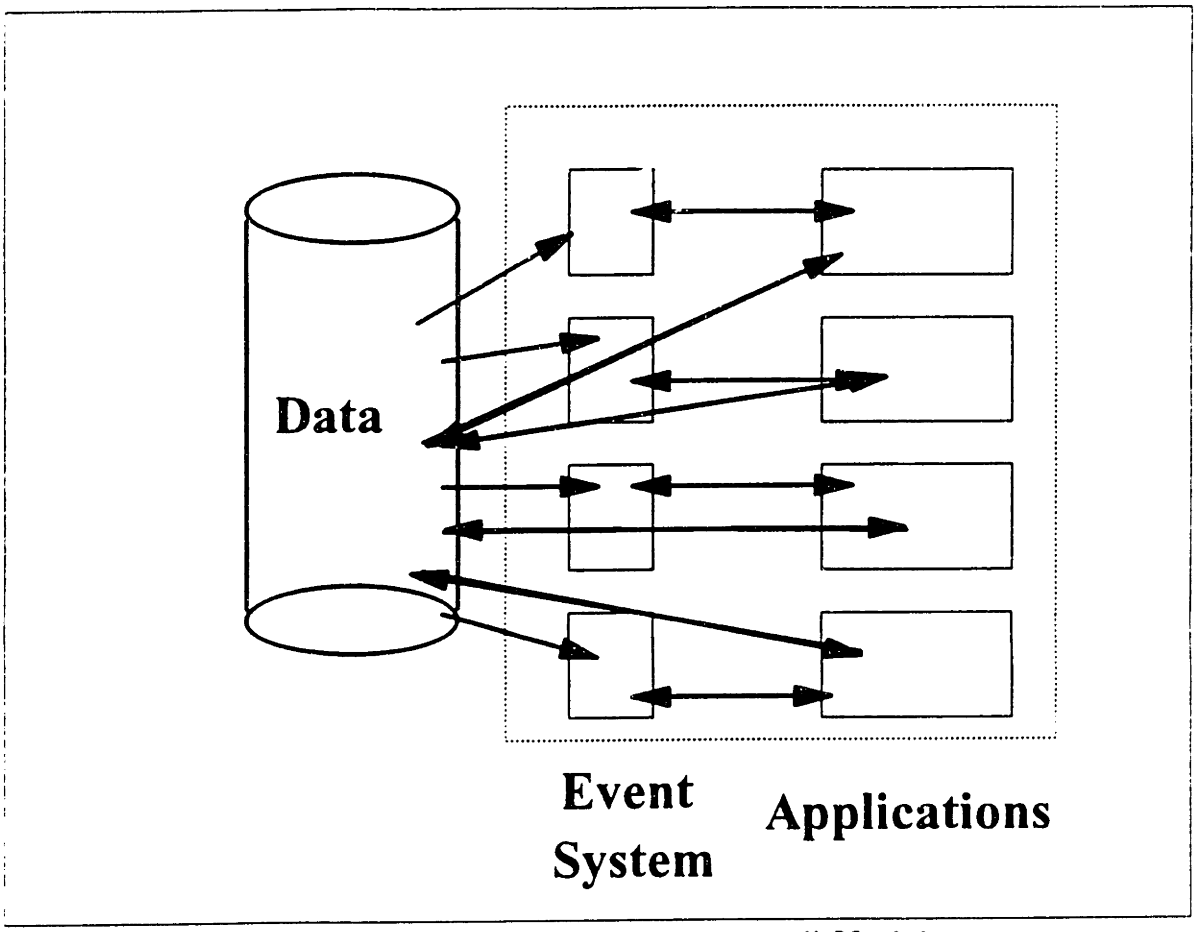


Figure 4-7: Event Interactions - Implicit (Integrated) Model

A depiction of such an integrated event environment is shown in Figure 4-7.

The interpretation of the interactions in this model can be summarized in the interaction matrix, shown in Table 4-1:

	Event	Data	Applications
Event	N/A	NO	YES
Data	YES	N/A	YES
Applications	YES	YES	N/A

Table 4-1: Interaction Matrix for Information Flow - Integrated Model

The interpretation of this matrix is accomplished by reading across each row, NOT down the columns. That is, in order to determine if a particular element ($element_i$) of the overall system is allowed a positive information flow with respect to another element ($element_j$), the matrix value (i,j) is used. For example, in order to determine if, in the integrated event architecture, the event system has a positive information flow to the data system, we read matrix value (1,2) which is "NO", indicating that the event system does not affect the data system directly.

In order to determine if a connection is uni-directional or bi-directional, we use the following rule:

IF (i,j) equals (j,i) THEN connection is Bi-Directional

IF (i,j) does not equal (j,i) THEN connection is Uni-Directional

The advantages of this architecture are (i) that there is a single

representation for design data, and (ii) that there is an integral communication mechanism between data and events since both are intrinsic in the given system.

The primary disadvantage of this architecture is that every element of design data must have natural knowledge of all of its potential uses. Since the event architecture may include conditional events, and since the development of the event system may be incremental, this would impose a significant systems maintenance overhead on the data/event information system. This implies a complex and complete configuration management capability, and the associated cost thereof.

Another negative feature of this system is the inability to interchange database and event representation mechanisms. The implications are that a commitment to one representation architecture is required, regardless of the applicability of different architectures for each component of the system, namely that design database and the event representation system.

4.3.2 Explicit Event Architecture: Discrete data and event systems

An alternate architecture is one in which a separation exists between the representation of design data and the system that manages events. In this architecture, the design data are represented independently, as if the event system did not exist. The hierarchies, classes, structures, and formats of the design data can remain declarative. A separate event system reads and writes to the design database, using only the declarative form of the data. This is illustrated in Figure 4-8. The event system would contain the active or functional segment of the system.

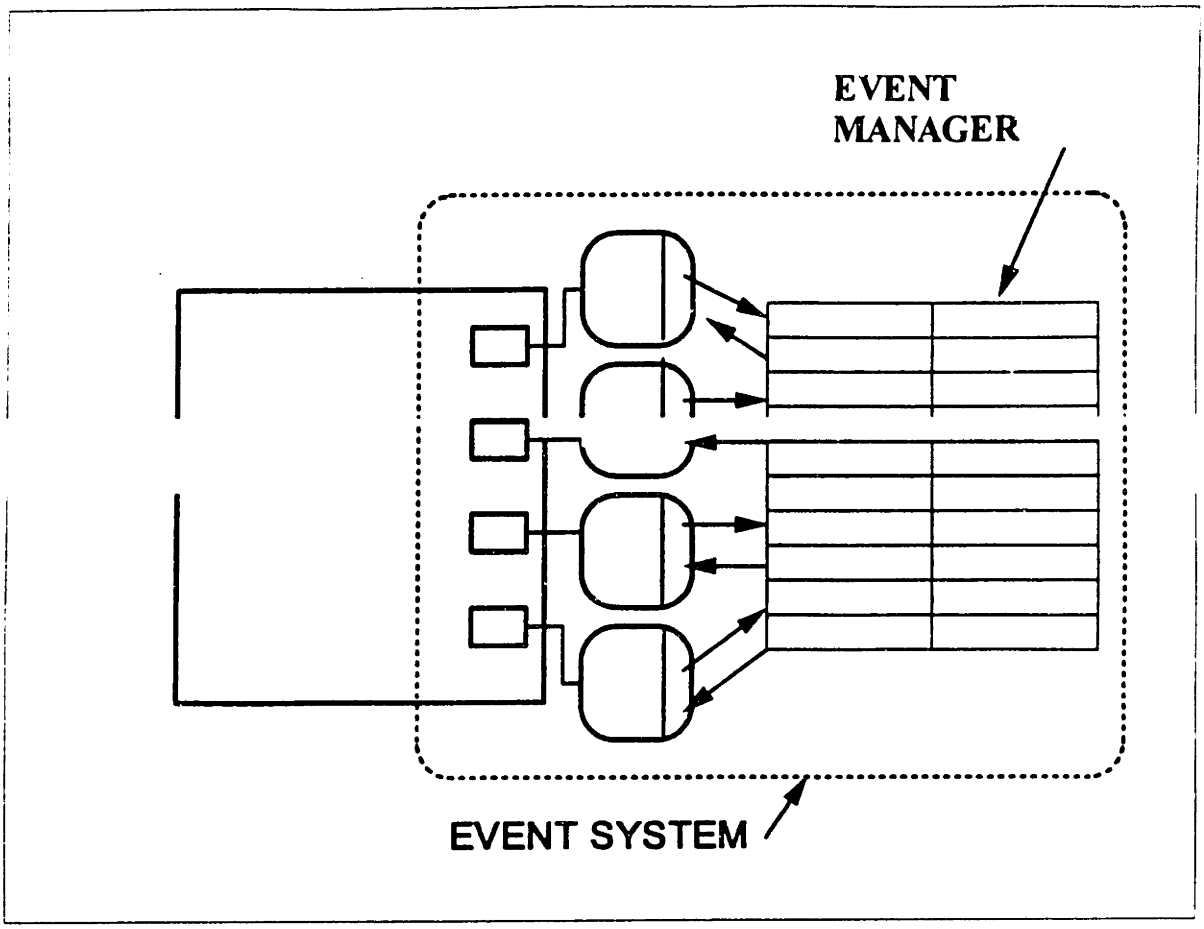


Figure 4-8: Explicit Architecture - Interaction Between Event System and Balance of Design Computing Architecture

Note that there are several characteristics of the inter-relationships between the event system and the other actors in this architecture. Specifically, the are uni-directional and bi-directional connections, falling into the behavioral model shown in Table 4-2:

	Event	Data	Applications
Event	N/A	NO	YES
Data	YES	N/A	YES
Applications	NO	YES	N/A

Table 4-2: Interaction Matrix for Information Flow - Discrete Mo...

The advantages of this architecture are: (i) that it is not necessary to know about how each piece of information is to be used when defining the information in the design database. It is well known that information can be used for a variety of reasons and by a number of individuals and systems. If each element of the design database must be defined as to its relationships to all other pieces of information, then the structure would become unwieldy and impractical. Rather, the relationships between data, in terms of events, are stored in the event system, keeping the structure of both simple; (ii) that changes in the event definitions would not require restructuring of the design database. The changes in the events would be reflected in changes in the interface boundaries between the event system and the design database, and the local effect of the event system changes would be maintained locally; (iii) Only a single bi-directional reference to data must be maintained within the event system for each event relationship between two pieces of design data, as compared to two complete event descriptions, one in each datum's structural representation in the database. This unique event description minimizes the representation requirement and the propensity for error, and (iv) that the implementation of the event system does not depend on the implementation of the design database system. In practice, this translates into the potential for using a relational database structure for the design data, and not requiring the use of an object-oriented database structure. The event system can be represented in a functionally disparate architecture from the design database, and the technologies need only support a boundary interface, allowing the event system to operate on the design database. Note that in the optimal case that a single technological representation is used, the inherent mechanisms within an object-oriented database can be exploited to take

advantage of the representation for the data, and the method structure for the events.

For the above reasons, the discrete event architecture (DEA) is recommended as the implementation architecture for event-driven knowledge-based design systems.

The event descriptions are contained in the object-oriented design database, and not in the applications. As such, people and systems contribute to the repository of information in the design database, and the design events are detected by the event management system. Actions are taken in terms of messages (descriptions of the events) passed to systems and people. For example, an event could consist of a design change. As soon as the design change is detected by the system, messages would be sent to knowledge-based systems to determine their effect on the process and the product, and messages would be sent to people, in the form of electronic mail or reports.

In this architecture, the event system consists of an *event manager*, which contains information about the events and their associated data. The event manager is notified by the design database system whenever data in the database are added, removed, or changed. The event manager then maps those changes to the appropriate events, which initiate their own activities. In this manner, the design database has no knowledge of the uses of the data themselves. The event manager communicates only the relevant information to each event class, and the event classes then take whatever action is appropriate. Note that the event manager can be built on the "event-loop model" basis, as described by Booch. This would be appropriate if the design database system is implemented in a passive architecture (e.g. relational database.) This would be implemented via a "polling" function, which repeatedly examined the database at regular intervals.

In the case that the design data are represented in a dynamic environment (e.g. object-oriented database) the event manager can be implemented using the event-callback model. In this model, the objects in the database send messages to the event manager whenever any change is made. This is facilitated by the methods or functions that are inherent in an object-oriented database architecture.

The event classes communicate directly with the database. The event manager simply notifies the event classes of changes in data that are relevant to each event class, and the event classes contain the intelligence to make decisions about whether and how to act in response to the changes in data. As such, an event class may be initiated because of a single elemental change in the design data, but it may require a variety of pieces of information that are available in other locations in the database. The event description is encapsulated within the event class itself, and as such, the instance contains the requisite access information for all of the data sources and active systems that pertain to the event.

4.4 Dynamic Strategy

Event systems can be thought of as having one of two dynamic strategies: (a) predispositive or prescriptive, or (b) post-analytical. The dynamic strategy defines the mechanism by which information becomes available from a contributor to a potential set of recipients. In a basic sense, these translate to whether the contributor specifically sends information to the event (or data) repository, or if the event system acts autonomously on the design database.

4.4.1 Predispositive

Whether design activity is being performed by a person or by a system, one can imagine a specific action taken by the designer to report design results to other designers. This requires that the originating designers have knowledge of all of those recipients who are interested in the design data. More specifically, the originating designer must have knowledge of the event-driven knowledge-based systems that have need for such data. Thus, the action of the event-driven knowledge-based systems is dependent on the diligence of the originating designers (people or systems) in terms of the fulfillment of their obligations to communicate information. Such a dependency is, of course, not practical.

An alternative architecture would be to "post" results or design data on a neutral data repository (such as a blackboard) and to have an analogous mechanism for "polling" or searching the blackboard for information - the existence thereof thus defining events.

The current limitation of this predispositive approach is the requirement that some knowledge of the event be present in both the originating and recipient designers. This translates into a manual mechanism for specifically calling out "posts" to the blackboard by designers (people or systems) or some yet-to-be-

defined software mechanism to alter systems and inform people such that the originating designers automatically included all specific "posting" actions required by the event system.

4.4.2 Reactive

With reference to the Discrete Event Architecture (DEA) described above, imagine the event system being capable of retrieving necessary information without a specific action on the part of the originating designers. That is, the event system must have some mechanism for determining the location of necessary information, and the ability to provide that information to the event description repository.

In such a system, the designers are aware neither that their actions are being observed or "used" by the event system nor of the specific source of information for the events that affect them. This approach reaches beyond the functionality of Product Data Management (PDM) systems, which provide generic access to heterogeneous databases. The event architecture could conceptually reside as a layer above a fully functional PDM system, acting as an active driver of the data retrieval and storage mechanisms.

4.5 Implementation

Three architectures are described here, each having characteristics of the Discrete Event Architecture, but with flexibility relative to the predispositive or post-analytical dynamic strategy selected.

For the purposes of consistency, the following definitions are proposed:

Designer: A person or knowledge-based system contributing as

a participant in the design process;

Contributor: Designer who generates information contributing to a design event, or otherwise causes (knowingly or unknowingly) the precipitation of a design event;

Recipient: Designer who is affected by or impacted by a design event;

Direct Link: Action to provide or retrieve information where the originating source or the specific recipient is known *a priori*;

Indirect Link: Action to provide or retrieve information where the originating source or the specific recipient is not known, but the general notion of the need for the information (in the case of a contributor) the availability of information (in the case of a recipient) is assumed, and

Blackboard: Global database through which independent knowledge sources communicate. (Engelmore, 1988)

Consider a contributor (c) and a set of contributors {C}, such that:

$$\{C\} = \{c_1, c_2, c_3, \ldots c_n\}$$

Also, consider a recipient (r) and a set of recipients {R}, such that:

$$\{R\} = \{r_1, r_2, r_3, \ldots r_m\}$$

where $n <> m$.

For any given event, a set of associated contributors and recipients exists. Note

that the contributors may be transparent in the Discrete Event Architecture, since the event system acts based on data, rather than direction from the applications. However, the general description of the event set includes both contributors and recipients. The total set of events is defined as:

$\{E\} = \{e_1, e_2, e_3, \ldots e_k\}$
where k = number of defined events.

For any specific event, a subset of the {O} and {R} sets is associated to the event:

let *i* = event number, then
e_i has associated subsets $\{C\}_i$, $\{R\}_i$
where $\{C\}_i$ and $\{R\}_i$ are the subsets of the general sets specific to event e_i.

In any design activity, each designer performs an action or a set of actions based on some input data and some operation. For each action, new information is generated, and this information may (together with other information from the design database or some external source) contribute to a significant design event. (Significance is defined, of course, in terms of those affected by the event, not those contributing to the event.) At any point in time, and with reference to any specific event, a designer may be an element of one or both of the above sets. That is, a designer can at once contribute to an event and also be affected by the event.

Note that, in the context of the event types described above, an event may be precipitated on the basis of one or more pieces of information or occurrences, from a single source or many sources. Similarly, the event may have one or

many resulting actions and recipients.

The morphology of an event is as follows:

a) designers perform actions that result in new information or other actions;

b) the occurrence of a single piece of information or many pieces of information in combination, combined with associated temporal factors, constitutes an event;

c) the event results in certain actions being taken, ranging from simple information communication (to a designer or set of designers) to the initiation of event-driven knowledge-based systems.

There are a variety of possible implementation architectures for event-driven systems (as described above) three specific implementation strategies are suggested here, for purposes of comparison and contrast. Note that the use of commonly accessible databases for design information is a general theme of all of these architectures. For the purposes of this discussion, they are all referred to here as blackboards.

> "The Blackboard Model. There is a global database called the blackboard, and there are logically independent sources of knowledge called the knowledge sources. The knowledge sources respond to changes on the blackboard. Note that there is no control flow; the knowledge sources are self-activating." (Engelmore, 1988)

Note that blackboard architectures have been used for integrating knowledge-based systems and data systems in real-time process control applications (Finn, 1989), as is illustrated in Figure 4-10. In these applications, the blackboard mechanism allowed a variety of knowledge bases to operate on process data that are collected in real-time from instrumentation in a process, chemical, or power plant, and for these knowledge bases to interact via the blackboard.

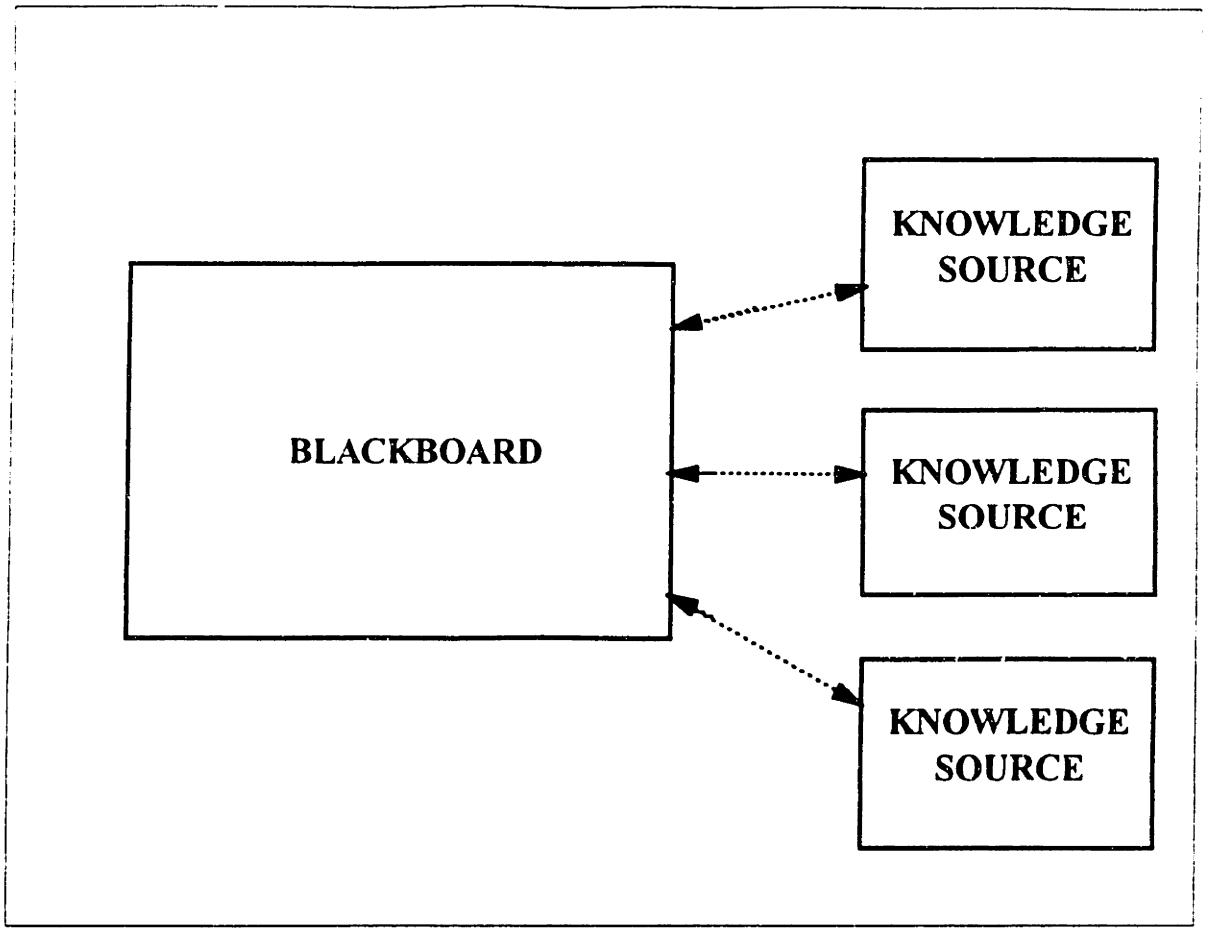


Figure 4-9: Blackboard Model (After Engelmore, 1988)

This mechanism allowed for a focus of the knowledge-based systems to be created, such that one type of system performed data interpretation, another system performed the data validation, and yet further systems performed diagnostics or prediction. (Racine, 1992) As such, these systems are event-driven, in the sense that they continuously poll the blackboard in anticipation of one or several events, and they take appropriate actions based on the events observed or derived from the data on the blackboard.

The analogy to design is such that the real-time data collection system translated into the collective accumulation of designers. While data are generated continuously, they do not become available to the design databases

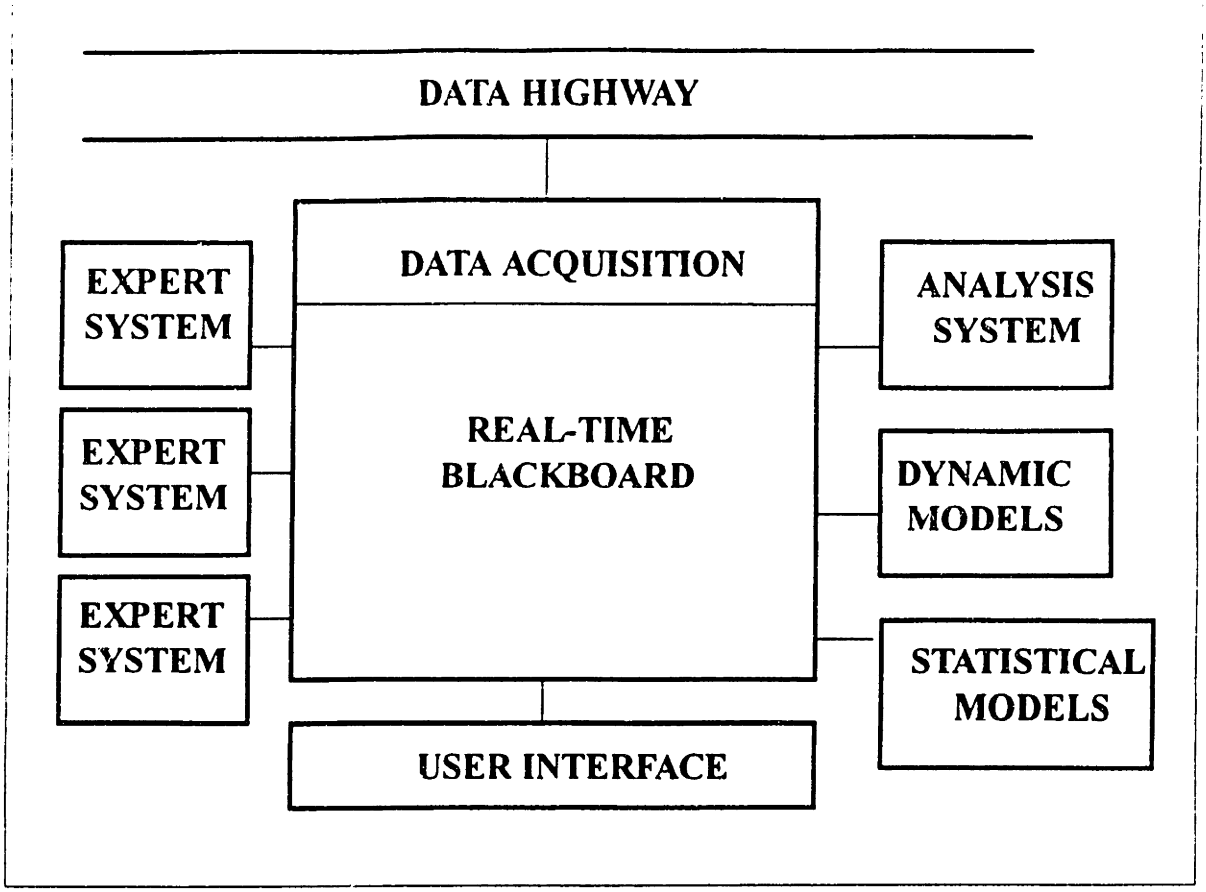


Figure 4-10: Real-Time Blackboard Architecture for Expert Systems (After Finn, 1989)

in a continuous fashion, rather, they are leased in packets of data at irregular intervals. Moreover, in the design context, it is necessary for the event-driven knowledge-based systems to communicate to the originators of the data, rather than feeding information to a central resource.

Note that blackboard architectures have also been used in integrated design infrastructures systems development efforts. One such project, Distributed and Integrated Environment for Concurrent Engineering (DICE), uses a blackboard architecture to

> " . . . 1) provide a means for storing information that is common to more than one KM [knowledge module]; 2) facilitate communication and

coordination, and 3) ensure that designs and plans generated during design and construction are consistent." (Sriram, 1992)

They provide a blackboard system divided into three components, these being Coordination, Solution, and Negotiation, as described in Figure 4-11.

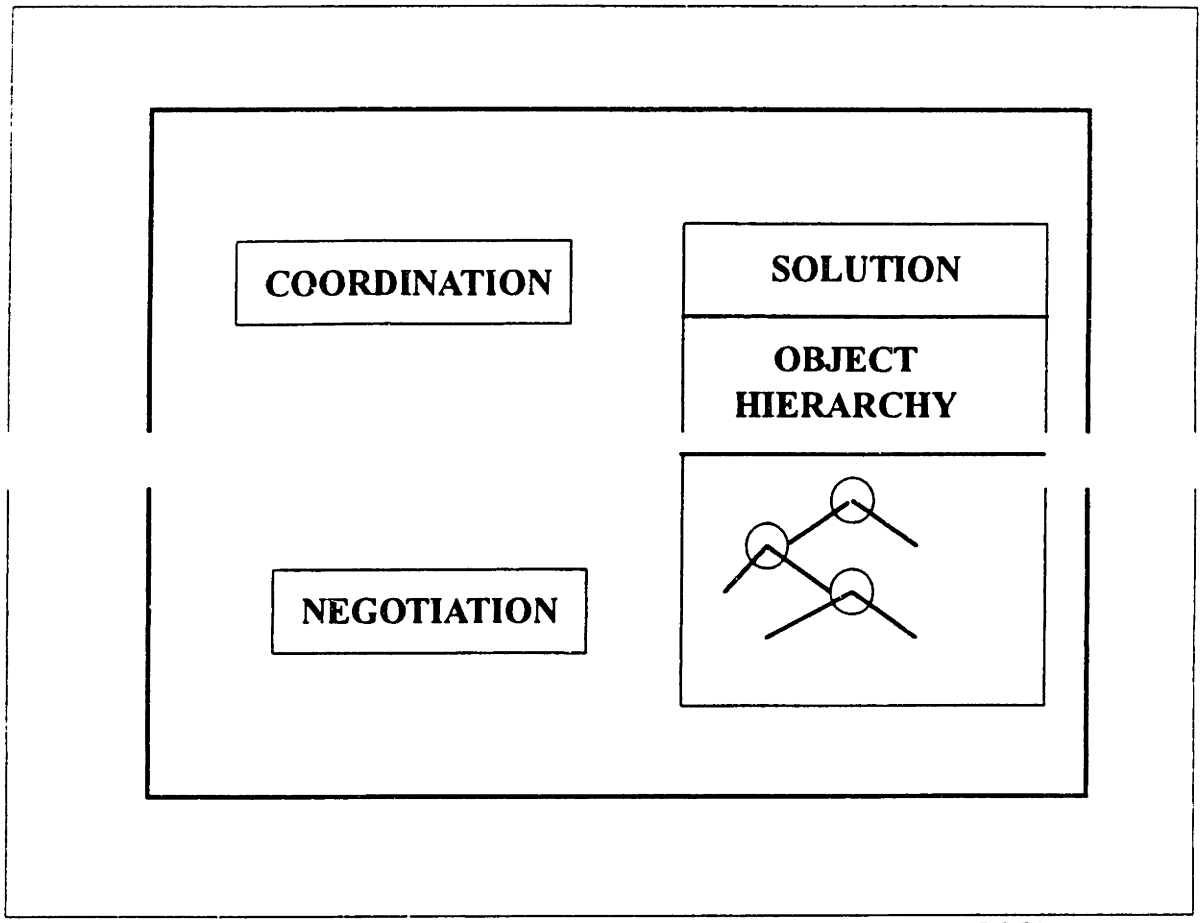


Figure 4-11: DICE Blackboard Architecture (After Sriram, 1992)

4.5.1 Implementation Architecture I - Pre-dispository

Using the general blackboard framework, we posit an architecture for the event-driven knowledge-based systems illustrated in Figure 4-12, as follows:

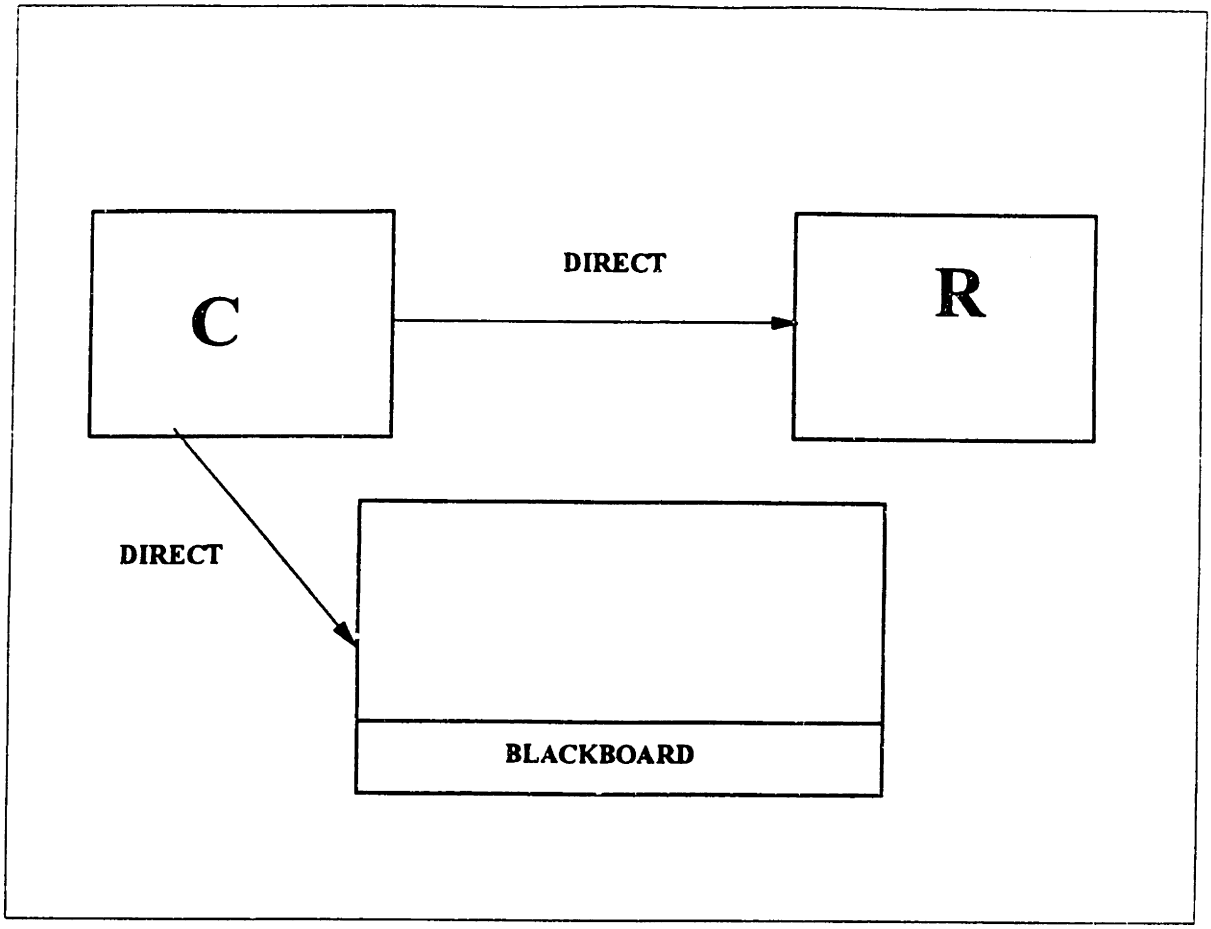


Figure 4-12: Pre-Dispository Architecture for Event-Driven Knowledge-Based Design

Characteristics of this architecture are that the contributing designers provide specific information to the blackboard. This specific information is required by one or more events, and is posted to the blackboard as a direct link between a designer (contributor) and an event.

Thus, each time an originating designer determines a value for a datum of interest to an event, that value is posted to the blackboard. (This can be accomplished one at a time, or in a packet of data collected prior to updating the blackboard.) The event-driven knowledge-based systems poll the blackboard at pre-determined intervals, searching for the

existence of all requisite data to satisfy the event conditions. Upon satisfaction of all event conditions, the knowledge-based system is executed.

The implications of this architecture are:

contributors must post all relevant data to the blackboard;

the event-driven knowledge-bases must "poll" the blackboard frequently enough to ensure that relevant events are recognized, and

contributors have built into their operations several specific actions that one or more events have listed as pre-requisite conditions.

This architecture has the advantage that the execution of the systems is relatively efficient, since only those pieces of information that have been defined as event conditions are posted to the blackboard.

The major disadvantage of this method is that it may be extremely difficult to determine, in highly complex systems, all of the points of origin of each piece of data required by events. This is essentially a manual process, and must be undertaken for every point of determination of all relevant data. Depending on the complexity of the systems involved, this may prove to be a limiting design factor. A second disadvantage of this approach is that an event may have relevance to more than one knowledge-based system or recipient designer. As such, there may be duplication in the blackboard polling, in

that more than one recipient designer may be attempting to satisfy the same conditions at the same time.

4.5.2 Implementation Architecture II - Intelligent Search

As in Architecture I above, we use the blackboard framework for this architecture. Here, however, we have an indirect link from the originating designers to the blackboard, and an intelligent search mechanism initiated by the recipients in order to determine the existence of sufficient conditions for each pre-requisite event. This method is illustrated in Figure 4-13.

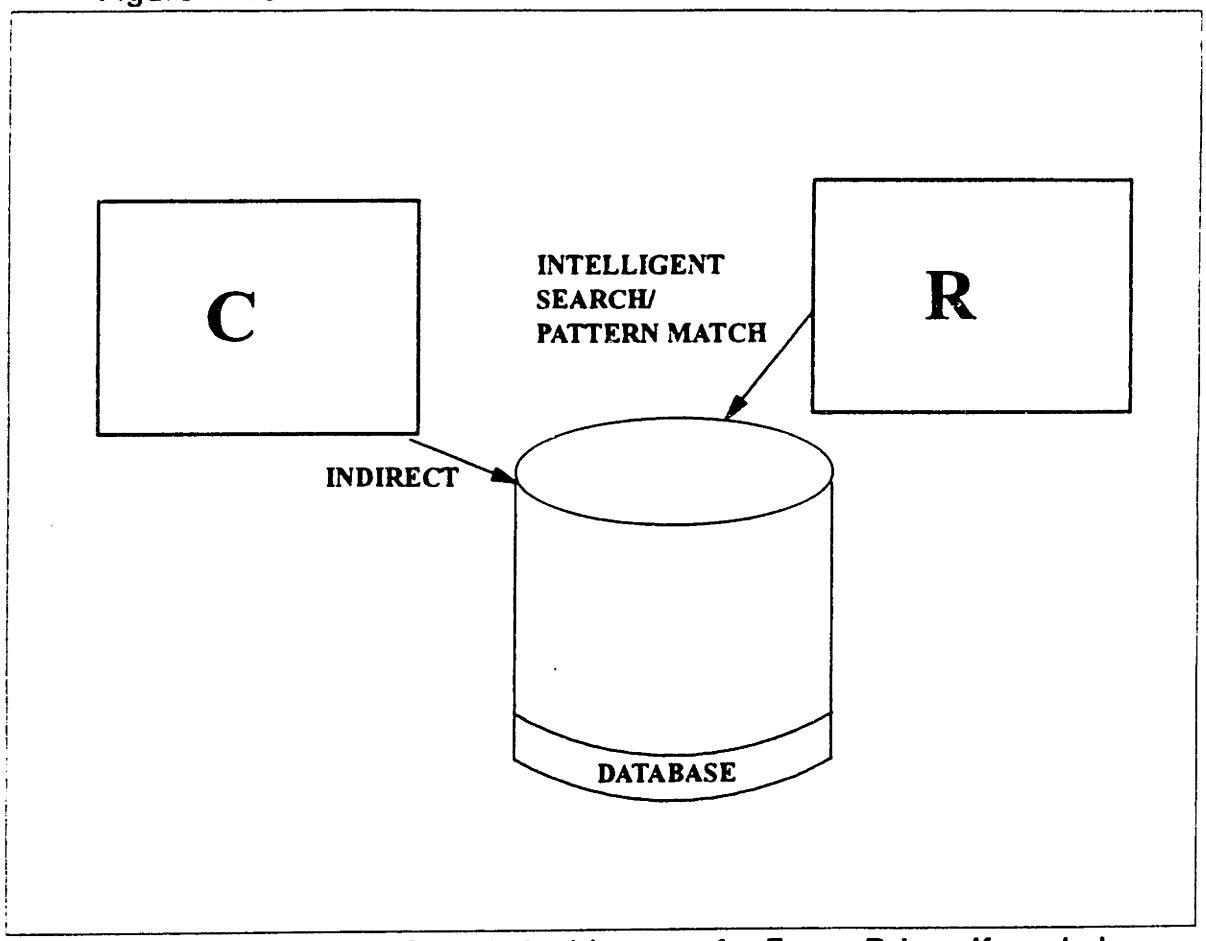


Figure 4-13: Intelligent Search Architecture for Event-Driven Knowledge-Based Design

Characteristic of this architecture is that the contributors are programmed to post all relevant design data to the blackboard, without regard for the potential use of such data. Thus, the originating designers have no specific alterations in their operations as a result of the event-driven design system, and operate in exactly the manner they would have in isolation. The single difference in their operation is that they provide their output data to the blackboard, rather than only to users or systems for which they were originally designed.

Also characteristic of this system is an intelligent search mechanism within each recipient designer for identifying the elements of the data on the blackboard that constitute sufficient satisfied conditions for the events required for their execution.

The advantage of this architecture is that the originating designers must have no *a priori* knowledge of the downstream use of their data, and their internal operations are not affected by the implementation of the event system.

The disadvantage of this system is that, as in Architecture I above, there may be a great deal of duplication in the blackboard polling, in that more than one recipient designer may be attempting to satisfy the same conditions at the same time. Particularly since the data have not been specifically tagged by each event, the search algorithm may require more complexity than any polling mechanism in the above architecture, and a substantial amount of data transfer and duplication.

4.5.3 Implementation Architecture III - Discrete Event Architecture Strategy

In order to provide a solution to the limitations of the two architectures described above, a third architecture is defined here. This method has a new component - the Event Manager. The Event Manager is a separate module from the blackboard, and it is separate from the knowledge-based systems.

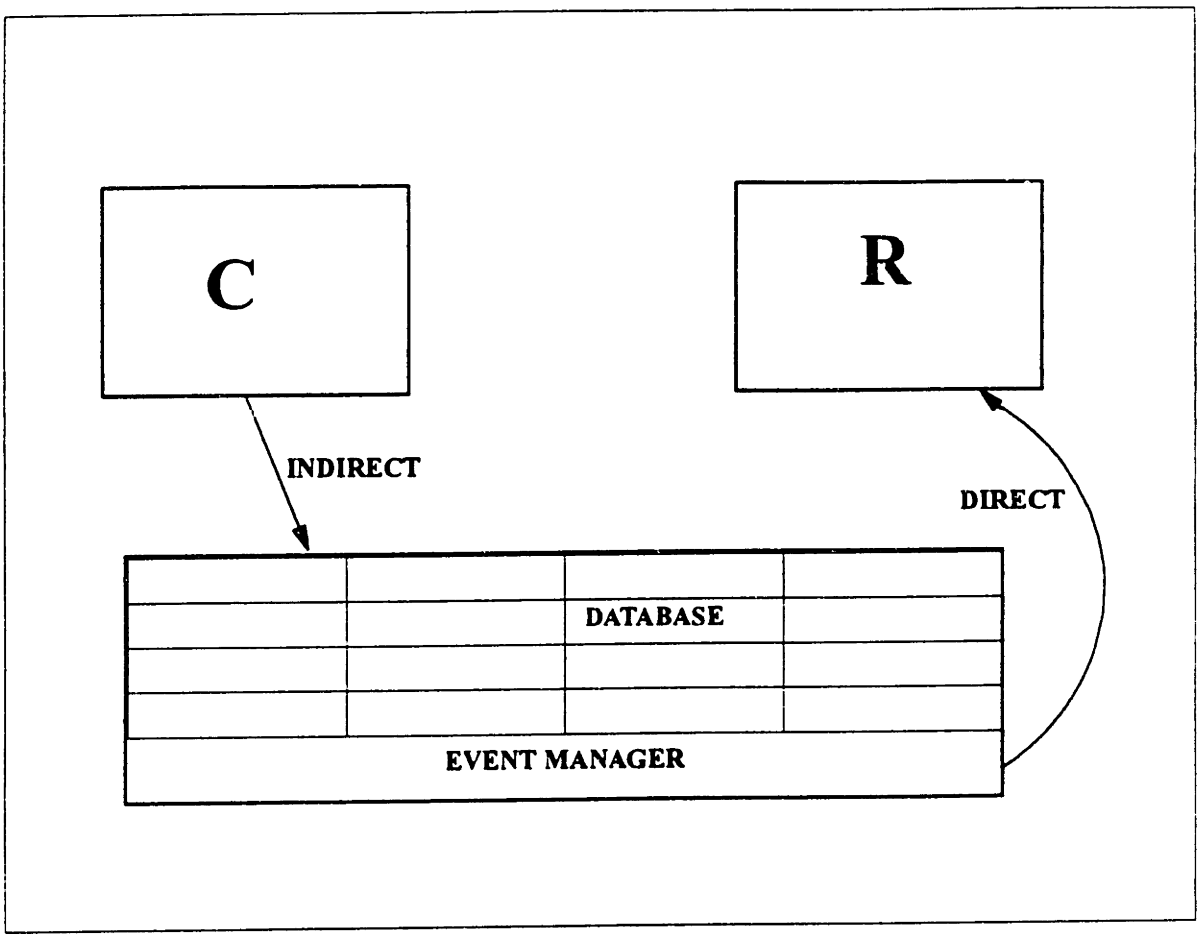


Figure 4-14: Discrete Event Architecture (Using the Event Manager) for Event-Driven Knowledge-Based Design

Characteristic of this architecture is the use of a system to monitor the existence of events, and to inform recipients thereof. As such, there are direct links between the event manager and the designers or knowledge-

based systems (recipients), but only indirect links to the blackboard from the contributors. The Event Manager is tasked with identifying the existence of any or all of the event conditions, and of managing the notification of all of the relevant recipients or affected parties.

The advantages of this approach are many:

(i) Independence of the knowledge sources (contributors and recipients) from the blackboard and from each other. The absence of direct links between these systems provides a flexibility and agility that allows for modularity of design, and ease of incremental development.

(ii) The Event Manager can take into account the state of the recipient at the time of the recognition of the event. For example, if a distributed design organization spans time zones, and an event occurs in one part of the organization while potential recipients are not available, records of the events can be made by the Event Manager and information relative to these events can be made available when the appropriate recipients are available. Conversely, if a recipient is in a specific state that makes the knowledge of a particular event critical, priority on the event can be increased by the event manager, so that appropriate dispatching of the event can take place.

(iii) New event-driven knowledge-based systems can be added to the design organization without necessarily changing the operation of any other designers. By informing the Event Manager of the important or associated events, the system can be adaptive without any re-design.

(iv) The Event Manager can span databases in terms of event recognition. That is, rather than limiting the scope of the definition of event conditions to those elements of the blackboard, a heterogeneous database can be used by the Event Manager. This database could include CAD files, Product Data Manager-generated or retrieved data, and blackboard data.

4.6 The Event Manager

The function of the event manager is to provide a vehicle for representation of events and for execution of the two primary active functions in event-driven systems. These two active functions are (a) event detection, and (b) event action initiation (reaction.)

4.6.1 Event Representation in the Event Manager

Based on the class definitions described above, the event manager must contain a class hierarchy representing events of the various types, and allowing for instantiation of events. One mechanism for event definition (and subsequent instantiation in the event manager's repository of events) is the manual entry or definition of events by a person. This would create a system of static instances of event definitions.

Following the basic definition of events within the specific design system's context, the system would be implemented whereupon events that take place are detected and instantiated. This mechanism would constitute dynamic instantiation. The difficulty with dynamic instantiation in the context described here is that the instances would be instantiated from the class definitions, not

the static contextual instances.

Hence, an alternate mechanism is proposed here. The basic hierarchy of event representations would follow the following sequence:

Provision of the generic event class definitions by the event management system;

Specialization of the event management classes into context-specific sub-classes, representing those events that are meaningful to the people and systems in the context of the design problem at hand;

Execution of the system, during which *dynamic instances* of the classes and sub-classes are created by the event manager.

This is depicted functionally in Figure 4-15:

This allows for the evolution of both new event types and on-line generation of event instances as the continuous progression of the design process occurs.

A further generic representation mechanism required in the Event Manager is that of an event domain or scope. (Allen, 1992) The event domain defines the range of contributors and recipients (or Allen's producers and consumers) for which each event is meaningful, and to which the event applies. As such, a registration process is necessary, in order to bound the effect of the event, and identify specific functions or actions that are to be taken upon the instantiation of an event. Note that contrary to Allen's requirement that both contributors and recipients register for the events in which they participate, the event representation here only requires that recipients are specified explicitly. That is, contributors are not necessarily aware that they are participating in, or have

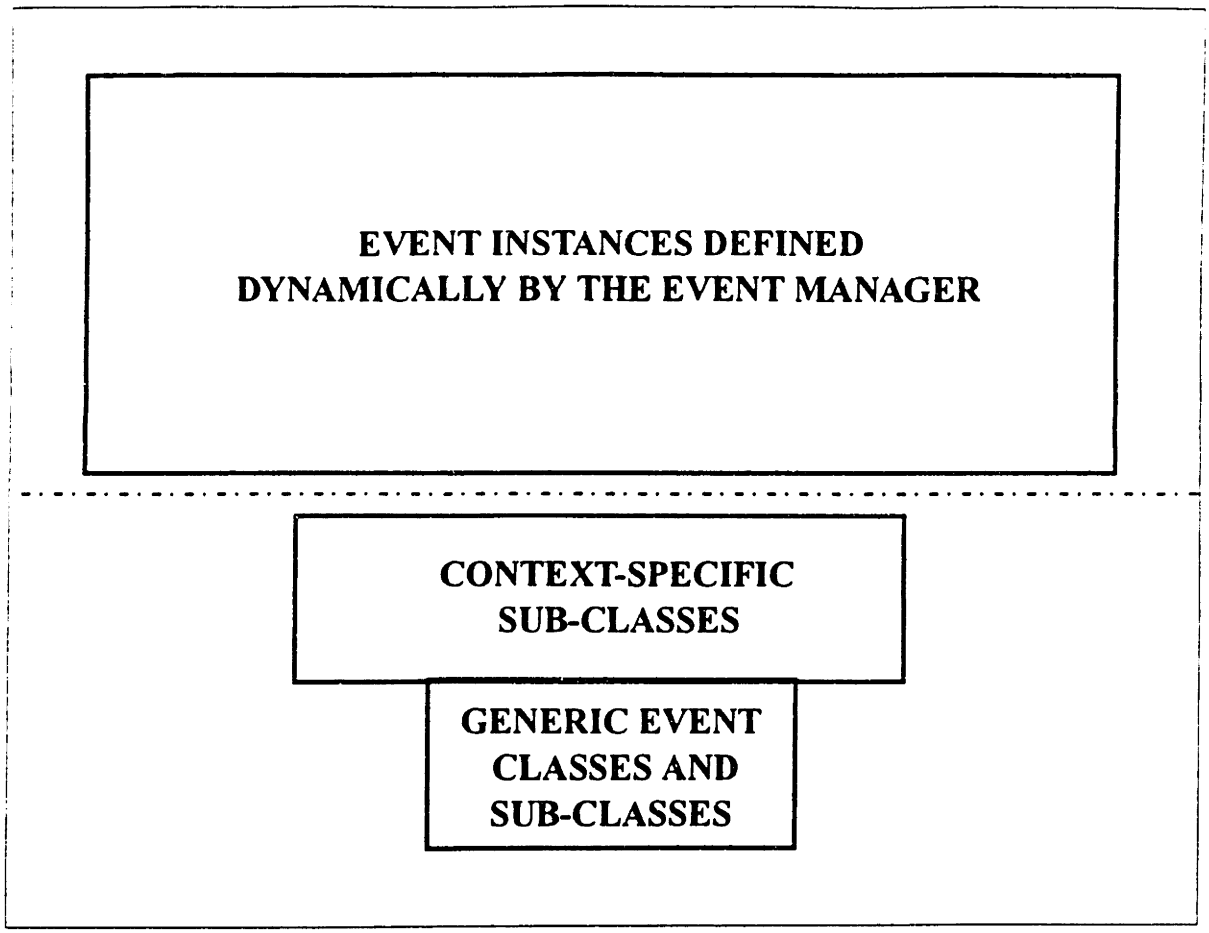


Figure 4-15: Event Instantiation Taxonomy

caused an event to occur. The nature of this morphology is such that the event is precipitated upon the *confluence of all relevant initiators or stimuli,* including data from a variety of disparate sources.

As such, the "conditions" of a general event are a set of data or other stimuli (such as the detection of a method or action) such that the contributors are actually creating the data, not the event itself.

It is incumbent upon the event manager to collect all of the relevant conditions for each event, and to make transparent the connection between recipients and contributors. This is necessary in order to accommodate the dynamics of the design process, in such a manner as to allow a variety of sources for data, and consequently to allow for a variety of original sources for events.

The representation of the event management dynamics requires a flexible initiation mechanism, such that events can be detected and resulting actions initiated without having to prescribe a sequence or schedule for such activity. The inherent object-oriented "method" provides such a representation. It is possible to construct a recipient registration mechanism that is dynamic, allowing events to be defined without "hard-coding" or fixing the responses. In this manner, additional recipients can be identified, and an evolution of the event can take place in an incremental manner.

A basic framework for such a registration process is provided in Table 4-3. In this framework, known as the Event Registration Mechanism (ERM), a template for the definition of the event and the recipient response is used to define the event name and the required action upon event detection.

Event Name:	Associated Actions:

Table 4-3: Event Registration Mechanism (ERM)

The event is structured according to the class structure as described above, with associated tools and utilities for adding conditions, actions, or modifying any specific event parameters (such as period, or threshold values, as appropriate.) The ERM described above is one example of a tool for

manipulating the event definition.

Note that because of the object-oriented structure of the events, it is possible to modify the event definitions in an on-line fashion. This, of course, creates a configuration management issue, but the programmatic mechanisms exist, since the event is not initiated until the appropriate conditions are recognized. As such, the list of actions can be dynamic; each instance of an event detection generates a specific action based on the elements in the action list at the time of the event occurrence. This is an important characteristic because it allows for *adaptive event development*, which would be a facet of a system that has meta-knowledge about event behavior and the progression of the design process, and would allow for a self-learning and adaptation of the event system. For example, if certain events were occurring frequently, and this was having a deleterious effect on the design process, an additional action would be added to the action list of the events to take some management action to correct the behavior.

4.6.2 Event Detection and Response

A variety of mechanisms have been proposed for event detection. These include the following three models:

(a) the event-loop model: The event loop observes a stack to find any pending events, and dispatches an appropriate event-handling routine;

(b) the event-callback model: The application registers a callback function for each pre-defined system that contains intrinsic knowledge of how to respond to certain events. Whenever the event occurs, the system is notified, based on its registration, and the system performs its appropriate action.

(c) the hybrid model: a combination of the above two models. (Berson, 1992)

In one implementation in the context of production planning and control, the event manager concept (called the "event router") is implemented on the basis of the event-loop model, and contains explicit knowledge of actions, objects, and systems that are affected by each event. (Joyce, 1988) This architecture is depicted in Figure 4-16.

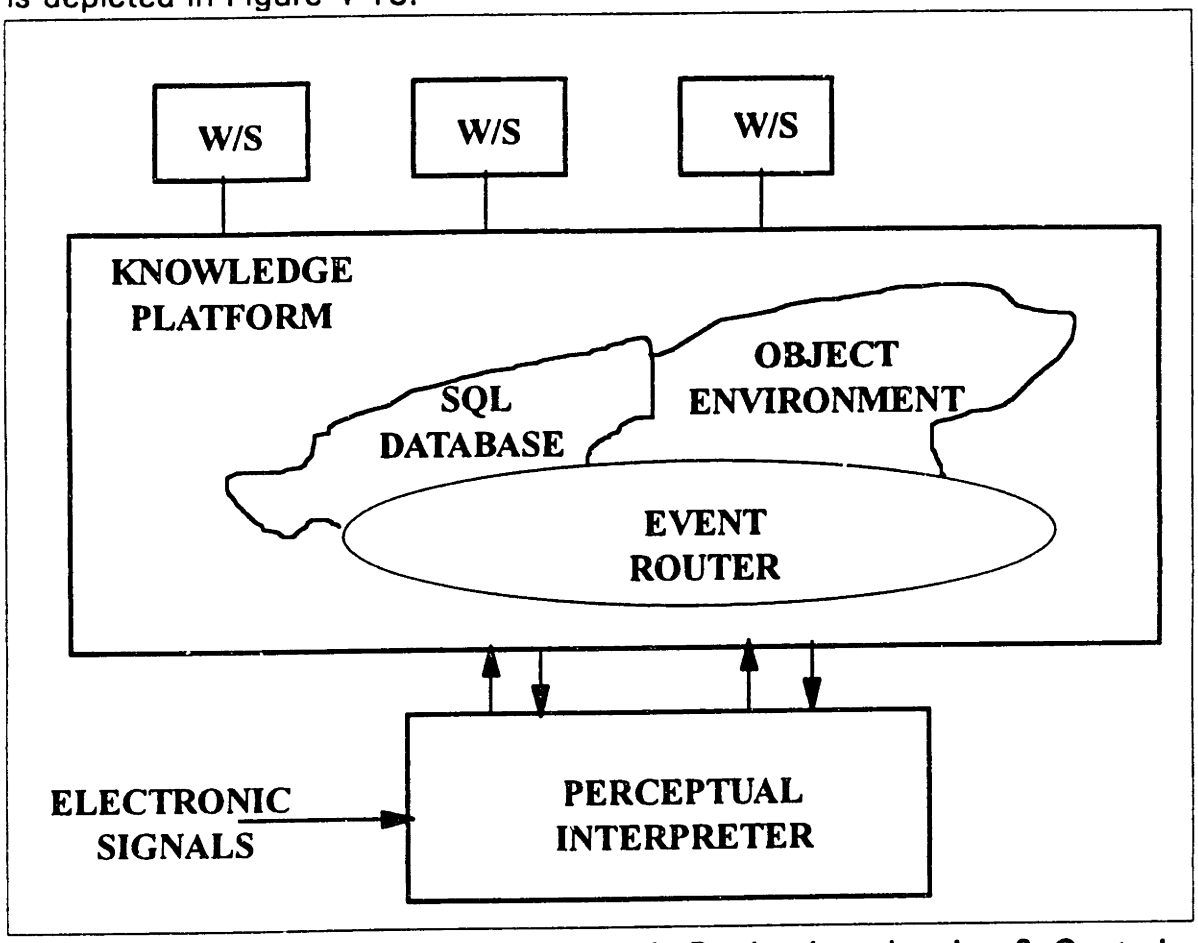


Figure 4-16: Event System for Joyce's Production planning & Control System (After Joyce, 1988.)

In another example implementation, the event system is based again on an event loop, which is described as follows:

"The structure of an event-driven program can be visualized as a central controlling module with a *main event loop* that continuously monitors and processes actions. .... Computer systems that support the event-driven approach often provide a toolbox for maintaining event-related information on a queue. The primary function of the main event loop within a system of this type is to extract events from the queue and to delegate work to event *handlers* on the basis of event types." (Forde, 1989) This architecture is depicted in Figure 4-17.

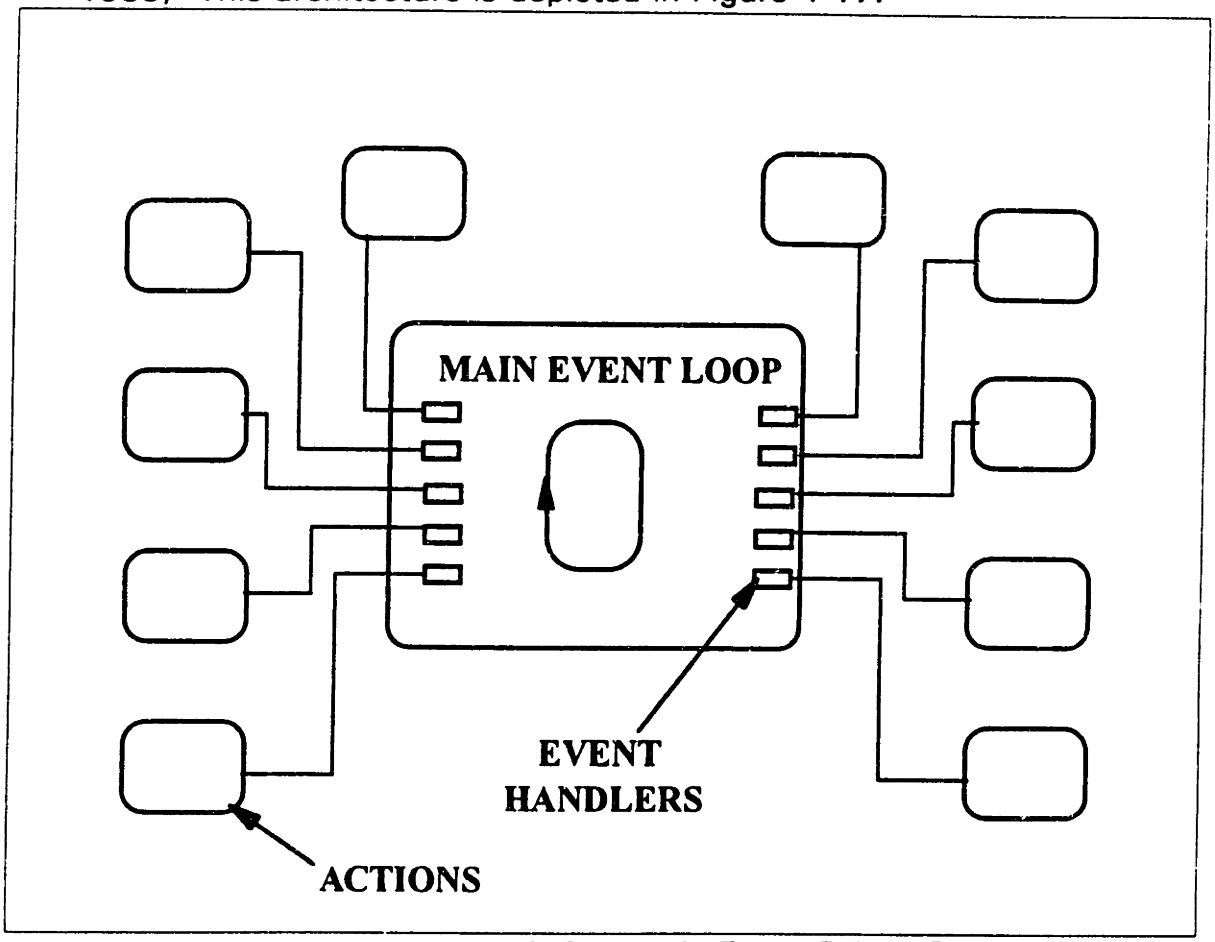


Figure 4-17: Structure of Forde & Steimer's Event-Driven Program

Note that in both of the above examples, several characteristics are evident:

a. These systems define a single-user orientation;

b. The events are localized within a given user's environment;

c. The event reaction mechanism depends on a sequential loop,

following a predictable path, even if the events follow an unpredictable path.

A much richer model of event management is presented in a framework known as CIMOSA, Computer Integrated Manufacturing Open System Architecture. (Goranson, 1992) This framework separates "internal events" from "external events." That is, external services or stimuli are dealt with in a different manner from internal events in the system. Further,

> " . . . an event need not include the transfer of information, or control of an object. The contents of an event are handled separately from the pure event itself. This purity allows the ID [integration domain] to track, manage, and record the event fabric without worrying about shuttling the contents around. The actual shuttling of the contents is performed by either EE [execution environment] or AA [application architecture] services as appropriate." (Goranson, 1992)

This architecture is depicted in Figure 4-18.

In the CIMOSA model, the focus on manufacturing generates a natural acceptance of the existence of events, and the need to react appropriately to internal and external events. In design, there is similar behavior, with respect to the generation of and need for an ability to react to events. In the CIMOSA definition, the Integrating InfraStructure ISS, defined here as the Event Manager,

> " . . . should accept events and be able to relate their definition to available domain models." (Querenet, 1992)

The framework developed here for event detection and response is the basic

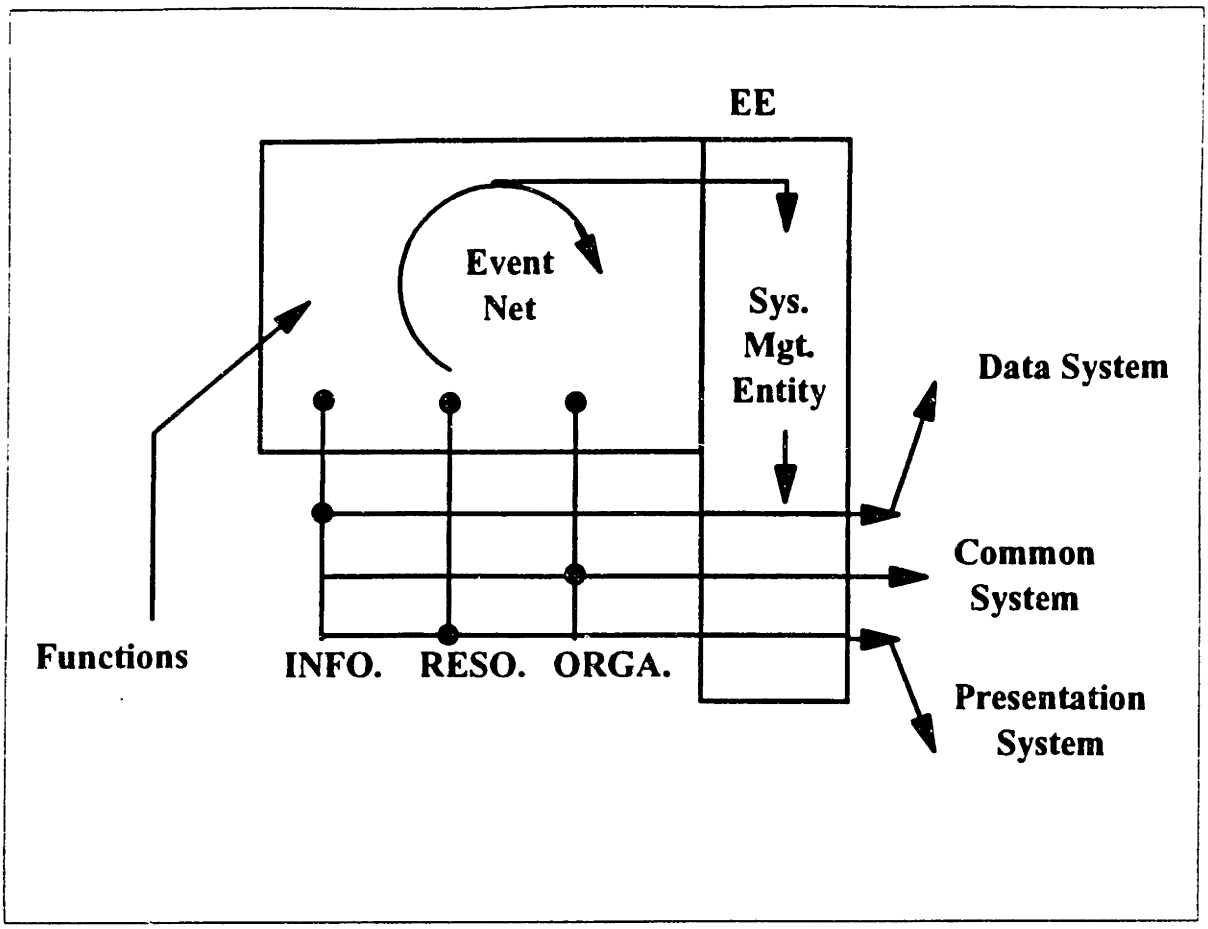


Figure 4-18: CIMOSA Application Architecture (After Goranson, 1992)

object-oriented mechanism for initiating "methods" based on the existence of certain conditions in the class definition. Invoking methods in object oriented systems follows different morphology from static languages. In typed object-oriented mechanisms, the methods to be invoked have the capability of being bound statically (equivalent to a procedural language function call) or dynamically, using a virtual procedure declaration. When actions are different for each sub-class, they are known as *specialized* methods. In order to provide an efficient run-time specialization mechanism, object-oriented systems use dynamic lists of pointers to virtual functions (i.e. dynamic methods.) In this manner, the specifics of the actions need not be known at the time the class is defined or stored. In this manner, there is no run-time searching of function tables because of the use of indirect references to pointers. In the case of "multiple inheritance," those object-oriented systems that allow sub-classes to

have multiple parent classes, the mechanism for dispatching methods requires a prioritization scheme. (Booch, 1994)

Each event class, then, operates independent of the other event classes. When an event is detected, the Event Manager creates a dynamic instance of the event, and initiates the requisite actions. As part of a housekeeping activity, it is necessary to perform some system-level functions, such as provide a date and time stamp for the instances.

4.7 System Infrastructure

It is necessary to provide a certain number of basic infrastructure capabilities within the framework of the event-driven knowledge-based design system. In order to support the broader organizational requirements of the design process, a system that allows for heterogeneous integration of design processes, systems, and events is required.

4.7.1 Local Event Management

In a local user environment, it is necessary to allow for the generation of, and response to events that are contained within the purview of a local user. Such events might include the violation of standards, completion of a sub-task, etc. The event management framework must at once coordinate between designers and also allow the individuals the freedom of working without constraints imposed unnecessarily by a system-wide manager. As such, the event management at the local level are contained within a user's workspace, in a single processor environment.

Local events are tracked by a local event system, resident on the user's workstation, and having a domain or scope limited to the actions of the individual user. As such, the granularity of events can be low-level, including such schemes as keystroke-driven events. Such schemes are predicated on the ability to provide event-management systems from within specific applications.

Since most user's applications in a design environment are commercial products, this capability varies greatly from application to application. For example, some CAD systems allow for a degree of interrupt-driven applications to be developed by users, and used in a dynamic manner. The definitions of interrupts range from user-selections from a menu, to keystroke trapping. Other CAD systems allow for certain limited application functions to be trapped, and user-developed programs to be called automatically. This enables the development of a certain degree of event-driven application invocation, and allows for flexibility of event-action or response based on the functionality of the user-developed application. Yet other CAD systems allow for a rich set of event-driven user-applications to be included, based on flexible event-trapping mechanism within the basic architecture of the CAD system itself. This allows for a full set of local event systems to be implemented.

These mechanisms all allow for the development of a local event system, which provides event-driven knowledge-based capability to the local user. Such an architecture is depicted in Figure 4-19.

Note that in this architecture, due to the constraints imposed by the CAD applications, the event system is partially defined within the application (via the application programming interface calls and programs) and partially external to the application, in the form of the user application. There is no explicit Event Manager in this context. Therefore, if management of events is to be

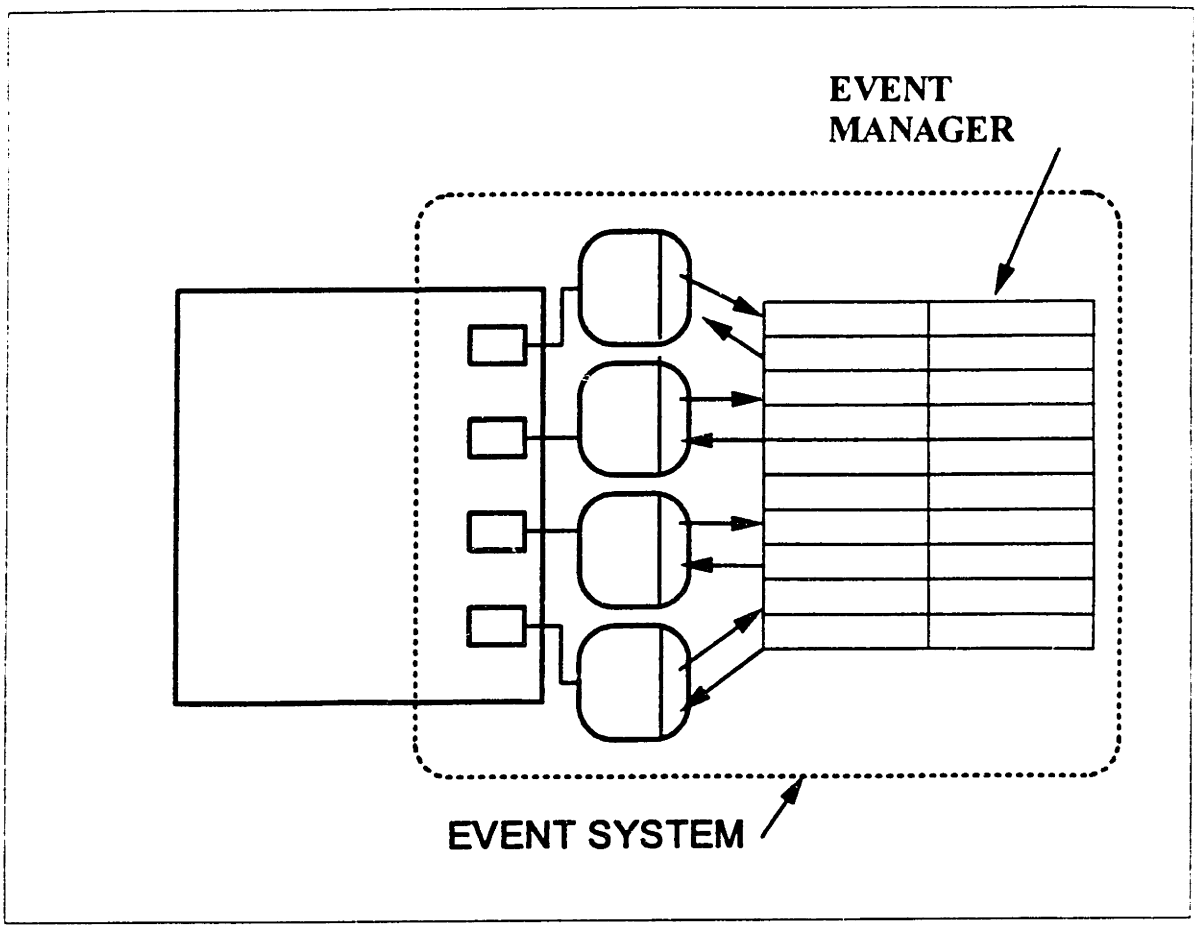


Figure 4-19: Local Event Management System

accomplished, each application that is invoked must have a mechanism to interface to an external event management system, residing locally. Such an architecture is depicted in Figure 4-20.

4.7.2 Regional or System Event Management

When designers must interact with each other, as is standard design practice, it is necessary to consider events that occur as a result of one or more user's actions, and for whom the recipients may be many. As such, it is necessary to provide a systems view of the event process, and to provide an infrastructure that supports such a view.

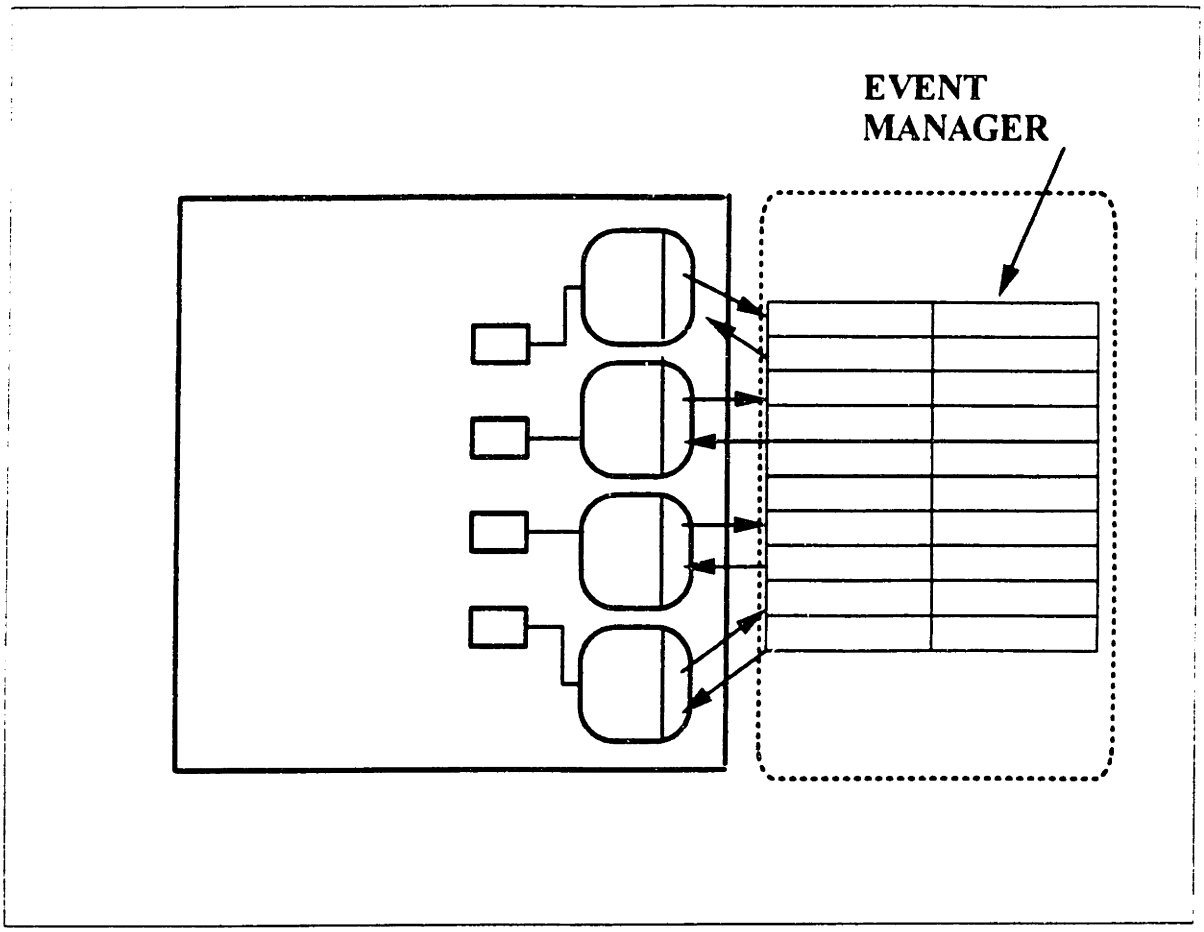


Figure 4-20: Partial Event Management Architecture With Internal Application Event Handlers

The basic mechanism for providing such an environment is to distribute the event manager's object-oriented database across a heterogeneous client/server architecture. This framework allows for the communication between designers and the systems, and allows for CPU-load balancing. Two alternate configurations are depicted in Figures 4-21 and 4-22.

In the centralized database model, the object database is located entirely within the server environment. This reduces the amount of distribution requirement left to the object manager, and increases the amount of communication required between clients and the server.

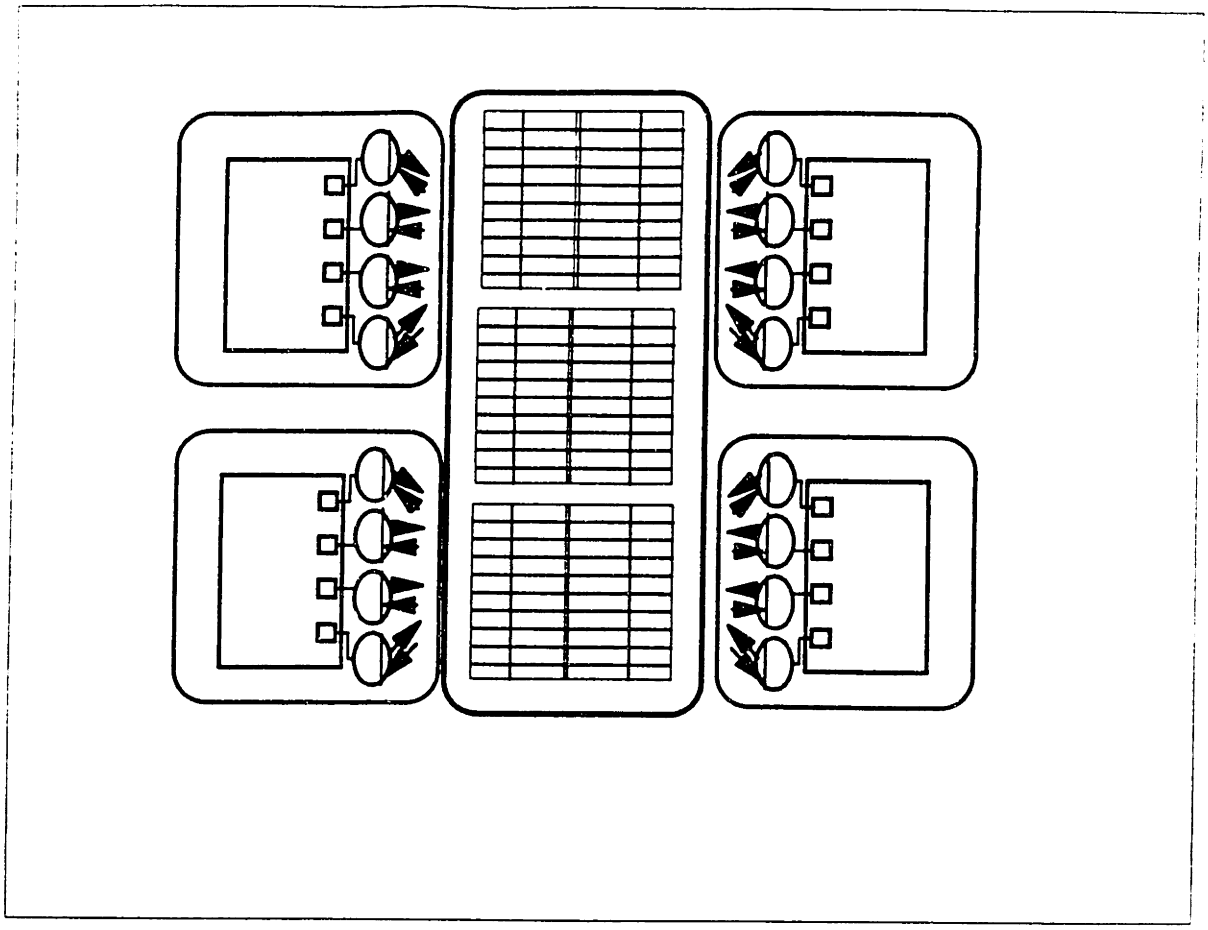


Figure 4-21: Event-Based Client/Server Architecture with Centralized Database

Notice that the event manager database in the distributed example is distributed between the clients and the server, allowing for localization of confined tasks, and the sharing of common tasks and objects.

The distributed model is preferred, since it offers the ability to provide a local event management system as well as a regional or system event capability, thus providing users with the ability to have self-contained event-driven knowledge-based systems as well as having the influence of others involved in the regional or system-wide design process.

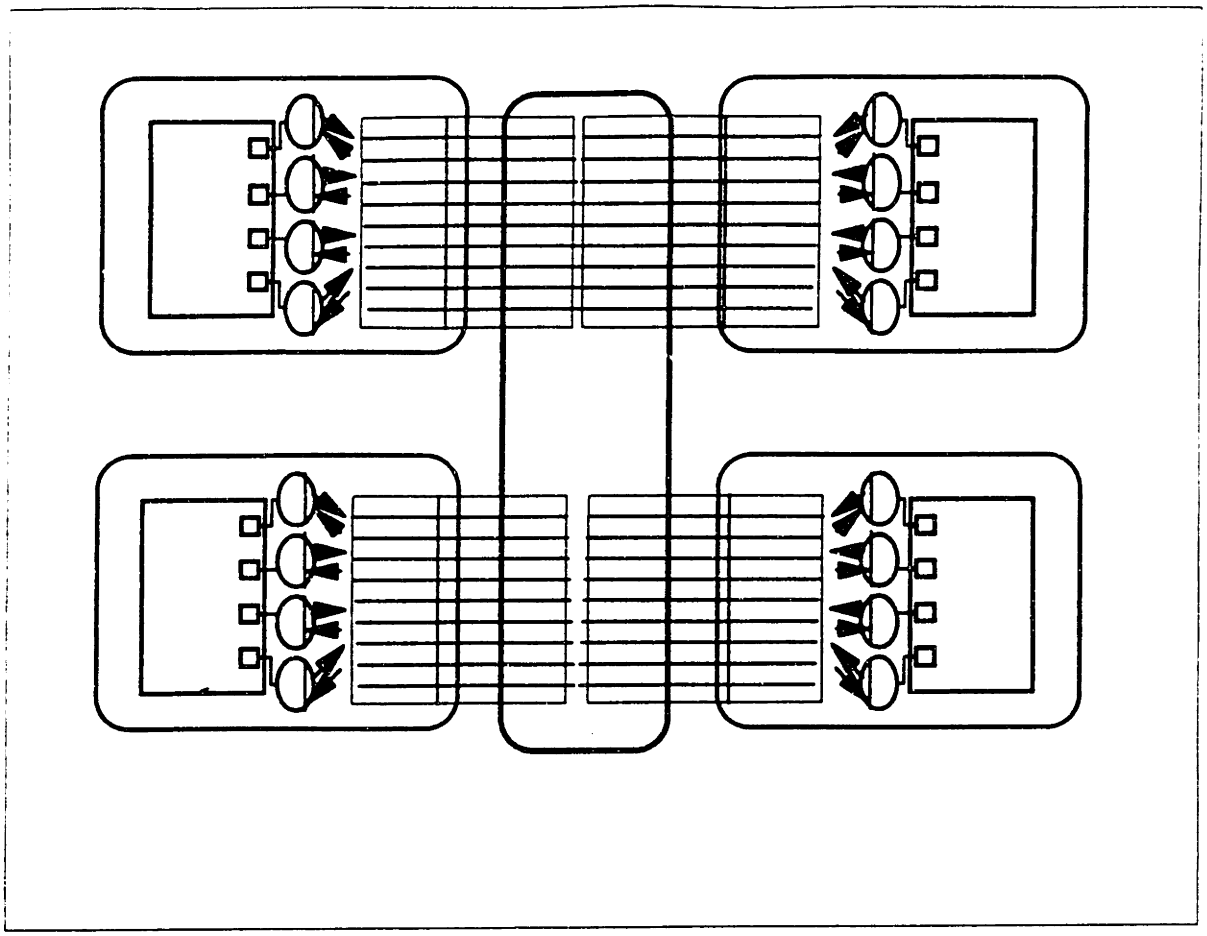


Figure 4-22: Event-Based Client/Server Architecture with Distributed Database

4.7.3 Global Event Management

Once the entire design enterprise is considered, it is necessary to manage the transition between system-level event management, and the global design process. This is accomplished through the definition of global servers, which contain meta-level capability about events that cross system boundaries. This is accomplished by distributing event management in a hierarchical fashion through the organization.

An architecture for representing a global event management structure is illustrated in Figure 4-23.

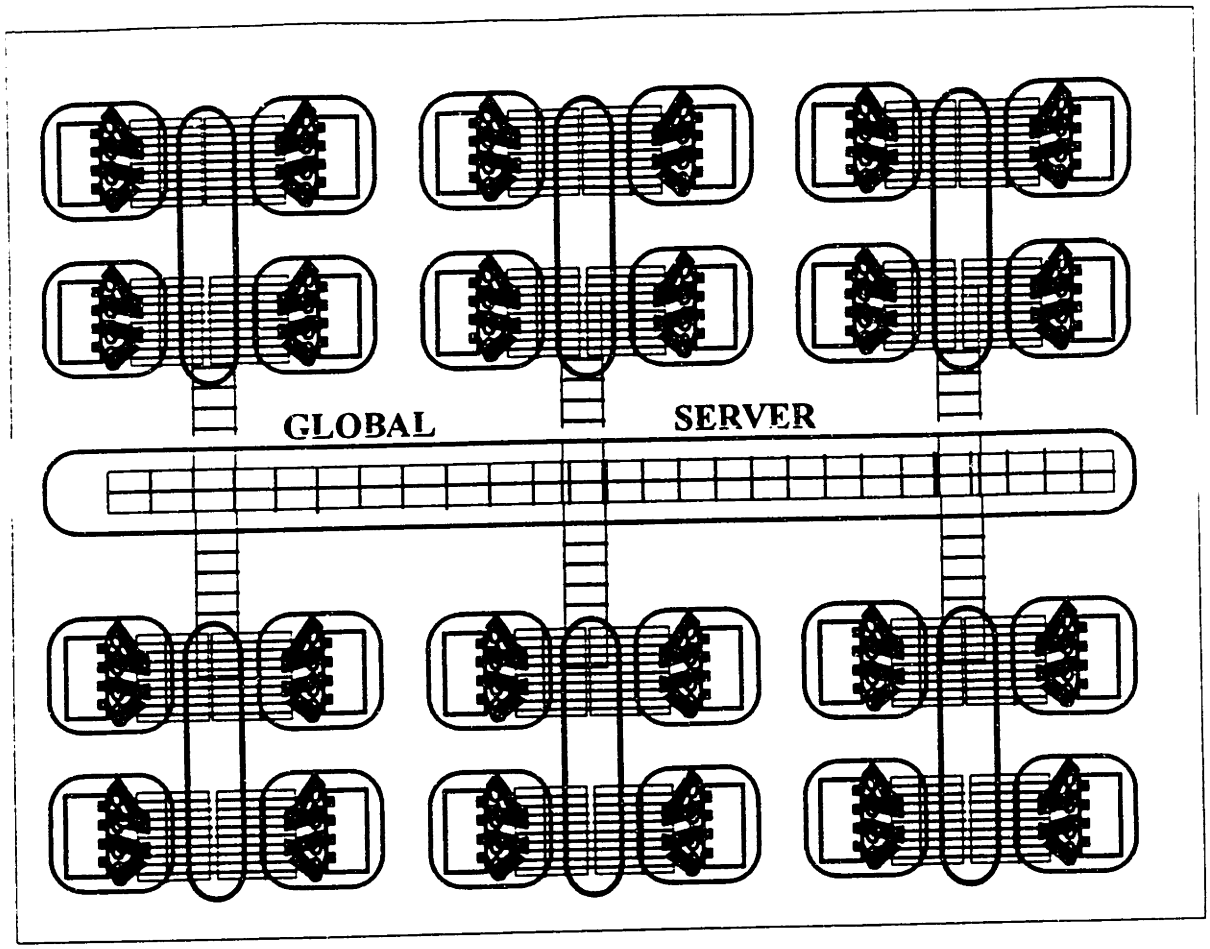


Figure 4-23: Global Event Management Architecture

Note that this environment supports both a single entity as the global organization or a variety of organizations, participating in the same design effort. This facet is significant because it allows for the distribution of events across corporate or organizational boundaries without sacrificing the integrity of the event management process.

4.8 Domains of Event-Driven Knowledge Bases

Most conventional knowledge-based systems act as a result of initiation by a user, whether directly or indirectly through the use of a call from one program to another. Event-driven knowledge based systems, by their nature, have the characteristic that they are intended to provide information or decisions that are predicated not on a user's initiation of a request, but rather on the occurrence of a particular event. Consequently, there are some types of event-driven knowledge-based systems that have functionally different roles from the user-initiated variety. While it is true that some knowledge-based systems may have value regardless of whether they are initiated by an event or by a conscious act of a user, others will be far more valuable as active participants in the design process.

Coordination

As a means of coordinating design activity between individuals, groups, and organizations, events represent significant changes of states that provide a universal protocol for understanding and coordinating the dynamics of design processes. This coordination can follow the hierarchical model, as described in Chapter 2, or it can follow an ad hoc process of asynchronous interaction.

Note that transcending the process of simply provide messages between people to activating contextual knowledge bases allows for the progression beyond making people aware of the significant events in the balance of the design process to an understanding of the *impact* of such events.

Feedback

One of the most difficult tasks in a multi-party design operation is the

consistent provision of contextual feedback. The ability of event-driven knowledge-based design systems to consistently provide appropriate feedback, regardless of whether the feedback is immediate, delayed, or dependent on subsequent activity, leads to a much higher degree of communication than is otherwise possible. The provision of automated generation of feedback to the appropriate designer is a technology that elevates the overall performance of the design team, due to the inevitability of collaboration between parties, as opposed to the potential isolation of individuals and groups.

Review

One of the most important, yet least interesting activities in an engineering design process is the review of the details and results of preceding design activities. For example, calculations must be checked, while drawings and other design documents must be visually examined. As a source of added cost in the engineering design process, mistakes and changes are a significant factor, both in terms of time and direct expenditure of resources. As such, the closer to the activity itself the checking or review process can occur, the smaller the impact of any mistake or change. Even with the availability of knowledge-based systems for review, it is still incumbent upon the engineer or designer to initiate this activity. While procedural steps can be taken to require that the automated review process be initiated, there is still a propensity for error or omission, albeit a much reduced risk.

Event-driven knowledge-based review systems provide the opportunity to ensure that not only are local checks made at the earliest possible time, but also that checks requiring the use of information outside of the scope of the engineer or designer's views can be made in a seamless manner. These systems' execution can be initiated based upon a release or saving of

information, or an active analysis of the state of a local designer's design context. For example, a review system can ensure that the engineer or designer does not exceed the envelope of acceptable design ranges for given design parameters, prior to the use of any unacceptable values in local downstream assumptions. Thus, if a design condition is exceeded inadvertently, that condition will be detected by the event-driven system and its impact determined. An automatic correction can be initiated, or the decision can be left to the engineer, and the incorrect state will not be inadvertently used as a basis for components or systems in the engineer's future design activity, or that of any other impacted design engineer.

Automation

As an extension of the conventional design automation knowledge-based system paradigm, in this context design automation knowledge-based design systems can be initiated spontaneously as a consequence of certain events. This automation activity can be significant, since it follows the concurrent process in as close a sense as is possible.

When design automation systems are activated as a consequence of events within the design system, they initiate high-productivity action at a minimal expense. For example, an automated tool design process can occur at the time of release of the part design information. This can result in almost simultaneous part and tool development, thus reducing the overall development effort both in terms of labor and in terms of duration.

Advisory Systems

Notification of events beyond the scope of a user's purview can provide significant benefits, including prevention of activities that are based on

incorrect assumptions, better definition of uncertain information, and other important capabilities.

The ability to analyze and contextualize such information may require a great deal of knowledge, and it is in this focus that the event-driven knowledge-based design framework provides value. The activation of a design advisory knowledge base as a result of other's actions can lead to a greater degree of control in the design process, a higher level of performance and accuracy, and a greater degree of consistency within the design itself. These advisory tasks are active, and are not dependent on the user's ability to recognize a particular need.

4.9 Discussion

Issues in the development of event-driven knowledge-based design systems have been addressed. A variety of architectures have been examined, and alternatives for the implementation of connectivity, dynamic strategy, distribution of tasks, and basic initiation and response strategies have been presented.

It is clear that there are several options with respect to the implementation of event-driven knowledge-based design systems. The appropriate strategy and architecture for a given design organization and problem may involve the harmonizing of some of the above described alternatives, particularly in the context of existing organizations, physical infrastructures and human societies of designers.

5. Implementation Example

In this Chapter, an example event-driven knowledge-based design case study is illustrated. This case study is based upon the preliminary design process for automotive body design.

The case study has been selected because it has the attributes of clarity, from the perspective of contextual understanding, and necessary complexity, from the perspective of knowledge and interaction. That is, the problem domain, while technically complex, can be easily understood by engineers of many different varieties without requiring the reader to be a specialist in automotive body design.

It should be noted that this example is described here for illustration purposes, and the case study is not intended to be, nor is it presented as, a complete production-ready system. The basic characteristics of the problem domain are represented, as are characteristics of the event-driven knowledge-based design methodology. This Chapter is provided as a description of some of the issues and specifics encountered in an implementation of an event-driven knowledge-based design scenario.

5.1 Problem Domain Overview

In the automotive design process, one of the major technical areas is that of body design. In this area of activity, all of the vehicle's structure is designed, including such elements as the frame, the door panels, the fenders, the hood and trunk panels, the bumpers, the roof sections, all of the pillars and the connections between the pillars and the balance of the frame, and all other components of the vehicle structure.

One of the critical areas of this process is the surface definitions for the body components and assemblies. The criticality of the surface definitions stem from two sources: the aesthetics of the vehicle, and the manufacturing requirements for the parts. The exterior surfaces are designed with respect to aesthetic functionality, and the surfaces are then converted into mathematic descriptions for further design purposes and for manufacturing. Since much of the part manufacturing process is performed using Computer-numerically-controlled (CNC) machinery, it has become a requirement that part designs be defined in both a visual form (for human examination) and mathematical form (for manufacturing purposes.)

The definition of surface requirements typically evolves as one of the earliest design activities, in some cases forming the first engineering activity within the development process. Note that the product development process begins with tasks such as market research, concept development, early design, etc. Not until concepts have been approved, and specific designs articulated is the engineering process begun. A time-line is associated with the product development process, and the engineering timeline often begins with the surface requirements definition. This may occur at time t_0 = 36 months, in a typical development process. The requirements definition process may last between five and seven months, and currently involves the use of a clay model of the proposed vehicle. The clay model is used for design purposes as well as mathematical surface definition validation. The latter process is accomplished through Coordinate-Measuring-Machines (CMM's) which measure physical coordinates and allow for comparisons between these physical coordinates and mathematically defined coordinates from a CAD model. The culmination of this process is the release of surface requirements to a further detailed body engineering process for actual surface layouts.

Some of the issues involved in this early surface requirements process include

the development of dimensional tolerances for surfaces, development of allowable radii for tangent continuity within parts and between adjoining parts, acceptable flushness standards between parts, and other details associated with the body surface. Based on the specifics of the vehicle design, critical areas relative to the surface design (and future potential problems) are identified, and design and manufacturing specifications are established. These specifications and standards can then be used by downstream engineering processes for the detailed design of the parts and assemblies. The direct downstream recipients of these data (and consequently, those directly affected by the quality and timing of the data) include metal stamping operations, assembly feasibility analysis, design engineering, design quality, manufacturing(fabrication and numerical control departments), surface layout, initial and detailed body design, systems engineering, engineering analysis, and program cost & scheduling.

The basic process is illustrated in Figure 5-1. Note that within each functional activity a great deal of specific engineering design tasks are implied.

5.2 Applicability of this Problem Domain to Event-Driven Knowledge-Based Design

During this process, a great deal of engineering analysis is required, since the details that are designed in this stage will affect many body engineering activities in the downstream development process, including both engineering design and manufacturing. As such, the effects of the decisions that are made by individuals and groups in this initial process are at once significant and also in some ways invisible to the participants in this early surface development process. That is, the individuals may know that the decisions that they are making are significant, but they are not necessarily aware of all of the specific consequences, with respect to other' design groups and individuals. As such,

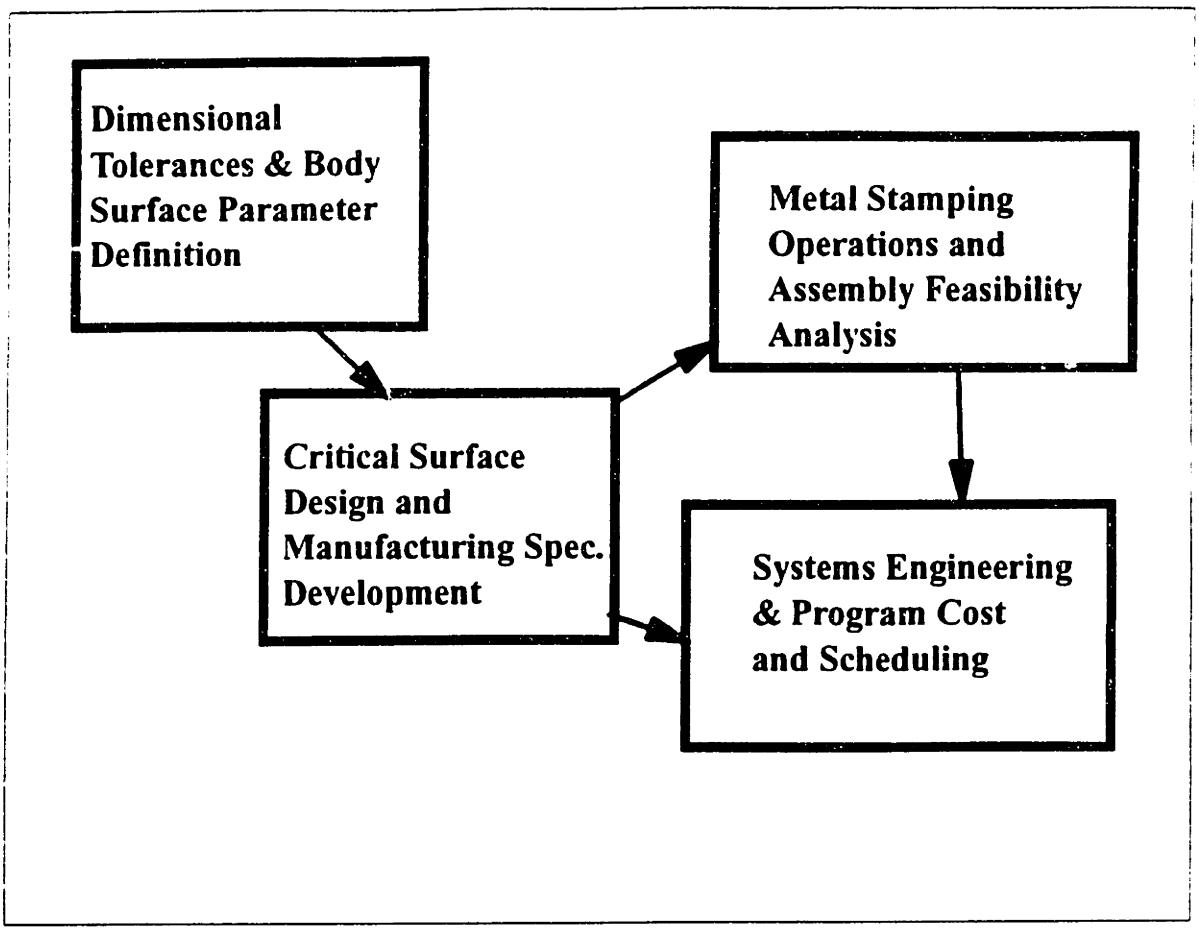


Figure 5-1: Basic Body Surface Definition Process

the ability to provide intelligence in the process at this stage can have highly leveraged effects. The basic decisions that are made at this stage are used as constraints and specifications for almost all body design activities.

The current methodology for providing review and checking functions for these decisions is to have a great deal of meetings, conferences, and physical sign-offs amongst the variety of engineering departments involved. This is both time-consuming and heavily bureaucratic, in that the signatures are treated as goals in the process, and the measure of progress and success is the approval cycle status relative to the time-line. While it is important to measure development chronology and progress, at least equal importance should also be afforded the context of the design activity.

Therefore, the process is a valid candidate for event-driven knowledge-based design, since it has the characteristics of having a distributed intelligence base (a variety of skilled engineers participating), a high value-added component (good decisions made in this process have a leveraged effect on the balance of the development effort), and timing of the identification of interactions and consequences is of primary importance since this task is on the critical path of the vehicle design process.

5.3 Specific Problem Decomposition

As an initial case study of process re-organization and automation in a part of the surface definition process, a small element of this process has been selected as an example of how a specific process can be changed with some degree of integrated knowledge-based design capability. This examination is performed from the perspective of the simulations described in Chapter 3.

The specific sub-process selected in this example is an element of the early surface requirements process described above. It is one of the support processes used to evaluate the feasibility of the design, from the perspective of the definition of the part surfaces. This process is the development of the test physical models of selected elements of the body surface. For the purposes of this examination, the process shall be defined as the Select Model Process (SMP).

The current SMP is defined in Figure 5-2. Note that the timeline for this process varies (by program and by manufacturer) between one month and three months. For this examination, an average of two months has been used as the baseline duration.

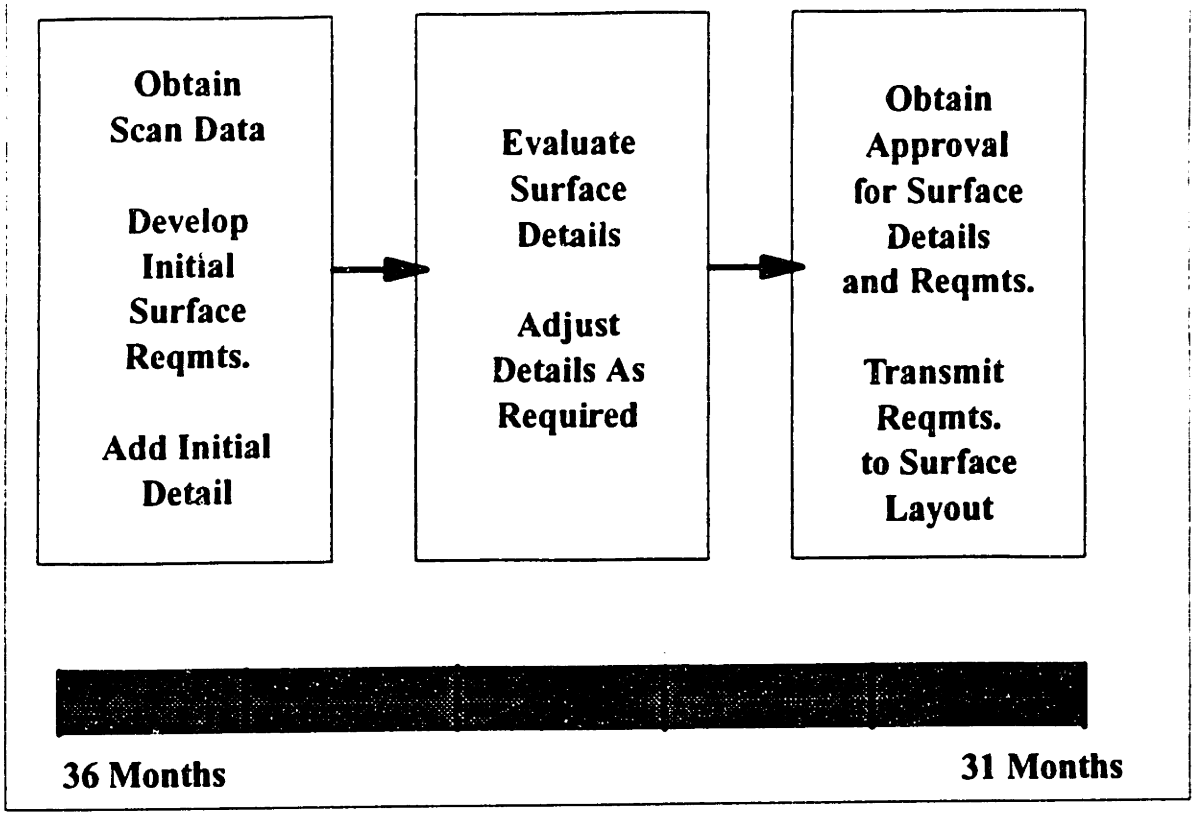


Figure 5-2: Basic SMP Process

The activities are defined as follows:

1. Identification of areas and parts for SMP process;
   Generation of part and area lists;
   Drawing generation;

2. Mark-up of previously generated drawings (from the CMM process);
   Detail of areas for SMP;

3. Generation of manufacturing drawings for SMP fixtures and assembly processes;

4. Assembly of physical models;

5. Mill physical models with CNC milling machines;

6. Complete physical model (surface finish) and prepare for presentation;

7. Review and approval of physical model;

8. Decision on acceptability of physical model;
   If OK, develop mathematical representations of the surfaces
   If NOT OK, then return to step 5 and revise milling procedure

Note that steps 5 through 7 are expected to be completed within a week. The quality of the final part surfaces has a significant impact on the schedule for this process. If the part is not accepted, then a full week can be added on to the development schedule. As such, the minimization of the duration of this process in other areas can be significant in the maintenance or improvement of the overall development time.

By examining the basic tasks in this overall process, a specific process has been identified for knowledge-based automation. This task, item #3 on the sequence of tasks, represents the design of fixtures and assembly sequences for the physical model. Since these fixtures follow a particular modular design, depending on the shape of the surface and the overall dimensions of the part, this "drawing generation" process can be performed by a knowledge-based system that would be initiated upon the release of the drawings of the areas required for physical mock-up (steps #1 and #2.)

This process has been simulated using the basic sequential model (as-is

process) and with the automated fixture design process. Figure 5-3 illustrates the simulation model for the sequential "as-is" process.

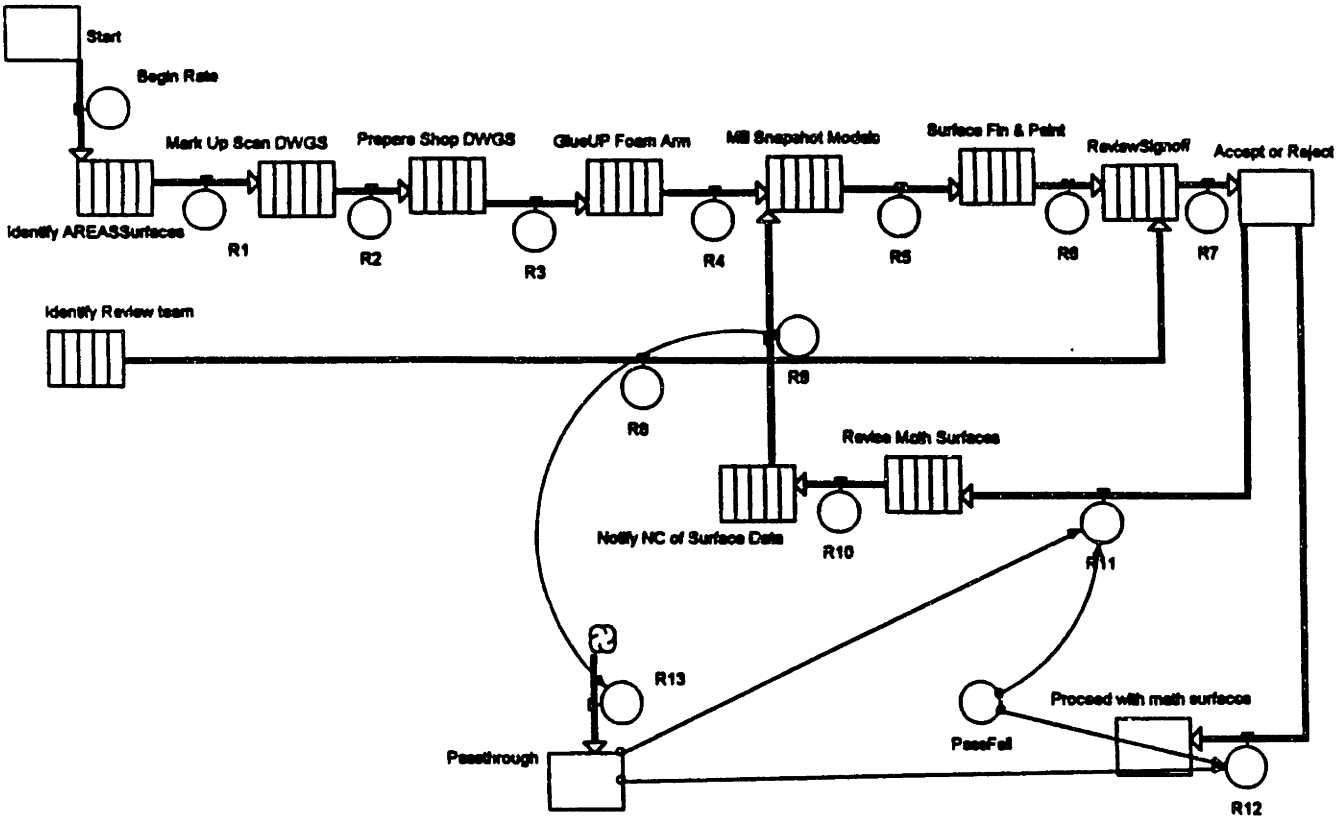


Figure 5-3: Simulation model for sequential SMP process

A simulation of the new process, in which the design of the fixtures for the physical models has been automated, has been developed. This simulation model is illustrated in Figure 5-4. Note that the automated task as having a very small duration, even though it is a part of the basic sequential process. This simulation provides the ability to visualize that a significant percentage of the duration of the task can be saved due to a re-organization of the process and automation of routine tasks. The savings for this simple process re-organization amount to more than one week, which is a 12.5% reduction.

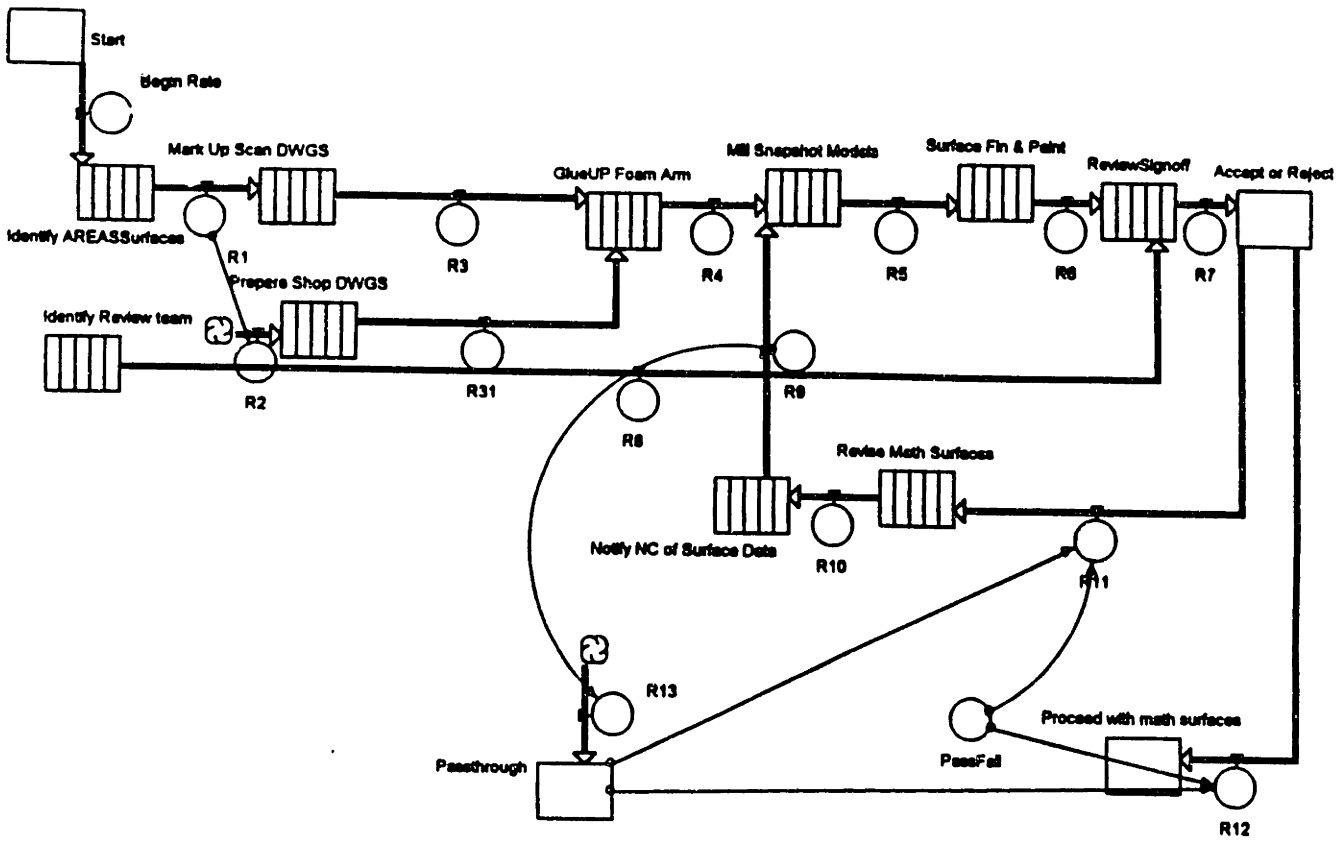


Figure 5-4: New SMP Process with automated knowledge-based design module for fixture design.

5.4 Bumper Beam Design Example

The design of the bumper beam for the front body section of the vehicle is influenced by a number of factors, including the required maximum impact force capability, the surface shape of the vehicle body, the outer dimensions of the vehicle, the maximum allotted weight of the bumper, the bumper fastening mechanism, and other design and manufacturing issues.

In this example, the influence of changes in the body shape and dimensions, combined with changes in the materials, configuration, and dimensions of the bumper will be examined, in the context of an event-driven design scenario.

Here, there are two design engineers, working on different parts of the vehicle concurrently, without direct knowledge of the influence of each other's activities over the other. As such, the example has been established with two CAD workstations, each with the basic toolkit of the CAD system, integrated knowledge-based design capability, and an event management system.

The architecture of the event management system is prototyped in three configurations, to illustrate the variety of mechanisms in which an event-driven knowledge-based system may operate, and the effects thereof.

The basic design tasks of the first designer involves the layout of the front section of the vehicle, including the basic surface shape of the body section immediately under the engine grille. It is this section that serves as the surface of attachment for the bumper. In addition, the first designer is responsible for the detailed outer body dimensions. Note that all of these dimensions and design criteria are part of the early surface design process, but the values arrived at in this stage of the design determine the major dimensions and characteristics of a variety of body (and non-body) components. One such influenced component is the bumper system.

The bumper system consists primarily of a bumper beam, which is used to absorb the impact load, and the bumper facia, which is the outer, more decorative part of the bumper, forming the primary visual identification of the bumper. Other important aspects of the bumper system include the fastening mechanisms. This example deals solely with the bumper beam, the structural

element in the assembly.

Note that in a parametric design situation, it is possible, through relations and assemblies, to create a purely mathematical relationship between one component in an assembly and other components in that assembly. As such, if the only variations in the parts involves geometric relationships, then many commercial CAD systems allow for the inclusion of such relationships, *vis-a-vis* placement and dimensioning of components in an assembly. Clearly, the bumper system can be thought of as an assembly whose design is driven by the surface shape of the exterior of the body. However, as described above, the bumper design is influenced by many factors, and it is not possible to draw a direct relationship between the major dimensions of the bumper beam and the geometry of the body surface alone. That is, while the geometry of the body surface is certainly an influencing factor, and certain cross-part dimensions are required to match, the values of the bumper beam dimensions and configurations are determined by a set of design rules and heuristics, as well as some rigorous mathematical and engineering equations defining load-carrying capability and crash behavior of the beam. As such, it is not possible to use the geometric relations built into many CAD systems to perform a bumper beam design. *Knowledge* is required - knowledge of specifications, constraints, requirements, manufacturing, performance, materials, and a host of other engineering considerations.

5.4.1 Design Problem

The complexity of this problem has been reduced here for the purpose of simplifying the context of the implementation case study. In essence, the bumper beam design problem involves the design of the beam geometry and the materials that comprise the structure. The basic design geometry consists of an exterior trajectory, and a cross-sectional configuration. These basic issues

are illustrated in Figure 5-5.

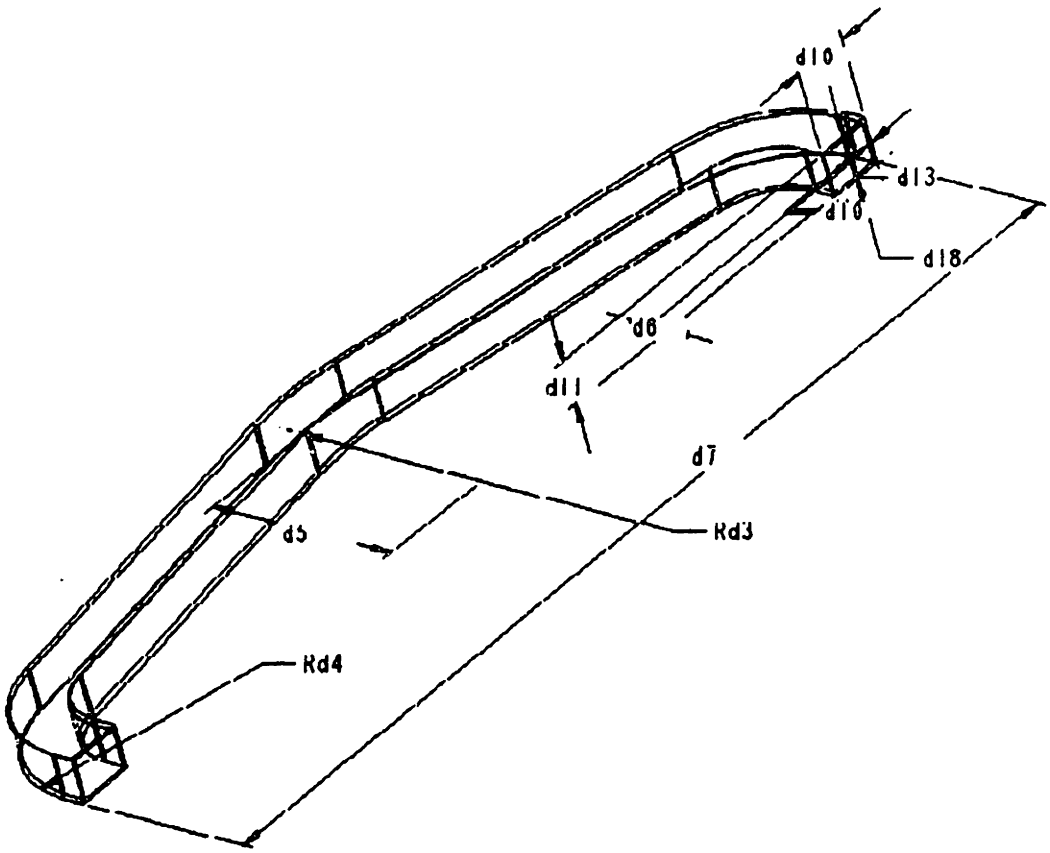


Figure 5-5: Basic Bumper Design Parameters

This problem can be defined in the following manner:

> Design a bumper beam that will withstand the impact of a collision of a specified speed, such that the bumper beam follows the geometric shape of the front of the vehicle, meets fastening requirements, and does not violate any specified constraints (such as maximum weight and cost.)

The bumper beam designer uses the shape of the front of the vehicle as a guideline for the development of a basic shape trajectory. Other design inputs

include a variety of vehicle specifications, such as maximum crash speed requirements, allowable bumper beam weight, expected cost, and manufacturing issues. Therefore, one of the primary issues in this design problem is the strength of the bumper beam, and its ability to withstand the required load and impact forces. Another major issue is the geometry of the bumper system, and the coordination of the shape of the bumper with the design of the balance of the vehicle body. (Note that the facia, or exterior, is designed separately, as a cover for the bumper beam, in order to provide a consistent aesthetic with the exterior vehicle body design.)

The four primary dimensions of interest in the initial beam design are:

major lateral dimension
vertex
cross-sectional width
cross-sectional height

The major lateral dimension and vertex are illustrated in Figure 5-6.

The cross-sectional depth and the cross-sectional height of the bumper beam are the outer dimensions of the cross-section. These are illustrated in Figure 5-7.

In addition to the basic dimensions, the configuration of the internal support elements within the beam's box shell is required. The configuration selected is essentially one of a set of allowable configurations, including:

Simple box beam
Box with single vertical support member

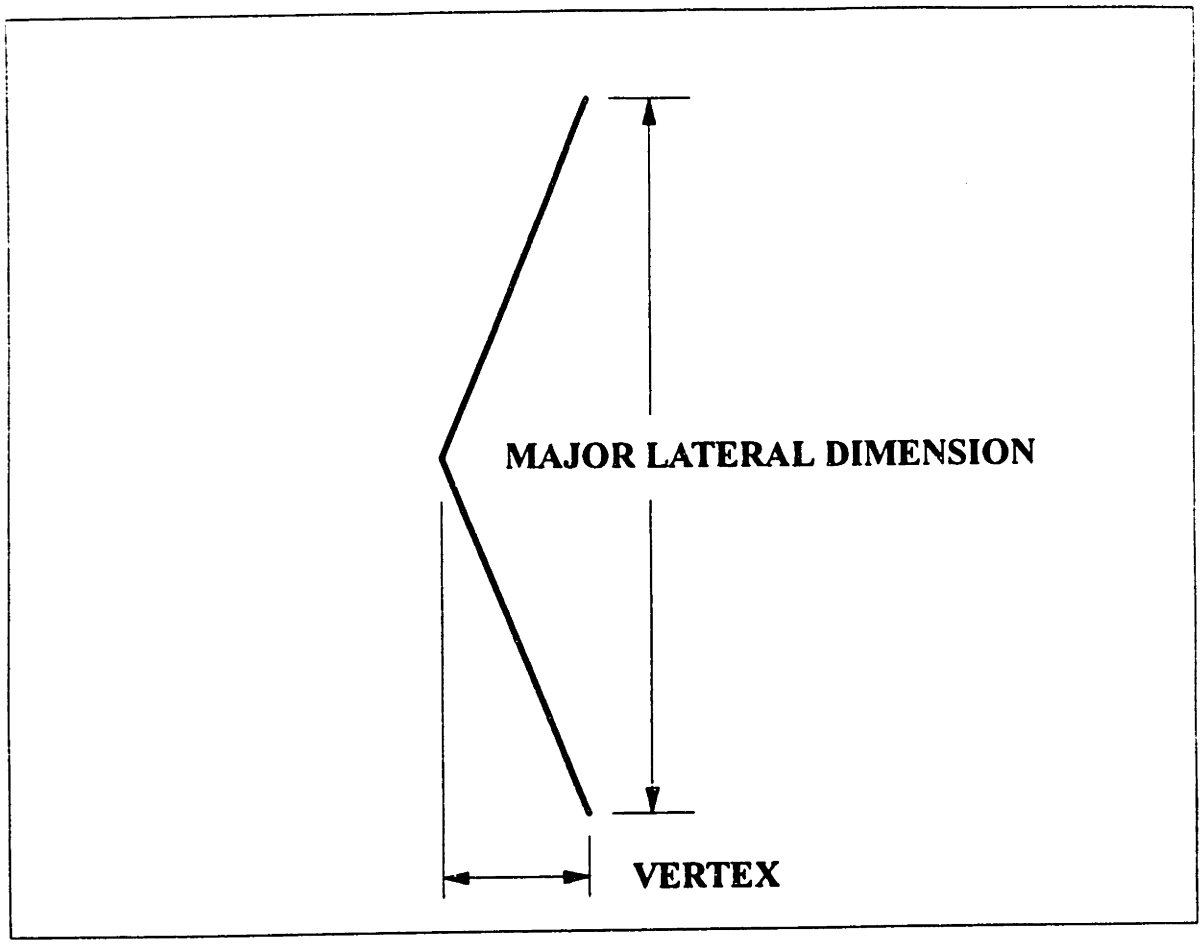


Figure 5-6: Primary dimensions of the bumper beam

Box with dual vertical support members

Box with diagonal support cross-members

These are illustrated in Figure 5-8.

The selection of a cross-sectional configuration and the dimensions associated with the cross-section and the beam itself are based on the strength requirements and capacity of the beam, and the implied weight and cost of the beam.

This simplified perspective of the problem does not address the issues of fastening mechanisms, manufacturing issues, and assembly requirements. However, the basic problem has sufficient elements of the general engineering

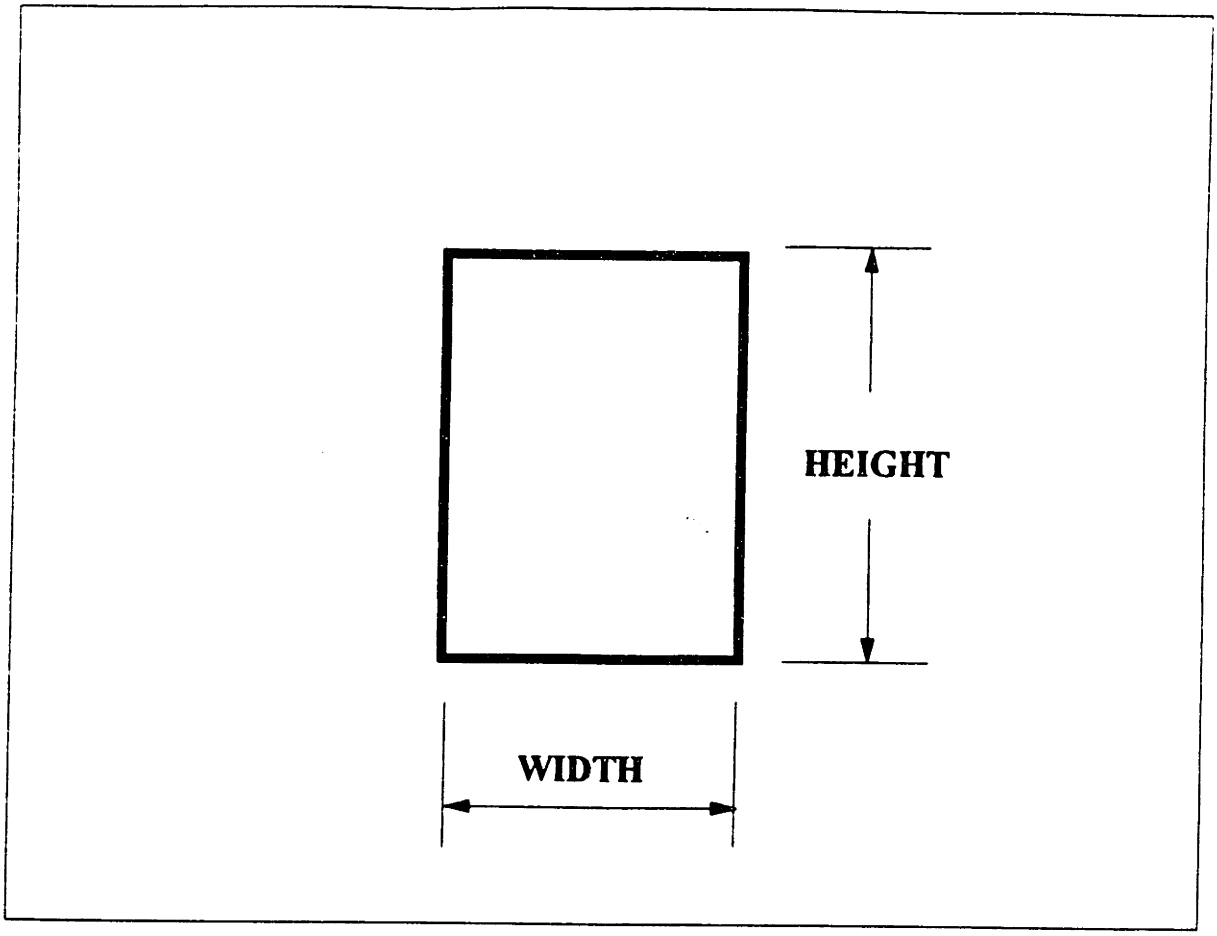


Figure 5-7: Cross-Sectional Beam Parameters

design context to be useful here. That is, this problem illustrates that, even with what appears to be a relatively simple component or systems design, there are issues of design specifications (e.g. weight and cost allowance, maximum crash speed,) inter-system coordination and cooperation (the bumper geometry must match the front vehicle body design even if this requires that the optimal bumper design be compromised) and multi-disciplinary analysis (the bumper beam must meet structural requirements as well as aesthetic constraints, fastener engineering requirements, etc.) In addition, this problem involves a physically and logically disparate team, in that the bumper engineering organization is not typically a part of the body design group, and the bumper requirements specifications are provided by yet a third group, namely vehicle engineering specifications. This organizational configuration is depicted in Figure 5-9.

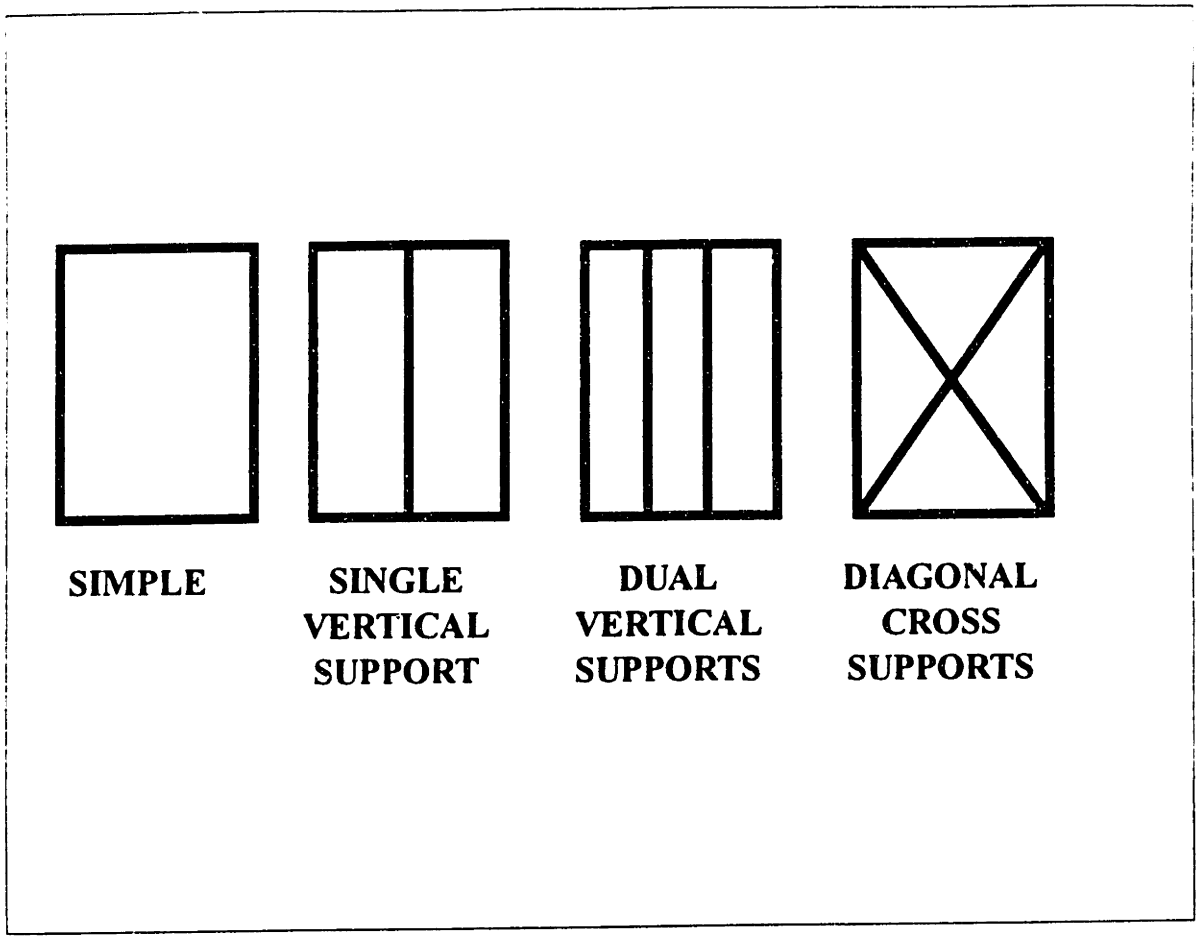


Figure 5-8: Sample Beam Cross-Section Configuration Options

As such, the problem is a non-linear design problem. The bumper engineering design process begins when enough information to initiate a preliminary layout is available. Further refinements of the design can only be made when a sufficient degree of information and knowledge is available to the bumper design engineer regarding the requirements and their implications. If changes are made to the initial body layout, these may imply that the initial bumper layout and configuration is no longer valid. This logic is complex, in that the validity of the initial layout and configuration may be compromised by the changes in the geometry of the body design, or by the location of the lights or other hardware, or by any changes in the design specifications. The reasons for the lack of compliance may be engineering (such as strength requirements) or geometric (the bumper beam no longer fits the changed body) or they may involve cost or weight constraint violations.

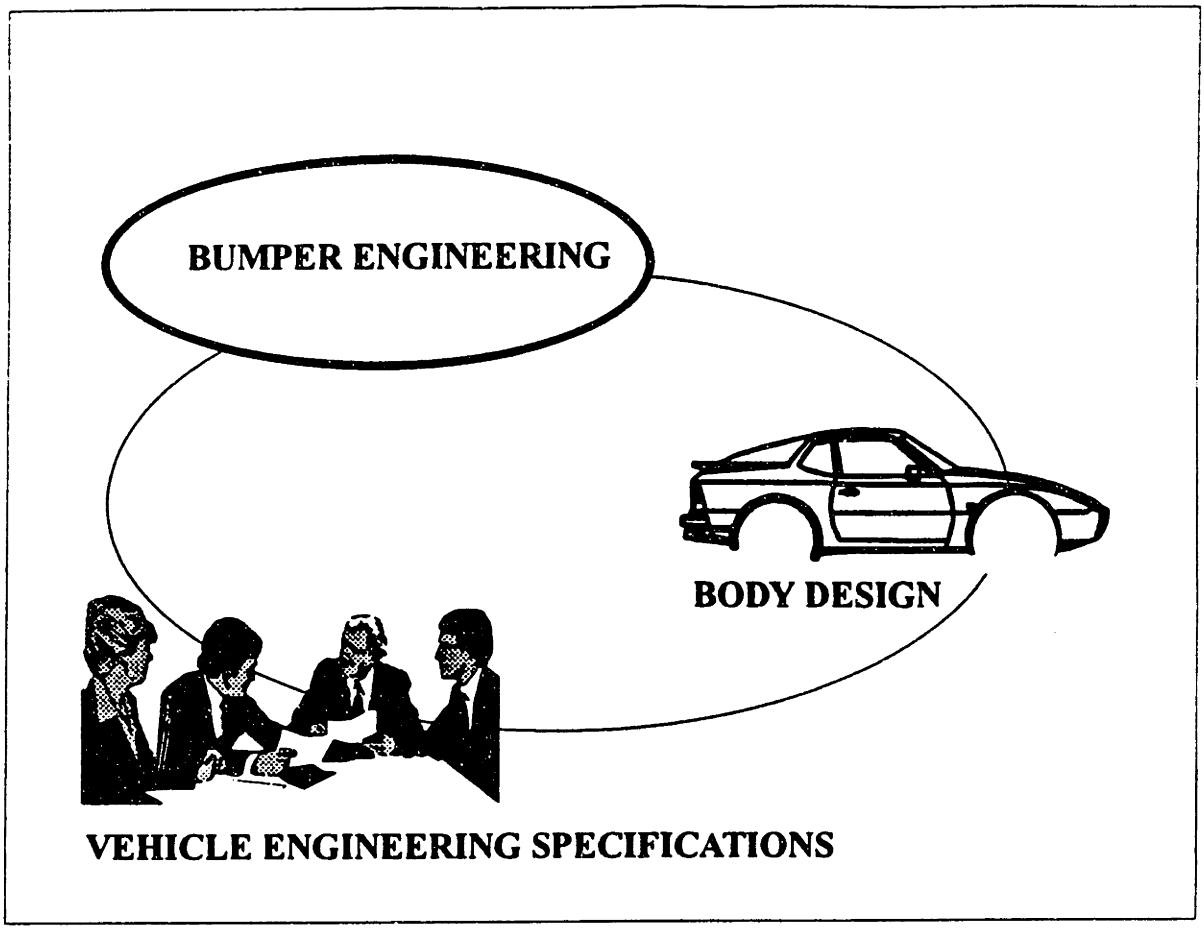


Figure 5-9: Organizations involved in bumper design

5.4.2 Bumper Design Knowledge-Based Design Architecture

In order to illustrate the Event-Driven Knowledge-Based Design concept, an example architecture has been constructed for this problem. This architecture consists of multiple design engineering workstations, and a separate engineering specifications module. The design engineering workstations consist of RISC and UNIX-based computers, configured with a three-dimensional solids-based CAD/CAM system, and a knowledge-based design system, connected via a network. The workstations are Silicon Graphics Irix and Hewlett-Packard HP-700 series systems. The CAD/CAM system is Pro/ENGINEER®, from

Parametric Technology Corporation, and the knowledge-based design system is STONE*rule*®, from Stone & Webster Advanced Systems Development Services, Inc.

Each engineering workstation has access to the CAD/CAM system, as well as to integrated knowledge-based systems. These knowledge-based systems have been built to operate in a variety of operational methodologies:

- stand-alone, passive advisory systems:
  invoked by the user, no modification to the design
- stand-alone, passive automation systems:
  invoked by the user, automated modification/generation of designs
- distributed, event-driven advisory systems:
  initiated automatically upon the occurrence of an event, no modification to the design
- distributed, event-driven automation systems:
  initiated automatically upon the occurrence of an event, automated modification/generation of designs

For the purposes of this example, the interactions between the parties is limited to one representative designer/engineer from each functional area, namely body engineering, bumper engineering, and vehicle engineering specifications. This interaction is depicted in Figure 5-10.

The basic architecture has been maintained through the variety of configurations of the event-driven design mechanisms. That is, three different event-driven design architectures have been prototyped, using the same basic design configuration. These event-driven design architectures are described in detail in the following major section.

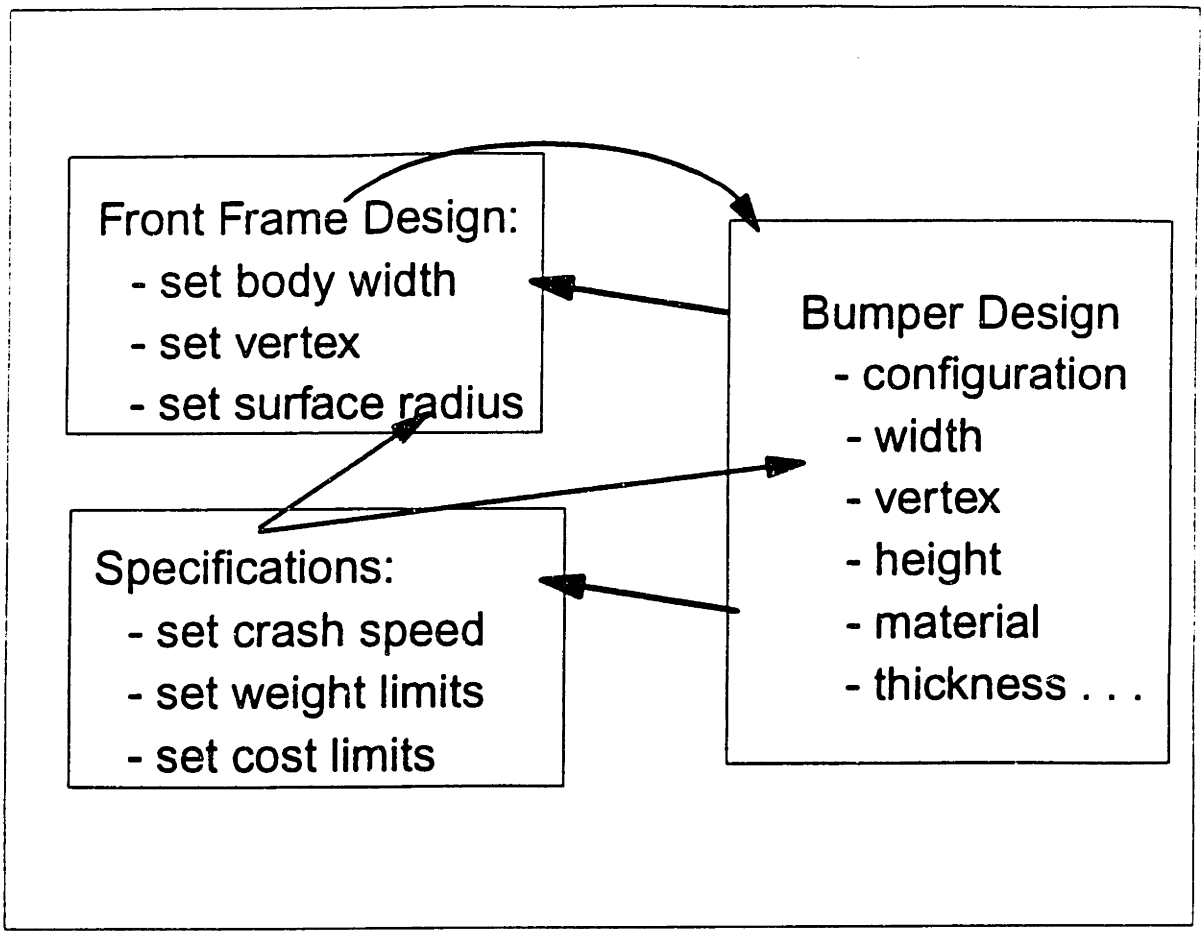


Figure 5-10: Basic design configuration for the bumper beam design example.

5.4.2.1 Bumper Design Knowledge Base

A bumper design knowledge-base has been developed, in order to assist the design engineer with the task of designing the basic layout and configuration of the bumper. The knowledge base operates by initially loading a bumper design model into the current workspace, and then it obtains the requisite inputs, and finally it performs the update to the design model.

The inputs include:

- the vehicle model number
- the basic dimensions of the front frame

The bumper design knowledge base has a class structure for the bumper beam, an illustrative portion of which is shown in Figure 5-11:

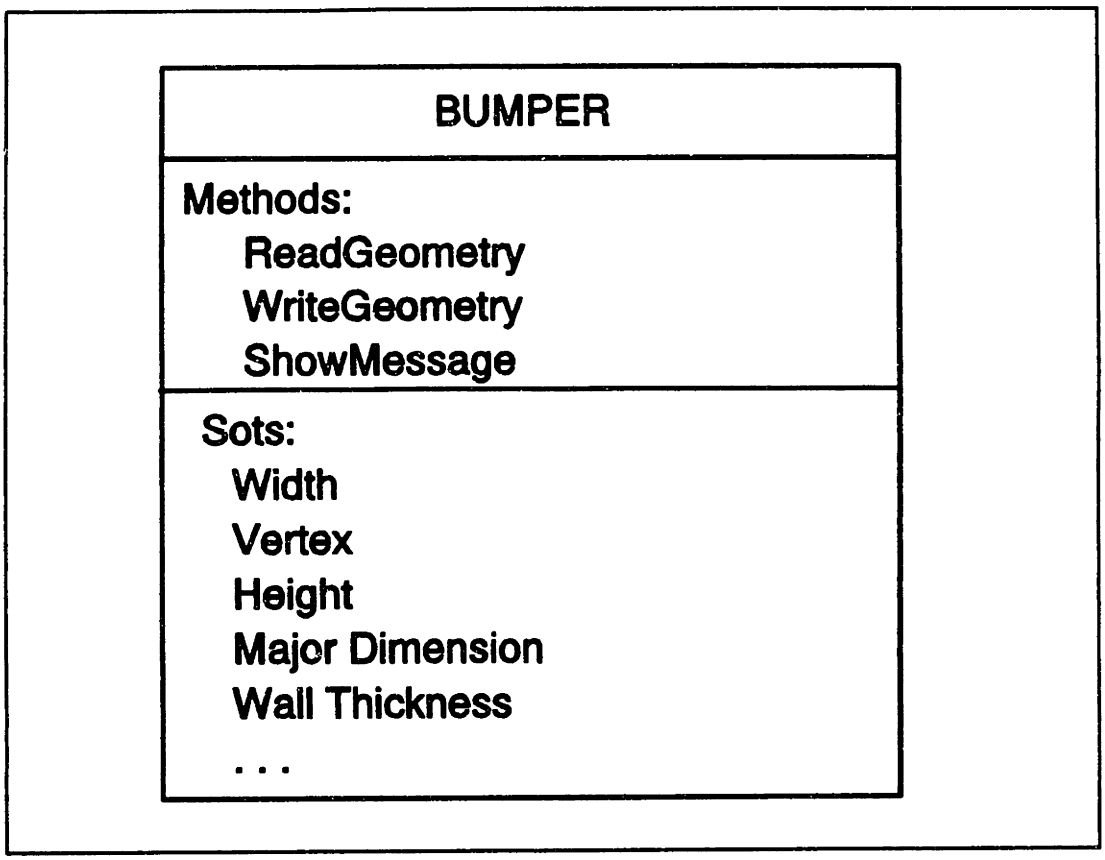


Figure 5-11: Bumper Beam Class Definition

A design is performed based on the class structure and the above primary parameters, in order to allow the design engineer the maximum flexibility in laying out multiple configurations. The interaction sequence follows the path of obtaining initial input from the specifications design engineer and the body frame dimensions from the frame designer, and then asking the user to specify the detailed design and functional parameters. The knowledge base determines the appropriate dimensions of the bumper based on the user's input and the

basic dimensions from the body frame design. An example user interaction is illustrated in the following set of figures:

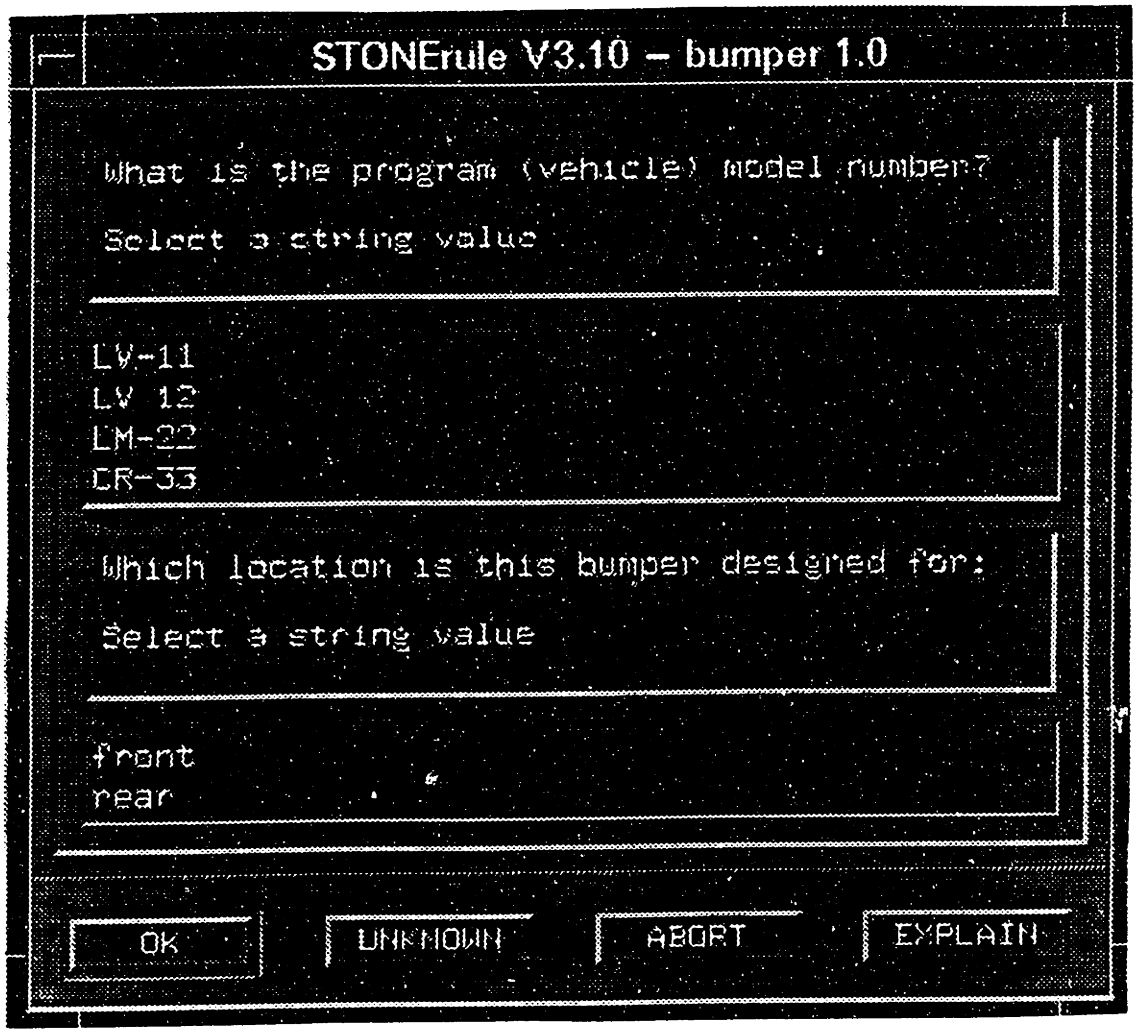


Figure 5-12: Initial input panel for bumper design knowledge base

Note that the information required for this knowledge base can be obtained from the design specifications database, from the geometric design database, or from the user directly.

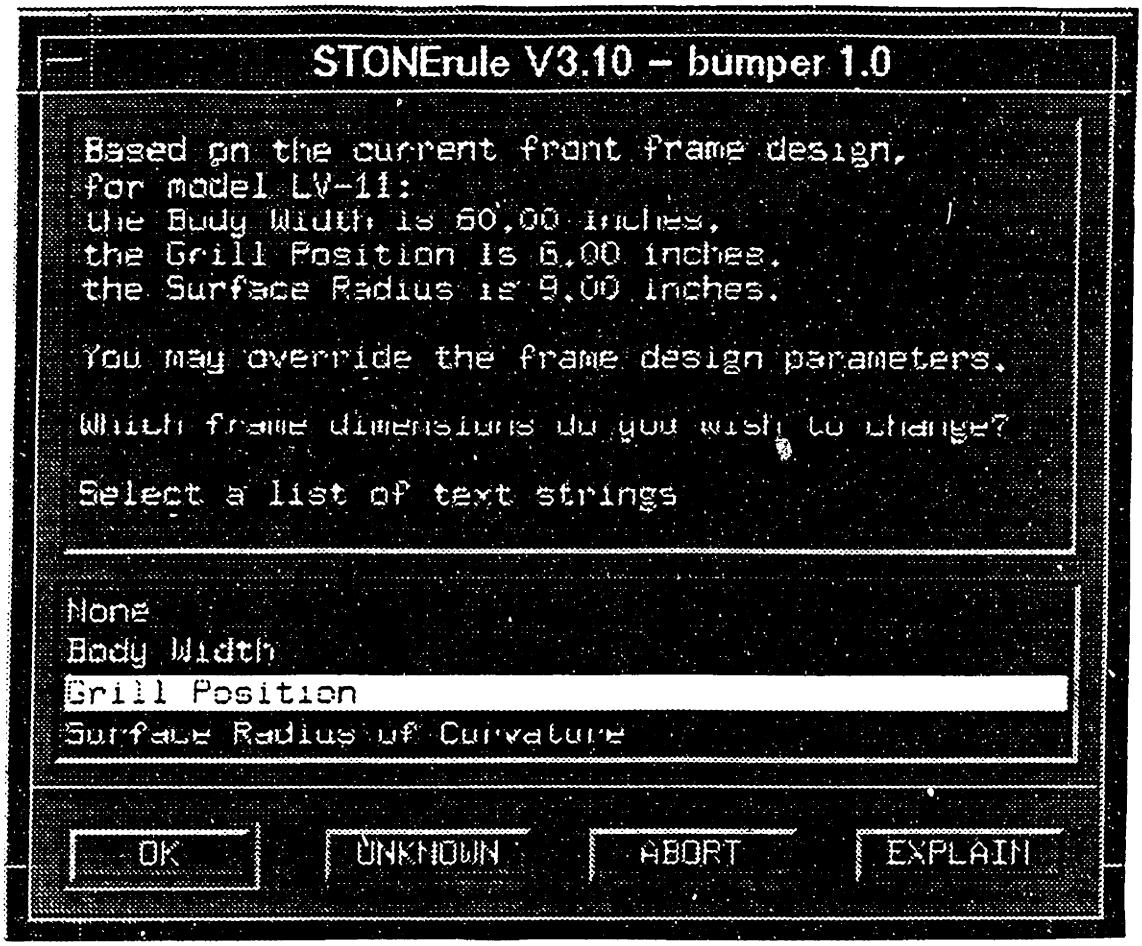


Figure 5-13: Input panel for bumper design knowledge base

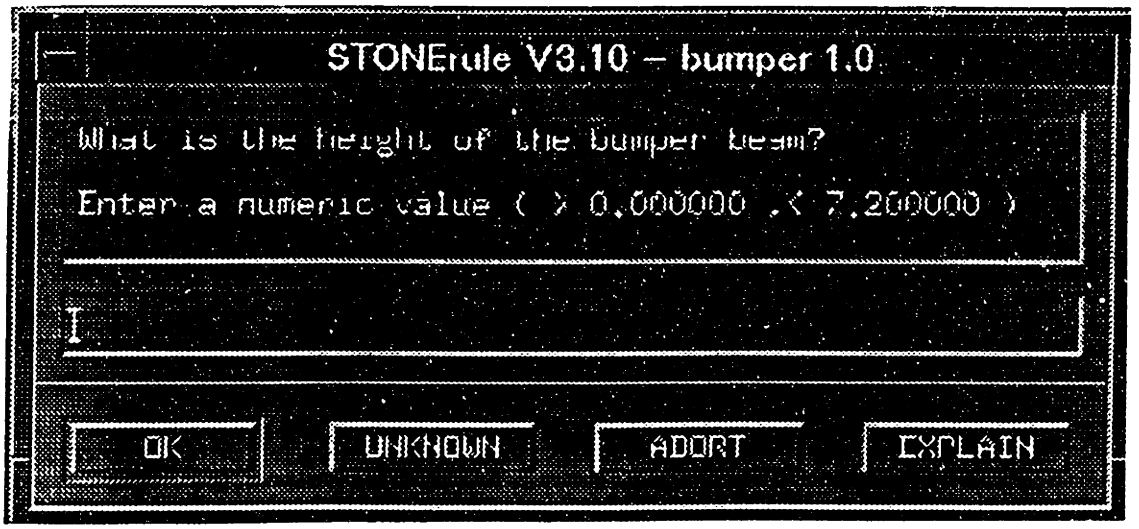


Figure 5-14: Input panel for bumper design knowledge base

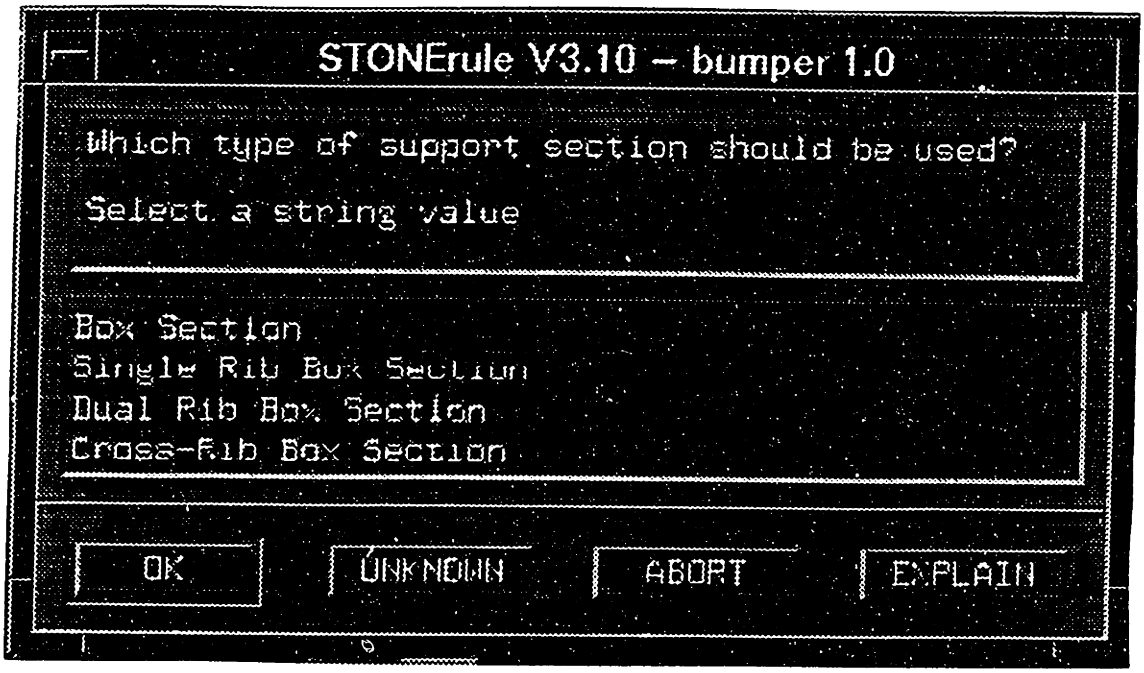


Figure 5-15: Input panel for bumper beam design knowledge base

5.4.2.2 Event-Driven Design Validation Knowledge Base

Once the bumper design is completed, a validation knowledge base is automatically initiated. This validation knowledge base consists of a strength evaluation, a configuration analysis, and a requirements validation.

This knowledge base is not initiated by the user. It is intended to serve as an event-driven system - one that is invoked at the occurrence of appropriate events. One such event is the release (or request for release) of design models, at which time the event-driven knowledge-base is automatically invoked. The event-driven knowledge-base is also accessible through external events, and is invoked automatically upon occurrence of the appropriate events.

This knowledge base is tasked with evaluating the proposed bumper configuration with respect to the structural integrity of the system, the ability of the bumper system to meet the specifications, and any improvements that could be made to reduce the cost or weight of the bumper beam.

The inputs to this knowledge base consist of the basic design configuration, the front frame configuration and dimensions, and the design specifications. The system reasons about the appropriateness of the proposed configuration, given the performance of the proposed design under the specified operating conditions. These analyses are performed with respect to the bumper *performance*, in the areas of stress, weight, and cost. Note that the initial bumper design is performed with respect to geometric issues of compatibility with the front body frame. The analysis, however, is performed with respect to (i) the ability of the bumper beam to withstand the stress resulting from the impact force, (ii) the weight of the bumper compared to the allotted weight, and (iii) the cost of the bumper with respect to the allowable bumper beam cost. The stress analysis is performed according to the following calculations:

F_I = Impact force applied to the bumper beam

F_y = Maximum allowable stress of the material

F_{max} = maximum applied stress to the bumper beam

A_m = Area of the bumper beam for maximum stress calculation

a = bumper beam vertex

b = bumper beam major dimension

h = bumper beam cross-sectional height

$F_{max} = (F_I/A_m)$

$F_{max} = F_I /(h \cdot ((b/2)^2 + a^2)^{1/2})$

The weight of the bumper beam is calculated as the product of the volume of

the beam and the material weight (in pounds per cubic foot.) The cost of the beam is calculated as the product of the weight of the beam and the material cost ($/lb).

The knowledge base has access to a materials database, which contains the AISC designation of the material (all steels), the value for maximum allowable stress (F_y), the weight of the material (lb/ft^3), the specific gravity of the material, and the cost of the material ($/lb).

Once the beam geometry is analyzed by the knowledge base, the appropriate materials (with respect to stress) are selected, and the user is given the option of selecting a material, or the system will select the lowest cost material for the list of those materials with $F_y > F_{max}$.

This knowledge base also has the capability to modify the design automatically, based on the results of its comparative analysis. The ability to modify the bumper design configuration include a shape modification, cross-sectional properties modification, material selection, and dimensional modifications.

An example interaction with this knowledge base is depicted in the following set of Figures, and a full set of interaction sequences is shown in Appendix B.

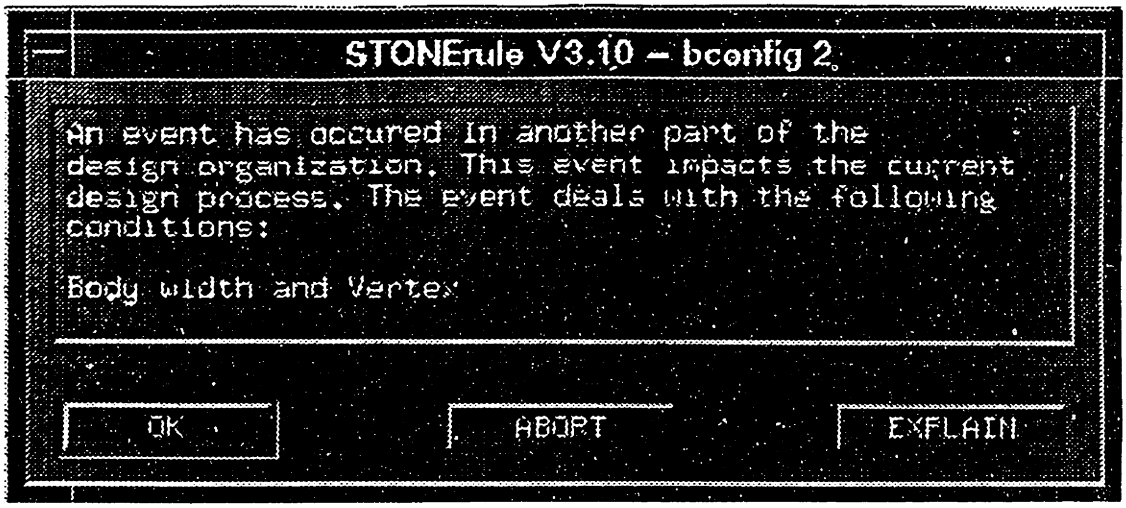


Figure 5-16: Sample interaction with event-driven design analysis and validation knowledge base.

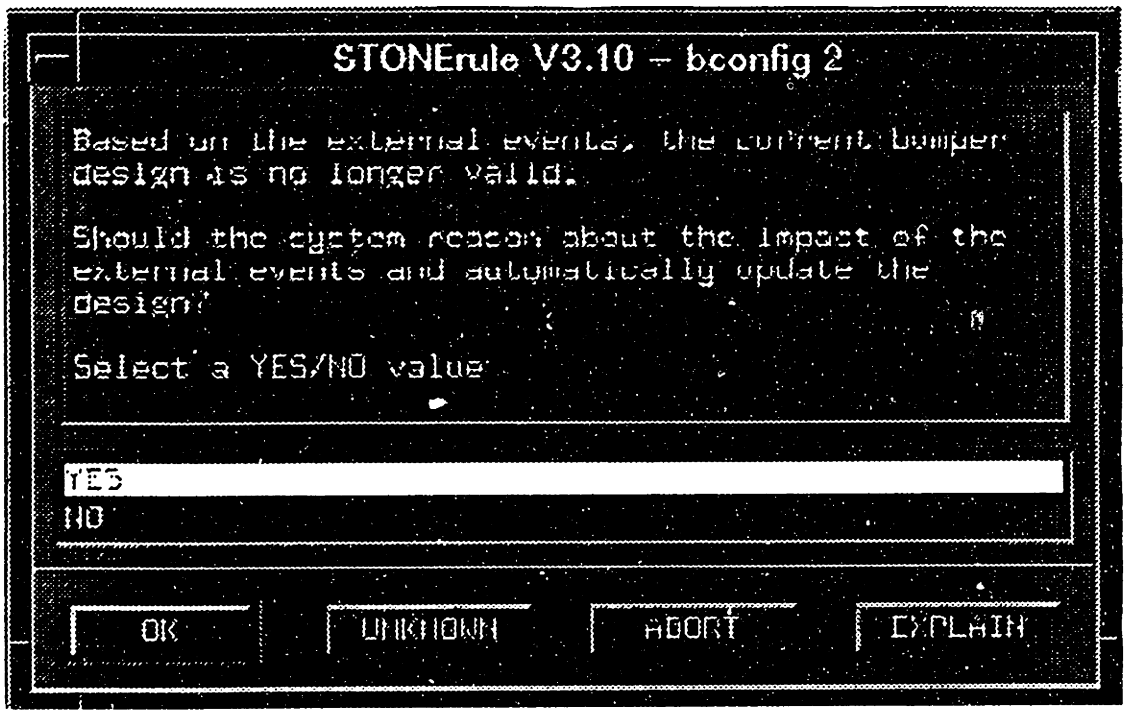


Figure 5-17: Sample interaction with event-driven design analysis and validation knowledge base.

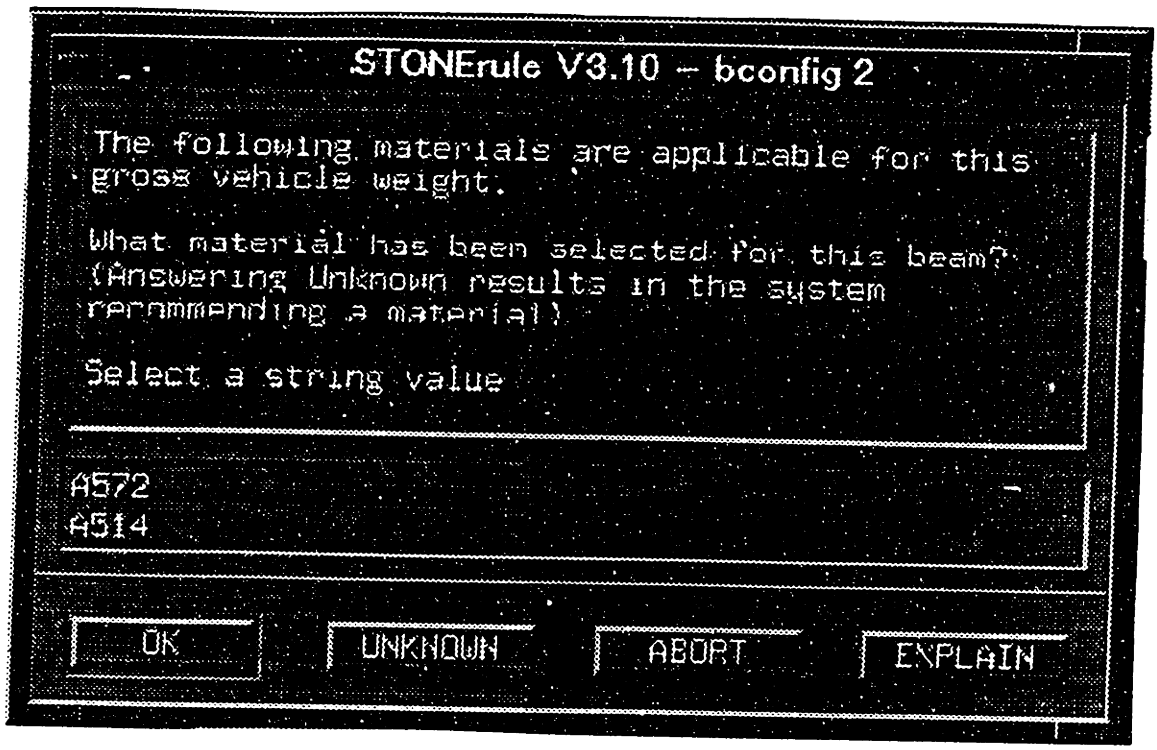


Figure 5-18: Sample interaction with event-driven design analysis and validation knowledge base.

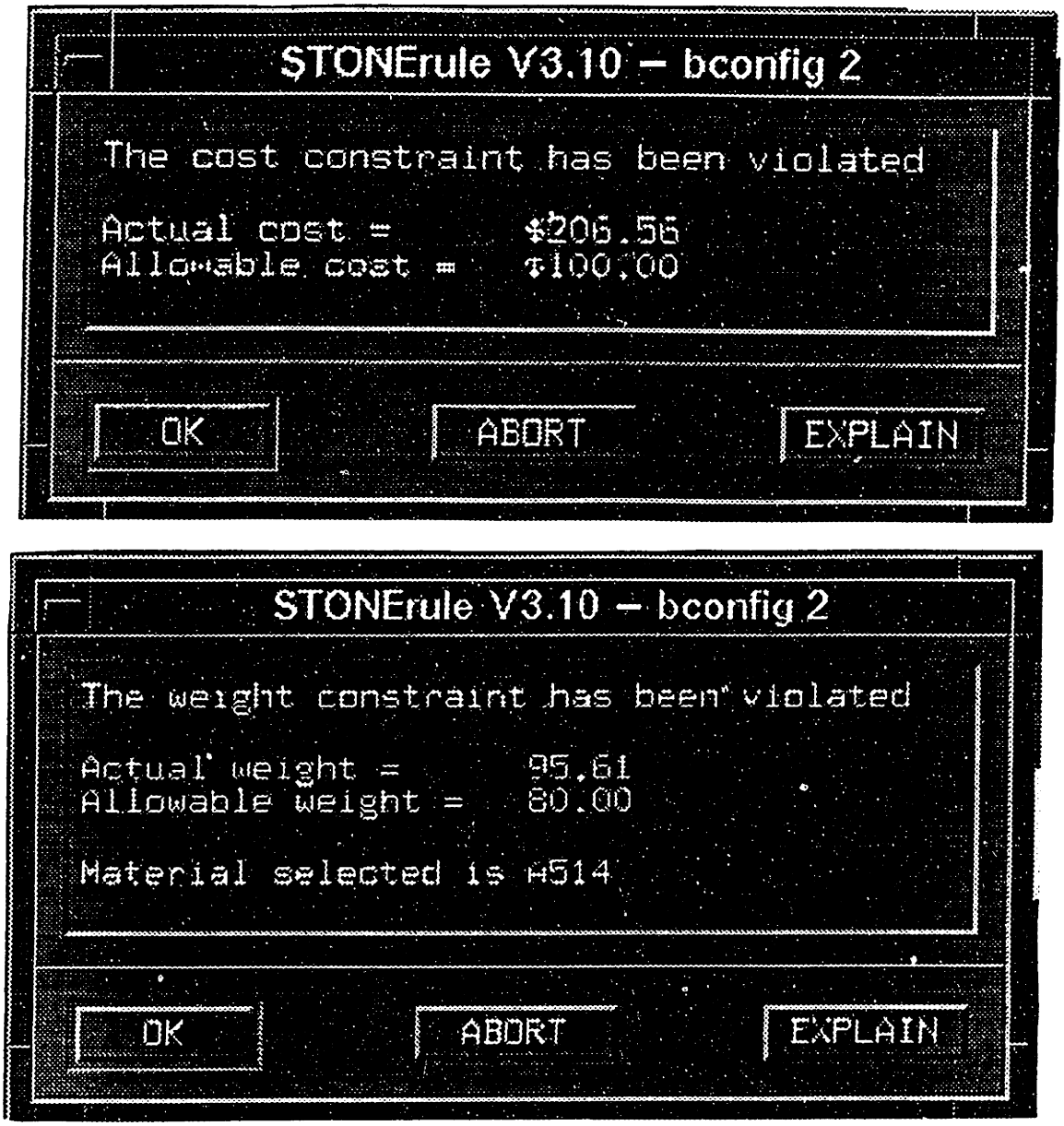


Figure 5-19: Sample Event-Driven Knowledge-Base Feedback

5.4.3 Event-Driven Design Architectures

The three basic architectures posited above have been investigated here. These are:

(i) The predispositive method

(ii) The intelligent search method

(iii) The discrete event architecture (DEA) method

5.4.3.1 Predispositive Architecture

In this architecture, the knowledge bases are aware of the existence of recipients or originators in the event cycle, and specifically and deterministically create information that is directed toward each other, based on the knowledge that a recipient requires the information that constitutes an event.

As such, in this case, the front frame designer has the role of an originator of conditions for the bumper engineer. Upon release (or modification) of front frame information, pertinent information is consciously sent to the recipient, in order that the recipient be made aware of the event and that the information presented to the recipient is current and valid.

As such, a knowledge base describing the pertinent front frame dimensions and other data has been developed. This knowledge base operates automatically upon the regeneration of the front frame design model. Once the design engineer saves the model, the pertinent information is automatically transmitted via the knowledge base, as a design event, directly to the bumper designer. This operation is depicted in Figure 5-20.

Note that in the predispositive architecture, specific events must be built into the design knowledge-base of the originators, such that the design events are

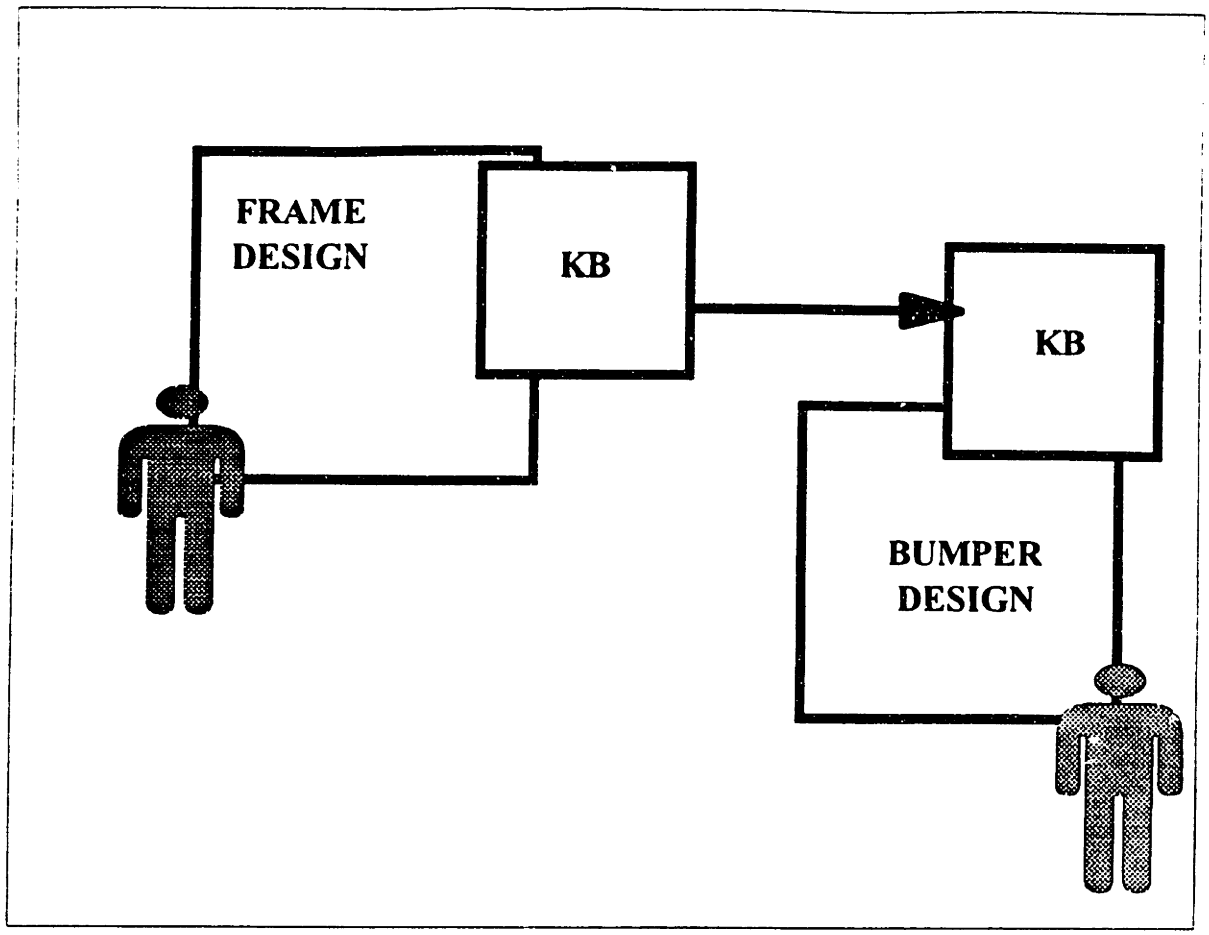


Figure 5-20: Predispositive event activity

captured at their origin, and transmitted directly and specifically to recipients. While this operates effectively here, note that as the number of recipients increases, this becomes a particularly complex and less feasible problem. More specifically, the capture of events becomes less feasible as the events increase in complexity. That is, when an event is originated entirely within a single source (originator) then it is possible to capture the event at its origin. As the number of contributing originators increases, this task becomes less feasible. In fact, it may not be possible to capture events at their origins since the origins may be disparate (both physically and logically) and they may meet only temporally at the initiation of the event. However, in the context of this example, the results proved successful, due to the simplicity of the environment.

The progression of this architecture follows the creation or modification of the frame design by the frame designer, the automated initiation of the event knowledge system in the frame design environment, and the transmission of the requisite event information to the bumper designer.

5.4.3.2 Intelligent Search Architecture

In this architecture, the recipients actively search for events by either polling the design database or by searching at critical point in the design process for the occurrence of events. This has been modelled in this environment by the modification and addition to the bumper design knowledge bases, in order to enable them to actively search for events in the design process that may have an impact on their specific design tasks.

In order to accomplish the first type of intelligent search, that is the polling of the design database at regular intervals for specified information and events, the bumper design knowledge base was modified to include an additional specific action. This action was the initiation of a memory-resident, continuously operational knowledge base whose function is to examine the design database (as a continuous flow of information) for the alteration of major dimensional changes in the front frame AND changes in the design specifications database. This polling knowledge base notifies the designer upon the detection of any relevant events, and the bumper designer is then able to execute the design validation knowledge base with respect to the changes that have been made external to the bumper design, and their impact on the bumper design process. This architecture is depicted in Figure 5-21. Note that the systems depicted by dashed lines are asynchronous memory-resident knowledge bases.

While this architecture has the advantage that while the bumper design process

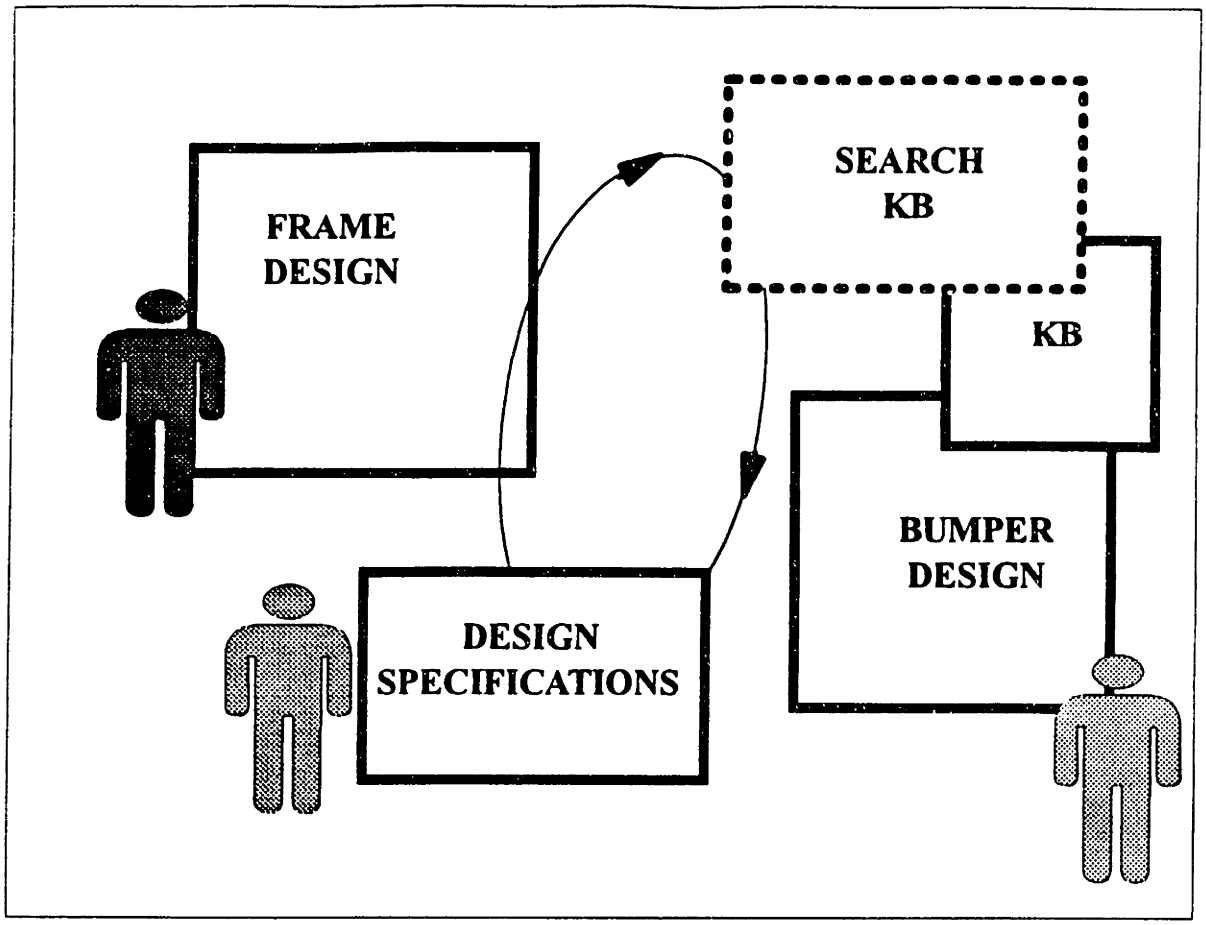


Figure 5-21: Intelligent Search Architecture - Implementation I (Polling Of Design Database)

progresses, any and all events identified in the search knowledge base will be detected and all relevant information provided immediately, the difficulty again is associated with scale. As the problems become more complex, the ability to operate in a continuous polling mode is diminished. This architecture has the advantage over the predispositive approach that recipients are responsible for the definition of their own events. Therefore, multi-origin events will be captured because the event recipient will have identified the origins and will be able to search for the events regardless of their origins. That is, in this architecture, we see events defined from the perspective of the recipient, which is the natural mode of event definition.

In the second intelligent search methodology, the knowledge bases of the recipients act as *agents,* responding to interactions of the designer by searching for events and information without having to have explicit direction regarding the origin of the events or the location of the information. As such, initiation of these searches depends on the need identified by the recipient at design time, and the life of the search agents terminates when the results of the search are transmitted back to the recipient designer.

This approach has been developed here, by initiating asynchronous knowledge bases that do not stay memory resident, but terminate upon completion of the search for the information that they have been directed to find. This architecture is represented by a specific knowledge base that searches the specifications database and the frame design database, and if no significant changes have been made since the bumper beam design assumptions or inputs were derived, then the system simply terminates. If pertinent events are detected, this information is transmitted immediately to the bumper designer. This architecture is illustrated in Figure 5-22.

This architecture is extremely effective because it is very specific and directed, this saving on computing resources and overhead, but it has the advantages of being recipient-oriented, as in the first intelligent search architecture described above. The issue of scale is less serious in this architecture, because the search mechanisms are only active when they are needed and specifically initiated by the design knowledge bases. However, these approaches suffer from the constraint of having to pre-define the events within the recipient systems.

5.4.3.3 Discrete Event Architecture

In the Discrete Event Architecture (DEA) neither the recipients nor the

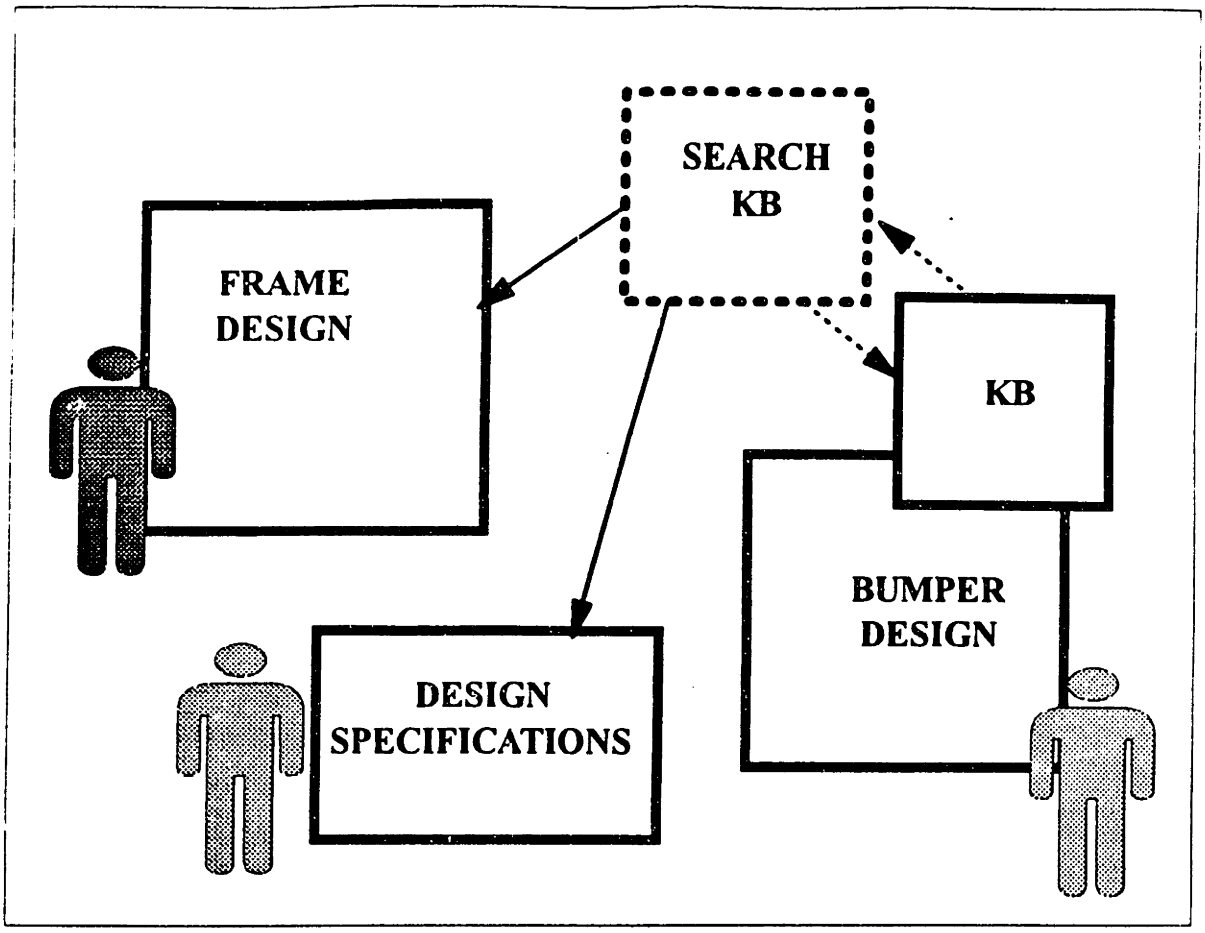


Figure 5-22: Intelligent Search Architecture II - Search Agents

originators represent events explicitly within their scope and domains. That is, the concept of an event exists only external to the design systems, and is contained within the ***event manager***. Thus, the generation of information, and the consequential occurrence of events, occurs as a natural part of the continuum of the design process. Events that are detected by the event manager are made visible to appropriate recipients.

In this architecture, design processes occur independent of one another. As described in Chapter 4, the event manager has class hierarchies defining events and their properties. In this case, events are defined as the modification of design parameters in the body frame design, and modification of vehicle specifications. The confluence of such actions forms events that are monitored by the event manager. This is implemented by means of a separate process,

which continuously examines the design database, and upon the occurrence of events, takes the appropriate action, as defined in the consequent of the event definition. This implementation is illustrated in Figure 5-23.

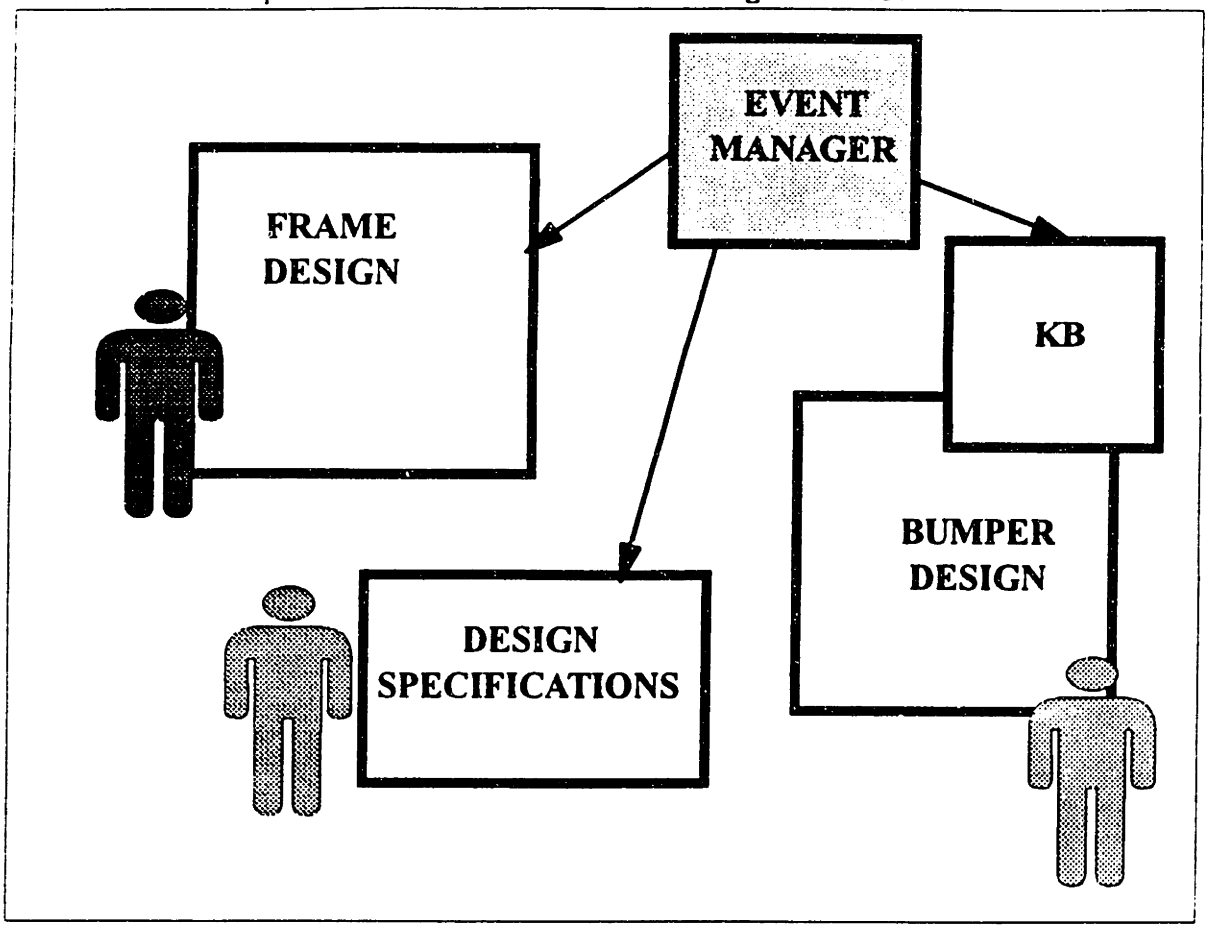


Figure 5-23: Discrete Event Architecture (DEA) Implementation Example

The DEA approach has the advantage that each design activity can proceed in accordance with its own process, regardless of the activities of events that may or may not exist. Specifically, these systems may be impacted by events that they are not aware of *a priori*, and as such, this approach has the potential for further development in the area of adaptive event management. That is, in such an environment, the event manager could determine which systems or designers are impacted by certain events without requiring that the recipients register with the event manager. This further leads to the notion of adaptive

event recognition, in that the next evolution of such an event manager would be a system that learns about events, and defines events dynamically.

5.5 Results

The three architectures have been implemented for the given exampie, and a sequence of simulated design sessions executed. All three architectures were built using event classes that were shared amongst the various systems, and were specialized in the cases that specific functions were required to read data or post data to the event system. The events were shared in a blackboard environment, such that each event was uniquely identified by means of a time and date stamp, the relevant data, and a currency flag. The structure of the event data was such that the data were parsed by the recipients and interpreted based on the recipient's interest. Since this was a limited implementation for a pilot study, the data structures on the blackboard were maintained as constant structures, but these could vary in a production implementation. Figure 5-24 illustrates a single row (event) from the blackboard.

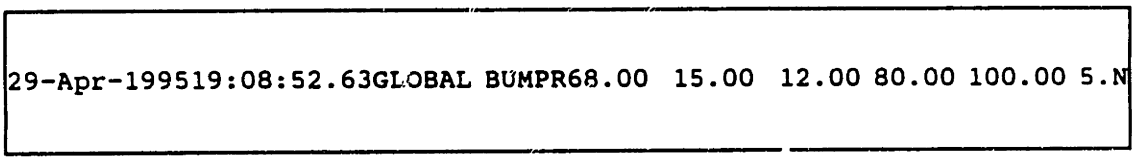


Figure 5-24: Example data structure of a single event from the blackboard

In all architectures, the concept of feedback was illustrated through the generation of events that were, in effect, responses to prior events. For example, in one case, the frame designer made changes to the geometry at the same time that the vehicle specifications were changed. This generated an event which resulted in modifications to the bumper design, but also resulted in the inability to arrive at a bumper solution that met the cost constraints imposed by the specifications. As such, an event was generated as a consequence of the bumper design having a design parameter that was in excess of the allowable value, and this event was recognized by the frame design system. In this manner, feedback was provided in a timely fashion.

Events were recognized in the bumper design environment via a knowledge base that interpreted the events with respect to contextual changes in the bumper beam design process. The events were characterized with respect to the effects that they had on the bumper beam design, and the system made appropriate changes to the beam design. This is illustrated in the following sequence in Figures 5-25 and 5-26.

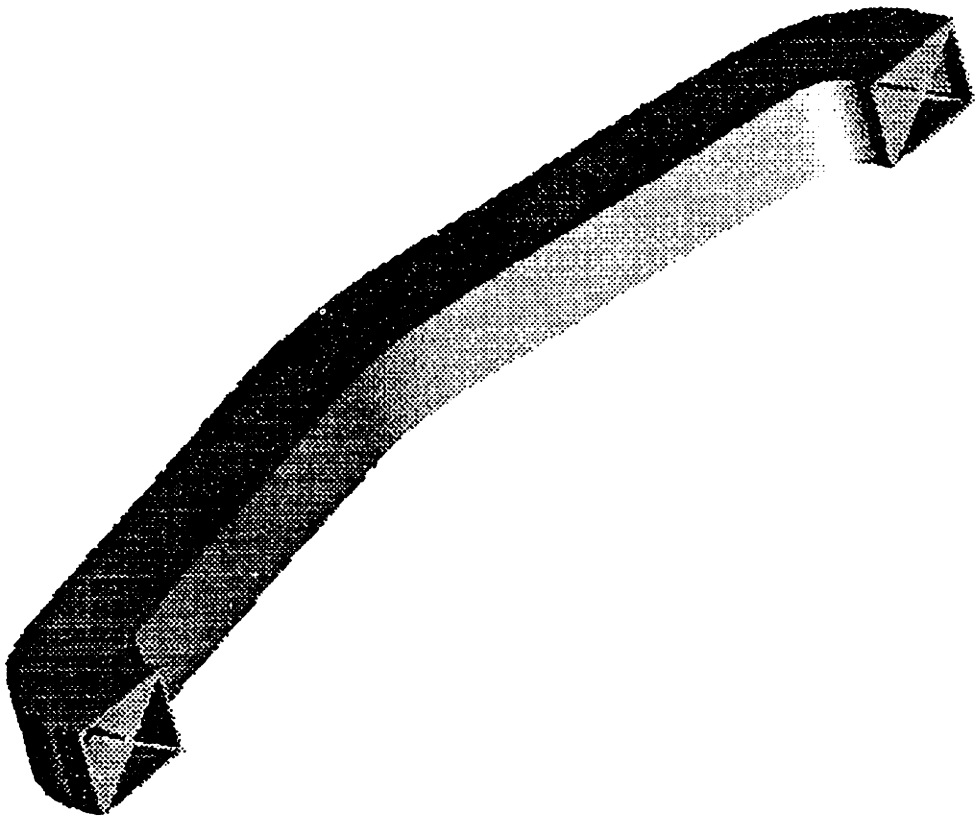


Figure 5-25: Initial Beam Configuration

5.5.1 Predispositive

The predispositive architecture was implemented by means of direct recognition of the recipients by the originators, and by specifying the recipients in the events. Events were created (initiated) from within the knowledge bases themselves, and posted to the blackboard. This followed the sequence of (a) recognition of events from within the knowledge base, (b) creation of dynamic event instances, and (c) writing the values for those instances to the blackboard.

In the examples, the frame design knowledge base was programmed to recognize specific events of interest to the bumper designer. These events were recognized at execution time, and the recipient was specifically named as the

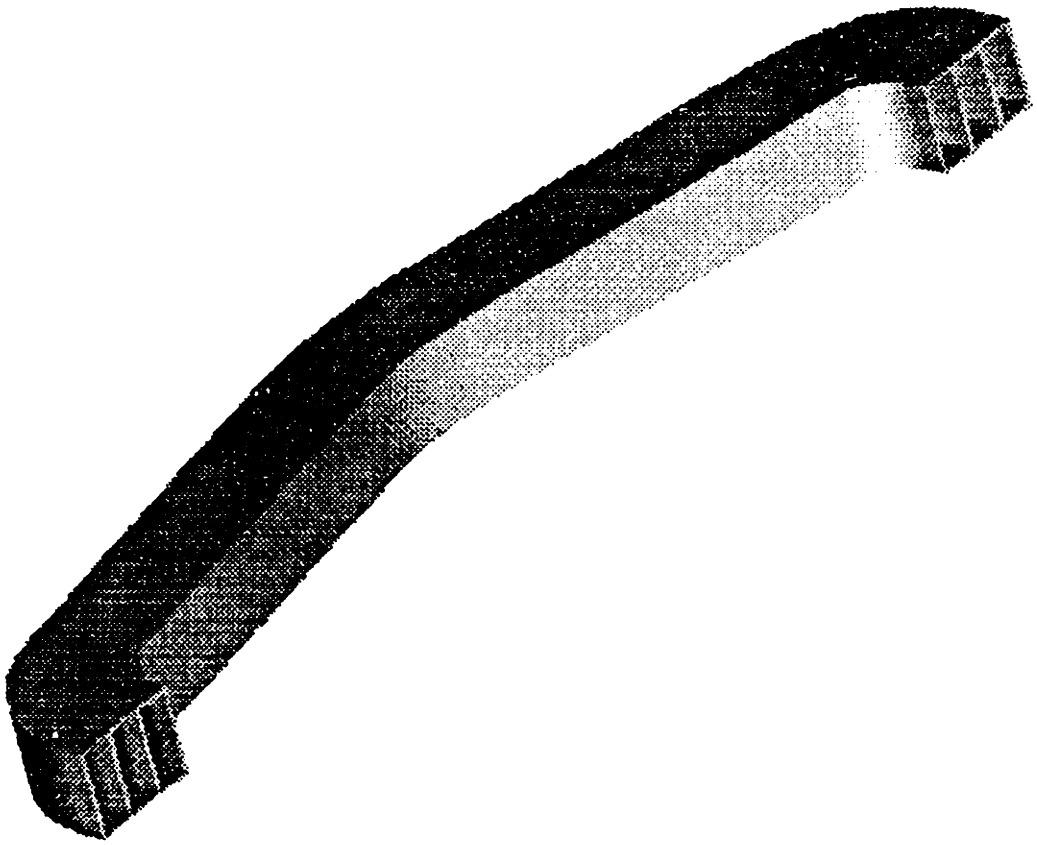


Figure 5-26: Final Beam Configuration

bumper designer. The specifications knowledge base was also programmed to recognize specific issues of interest to the bumper designer, and this knowledge base also has specific event generation mechanisms built into the system.

The bumper knowledge base had a specific **GetEvent** method within its event class hierarchy, and it was the initiation of this method that caused the events to be recognized. The bumper knowledge base was not specifically aware of the contributors to the events, in that the event data were used without knowledge of the architecture being exercised at the time. As such, the events existed only from their explicit contributor inception until the recipient knowledge base read the event.

The behavior of the system was highly predictable, and was explicitly visible through examination of the blackboard and observation of the progression of

the various systems involved. The GetEvent method is illustrated in Figure 5-27.

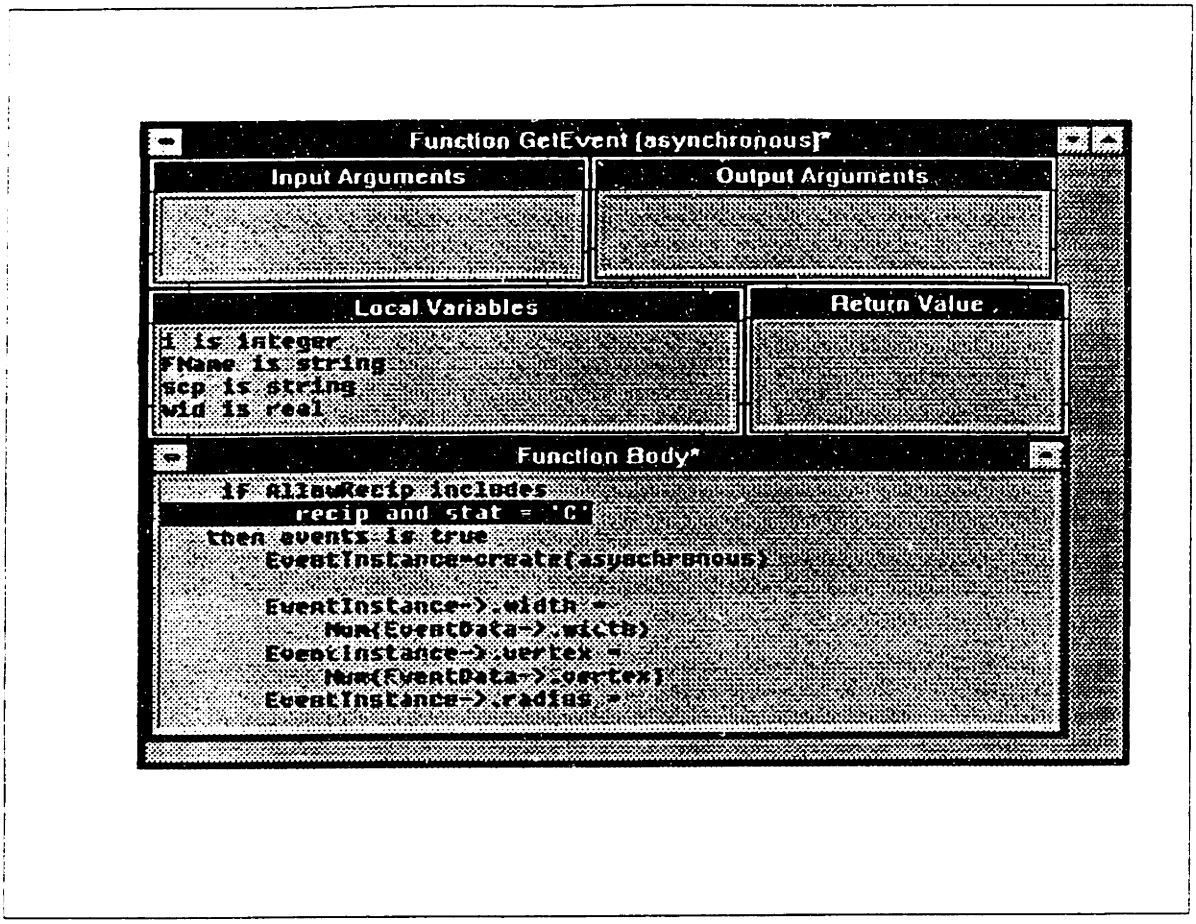


Figure 5-27: Partial Illustration of GetEvent Method

5.5.2 Intelligent Search

The Intelligent search architecture was implemented by means of an intelligent agent (in the form of a knowledge base) that has specific directions regarding its mission. It is programmed to search the various databases and to discover any changes in the data that reflect significant events from the perspective of the recipient. The intelligent agent determines if any events have occurred, and

if so, it updates the blackboard and terminates its execution.

As an alternate architecture, the intelligent agent was initiated as a memory-resident system, and was only terminated at the explicit termination instructions of a user. This architecture can be viewed as a local event management mechanism, with the intelligent agents having limited scope and vision, relative to the plethora of data and events in a global design organization. The operation of the agents is precisely the same as the event managers', except that the recipients are limited to single (or a limited few) systems.

In this architecture, the contributors have no knowledge of an event system whatsoever. The contributors act in the same manner as they would have without any notion of events. That is, they update the design database periodically, and it is the design database that forms the source for the data that may contribute to events. The intelligent agents make no contact with the contributing designers, but act solely on the design database.

In this architecture, the behavior of the system as a whole was similar to the predispositive architecture, with one notable exception. Events had an extremely limited currency, since they were generated immediately prior to being made available to the recipients. That is, the events did not wait on the blackboard for recipients to notice them, since they were generated by agents who had specific instructions to notify the recipients upon generation of the events.

5.5.3 Event Manager

The event manager architecture was implemented via an object-oriented system

that behaved in a memory-resident, continuous search mode. This system was provided with instructions as to the events of interest (and their recipients) and access to the design databases. As in the case of the Intelligent Search architecture, the design systems had no notion of events, except inasmuch as they were able to respond to events for which they were recipients. The generation of events was left entirely up to the event manager, with respect to timing (when the event occurs) as well as when to generate notification of the events.

During the course of the development of the systems for this architecture, it was noticed that the events that were generated by the event manager were duplicated by the event manager after some short period of time. After investigation of this behavior, it was determined that this was caused by the continuous nature of the event manager's existence. As soon as the event was recognized by the recipient, it was deemed non-current by the event manager. This behavior was characteristic of all three architectures. However, in the case of the event manager, the continuous exploration of all databases lead to the determination that the event was once again of interest, since the recipients had not responded. In reality, the recipients had not had time to respond, since immediately after the notification of the event to the recipients, the event manager was on its search mission, and determined that there was a disparity in the design databases of the contributors and recipients. This behavior lead to the development of a feature which identifies events that have been recognized, and prevents duplication of events.

The event manager proved to be a powerful tool for the dissemination of pertinent knowledge about changes throughout the design organization, and was a central control mechanism for events in the design process.

5.6 Comparison Between Architectures

As can be seen from the above analysis, there are many similarities in the overall system behavior regardless of the choice of architecture. Specifically, the designers behave in very similar fashions across architectures. In addition, the substance of the events is common, as is the structure of the blackboard and the event class hierarchy.

The differences appear upon examination of the knowledge systems themselves, and the degree to which they must act and react to the existence of an event system. In the case of the predispository architecture, the designers must have knowledge of those issue which constitute events, and they must also have mechanisms for generating events. In the case of the intelligent search architecture, the designers that wish to be notified of significant events must consciously act to initiate such notification, even in the case of the memory-resident agent, which must be initiated. In the case of the event manager architecture, the designers do not need to initiate the generation of, nor the receipt of events.

It is likely that these three architectures could co-exist within the same design organization. For example, a direct mechanism can create events for those designers that are intimately tied together, and for whom notification and feedback amongst each other is a specific and integral part of the local design process. In those cases that less direct ties bind designers, the event manager or intelligent search architectures could be used.

It is probable that the event manager architecture would suit those organizations that maintain centralized approaches to system integration, and for which there may exist many inter-dependencies between designers and data across the organization. Additionally, this architecture would be appropriate for

situations in which there were a one-to-many relationship between events and recipients. That is, when many designers are affected by the same event, then it is more efficient to use an event management system in order to reduce programming requirements and computing (execution) cycles.

The intelligent search architecture is highly attractive in the sense that it offer the opportunity to create highly focused, yet broad-reaching agents for periodic or spurious event-checking. This architecture offers the added opportunity to allow for growth in technology, with respect to the capabilities of the agents themselves. As agent technologies evolve, it is conceivable that these systems could *learn* about the needs of the designers, and could determine the significance of the dynamics of the design data without having to have a comprehensive pre-determined set of events. In this scenario, one might imagine an agent that learns about what issues affect a particular design problem, and would initiate events perhaps from the confluence of data that may appear to be obscure, if unrelated to the design problem at hand, but which prove to be influential.

The predispositive architecture is attractive for those data, events and designers that act in a manner that is highly correlated amongst one other. This architecture eliminates the need for searching, and could be efficient in terms of computing resources. It is limited, however, in terms of the need to have systems that are aware of all of the implications of their results. This is not a practical assumption, because in reality, in the design of complex products, it is highly unlikely that designers are aware of all of the implications of decisions that they make; nor are they aware of all of the uses of their data.

With respect to computing efficiency, the predispositive architecture performs the best for small, limited sets of events. It is expected, however, that as the number of designers increases, the inter-connections between designers

increases, and the computing efficiency of this approach would decline dramatically. The agent technologies perform in an efficient manner when the agents terminate after their specified mission expires. This efficiency is lost when the agents remain memory-resident, and consume cycles continuously. The event manager consumes the most computing cycles in the comparison of the three architectures for the pilot implementation described here. It is expected, however, that as the volume of designers increases, and the traffic between agents and the databases increases, the event manager will prove to be a more efficient architecture from a computing perspective. The computing efficiencies of the three architectures are compared in Figure 5-28.

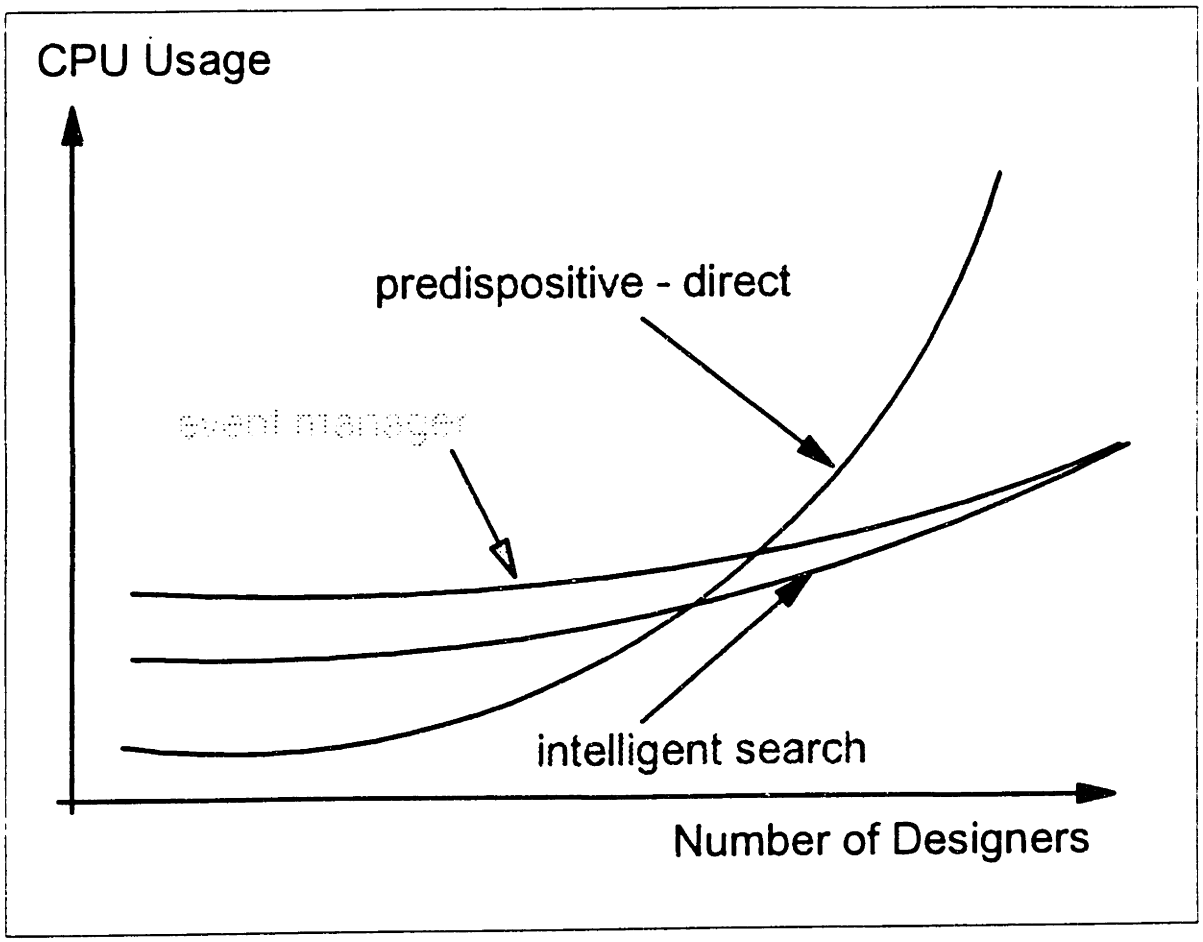


Figure 5-28: Comparison of computing efficiencies of the three event architectures

From the perspective of development, the predispositive architecture requires the most programming-type work, since each event must be described from within the contributors as a specific action to a specific recipient, or set of recipients. The intelligent search approach requires the development of individual systems, or minimally, event class definitions for each recipient. From the perspective of the recipient, this is neither more nor less work than the event management approach. However, from the organization's perspective, many events would be common amongst recipients. Therefore, it is necessary to create an event library, which can be shared amongst recipients, such that they could be used by any number of search agents. This latter approach provides for the same amount of development effort as would be required by the event management approach. in the latter approach, recipients register with respect to existing events, or define new events to the event manager.

In summary, the variations in architecture appear small, in the context of the implementation example described here. It is expected that the limitations and strengths of each architecture, will become more pronounced in a design organization with many designers and a great many events. It is further expected that a hybrid approach incorporating all three architectures would be a legitimate strategy for a full-scale implementation.

6. Summary and Conclusions

In this work, a variety of technical issues have been addressed. These technical developments have been driven by the need to provide technologies that distribute knowledge in a timely fashion across a disparate engineering design organization. The ability to access cross-functional knowledge in a context-sensitive environment, it has been shown, has a positive significant effect in reducing the amount of time that the engineering design process takes.

6.1 Summary of Technical Effort

Various design process configurations have been examined and simulated, with respect to the duration of a hypothetical design process. These organizational configurations were compared, in terms of their costs and cycle-times. The models examined were sequential, parallel, coordinated, and concurrent design. These simulation studies showed that there is a significant variation in the performance of the design organization, depending on the configuration of the organization and the process involved. The sequential process was found to be the most costly, in terms of direct labor expenses and imputed costs of time-to-market, while the other models showed reductions in costs and durations.

The concurrent design process, while popular in terms of modern process re-organization efforts, was found to be heavily dependent on knowledge access amongst the participants in the design process. That is, the concurrent process requires timely access to knowledge of a variety of disciplines at the time that design alternatives are proposed and studied. Using conventional methodologies for accessing such knowledge produced longer design durations than those of the coordinated design model, due to the high volume of time spent in communication between specialists in the various disciplines. In order to reduce such time-intensive processes, and automated knowledge-access mechanism

is required. When such an automated knowledge-access mechanism is used in the simulation models, dramatic reductions in cost and durations as compared with the sequential process, result. This design organization and infrastructure proves to be the lowest cost and duration model studied.

The use of Event-Driven Knowledge-Based Design systems as such an automated knowledge access mechanism has been posited. Event-Driven Knowledge-Based Design systems are knowledge-based systems that are activated as a result of events that occur in the course of the design process, rather than as a result of conscious operator system initiation. By enabling such systems to be sensitive to events in the design process, and as such, to be stimulated automatically at the occurrence of significant events, knowledge is distributed throughout the design organization in such a manner as to provide useful cross-disciplinary analysis and understanding as a central activity in the design process.

The design organization has been characterized in terms of event systems, into designers that contribute to a specific event, and recipients, or those that are affected by the event. Methods for event representations have been developed, in terms of object models and architectures. Detection and reaction mechanisms have been defined using three architectures:

Pre-dispositive: Knowledge of events is embedded within each contributor and each recipient, and event information is transmitted directly between contributors and recipients.

Intelligent Search: Recipients initiate intelligent agents who search through the design database for the conditions of events that are significant to the recipient, and the events are generated by these agents. Contributors

have no notion of events; rather, they simply update the design database.

Event Manager: A Discrete Event Architecture (DEA) monitors the design database in a continuous fashion, and generates events that have been pre-defined and registered by potential recipients. Neither the contributors nor recipients perform any event-specific actions.

In all of these architectures, the knowledge-based systems that are initiated by the events have been designed to provide knowledge to their recipients regarding the contextual impact of the events on the design process in question. As such, these methods transcend message-passing systems because of their ability to assess the impact of the events based on the current state of the recipient's design process relative to the event and the context of the balance of the design process.

An example implementation has been developed, in which the various architectures have been constructed. The implementation involves knowledge access mechanisms for understanding the impact of designs and design modifications that affect continuing efforts in other parts of the design organization. Specifically, the example focuses on the structural design of a bumper beam for automotive design applications, and the events that impact the configuration and detailed design of the structural member. A variety of knowledge bases were developed for this implementation study, including a knowledge base for bumper beam design automation, and an event-driven knowledge base for design analysis. The event-driven knowledge base has the capability to assess the proposed bumper beam design in the context of the vehicle body dimensions, and the vehicle performance specifications. These specifications include weight and cost requirements, as well as design

performance characteristics, such as impact speed.

The event-driven knowledge-based system was shown to be initiated upon the occurrence of significant events that originated in other parts of the design organization. The impact of the events was assessed by the knowledge-based system, and this was transformed into recommendations and automated actions for modifications to the proposed design.

The event systems developed leveraged modular, re-usable object-oriented methodologies that were shared across all systems in the process. That is, event classes with universal slot or attribute definitions and methods were distributed to all participants in the event system. Methods were then specialized locally within each specific system, to allow for system-specific issues. These shared classes were applied to all three architectures, and were activated in different ways, depending on the architecture in use at the time.

This pilot implementation illustrated that regardless of the event architecture, the ability for event-driven knowledge-based systems to participate in an active way in the design process could dramatically improve a design organization's ability to access knowledge in a timely fashion, and thus raise the performance level of the organization as a whole.

6.2 Conclusions

In performing the theoretical development of the architectures for event-driven knowledge-based systems, both extrinsic and intrinsic event systems were generated. This duality was based on the premise that there were potential advantages to both methods, and that the study would identify these, as well as potential pitfalls. Based on the strategy of utilizing commercial CAD systems, however, it became clear that the implementation of intrinsic event

architectures would not be possible. This is due to the structure and architecture of the commercial systems as they exist today. They are built primarily for single users, with interaction to other databases and applications occurring as support activities for the user. The internal structure of these software programs, therefore, is oriented to the user's interactions (and possible local events) but not to sensitivity to events generated exogenously. As such, one major finding is that the limitations of current CAD systems require the use of extrinsic event architectures (such as the event manager method) in any current implementations of Event-Driven Knowledge-Based Design.

In terms of the architectural approaches, it has been shown that the effect of generating and detecting events can be accomplished in all of the methods developed herein. The selection of an approach is, therefore, best made on the basis of the specific design organization and the applications of the knowledge based systems. It is probably useful, for example, to consider direct (or pre-dispositive) architectures for designers that perform highly coupled design activities. This would facilitate the optimal development process by reducing or eliminating unnecessary iterations. For interactions between designers based on less direct dependencies, it is probable that intelligent search or event management architectures would provide the necessary infrastructure for event generation, detection, and consequential activity initiation.

The event architectures developed here are not mutually exclusive. That is, the event-driven knowledge-based systems can react to, and can support any of the architectures without modification to the applications themselves. As such, a hybrid architecture within a large engineering design organization is not only feasible, but probably desirable, based on the methodology selection criteria outlined above.

Based on the experimental studies performed in this research, it can be concluded that an immediate, knowledge-based and contextual response to proposed or actual changes or design decisions can be achieved, despite functional, physical, or temporal disparities in the design organization. The effect of such immediacy in response to such decisions or changes is to reduce the amount of iteration in the design cycle that would otherwise have arisen due to unforeseen or undetected effects of individual actions or decisions on other parts of the design process or the product. This is accomplished because of two major technological contributions: (i) the event system, and (ii) the distributed knowledge-based systems. The combination of these technologies provides the means to access knowledge (rather than simply data) in a contextual fashion (how the context applies to the current state of the design) at the appropriate time (when the events occur).

The framework for implementation of such systems has been laid, whether current infrastructures and technologies are used, or within the context of new event-driven applications and networks.

6.3 Future Work

The issue of scale was only briefly addressed in this work. When these event architectures and applications are applied to a large number of designers, a variety of consequences are likely to arise. These include an increase in CPU utilization (as described in Chapter 5), an increased level of development requirement with respect to the events themselves, and more emphasis on the management of the interactions between designers. As the number of designers in the event network increases, the relevant events will also increase. A study of such issues of scale would provide beneficial results, in that the large-scale implementation issues would be identified and quantified in a theoretical mode prior to actual implementation. One potential environment for such a study would be a classroom collaborative design exercise, using students across a variety of disciplines (potentially from different educational institutions).

A great deal of work remains in the areas of applications development, relative to domain-specific event-driven knowledge-based systems. For example, a set of generic applications for the automotive, aerospace, AEC, or other industries could be developed. These could be applied, with some organization-specific customizations to the general industry engineering design situation.

A further set of research areas consists of the development of generic events - whether industry-specific, or more general. This would be beneficial since these event definitions could be used across applications and organizations, and would provide a framework upon which problem-specific event definitions would be developed.

Appendix A

Simulation Model Equations for Concurrent Design Organization and Process

```
COMPLETED_DESIGNS(T) = COMPLETED_DESIGNS(T - DT) + (OUTPUT) * DT
INIT COMPLETED_DESIGNS = 0

OUTPUT = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
COMPLETED_DESIGNS_2(T) = COMPLETED_DESIGNS_2(T - DT) + (OUTPUT_2) * DT
INIT COMPLETED_DESIGNS_2 = 0

OUTPUT_2 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
COMPLETED_DESIGNS_3(T) = COMPLETED_DESIGNS_3(T - DT) + (OUTPUT_3) * DT
INIT COMPLETED_DESIGNS_3 = 0

OUTPUT_3 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
COMPLETED_DESIGNS_4(T) = COMPLETED_DESIGNS_4(T - DT) + (OUTPUT_4) * DT
INIT COMPLETED_DESIGNS_4 = 0

OUTPUT_4 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
COMPLETED_DESIGNS_5(T) = COMPLETED_DESIGNS_5(T - DT) + (OUTPUT_5) * DT
INIT COMPLETED_DESIGNS_5 = 0

OUTPUT_5 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
COMPLETED_DESIGNS_6(T) = COMPLETED_DESIGNS_6(T - DT) + (OUTPUT_6) * DT
INIT COMPLETED_DESIGNS_6 = 0

OUTPUT_6 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
DESIGN(T) = DESIGN(T - DT) + (RATE - OUTPUT - FEEDBACK) * DT
INIT DESIGN = 0
   TRANSIT TIME = VARIES
   INFLOW LIMIT = INF
   CAPACITY = INF

RATE = DESIGNERS
OUTPUT = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
FEEDBACK = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
```

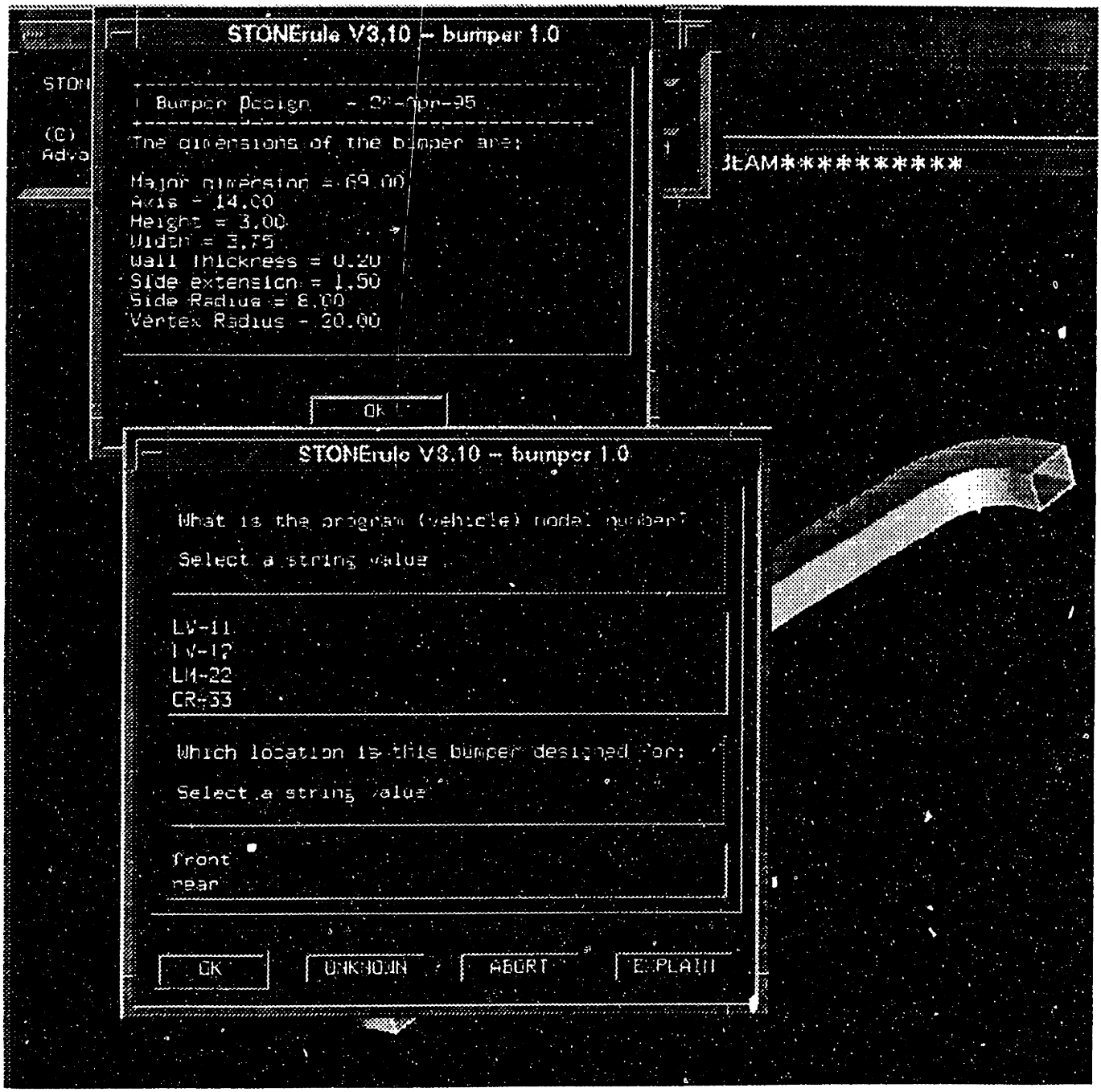
```
   NO-LEAK ZONE = 50%
DESIGN_2(T) = DESIGN_2(T - DT) + (RATE_2 - OUTPUT_2 - FEEDBACK_2) * DT
INIT DESIGN_2 = 0
   TRANSIT TIME = VARIES
   INFLOW LIMIT = INF
   CAPACITY = INF

RATE_2 = DESIGNERS
OUTPUT_2 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
FEEDBACK_2 = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
DESIGN_3(T) = DESIGN_3(T - DT) + (RATE_3 - OUTPUT_3 - FEEDBACK_3) * DT
INIT DESIGN_3 = 0
   TRANSIT TIME = VARIES
   INFLOW LIMIT = INF
   CAPACITY = INF

RATE_3 = DESIGNERS
OUTPUT_3 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
FEEDBACK_3 = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
DESIGN_4(T) = DESIGN_4(T - DT) + (RATE_4 - OUTPUT_4 - FEEDBACK_4) * DT
INIT DESIGN_4 = 0
   TRANSIT TIME = VARIES
   INFLOW LIMIT = INF
   CAPACITY = INF

RATE_4 = DESIGNERS
OUTPUT_4 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
FEEDBACK_4 = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
DESIGN_5(T) = DESIGN_5(T - DT) + (RATE_5 - OUTPUT_5 - FEEDBACK_5) * DT
INIT DESIGN_5 = 0
   TRANSIT TIME = VARIES
   INFLOW LIMIT = INF
   CAPACITY = INF

RATE_5 = DESIGNERS
OUTPUT_5 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
FEEDBACK_5 = LEAKAGE OUTFLOW
```

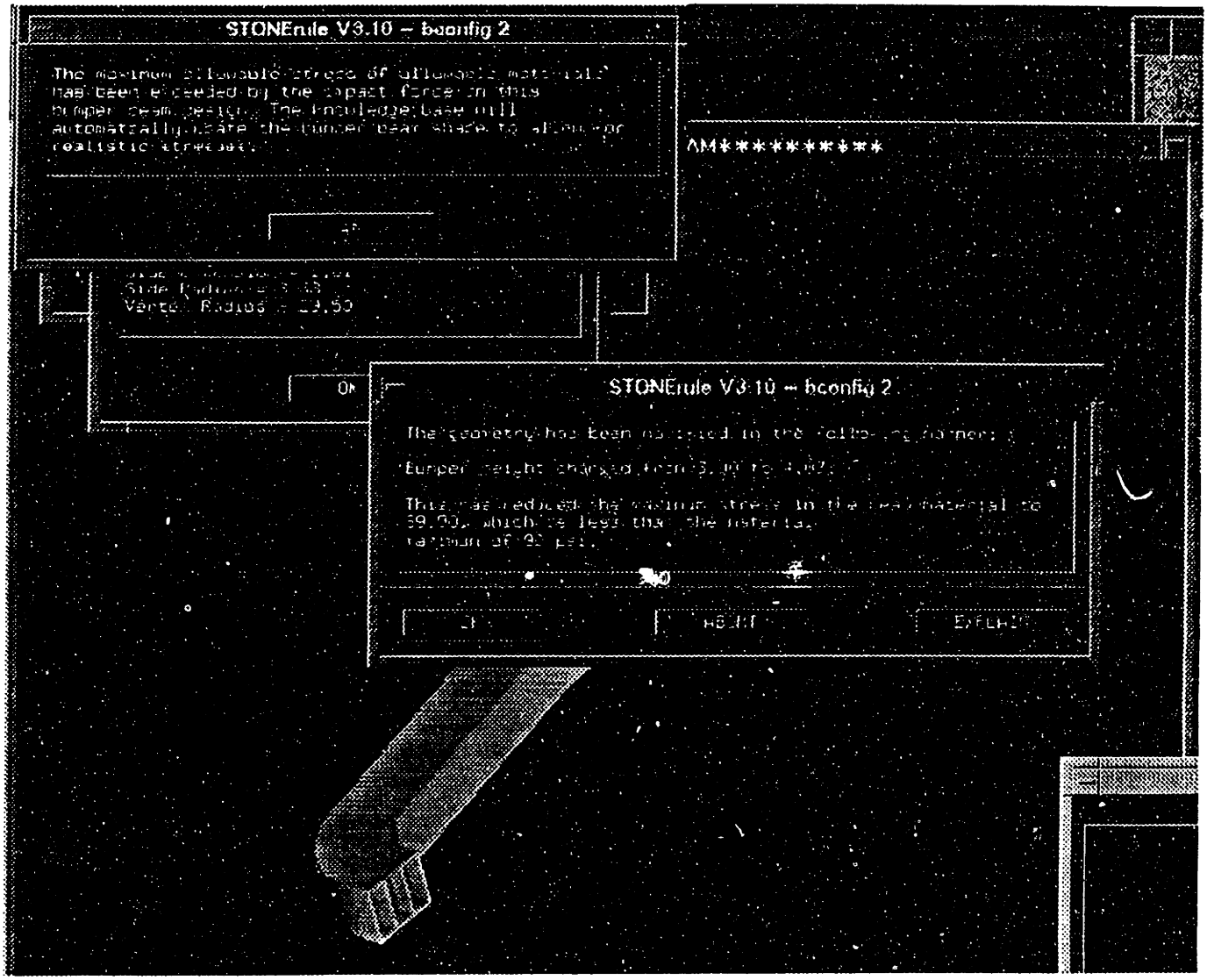
```
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
DESIGN_6(T) = DESIGN_6(T - DT) + (RATE_6 - OUTPUT_6 - FEEDBACK_6) * DT
INIT DESIGN_6 = 0
   TRANSIT TIME = VARIES
   INFLOW LIMIT = INF
   CAPACITY = INF

RATE_6 = DESIGNERS
OUTPUT_6 = CONVEYOR OUTFLOW
   TRANSIT TIME = PRODUCTIVITY
FEEDBACK_6 = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
DESIGN_INVENTORY(T) = DESIGN_INVENTORY(T - DT) + (COMPLETION_RATE) * DT
INIT DESIGN_INVENTORY = 0

COMPLETION_RATE =
 (OUTPUT + OUTPUT_2 + OUTPUT_3 + OUTPUT_4 + OUTPUT_5 + OUTPUT_6) * (1-FEEDBACK_RATIO)
INITIAL_INVENTORY(T) = INITIAL_INVENTORY(T - DT) + (- DRAWDOWN_RATE) * DT
INIT INITIAL_INVENTORY = 1000

DRAWDOWN_RATE = DESIGNERS
INVENTORY(T) = INVENTORY(T - DT) + (FEEDBACK + IN_1' - RATE) * DT
INIT INVENTORY = 0

FEEDBACK = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
IN_1' =
 DRAWDOWN_RATE/6 + (OUTPUT_2 + OUTPUT_3 + OUTPUT_4 + OUTPUT_5 + OUTPUT_6) *
  DIRECTED_FEEDBACK
RATE = DESIGNERS
INVENTORY_2(T) = INVENTORY_2(T - DT) + (FEEDBACK_2 + IN_2 - RATE_2) * DT
INIT INVENTORY_2 = 0

FEEDBACK_2 = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
IN_2 =
 DRAWDOWN_RATE/6 + (OUTPUT + OUTPUT_3 + OUTPUT_4 + OUTPUT_5 + OUTPUT_6) *
   DIRECTED_FEEDBACK
RATE_2 = DESIGNERS
INVENTORY_3(T) = INVENTORY_3(T - DT) + (FEEDBACK_3 + IN_3 - RATE_3) * DT
INIT INVENTORY_3 = 0
```

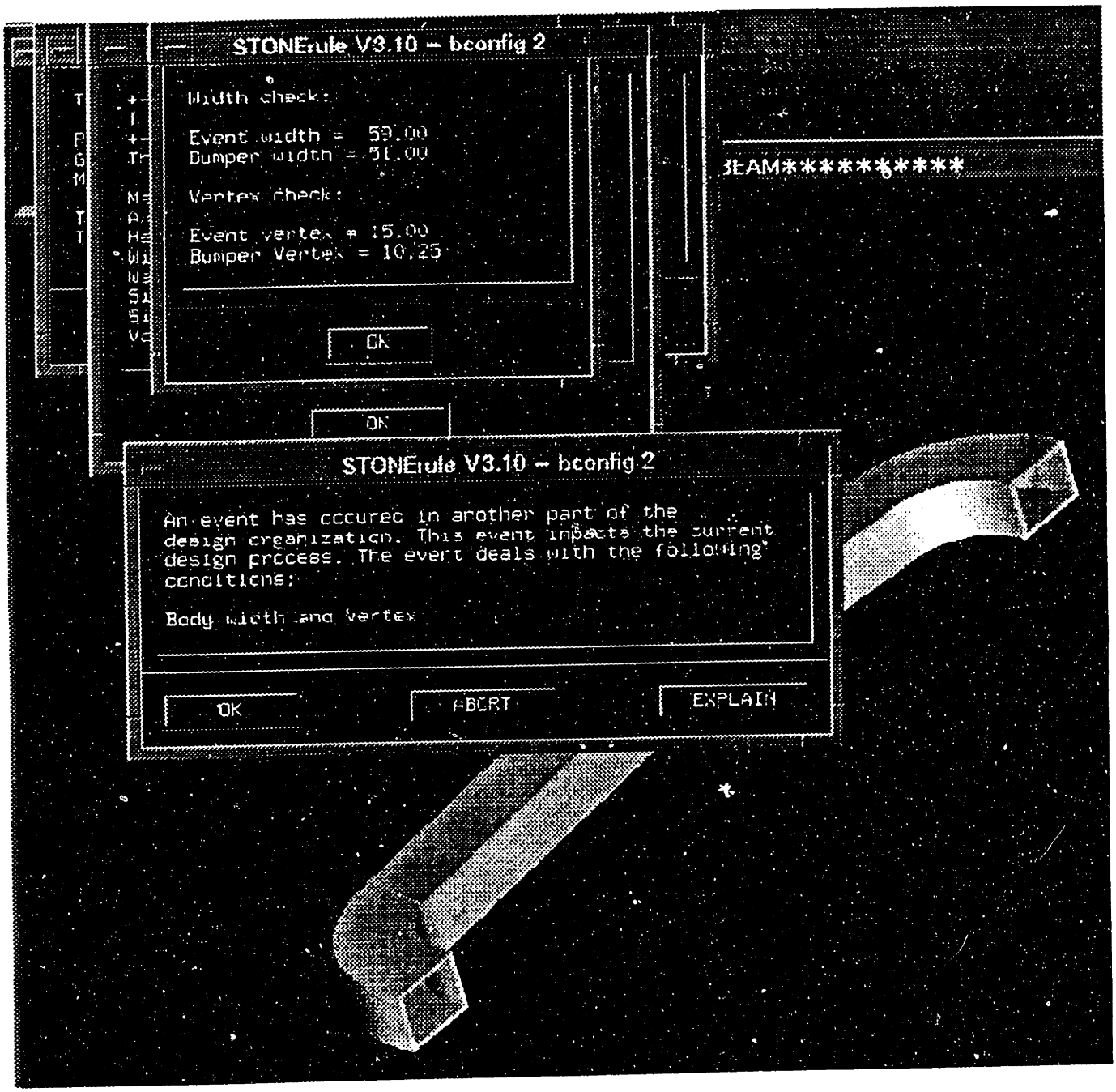
```
FEEDBACK_3 = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
IN_3 =
 DRAWDOWN_RATE/6 + (OUTPUT + OUTPUT_2 + OUTPUT_4 + OUTPUT_5 + OUTPUT_6) *
   DIRECTED_FEEDBACK
RATE_3 = DESIGNERS
INVENTORY_4(T) = INVENTORY_4(T - DT) + (FEEDBACK_4 + IN_4 - RATE_4) * DT
INIT INVENTORY_4 = 0

FEEDBACK_4 = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
IN_4 =
 DRAWDOWN_RATE/6 + (OUTPUT + OUTPUT_2 + OUTPUT_3 + OUTPUT_5 + OUTPUT_6) *
   DIRECTED_FEEDBACK
RATE_4 = DESIGNERS
INVENTORY_5(T) = INVENTORY_5(T - DT) + (FEEDBACK_5 + IN_5 - RATE_5) * DT
INIT INVENTORY_5 = 0

FEEDBACK_5 = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
IN_5 =
 DRAWDOWN_RATE/6 + (OUTPUT + OUTPUT_2 + OUTPUT_3 + OUTPUT_4 + OUTPUT_6) *
   DIRECTED_FEEDBACK
RATE_5 = DESIGNERS
INVENTORY_6(T) = INVENTORY_6(T - DT) + (FEEDBACK_6 + IN_6 - RATE_6) * DT
INIT INVENTORY_6 = 0

FEEDBACK_6 = LEAKAGE OUTFLOW
   LEAKAGE FRACTION = FEEDBACK_RATIO
   NO-LEAK ZONE = 50%
IN_6 =
 DRAWDOWN_RATE/6 + (OUTPUT + OUTPUT_2 + OUTPUT_3 + OUTPUT_4 + OUTPUT_5) *
   DIRECTED_FEEDBACK
RATE_6 = DESIGNERS
DESIGNERS = 100
DIRECTED_FEEDBACK = FEEDBACK_RATIO/5
FEEDBACK_RATIO = .25
PRODUCTIVITY = 1
```

Appendix B

Screen Interaction For Bumper Beam Design Knowledge-Based System and Event-Driven Knowledge-Based System




STONEmle V3.10 – bconfig 2
STONEmle V3.10 – bconfig 2


STONErule V3.10 – bconfig 2
Width check:
Event width = 59.00
Bumper width = 51.00
Vertex check:
Event vertex = 15.00
Bumper Vertex = 10.25
OK
OK
STONErule V3.10 – bconfig 2
An event has occured in another part of the design organization. This event impacts the current design process. The event deals with the following conditions:
Body width and Vertex
OK
ABORT
EXPLAIN

References

[Allen, 1976] Allen, Thomas J., and Fusfield, Alan R., "Design for Communication in the Research and Development Lab," *Technology Review*, Massachusetts Institute of Technology, May 1976.

[Allen, 1992] Allen, Wayne, *GEM: Global Event Management in CAD Frameworks*, in Enterprise Integration Modeling: Proceedings of the First International Conference, Charles Petrie, ed., MIT Press, Cambridge, MA , 1992.

[Archer, 1984] Archer, Bruce, "Systematic Method for Designers", 1965, reproduced in *Developments in Design Methodology*, Nigel Cross, ed., John Wiley & Sons, Chichester, U.K., 1984.

[Asimow, 1962] Asimow, M., *Introduction to Design*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1962

[Bahk, 1993] Bahk, B. H., and Gort, M., "Decomposing Learning by Doing in New Plants", *Journal of Political Economy*, Volume 101, Number 4, 1993.

[Becker, 1973] Becker, J. M., *A Structural Design Process Philosophy and Methodology*, Report No. UC SESM 73-13, Doctoral Dissertation, Department of Structural Engineering, University of California Berkely, September 1973.

[Berson, 1992] Berson, A., *Client/Server Architectures*, McGraw Hill, New York, 1992.

[Booch, 1994] Booch, Grady, *Object-Oriented Analysis and Design*, the Benjamin/Cummins Publishing Company, Redwood City, CA, 1994.

[Boothroyd, 1994] Boothroyd, G., Dewhurst, P., Knight, W., *Product Design for Manufacture and Assembly*, Marcel Dekker, Inc., New York, 1994.

[Brogan, 1982] Brogan, William L., *Modern Control Theory*, Prentice Hall, 1982.

[Brynjolfsson, 1994] Brynjolfsson, Erik, and Hitt, Lorin, *Computers and*

Economic Growth: Firm-Level Evidence, Working Paper 94-005WP, Industrial Performance Center, Massachusetts Institute of Technology, Cambridge, MA 1994.

[Christian, 1995] Christian, Nicole, M., *Detroit in 3-D: Car Design Gets New Dimension*, The Wall Street Journal, New York, Dow Jones & Company, March 24, 1995.

[Coles, 1994] Coles, B. C., and Reinschmidt, K. F., *Computer-Integrated Construction*, Civil Engineering Magazine, American Society of Civil Engineers, New York, June, 1994.

[Cope, 1913] Cope, E.A., *Filing Systems: Their Principles and Their Application to Modern Office Requirements*, London, Sir Isaac Pittman & Sons, 1913, as quoted in Information Technology and Organizational Transformation, Scott Morton, ed., Oxford University Press, Oxford, 1991.

[Card, 1983] Card, S. K., Moran, T. P., and Newell, A., *The Psychology of Human-Computer Interaction*, Lawrence Earlbaum Associates, New Jersey, 1983.

[Chapman, 1994] Chapman, C.D. and Jakiela, M.J., *Genetic Algorithm-Based Structural Topology Design with Compliance and Manufacturability Considerations*,, American Society of Mechanical Engineers Journal of Mechanical Design, November 1994.

[Christainsen, 1993] Christiansen, T. R., *Modeling Efficiency and Effectiveness of Coordination in Engineering Design Teams: VDT- The Virtual Design Team*, Doctoral Dissertation, Department of Civil Engineering, Stanford University, September 1993.

[Cinquegrana, 1990] Cinquegrana, David, *Intelligent CAD Automates Mold Design*, Mechanical Engineering, July 1990.

[Clark, 1994] Clark, R. A., Reinschmidt, K. F., and Finn, G. A., *Automotive Applications of Knowledge Based Systems for Design and Manufacturing*, Presented at the Fall 1994 ORSA/TIMMS Conference, Detroit, MI, 1994

[Cross, 1984] Cross, Nigel, *Developments in Design Methodology*, John Wiley & Sons, Chichester, U.K., 1984.

[Crowfoot, 1992] Crowfoot, N., Hatfield, S., and Swank, M., *A Knowledge-based Cost Estimating Tool*, Xerox Corporation Internal Report, Webster, NY, April 1992.

[Cusumano, 1990] Cusumano, M. A., and Nobeoka,K., *Strategy, Structure, and Performance in Project Development: Observations from the Auto Industry*, Working Paper 3150-90-BPS, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, April 1990.

[Davenport, 1993] Davenport, Thomas H., *Process Innovation, Reengineering Worth through Information Technology*, Harvard Business School Press, Boston, MA, 1993.

[Dong, 1993] Dong, X., Gilman, C., and Wozny, M., *Feature-based Fixture Design and Set-Up Planning*, in Artificial Intelligence in Design and Manufacturing, Dong, Z. (ed), Prentice-Hall, Englewood Cliffs, N.J., 1993

[Dunker, 1935] Dunker, K., "On Problem-Solving," [Translated by L.S. Lees], Psychological Monographs, Col 58, No. 5, berlin, 1935.

[Dym, 1991] Dym, C. L., and Levitt, R. E., *Knowledge-Based Systems in Engineering*, McGraw-Hill, Inc., New York, 1991.

[Dym, 1994] Dym, Clive L., *Engineering Designm A Synthesis of Views*, Cambridge University Press, Cambridge, U.K., 1994.

[Engelmore, 1988] Englemore, R. and Morgan, A., *Blackboard Systems*, Addison Wesley, Reading MA, 1988.

[Eppinger, 1990] Eppinger, S., et al., *Organizing Tasks in Complex Design Projects*, Working Paper Number 3183-89-MS, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, June 1990.

[Eppinger, 1993] Eppinger, S.D,. and McCord, K. R., *Modeling the Impact of Organizational Structure on Design Lead Times,* Working Paper 0893-259, Sloan School of Management, Massachusetts institute of Technology, Cambridge, MA, August 1993.

[Favela, 1993] Favela, J., and Connor, J. J., *Organizational Memory*

Management, Report Number IESL93-22, Intelligent Engineering Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA, December 1993.

[Feigengbaum, 1988] Feigenbaum, E. McCormack, P., and Nii, H.P., *The Rise of the Expert Company*, Times Books, New York, 1988.

[Fergusen, 1992] Fergusen, E. S., *Engineering and the Mind's Eye*, The MIT Press, Cambridge, MA, 1992.

[Field, 1992] Field, John D., *A Rule-Based System for Aircraft Engine Tooling*, S. M. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, June 1992.

[Finn, 1986] Finn, G. A., and Reinschmidt, K. F., *Expert Systems in an Engineering Construction Firm*, Symposium on Expert Systems in Civil Engineering, American Society of Civil Engineers, Seattle, WA, March 1986.

[Finn, 1988] Finn, G. A., Reinschimdt, K. F., and Bassler, E. J., *Expert Systems for Control of Operations in Real Time*, Proceedings of the Spring National Meeting of the American Institute of Chemical Engineers, New Orleans, LA, March 1988.

[Finn, 1989] Finn, G. A., and Reinschmidt, K. F., *Applications of Expert Systems for Plant Operations and Maintenance*, Pproceedings of the American Institute of Chemical Engineers 1989 Summer National Meeting, Philadelphia, PA, August 1989.

[Fisher, 1986] Fisher, Thomas, *Intelligent Computers*, Progressive Architecture, June 1986.

[Forde, 1989] Forde, B. W. R., and Stiemer, S.F., *Development of Engineering Software Within a Generic Appliction Framework*, in Microcomputers in Civil Engineering, Volume 4, No. 3, Elsevier Science Publishing Company, New York, September 1989.

[Forrester, 1962] Forrester, J, *Industrial Dynamics*, MIT Press, Cambridge, MA, 1962.

[Galbraith, 1973] Galbraith, Jay, *Designing Complex Organizations*, Addison Welsey, Reading, MA., 1973.

[Galbraith, 1974] Galbraith, J., *Organization Design, An Information Processing View*, Harvard Business Review, Boston, MA, May 1974.

[Gammons,1992] Gammons, Richard A., *Eskimo: An Expert System for Kodak Injection Molding Operations*, in Tong, Chris and Sriram, Duvvuru, Artificial Intelligence in Engineering Design - Volume III, Academic Press, Inc., San Diego, CA, 1992.

[Garvin, 1987] Garvin, D. A., *Competing on the Eight Dimensions of Quality*, Harvard Business Review, Boston, MA., November-December, 1987.

[Gebala, 1991] Gebala, D. A., and Eppinger, S. D., *Modeling the Impact of Organizational Structure on Design Lead Time and Product Quality*, Working Paper No.3301-91-MS, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, May 1991.

[Gleckman, 1993] Gleckman, *et al, The Technology Payoff*, Business Week, June 14, 1993, p57.

[Goldman, 1995] Goldman, S. L., Nagel, R. N., and Preiss, K., *Agile Competitors and Virtual Organizations, Strategies for Enriching the Customer*, Von Nostrand Reinhold, New York, 1995.

[Goranson, 1992] Goranson, H.T., *The CIMOSA Approach as an Enterpise Integration Strategy*, in Enterprise Integration Modeling: Proceedings of the First International Conference, Charles Petrie, ed., MIT Press, Cambridge, MA, 1992.

[Gottschalk, 1994] Gottschalk, Mark A., *How Boeing got to 777 Heaven*, Design News, September 1994.

[Haase, 1992] Haase, Bruce, *Who and What is Smart?*, DesignNet, June 1992.

[Haase, 1994] Haase, Bruce, *The Neon: A Shining Example of Design Efficiency*, Computer Graphics World, Penwell Publishing Company, February 1994.

[Haimes, 1973] Haimes, Y. Y., "Multilevel Dynamic Programming Structure for Regional Water Resource Management, in Himmelblau, D. M., *Decomposition of Large Scale Problems*, North-Holland Publishing Company, American Elsevier Publishing Company, New York, 1973.

[Halgamuge, 1993] Halgamuge, S.K., Poechmueller, W., and Glesner, M., *A Rule Based Prototype System for Automatic Classification in Industrial Quality Control*, IEEE International Conference on Neural Networks, July 1993.

[Hansen,1977] Hansen, F., "Systematic Design", 2nd Ed., VEB-Verlag Technik, Berlin, 1965, as referenced in Pahl, G., and Beitz, W., *Engineering Design: A Systematic Approach*, Springer Verlag, Berlin 1977.

[Hapgood, 1995] Hapgood, F., *Beneath the Big Dig*, CIO Magazine, international Data Group, Framingham, MA, February 1995.

[Hauser, 1988] Hauser, J. R., and Clausing, D., *The House of Quality*, Harvard Business Review, Boston, MA, May-June 1988.

[Hax, 1991] Hax, Arnaldo C., and Majluf, Nicolas S., *The Strategy Concept and Process: A Pragmatic Approach*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1991.

[Helfner, 1995] Helfner, Robert S., *Knowledge-Based CATIA Model Checking*, Spring Conference of the CATIA Operator's Exchange, Colorado Springs, Colorado, March 1995.

[HPS, 1994] High Performance Systems, Inc., "think" User's Guide", 1994.

[Himmelblau, 1973] Himmelblau, D. M., *Decomposition of Large Scale Problems*, North-Holland Publishing Company, American Elsevier Publishing Company, New York, 1973.

[Hood, 1994] Hood, William, *Knowledge-Based Design at Lockheed*, Proceedings of the Spring CATIA Operator's Exchange, Atlanta, Georgia, 1994.

[Hoffman, 1993] Hoffman, G.E., and Oleksy, C.A., *Expert System Strategies for Planning and Scheduling Software*, American Production and Inventory Control Society Process Industries

Symposium, Cincinnati, OH, June 1993.

[Hopper, 1990] Hopper, Max, D., *Rattling SABRE - New Ways to Compete on Information*, Harvard Business Review, Boston, MA, May-June 1990.

[Hughes, 1995] Hughes, David, *CATIA Revision Covers Non-IBM Workstations*, Aviation Week and Space Technology, March 20, 1995.

[Hummel, 1988] Hummel, K.E., and Brooks, S. L., *Using Hierarchically Structured Problem-Solving Knowledge in a Rule-Based Process Planning System*, in Expert Systems and Intelligent Manufacturing, Oliff, M.D. ed., Elsevier Science Publishing Company, 1988.

[Jakiela, 1989] Jakiela, M. J., *Intelligent Suggestive CAD Systems: Research Overview*, Paper Number LMP-89-021, Laboratory for Manufacturing and Productivity, Massachusetts Institute of Technology, Cambridge, MA, February 1989.

[Jones, 1963] Jones, J. Christopher, "A Method of Systematic Design", in Jones, J.C. and Thornley, D. (eds) *Conference on Design Methods*, Pergamon Press, Oxford, U.K., 1963.

[Joyce, 1988] Joyce, Richard D., *Applying Today's Technologies to Production Planning and Control*, in Expert Systems and Intelligent Manufacturing, Oliff, M.D. (ed), Elsevier Science Publishing Company, New York, 1988

[Keigler, 1992] Keigler, Arthur L., *Improving Performance in Product Deployment by Improving Coordination*, Working Paper #3506-92, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, December 1992.

[Keyes, 1990] Keyes, Jessica, *The New Intelligence*, Harper Business Press, New York, 1990.

[Kurland, 1992] Kurland, R. H., *Understanding Variable Driven Modeling*, Technicom, Inc. Technical Publication, Clifton, N.J., December 1992.

[Lauzun, 1992] Lauzun, David K., *Simultaneous Engineering for Manufacturing Process Development*, Master's Thesis,

Massachusetts Institute of Technology, Cambridge, MA, June, 1992.

[Liker, 1992] Liker J. K., Fleischer, M., and Arnsdorf, D., *Fulfilling the Promises of CAD*, Sloan Management Review, Massachusetts Institute of Technology, Cambridge, MA, Spring 1992.

[Lindgren, 1993] Lindgren, R., and Luray, M., *Artificial Intelligence Gets Real*, Enterprise, Digital Equipment Corporation, Merrimack, NH, October 1993.

[Malone, 1991] Malone, Thomas W., and Crowston, K., *Toward and Interdisciplinary Theory of Coordination*, Working Paper Number CCS TR# 120, Massachusetts Institute of Technology, Cambridge, MA, April 1991.

[March, 1958] March, James G., and Simon, Herbert A., *Organizations*, (New York, John Wiley, 1958.

[Marefat, 1992] Marefat, M., and Kashyap, R. L., *Automatic Construction of Process Plans from Solid Model Representations*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 5, September 1992.

[Maus, 1991] Maus, R., and Keyes, J., *Handbook of Expert Systems in Manufacturing*, McGraw Hill, New York, 1991.

[McCord, 1993] McCord, K.R., and Eppinger, S.D., *Managing the Integration Problem in Concurrent Engineering*, Working Paper #3594-93-MSA, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, August, 1993.

[McFarlan, 1984] McFarlan, Warren, *Information Technology Changes the Way You Compete*, Harvard Business Review, Boston, MA, May-June 1984.

[Mesarovic, 1970] Mesarovic, M.D., Macko, D., and Takahara, Y., *Theory of Hierarchical, Multilevel Systems*, Academic Press, New York, 1970.

[Meyer, 1991] Meyer, Mark H., and Curley, Kathleen F., *Putting Expert Systems Technology to Work*, Sloan Management Review, Massachusetts Institute of Technology, Cambridge, MA,

Winter 1991.

[Minsky, 1985] Minsky, Marvin, *The Society of Mind*, Simon & Schuster, New York, 1985.

[Moeller, 1994] Moeller, M., *Boeing Goes On-Line with 777 Design*, Computer-Aided Engineering, Penton Publishing Co., Cleveland, OH, August 1994.

[Mosow, 1985] Mosow, J., *Towards Better models of the Design Process*, Artificial Intelligence Magazine, Volume 6, Number 1, 1985.

[NRC, 1991] National Research Council, *Improving Engineering Design*, National Academy Press, Washington, D.C., 1991.

[Nevins, 1989] Nevins, J. L., and Whitney, D. E., *Concurrent Design of Products and Processes*, McGraw Hill, New York, 1989.

[Norman, 1986] Norman, Donald. A., *Cognitive Engineering - Cognitive Science*, McGraw Hill, New York, 1986.

[Oshuga, 1989] Ohsuga, Setsuo, *Toward Intelligent CAD Systems*, Computer-Aided-Design, Vol 21, Number 5, Butterworth & Co., June 1989.

[Pahi, 1977] Pahl, G., and Beitz, W., *Engineering Design: A Systematic Approach*, Springer Verlag, Berlin 1977.

[Piore, 1989] Piore, M. J., *Corporate Reform in American Manufacturing and the Challenge to Economic Theory*, Paper No. 533, Department of Economics, Massachusetts Institute of Technology, Cambridge, Massachusetts, September, 1989.

[Porter, 1980] Porter, Michael E., *Competitive Strategy, Techniques for Analyzing Industries and Competitors*, The Free Press, New York, 1980.

[Porter, 1985] Porter, Michael E., *Competitive Advantage, Creating and Sustaining Superior Performance*, The Free Press, New York, 1985.

[Pounds, 1963] Pounds, William F., "The Scheduling Environment", in Muth and Thompson (eds), *Industrial Scheduling*, Englewood

Cliffs, N.J., Prentice Hall, 1963.

[Procter, 1994] Procter, P., *Boeing Rolls Out 777 to Tentative Market*, Aviation Week and SpaceTechnology, McGraw Hill, New York, April 11, 1994.

[Querenet, 1992] Querenet, B., *CIMOSA - A European Development for Enterprise Integration: Part III Enterprise Integrating Infrastructure*, in Enterprise Integration Modeling: Proceedings of the First International Conference, Charles Petrie, ed., MIT Press, Cambridge, MA , 1992.

[Racine, 1992] Racine, J. P., and Borsje, H. J., *The Use of Expert Syetems for Recovery Boiler Tube Leak Detection*, Proceedings of the TAPPI Process Control Conference, Atlanta, GA, 1992.

[Reinschmidt, 1992] Reinschmidt, K. F., and Finn, G. A., *Integration of Expert Systems, Databases, and Computer-Aided Design*, in Intelligent Design and Manufacturing, Kusiak, A. (ed), John Wiley & Sons, New York, 1992.

[Reinschmidt, 1994] Reinschmidt, K. F., and Finn, G. A., *Knowledge-Based Systems Embedded in CAD*, IEEE Expert, Volume 9, Number 4, IEEE Computer Society, Los Alamitas, CA, August 1994.

[Reinschmidt, 1995n] Reinschmidt, Kenneth F., Personal Communications and Unpublished Notes, 1995.

[Rich, 1994] Rich, B. R., and Janos, L., *Skunk Works*, Little, Brown, and Company, Boston, MA, 1994.

[Robertson, 1990] Robertson, D. C., and Allen, T. J., *Evaluating the Use of CAD Systems in Mechanical Design Engineering*, Working Paper Number 3-90, International Center for Research on the Management of Technology, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, January 1990.

[Robertson, 1991a] Robertson, D. C., and Allen, T. J., *CAD System Use and Engineering Performance in Mechanical Design*, Working Paper 32-91, International Center for Research on the Management of Technology, Sloan School of Management, Massachusetts Institute of Technology,

Cambridge, MA, January 1991.

[Robertson, 1991b] Robertson, D. C., Ulrich, K., and Filerman, M., *Cognitive Complexity and CAD Systems: Beyond the Drafting Board Metaphor*, Working Paper 31-91, International Center for Research on the Management of Technology, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, January 1991.

[Robinson, 1990] Robinson, P., *The Emergence of Smart CAD Software*, Computer Graphics World, July 1990.

[Rodenacker, 1970] Rodenacker, W. G., "Methodisches Konstruieren", Springer, Berlin, 1970, as referenced in Pahl, G., and Beitz, W., *Engineering Design: A Systematic Approach*, Springer Verlag, Berlin 1977.

[Sage, 1977] Sage, Andrew P., *Methodology for Large-Scale Systems*, McGraw-Hill Book Company, New York, 1977.

[Scott Morton, 1991] Scott Morton, M. S., *The Organization of the 1990's, Information Technology and Organizational Transformation*, Oxford University Press, Oxford, 1991.

[Schay, 1991] Schay, Peter, *Managing the Changing Enterprise*, The Consultant Forum Special Edition, Digital Equipment Corporation, Nashua, NH. 1991.

[Schultz, 1993] Schultz, G., *Chrysler Leverages CAD*, Managing Automation, August, 1993.

[Shannon, 1962] Shannon, C., and Weaver, W., *The Mathematics of Communication*, University of Illinois Press, Urbana Ill., 1962.

[Simon, 1973] Simon, Herbert, *The Structure of Ill-Structured Problems*, Artificial Intelligence, Volume 4, North-Holland Publishing Company, 1973.

[Simon, 1981] Simon, Herbert A., *The Sciences of the Artificial*, 2nd Edition, MIT Press, Cambridge, MA, 1981.

[Smith, 1880] Smith, Adam, *An Inquiry into the Nature and Causes of the Wealth of Nations*, Oxford, Clarendon Press, 1880.

[Springer, 1995] Springer, D., *Bell Knowledge-Based System Provides Producibility Analysis and Automated Tool Design for the V-22 Integrated Wiring System*, Proceedings of the 51st Annual Forum of the American Helicopter Society, Fort Worth, TX, May 1995.

[Sriram, 1992] Sriram, D., et. al., *DICE: An Object-Oriented Programming Environment for Cooperative Engineering Design*, in Artificial Intelligence in Engineering Design - Volume III, Tong & Sriram eds., Academic Press, Inc., San Diego, CA, 1992.

[Stephanou, 1992] Stephanou, S. E., and Spiegel, F., *The Manufacturing Challenge*, Van Nostrand reinhold, New York, 1992.

[Straub, 1964] Straub, H., *A History of Civil Engineering: An Outline From Ancient to Modern Times*, The MIT Press, Cambridge, MA., 1964.

[Suh, 1990] Suh, Nam P., *The Principles of Design*, Oxford University Press, New York, 1990.

[Sutherland, 1963] Sutherland, Ivan E., *SKETCHPAD*, Proceedings of the AFIPS, Vol 23, 1963.

[Thurow, 1991] Thurow, Lester, *Foreword*, in The Organization of the 1990's, Information Technology and Organizational Transformation, Scott Morton, ed., Oxford University Press, Oxford, 1991.

[Tong, 1992] Tong, Chris and Sriram, Duvvuru, *Artificial Intelligence in Engineering Design - Volume III*, Academic Press, Inc., San Diego, CA, 1992.

[Ullman, 1988] Ullman, D. G., Wood. S., and Craig, D., *The Importance of Drawing in the Mechanical Design Process*, Computers and Graphics, Volume 2, Number 1, 1988.

[Ulrich, 1995] Ulrich, K. T., and Eppinger, S. D., *Product Design and Development*, McGraw Hill, New York, 1995.

[Wilson, 1995] Wilson, J., and Bryan, P., *3-D modeling as a tool to Improve Integrated Design andConstruction*, Construction Industry Institute Source Document No. 104, November

1994.

[Womack, 1990] Womack, J. P., Jones, D. T., and Roos, D., *The Machine that Changed the World, The Story of Lean Production*, Macmillan Publishing Company, New York, 1990.

[Wong, 1970] Wong, A. K. C., and Bugliarello, G., *Artificial Intelligence in Continuum Mechanics*,Journal of the Engineering Mechanics Division, Proceedings of the American Society of Civil Engineers, December 1970.

[Woodruff, 1993] Woodruff, David and Miller, L., *Chrysler's Neon: Is this the small car Detroit couldn't build?*, Business Week, May 3, 1993.

[von Hardenberg, 1995] von Hardenberg, P., and Finn, G. A., *Use of Knowledge-Based Systems for Helicopter Design*, Proceedings of the 51st Annual Forum of the American Helicopter Society, Fort Worth, TX, May 1995.

[Yates, 1989] Yates, JoAnne, *Control Through Communication: The Rise of System in AmericanManagement*, The Johns Hopkins University Press, Baltimore, MD, 1989.