TECHNICAL MEMORANDUM Copy No. 226

COMPUTER-AIDED DESIGN: A STATEMENT OF OBJECTIVES

by

Douglas T. Ross

8436-TM-4

September, 1960

Contract No. AF-33(600)-40604

The work reported in this document has been made possible through the support extended to the Massachusetts Institute of Technology, Electronic Systems Laboratory, by the Manufacturing Methods Division, AMC Aeronautical Systems Center, United States Air Force, under Contract No. AF-33(600)-40604, M. I. T. Project No. 8436. Part of the work is being performed by the Mechanical Engineering Department, Design and Graphics Division, under M. I. T. Project No. 8477. The report is published for technical information only and does not represent recommendations or conclusions of the sponsoring agency.

Approved by: Douglas T. Ross

Douglas T. Ross, Project Engineer
Head, Computer Applications Group

Electronic Systems Laboratory
Department of Electrical Engineering
Massachusetts Institute of Technology
Cambridge 39, Massachusetts

0582119

ABSTRACT

The objective of the Computer-Aided Design Project is to evolve a man-machine system which will permit the human designer and the computer to work together on creative design problems. This document states the philosophy of approach being used by the computer applications group of the project. A companion document, 8436-TM-5, states the philosophy of the design and graphics group.

From the computer applications point of view the primary problem is not how to solve problems, but how to state them. It is proposed that outside-in problem statement, in which a problem is described first in general terms and then refined and made precise by further elaborative statements, is required, rather than the inside-out problem statement form which characterizes present computer programming. General problems are viewed as internally structured by means of interconnected "objets". An objet is an abstract entity of meaning, and the computer's "understanding" of a problem is represented by the structure connecting the objets of the problem. The human's understanding is in terms of a language which is isomorphic to the structure of objets. This language for problem statement will consist of pictorial as well as alphabetic representations, and can be molded to suit particular problem areas. The various project activities required to establish a proper research environment are also outlined.

PREFACE

The computer-aided design project at M. I. T. is a joint endeavor between the Computer Applications Group of the Electronic Systems Laboratory, Electrical Engineering Department, and the Design and Graphics Division of the Mechanical Engineering Department. Based upon their backgrounds in different disciplines, these groups are taking complementary approaches to the problem of how to use computers to assist humans in the design process. The long-term goal is automatic manufacture once the human-computer "design team" has established the features of a design. It is not contemplated that fully automatic design without human guidance and decision is a possibility for the foreseeable future. However, the possibility of having a computer be an active partner to the designer, to accept and analyze his sketches and perform all or a substantial amount of the necessary design calculations, does seem reasonable for the near future.

Based upon the first six months work, each group has made an initial formalization of its philosophy in approaching the problem, and these are published in two companion Electronic Systems Laboratory documents (of which this is one):

> "Computer-Aided Design: A Statement of Objectives," D. T. Ross, Technical Memorandum 8436-TM-4, September, 1960

> "Computer-Aided Design Related to the Engineering Design Process," S. A. Coons and R. W. Mann, Technical Memorandum 8436-TM-5, October, 1960

Details of the work conducted under these programs are contained in a series of Interim Reports beginning with 8436-IR-1, covering the period December 1, 1959 to May 30, 1960. Technical reports and memoranda are also issued on specific subjects where appropriate.

John E. Ward
December, 1960

TABLE OF CONTENTS

CHAPTER I

A PHILOSOPHY OF COMPUTER-AIDED DESIGN

A. INTRODUCTION

On December 1st, 1959 under sponsorship of the Manufacturing Methods Division, Air Materiel Command of the United States Air Force, the Computer Applications Group of the Electronic Systems Laboratory, M. I. T., in cooperation with a group from the Mechanical Engineering Department, M. I. T., officially began an investigation of the possibilities for computer-aided design. This memorandum is an attempt to set down, in more or less explicit form, the type of man-machine system which is the goal of the project, and to indicate some of the fundamental reasons for the establishment of a goal of such magnitude. Descriptions of progress and achievements will be found in Interim Reports issued every six months by the project and in Technical Memoranda and Reports published as major phases of the work are completed. Undoubtedly the philosophy which is espoused here will be subject to many modifications as the work proceeds, so that this memorandum should not be considered to be a definitive specification for the work to be done, but rather an indication of the approach which will be followed.

B. WHAT IS NOT COMPUTER-AIDED DESIGN

It is very difficult to define what is meant by computer-aided design since the complete definition is, in fact, the sum and substance of the total project effort which has only begun. It is much easier to describe, what is not computer-aided design as we mean it. So we begin by describing several ways in which computers have already been employed in design--work which we do not intend to duplicate or mimic. In the final analysis, after a complete definition of what we do in fact mean by computer-aided design has been shown and demonstrated through an actual working system of some sort, almost all of these ways of using computers in the design process, which we now disown, will then be incorporated and used in one form or another. These are all useful techniques and ones well deserving of further study, but all seem to be subsidiary to the more elaborate problems to which we wish to address our efforts. What, then, are these things that computer aided design is not?

Computer-aided design is not automatic design, although it must include many automatic design features. By automatic design we mean design procedures which are capable of being completely specified in a form which a computer can execute without human intervention. There is a growing number of examples of automatic design programs of this type for designing electrical motors, transformers, electronic circuit layouts, cut and fill procedures in civil engineering, and many others. In all these systems there is an essentially fixed design or class of designs which are already completely worked out. The problem which is solved by the computer program is that of varying the parameters of the generalized design to match the constraints imposed by a particular application. Programs of this type can have a very important economic advantage over manual solution of the same problem, and in some cases can be used to yield optimum designs rather than ones which are merely satisfactory or adequate. Automatic design procedures of this type, however, offer little opportunity for modifying the basic features of a design and in general are applicable only to very restricted design areas.

Computer-aided design also is not automatic programming, although automatic programming techniques must necessarily play an important role in computer-aided design. Automatic programming systems such as APT have the features of flexibility and adaptability in problem specification by the human which are found wanting in the automatic design area. However automatic programming systems usually represent only an aid in machine coding, and in general contain very little of a programming, problem-solving, or design nature. In an automatic programming system the human is still left with the burden of solution, which is why automatic programming cannot be considered computer-aided design.

Thus we see that automatic design has the computer do too much and the human do too little, whereas automatic programming has the human do too much and the computer do too little. Both techniques are important, but are not representative of what we wish to mean by computer-aided design.

Before launching into a description of what we think computer-aided design ought to mean, let us mention one other thing which it should not be. Computer aided design should not just be a mirror of present design practice, although it should of course be related to it. Present design practice has many forms, and the art of design is practiced differently in all fields and in all industries. Many features of present design practice are undoubtedly of immortal worth and are good whether or not the computer is employed. Others, however, can quite probably be replaced by more efficient, more natural, more expressive techniques which are made possible only through the combining of man and machine into a design system. Therefore, computer-aided design, whatever its ultimate meaning may be, is not the same as our present understanding of design as it is practiced today.

C. DESIGN VS. PROBLEM-SOLVING

Is design a special case of problem-solving or is problem-solving a special case of design? This is a moot question; for it is possible to argue that one only works out a design as a solution to a problem, or one can argue equally strongly that the problem-solving portion of the solution to a problem has all of the characteristics of creativity, esthetics, and adjustment of relevant factors which are normally associated with design. In this discussion we shall use the two terms almost interchangeably. In one instance the terminology of problem-solving may be more conducive to agreement, while in another, the terminology of design may yield a more lucid exposition. It is quite important to recognize that these terms are purposefully being confused. The appropriate quote from Webster's is "to plan mentally; to outline, to scheme; - distinguished from execute". In Webster's this definition is listed under design. It should be borne in mind that we wish to apply it also to problem-solving. As soon as the primary objective of a procedure ceases to be scheming, and changes to execution or carrying out of schemata, then the problem-solving or design terminology may no longer properly be applied.

At this point it does not seem advantageous to belabor further the definition of design and problem-solving as individual words. Any attempt to be more explicit would be ill suited to our purpose. Extended discussion of specifics would be patently specious, since our primary purpose here is not to profound proposed dogma, but rather to express a philosophy in a general but compelling way. Bearing these thoughts in mind-- the nebulous difference between scheming and executing schemata, and that the discussion is of necessity philosophical in nature-- we may now resume our pursuit of the elusive meaning of computer-aided design.

D. TO SOLVE OR TO STATE

We begin with a paradoxical twist. We have just finished pointing out the essential equivalence of design and problem-solving, which would seem to indicate that we were going to turn our attention to the solution of problems. Instead, however, we now declaim that our main objective is not to solve problems, but to state problems.

In actuality, this apparent exercise in mental gymnastics is well founded. At the grand level in which we want to consider problems, we don't really know what problem to solve in the first place. Take for example any cogent design problem such as designing an airplane. Admittedly there are many very difficult problems associated with the design of an airplane, many of which can at least partially be solved with the aid of a computer. But if all we say is "I want to design an airplane", there is no possible way of proceeding. Such a situation is reminiscent of certain instances of childhood in which a youngster will express vehemently the demand that some want be satisfied, without adequately describing the difficulty. Whether the designer is computer-aided or people-aided, he must certainly be more explicit than this.

This contention that the primary objective is not to solve problems but to state problems is further supported by the fact that we know many problems that we don't know how to solve. We also know many problems we can "solve" in many ways with or without computer programs. In a particular case we don't know which solution method to use. Furthermore we don't want to be restricted to solutions we already have even though they at first seem adequate. A new problem environment may stimulate better solutions. In all of these instances the basic inadequacy-- the first problem to be solved-- is to achieve an adequate statement of our problem.

E. OUTSIDE-IN PROBLEM STATEMENT

The manner of stating problems is also of primary importance. We must be able to state problems from outside-in, not inside-out. The normal practice of being explicit about a problem is to build a description of the solution procedure in small steps, i.e. to code the problem. This is particularly evident if a computer is being used, but is also true in activities among people when those activities are organized into a well-structured system. In other words, whether a "machine" to assist in solving problems is composed of mechanical and electronic hardware or of people with specialized training and talents, the method of employing this machine in the solution of the problem is essentially the same. The solution procedure must be built up step-by-step to achieve the desired result. The most complete statement of the problem is, in fact, its solution procedure.

Stating a problem step-by-step from the inside-out in this manner is satisfactory if the problem is well understood to begin with. But there is an alternate procedure which is frequently used among people, but is virtually nonexistent as regards computers. In this alternate procedure, which we call stating a problem from the outside-in, we state the problem originally in general terms and then explain further and in more and more detail what we mean. In the case of designing an airplane, for example, we might add a general statement such as "It must be a high-performance commercial aircraft". As the specification of the design proceeds the general terms "high-performance" and "commercial", and possibly even "aircraft", will be elaborated further.

It is quite apparent that this procedure of stating a problem from the outside-in by means of further and further refinements is the only feasible way to attack grand problems. The inside-out method of stating problems, which is normal practice with computers, cannot possibly be carried out until a problem is stated in sufficiently detailed form, and this statement itself can come only from an outside-in consideration of the problem. Normally this preliminary outside-in study of the problem must be carried out entirely by people. A principal aim of computer-aided design as we mean it is to permit the computer to play a part in this scheming portion of problem-solving by providing a mechanism for performing outside-in problem statement.

F. THE NECESSITY FOR IDEALIZATION

What is a problem? If we are to design a procedure for stating problems from the outside-in (or even from the inside-out) we should first have a firm idea of what a problem consists of, so that our efforts may have a solid foundation. What, then, really is a problem? A seemingly irrefutable argument may be made that if we really knew what a problem is, in a profound way, we would have no real problems. The argument returns again and again to the proposition with which we are already laboring--that the statement of a problem is of paramount importance. Therefore, we will not attempt to invent a rigorous definition of "problem", since any such definition which could be stated precisely would of necessity be so restrictive as to leave us playing games of little consequence.

If we are unwilling to define precisely what we mean by problem, it might seem that a reasonable substitute would be to be specific about the kinds of problems we wish to consider. This, too, is not an appropriate path to follow. Initially we don't care about detailed solutions to problems, nor even tightly circumscribed problem areas. We can only get to setting up explicit sub-problems at the very end of the problem statement sequence. As far as the outside-in problem statement sequence is concerned, an abstract or idealized problem is treated in the same way as a detailed practical problem, and in fact is more pertinent to the study. We only arrive at explicit problems after a long series of hazy, incomplete, ambiguous refinements of the original goal statement, which taken together acquire precision, so that our main emphasis must be on the consideration of problems which are in a sense abstractions or idealizations of practical problems.

Neither of the terms "abstraction" or "idealization" really conveys the concept which is important for our study. We do not wish pertinaciously to insist upon the abstract or the ideal due to any whimsical misconception of our role, or of the magnitude or importance of our task. The plain fact of the matter is that problems being what they are, and the outside-in form of problem statement having inherently the characteristics it possesses, we have no alternative but to concentrate our attention on things which are poorly, incompletely, and for the most part actually incorrectly defined, if they are considered relative to a "complete and definitive" problem statement. In the sense that these things do not accurately mirror the real world of our problem, they are abstract or ideal, but it must clearly be recognized that these traits are not arrived at by choice, but are forced upon us by circumstances. The very reality of our problems forces them to be idealized. We will later see how concomitant idealizations can yield solutions to very real, practical, and difficult problems.

G. REPRISE

We now appear to have worked ourselves into a position which is even worse than that from which we started. Originally we were concerned with a description of what we meant by computer-aided design, a topic which admittedly needed some discussion, but one which appeared to be reasonably concrete and tangible. Now, however, by insisting on outside-in problem statement and by being painfully honest about how little we know about problems in general, we find ourselves concerned about things which are not really abstract, not really ideal, but are nebulous and incomplete, ambiguous and ill-defined.

This very lack of form, however, this very chaos, which seems to limit and almost preclude any chance of success is, in fact, our principle strength. By discarding all pretense concerning our ability to force tractability on basically intractable problems, we must consider the world of problems in its barest essentials, in terms of what it really is. We admit freely that we cannot define succinctly what a problem is, but we admit to no such incapacity with regard to the elements out of which problems are composed.

H. REPRESENTING THE STRUCTURE OF PROBLEMS

Our goal at this point in the discussion is to devise a scheme for representing the elements out of which problems are composed. All substantive problems are internally structured. We recognize that they are made up of sub-problems, which in turn have internal structures of their own. The sequence of sub-structures is terminated finally in some elemental quantities which are intimately related to the particular aspect of "reality" with which the overall problem is concerned. We wish to devise a scheme, (a mathematical model, if you will), which we will consider in terms of a computer structure, for manipulating problem elements. Problem elements may be of arbitrary form, and our primary objective is to have a computer structure capable of expressing relationships between general objects in a natural, non-artificial way.

The initial object for any particular problem will be the general goal statement. Successive objects will be the constituents of subsequent refinement or explanation statements concerning the problem. Since the problem elements with which we must deal are so varied, we prefer not to attempt to convey what we mean by attaching very specialized meanings to existing words such as element and object, and instead we borrow the French word objet, meaning thing or object, and assign specialized meaning to this distinctive but mnemonic word.

Objets are merely "things" with properties we will define. Some properties are measurable and quantitative, some properties are qualitative, some properties are expressions of relationships with other objets. Objets may range in meaning from very concrete, physical things, to the utmost in abstraction. Objets may, themselves, be properties of other objets and conversely any specific property of an objet may itself be considered an objet.

In general an objet will be some entity of meaning, the meaning being made implicitly explicit through the list of properties which relate the objet to measuring scales or other objets. Thus an objet is literally anything you wish it to be. There is no standard or stock procedure for defining the properties of an objet. One merely describes how an objet is related to already established things such as units of measure, colors, qualities such as big and small, good or bad, or in terms of other objets, whether they have previously been defined or will be defined subsequently. The total combination of statements about an objet constitute the definition of the objet. The computer structure we seek must be capable of representing all of these complex interrelationships in an efficient and workable fashion.

I. A FLEXIBLE LANGUAGE STRUCTURE

Equally important as a primary objective is a natural language for expressing statements concerning objets. The way in which the computer structure will be set up and adapted to fit a particular problem will be the result of a sequence of statements made by the human concerning the problem.

As a matter of fact the language will represent the computer structure to the human and will be parallel, or in a sense isomorphic, with this structure. Thus each objet has a corresponding word, phrase or other means of referencing the objet from the language, and this word or phrase obtains its definition in turn from the property list of the objet, i.e. the word and the objet are equivalent representations of the same "thing". The human understands the thing in terms of the language representation; the computer understands the same thing in terms of the property lists stored in the computer structure.

The language, itself, and the rules for its meaning and translation will be structured and defined by means of objets. In other words, the language is self-defining. It can express general relationships about general things, and in particular it can express its own rules and structure. The human's understanding of the meaning of the language is mirrored by a corresponding understanding on the part of the computer in the form of interrelated property lists among the objets of the language stored in the computer structure. Just as new objets are introduced as the various aspects of a problem are unfolded, so also the objets of the language itself may be modified and altered through use. As long as the human and the computer maintain at all times a mutual understanding of the meanings of everything, nothing is lost and a great deal is gained by permitting the basic definitions of the language itself to change with the problem.

The combination of a language of this type with outside-in problem statement is equivalent to a language which can be continuously modified and molded to suit a particular problem. Successive refinement statements elaborate and particularize meanings of words already used, by amplifying the interrelationships between technical words representing the objets of the problem and the standard words represented by measuring systems. As words change meaning, translation procedures also change, since a translation procedure for understanding one meaning of a word may no longer be valid when a new meaning for that word is obtained by additional modifying relationships with other words.

J. SUMMARY OF RESULTS

We are now in a position where we can evaluate our progress. By following the rather tortuous path from computer-aided design, to problem-solving, and then to problem statement not from the inside-out but from the outside-in, followed by a frank recognition of how little we know about problems in general, we have finally ended up at a position where we can see that "objets" combined with an appropriate "language" can form the basis for a workable and powerful philosophy of computer-aided design. To be sure we have not yet been specific about the computer structure for representing and manipulating objets, nor have we specified a vocabulary or syntax of the language, but these are the tasks that must be attacked by the project in the coming months. A significant amount of progress in this direction has already been made, and has lead directly to the concepts and philosophy which are being presented here.

The principle reason that the philosophy of a fluid language which can be molded to fit individual problems can be anticipated to form a significant contribution to computer-aided design is that so little depends upon the problem itself. We have specifically avoided the pitfall of attempting to match the language and features of the system to any particular class of problems, since the concept of a general problem is so incompletely understood, and instead have concentrated upon pure structure. The objets and the words corresponding to them may be completely arbitrary; the only things we will explicitly be concerned with are the structural relationships among objets. Therefore, if a person can begin talking about or describing in a general way the problem with which he is concerned, he can begin on a solution of that problem.

In fact, the statement of a problem is indistinguishable from its solution, in the sense that once the problem has been completely stated, the execution of resulting schemata can be treated routinely. This fact was alluded to earlier when it was said that the clearest statement of a problem is its solution procedure. For a problem which is truly new, i.e., the type of problem in which we are interested, new words are introduced by means of hazy or incomplete definitions. Possibly we specify only certain parts of a definition realizing that there are many other parts left unsaid. As the meanings of words are elaborated, the corresponding changes are made among the objets of the problem until finally all of the interrelationships necessary to convey our meaning have been specified. At this point the problem may actually be "solved" by executing appropriate automatic calculation routines. On the basis of results thus obtained, we may continue with further elaborations.

The difference between what we are here proposing and the normal relationship between humans and computers is that we propose to work toward the solution from the outside-in, not from solution to the statement from the inside-out. In ordinary programming the human must solve the problem alone through a sequence of outside-in considerations, and if the program written as a result of this does not end up equivalent to the problem statement he had in mind, he must reprogram the problem. Under the new philosophy, successive stages of problem statements are greater and greater refinements of the original statement of the problem. Each stage is represented in and materially assisted by the language and computer structure. The end result is a sufficiently refined solution to the original goal, achieved by a sequence of elaborations, modifications, tests and evaluations, all of which taken together constitute the evolution of an ever-clearer idea of just what the problem is that is to be solved. Since the human need not be aware of any of the detailed structure of the problem to begin with, and need not analyze separately and program the problem beforehand--since the computer is able to work in partnership with the human at all levels of consideration on the problem--the process is truly computer aided. Whether it is called computer-aided design, computer-aided problem-solving, or if some other term is used, the successful completion of even a rudimentary system based upon this philosophy, will represent a significant advance in utilization of the combined talents of men and machines.

CHAPTER II

AN OUTLINE OF SYSTEM FEATURES

The previous chapter has described a broad philosophy for combining the talents of a human and a computer into a man-machine system for the effective solution of complex problems. Such a philosophical discussion is necessarily concerned with intangibles. The type of man-machine system which we propose should actually be quite tangible, however, so that in this chapter we attempt to describe some of its features in more concrete terms, and show how they are related to those of present day automatic programming techniques. Once again, these remarks should be interpreted only as an attempt to indicate the type of system being proposed. The actual formulation of the system and the precise features which it will have will be the result of further project activity.

A. THE PROBLEM-STATEMENT LANGUAGE

Initially, the system will look little different from any other modern automatic programming or compiling system. A basic set of routines will permit coding directly in machine language with free use of symbolic names for quantities and locations. A macro instruction capability will also be present so that arbitrary collections of statements in the language can be used to define new functions (possibly recursive) which can be used in the same way as basic machine instructions. The system will also permit ordinary algebraic notation to be employed as in Fortran and other automatic coding systems. In general all of the widely used techniques of modern programming will be incorporated into the system.

The principle difference between the new system and others will be internal, and the effects of this internal difference will not show externally until the system is farily well developed. The entire system will be organized on the basis of objets with property lists, and therefore, all of the features described in the previous chapter will be available for modifying and expanding the basic language itself. By using the features such as algebraic notation and macro instructions as described in the previous paragraph,which will be provided by the basic objets, it will be possible to define new objets which, when taken in conjunction with themselves and the basic objets, will permit essentially arbitrary expansion of the language.

In particular it will not be necessary to define an objet completely before it is used within a statement. This feature is already widely available with respect to symbolic addresses, in almost all programming systems, but in the new system the same feature will apply to operators, functions, and entire programs. If the computer is called upon to actually perform an undefined function, then it will stop and ask for a suitable definition before proceeding. Depending on circumstances this definition may or may not be the final definition of the function. It will be perfectly possible to put in a quick approximation for testing purposes, and replace the definition later by one that is more complete.

In human languages most words have many possible meanings all of which are listed in a dictionary, and the appropriate meaning is selected by considering the context in which the word is used. In the system it will also be possible to assign multiple meanings to individual words. For example the word ADD may be used to add fixed point numbers, floating point numbers, vectors or matrices, since the context within which ADD is used will automatically determine the appropriate corresponding computer program.

There are many other features which are planned and could be mentioned at this time. Each one taken by itself seems inconsequential and many of them already have exact or at least very similar counterparts in other systems. The difference in internal organization of the new system, however, will combine all of these small effects into an overall performance capability which should be a significant improvement over present day systems.

B. THE LANGUAGE OF CONTROL AND EXPERIMENTATION

Since the new system is considered as a man-machine system and not just an automatic programming system, all of its features will be capable of being executed on-line from typewriter or pushbutton language control as well as automatically from punched card or magnetic tape control. This philosophy has important consequences in terms of the point of application of statements made in the language. The same language which is used for stating and solving problems may be used for controlling the operation of the system itself. Thus, the full range of algebraic notation, machine coding and arbitrary macro instructions as are used for creation of an object program may be applied also to the setting up of individual arguments for functions or operations. Thus, instead of laboriously figuring out the precise symbolic or numerical address of a quantity which is to be used as the argument of a function, it will be possible to merely describe a procedure by which the system itself can determine the appropriate location of the argument. This feature, which is related to that of nested definitions, is used in several other systems, but frequently there are restrictions placed on the language when it is used within the definition of an argument, since simplified special - purpose techniques are used.

An important aspect of problems of the degree of difficulty in which we are interested, is that it is extremely unlikely that the user knows all about his problem even at a very general level. He must learn about the problem as he attempts to state and solve it. Much of this learning comes as a result of testing hypotheses. In order to assist this process materially the entire sweep of the scientific method must be built into the system, in a sense. The user must be able to establish a controlled environment, set up an experiment, try test cases, analyze results and modify whatever is appropriate, all by simulation on the computer. All of these various facets of the actual solution of the problem must be carried out efficiently and conveniently by statements back and forth in the language, in a conversation or discussion about the problem, between the man and the computer. Therefore the system must be an operator system. Here again the operator features must be capable of either automatic or manual control without the use of restrictive or unnatural control techniques. In other words, the problem of setting up, executing and analyzing results must be handled with the same facility as is available for programming of the object problem itself.

C. SUMMARY

A number of features of the new man-machine system have been described. There undoubtedly are other aspects of the system which could be seen fairly clearly in outline at this time in addition to those already covered, but sufficient "feel" for the operation of the system has been given already. The primary cohesion among all of the things described comes back to the basic philosophy of the previous chapter: Everything, be it stating an algebraic formula, setting up a table of data, controlling the operation of a program, translating language in written or picture form--everything is a problem of some sort. Every problem is structured and may be represented by collection of interrelated objets. Therefore, once again, if the structure and the meaning of general abstract objets can be handled in an elegant fashion, all else will follow in due course.

Starting out with the basic language and way of handling objets, the entire system will grow and expand with use to encompass problems of greater and greater scope. For each problem or problem area, appropriate new terms can be introduced and defined, and later redefined and employed in new contexts. Each new problem will make use of some of the language capabilities already in the system or left over from other problem areas, and from these the language will branch out to establish appropriate terminologies and operations so that the new problems can also be stated and solved rapidly and efficiently.

Special note should be taken of the following disclaimer: We do not claim that the original formulation of objets and language will itself have the power of solving essentially arbitrary problems directly. But we do claim, and claim confidently, that the original structure will start from a firm foundation and lead in the right direction so that significant advances towards these goals can be made.

CHAPTER III

A COORDINATED PROGRAM OF RESEARCH

A philosophy in a vacuum is of little value. In order to be effective a philosophy must not only indicate the establishment of appropriate goals, but must also materially assist in the achievement of these goals. We, therefore, close with a few remarks showing how the espoused philosophy provides concrete guidance in the selection of appropriate project activities.

A. THE EFFICACY OF A COSMOPOLITAN APPROACH

It might at first appear that direct application of the philosophy to the selection of appropriate project activity would dictate that the project should concentrate exclusively upon the design and construction of the compiler and programming system so essential to its realization. At some appropriate future date, after the system was more or less fully developed and functioning, it could be applied to various problems of design, and computer-aided design would then be a reality. Although such an approach might lead to a functioning system of some sort fairly quickly, the resulting system in all probability would not be representative of what computer-aided design ought to be. Such a premature advance to a supposedly finished system would ignore completely the fact that despite the high-sounding phrases of the overall philosophy, there is much more still to be discovered about computer-aided design than we already know at this time. It would indeed be possible to commence immediately on an all out effort to construct a complete system along the lines which we presently envision, but a much more sensible approach is to consider the actual construction of the computer system as only a part of an overall program of research aimed at a more complete understanding of computer-aided design.

Perhaps the best illustration of the validity of the contention that a system should be developed in conjunction with other project activities is that the philosophy as it stands today is in fact the end product of just such a situation. Both the broader aspects of the philosophy, as well as the detailed computer programming considerations which are now actively being pursued, are the result of innumerable influences from past activities of the Computer Applications Group. Ideas, concepts, and new approaches cannot be generated spontaneously. They must result from the interplay of research goals with actual concrete problems of significance. By working with problems which are so complex and difficult that they cannot be solved acceptably with existing techniques, the generation of new approaches, or more frequently the modification of old approaches and then re-application with a slight twist, is fostered.

B. THE IMPORTANCE OF PROBLEM ENVIRONMENT

There is little challenge and virtually no prospect of significant achievement in solving problems which are known immediately to lie well within the scope of existing techniques, and the same also applies to solving pseudo problems which are merely dreamed up to lend support to some otherwise untenable hypothesis.

Most of the problems which have been attacked by the Computer Applications Group have been of the type which fosters sound research. They have been difficult problems from the real world sufficiently beyond the capabilities of existing techniques to cry out for new techniques, and yet capable of being formulated in such a way that bridging from the old to the new was possible of achievement. The net effect of these past experiences has led almost directly to the present outline of a philosophy for computer-aided design. Our new task is to elaborate on this philosophy by maintaining a fertile problem environment in which to work and think.

What are the attributes which a suitable problem environment must exhibit? The problem environment must be varied but cohesive; it must be sufficiently difficult to be challenging, and yet not so difficult as to be unrealistic; it must be sufficiently detailed to stimulate concrete advancement, but broad enough to avoid fruitless endeavors to resolve parochial oddities. Above all the problem environment must be dynamic and vital; capable of being molded to suit various purposes; pliable enough to be compressed for sweeping generalized attacks, and yet able to be stretched taut for penetrating attention to detail without coming apart. Design and problem-solving are areas of sweeping scope, so that care must be exercised in selecting from them those aspects which will establish the proper problem environment conducive to worthwhile research.

C. AN OUTLINE OF PROJECT ACTIVITIES

The mandate of the project is to carry out research leading toward the formulation of a procedure for computer-aided design, with emphasis on mechanical design. It is hoped that the result of this effort can be blended with the APT System for automatic programming of numerically controlled machine tools to provide an efficient mechanism for going almost directly from the requirement for a machined part to the finished product. Although investigations will concentrate on design problems in this rather restricted area, if other types of design appear more appropriate for the generation of ideas for handling certain types of problems, they will not be excluded.

On the basis of this mandate and the remarks of the preceding sections, project activity may be broken into five general areas.

1. Computer programming techniques
2. Mathematical and analytical techniques
3. Man-machine communication devices and language design
4. Analysis of the design process itself
5. Extension and expansion of the APT System.

We will now elaborate briefly on each of these general areas.

Under the heading of computer programming techniques would come the basic work of designing and programming the computer system itself. This topic has received sufficient discussion previously to require no further elaboration here. Also under this heading, however, would come investigations into computer application techniques which are peculiar to artificial intelligence and symbol manipulation programming. The problem of computer-aided design, at least when it is approached on the grand scale which is contemplated here, requires the elaboration and extension of these rather esoteric techniques to achieve satisfactory solutions. Many of the features of the compiler and computer system will be strongly influenced by considerations from these areas.

Investigations of mathematical and analytical techniques will be concerned primarily with procedures for solving some of the extremely complex problems of mechanical design in ways which are appropriate for computer solution. Special emphasis must be placed upon efficiency of computation and extensive use of approximations, since we are considering using these computational techniques in an essentially real-time application. Many of the computer programs which are now in use for stress analysis and other problems of design are far too cumbersome and expensive to operate to be considered for use on the incomplete or faulty designs which will be an inherent feature of a design system. It therefore seems probable that we will concentrate on techniques for getting good approximate results efficiently, and then incorporate more elaborate and exact techniques to be applied as a check to a final analysis of the resultant design. There is a remote possibility that an approach to problems of this type can be found which can be applied either coarsely at high speed or finely at low speed, so that the necessity for alternate techniques would not be required. Also under the heading of mathematical an analytical techniques will come more general problems of the efficient handling of computations of functions of many variables of differing dimensions, as well as development of generalized techniques for speeding the convergence of iterative solutions, etc.

Since the computer-aided design system which is our goal is a man-machine system, the problem of obtaining a good transfer of ideas and information between the human and the computer is of paramount importance. In addition to typewriter and pushbutton languages, the system will employ picture languages, for both input and output, as well as physical manipulations of pictures. Initial developments will concentrate on output oscilloscope and light pen input techniques because of their availability and versatility, but later investigations may branch out to consider other devices as appropriate. A great deal remains to be learned concerning the language form represented by picture devices and the correspondence of this language form with other language forms. All of these language considerations will of course have an important interplay with the developments of the computer programming system itself.

Although it is anticipated that development of a system for computer-aided design will introduce many techniques and approaches to design problems, it is nonetheless likely that the majority of the techniques used in the application of the system will be very closely related to, if not identical with, those that are used now in manual design. For this reason the procedures used by mechanical designers, and problem-solvers in general, will be investigated to determine how the burden can be shared between man and machine, and also to act as a check that chosen techniques for computer-aided design are indeed suitable for actual design problems. Also, under this heading will come studies of the organization of design information as it is presently available. Many parts of design are determined not so much by mathematical and analytical considerations as the availability of standard items or the results of empirically determined test results contained in a designer's handbook. Consideration of these various forms in which information about design exists may very well influence in a fundamental way the organization of the overall computer-aided design system.

The extension and expansion of the APT System for the automatic programming of numerically controlled machine tools may at first appear to be a peripheral item which is considered in the overall project activities purely for historical reasons. The actual purpose of continuing these studies, however, is that the problems involved in expanding the APT System are ideally suited to the stimulation of techniques of all types which are likely to be of use in computer-aided design. The problems of the APT System are a good instance of difficult problems which lie just beyond the reach of existing techniques, and yet which are well enough understood to yield many hints and suggestions for improving the state of the art. Already there are indications that many of the computational techniques which have been pioneered in the APT System will be very useful in design work, and there is every reason to anticipate that this valuable cross-fertilization will continue. An additional important reason for continuing work on the APT System is that it provides an ideal outlet for results of the project. It is expected that many of the techniques which are worked out on the basis of computer-aided design will not be of practical use in design for several years at least, since to apply them to actual design problems may require as yet uncompleted portions of the system. These same techniques, however, may have immediate application in improving the performance of the APT System. Therefore the ideas can be put to work and given extensive testing in the APT framework, even though in terms of design they may have been completed prematurely.

D. A PILOT STUDY AS FOCAL POINT

A final project activity which is planned and which has not been mentioned previously is the carrying out of a simple pilot study of a computer-aided design system. The precise area of application for this pilot study has not yet been decided upon, but it is hoped that a class of problem can be selected which will bring into focus all of the various aspects of the project in a simple way, so that experience can be gained with the different parts working together in a system. It is not intended that this pilot study be in any definitive way representative of what we will ultimately mean by computer-aided design, but in addition to providing a vital connecting link between the various project activities as they proceed, it will also serve a useful function in exposing our nascent ideas to public criticism.

The complexion of the project will undoubtedly change as the work proceeds, but it is hoped that this brief summary of planned project activities indicates how the broad philosophy of computer-aided design, which will continue to develop as physical realizations of its various aspects are completed, serves as a goal toward which the project can work. It is hoped that initial efforts in all these areas can begin to exert influences on each other at an early date, so that the evolution of both the philosophy and the actual man-machine system can proceed in an orderly fashion toward the desired result.