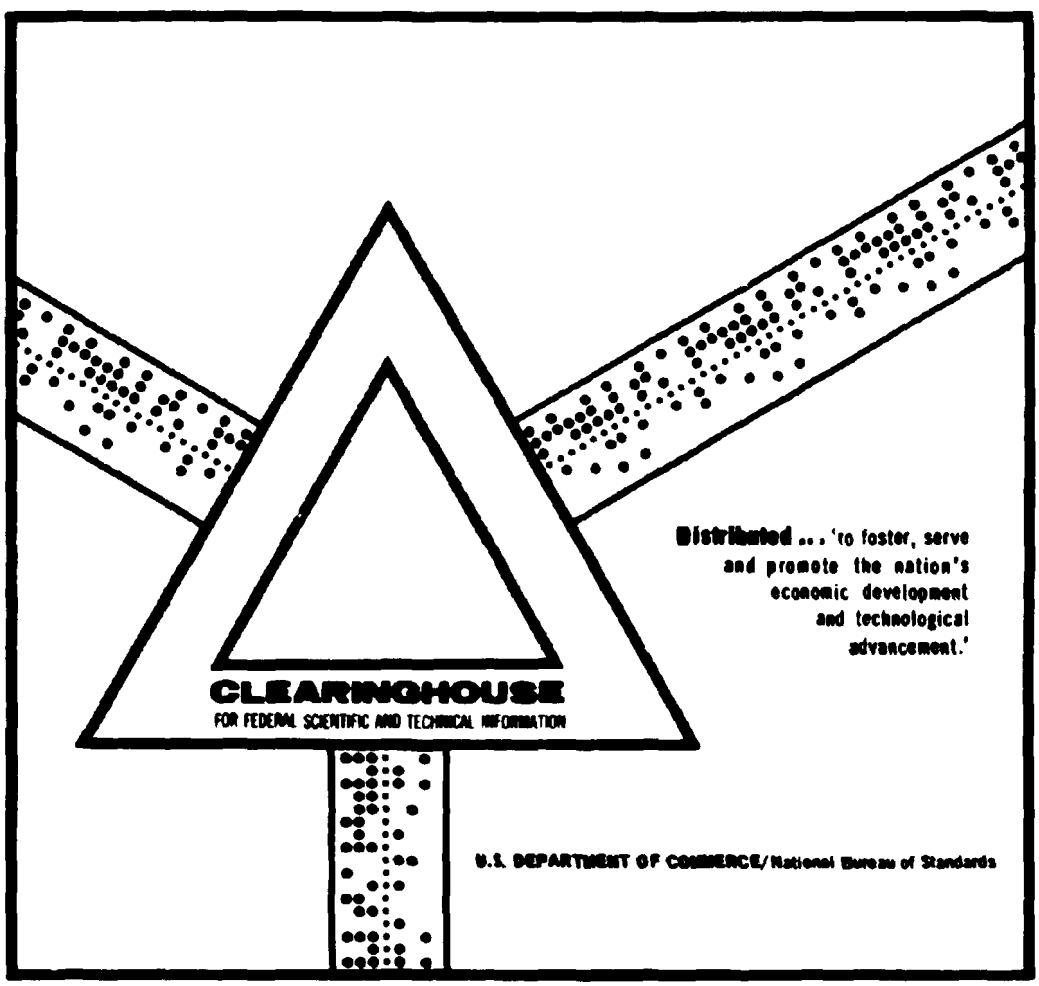
DESIGN - THEN AND NOW

George E. Forsythe

Stanford University
Stanford, California

September 1969



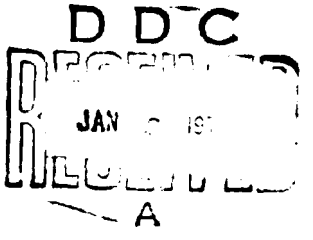
CS 140

DESIGN - THEN AND NOW

BY

GEORGE E. FORSYTHE

TECHNICAL REPORT NO. CS 140
SEPTEMBER 1969



COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

17

DESIGN -- THEN AND NOW

George E. Forsythe*
Computer Science Department
Stanford University
Stanford, Calif. 94305

Author's apology

The author must confess that he has never designed anything more than short computer algorithms. Perhaps this gives him an unbiased vantage point from which to open this conference.

What is design?

"Design," in the author's opinion, is the application of all knowledge to the specification of some structure that best, or nearly best, satisfies a given criterion. The structure may be material (e. g., a bridge), social (e. g., a traffic control system), or intellectual (e. g., a data structure in a computer). Moreover, it may itself be a tool for the design of something else.

There are three distinguishable parts of design: analysis, synthesis, and optimization. Assume the desired structure has been parametrized so it is specified by a certain vector-valued "input" variable, and its performance is measured by another

---

*The preparation of this paper was sponsored at Stanford University by the Office of Naval Research and the National Science Foundation.

---

vector-valued "output" variable. Then the total process of design is to determine a value of the input variable that brings the output variable close to the given target value, while satisfying any constraints that are given with the problem. The process of analysis starts from given values of the input variable, checks whether the constraints are satisfied, and determines the resulting value of the output variable. The process of synthesis starts from the desired value of the output variable, and determines appropriate values of the input variable. The process of optimization finds a value of the input variable that minimizes the distance (in a given metric) between the corresponding output variable and its desired value. In those rare cases where perfect synthesis is possible, it becomes the core of the design process. More commonly, the process of design proceeds by iterative steps--the analysis of the performance resulting from a given input variable, the synthesis of changes to the input variable that will decrease the distance of the output variable from its target value, and repeat. Thus design is most often a process of iterated analysis.

In a broader sense, the process of design includes the original presentation of the problem, the method of parametrization, methods of analysis, synthesis, and optimization, and the presentation of the resulting design to the recipient. It is very clear from papers to be presented to this conference that all these broader aspects of design are acquiring increasing importance.

DESIGN THEN

"Then" means when the author was a young mathematician

working at the Boeing Aircraft Company. To me that is just a few years ago, but to you it may be part of U. S. history (I hope it's modern U. S. history). Let's call it 1946, to be definite.

What was being designed then?

"Then" was a period when mechanical engineering was still fashionable, and electric power machinery was still taught in every engineering school. But electronic engineers were in the ascendancy. The kinds of things being designed were bridges, highways, buildings, of course, and generators and power distribution networks. Also jet engines, airplane frames and even pilotless aircraft. But the biggest action was on the electronic front, where being designed were radars, guidance systems, filters, counters, analog computers, and so on.

The main point is that the kinds of things being designed were material, physical things out of concrete or vacuum tubes or metal.

What tools were used for design then?

Not being an engineer, I can hardly speak with authority on what the real design tools were. There seemed to be a lot of handbooks that told engineers about the properties of existing materials. In much of engineering the deepest mathematical tool seemed to be calculus, or a little bit about differential equations. When calculations were needed there was that universal computing machine, the slide rule, of which there must have been thousands in any large company.

The most potent calculational tool was the desk calculator. Manned by young women, mostly, there were large batteries of desk

•calcu ators. They were used for a variety of computations, from inverting matrices to evaluating rational functions of a complex variable z for many values of z, or perhaps integrating the ordinary differential equation of aircraft motion. Any one who recalls the drudgery of just dividing two complex numbers would say that asking some one to man a desk calculator was immoral exploitation of human beings. Yet many of the girls liked it, and resented losing their work to machines in later years. It required a lot of creativity, to weed out the inevitable numerous blunders of calculation before they could spoil a lot of subsequent work.

A principal tool of the electronics engineer was the Nyquist diagram for determining the stability of a linear feedback circuit with lumped parameters. About all that was happening was that the zeros of certain rational functions of modest degree were being localized, although somehow a little more was being learned. Nowdays a few seconds of computer print-out of a root locus plot would probably replace a week's work "then."

These tools for the design of feedback circuits were beginning to be applied also to the design of pilotless aircraft, and this is where the electronics engineer reigned supreme. The mechanical engineers normally couldn't figure it out.

Other design tools were the wind tunnel, the analog computer (locally designed, of course), and a lot of pencil-and-paper algebra. To me, a young mathematician, there was unending amazement in seeing an algebraic formula that was 20 pages long. I very much doubted that all terms were correct, and I was probably right.

A few ambitious engineers had learned something of the

methods of Southwell for solving two-dimensional field problems by pencil-and-paper "relaxation," an informal iterative process for solving finite-difference equations. Some specialized engineers were designing motors by a certain technique for graphical solution of Laplace's equation.

In 1946 and 1947 at Boeing's Physical Research Unit we were reading a lot of theodolite pictures and discovering missile velocities and accelerations by numerical differentiation and smoothing. The process was very tedious. A look at W. J. Eckert's PUNCHED CARD METHODS IN SCIENTIFIC COMPUTATION convinced me that it would be possible to use a punch-card tabulating machine to apply simple linear formulas to a long table of data. So we spoke to the tab man who wired the boards for these precursors of the IBM tabulating machine, type 402. He was very willing and helpful. Our main problem was convincing him that a negative number multiplied by a negative number should be negative. He thought we were crazy. In turn, he had his problems representing negative numbers on the board. As well as I can remember, he associated a digit 0 with positive numbers, and a digit 5 with negative numbers. Then he added those digits, ignoring carries, to determine the sign of a product. So the answers came faster, and with much less wear and tear to our desk calculator group, who could then spend more of their time reading those fascinating theodolite pictures. This needed their full attention, because our theodolites had so much "slop" in them that the reading differed significantly, depending on which direction the instrument was turning.

Nowdays, when I read that Boeing has over $100,000,000 worth of digital computers, it's hard to remember that I introduced

automatic computing to the company only 22 years ago.

Though I was elsewhere by then, it was only two or three years before the card-programmed calculator (CPC) arrived on the computing scene. This consisted of some electronic tube multiplying units, some storage units called ice boxes (because they looked like little refrigerators), and a permanently wired program board. The program itself was on punched cards, so it could be altered without rewiring the board. This was a great advance. However, it was pretty likely that the CPC would make an arithmetic error (or more) in any computation lasting over five minutes. Most people didn't notice this, but I was frequently carrying out computations for which I knew what ought to happen. The fact that the CPC was generally wrong when I knew the answer made me wonder what it was like for some one who didn't know what to expect. It seemed a marvel that planes nevertheless did (usually) fly.

At any rate, we learned to get correct answers from machines that usually made errors, programmed by people who usually did also. It was good practice in debugging.

One day I learned something new about how design was really done in the exalted circles of the high-ranking staff engineers. The Boeing ground-to-air pilotless aircraft (GAPA) was in the early stages of testing. On test flights it usually blew up, scattering parts among the men manning the theodolites and radar range finders. When the films came back to Seattle, our group spent days reading data off the films and computing velocities and so on. One time the situation was desperate and clearly called for a big change. The staff engineer took out some paper and designed a new shape for the tail fins with two or three

snips of his scissors. I was much impressed with this direct approach to design.

DESIGN NOW

What is being designed now?

To get a quick feeling for the kinds of things designed now one has only to read the program of this meeting. Whereas "then" the only things designed were material or physical things, one now sees designs of material things, social things, and intellectual things. Any of these can, in turn, be a tool for the design of other things. Moreover, whereas in 1946 the things designed tended to be at the component level, today we see more and more designs of total systems.

Despite the great diversity of things designed today, the invited talks at this conference seem to deal mainly with the design of physical things, and with tools for designing them. Thus, among the things discussed are the design of aerospace vehicles, optical elements, networks, elastic-plastic structures, nonlinear circuits, electronic circuits, and logic circuits--all of which seem to be in the realm of technology. The invited talks on speech processing and fault detection are in the intellectual realm.

The tools presented in the invited talks have a wider application. Those on differential equations and on free form surfaces seem directed only to physical objects. But those on nonlinear programming, matrix problems, and machine-aided mathematics deal with tools as well suited to the design of social things as to physical things. And the talk on algebraic manipulation is surely directed to a tool for intellectual

matters. Finally, the graphical systems discussed in two invited talks are useful for the design of physical, social, and intellectual things.

To get a feeling for the wider class of things being designed today, one should look at the papers contributed to this conference, or look under "design" in the permuted (keyword) index of COMPUTING REVIEWS. (I would like to have done the same for MATHEMATICAL REVIEWS but, unfortunately, there is no permuted index for that journal.) Among the material things being designed we find pipe systems, circuit boards, air cushion vehicles, antennas, chemical plants, aircraft frames, printing heads, and so on. Among the social things being designed we see cities, air traffic control systems, time-sharing systems, universities, communication nets, employment services, a census, school systems, etc.

Here are some of the intellectual things being designed today: time-tables, robots, mathematical assistants, operating systems, algorithms for solving X (for all manner of X), file systems, arithmetic units, abstract data structures, management procedures for a competitive firm, computers, stock portfolio management procedures, libraries, sequential testing procedures, methods for making perspective drawings, and so on.

What tools are now being used in design?

The most obvious new design tool to appear in recent years is the automatic digital computer. Its impact is all-pervasive. Its breadth of application (but not its depth) already far exceeds that of mathematics, and the field of computer science is only in its infancy. Few of the papers presented to this

conference fail to mention a computer program as a major tool. What are the special computer tools in wide use?

First are the various general-purpose languages for the expression of algorithms: Cobol, PL/1, Algol, Fortran, etc. Then we have special languages for symbolic and algebraic manipulation: Lisp, Reduce, Formac, etc. Then there are special languages for simulation: GPSS, Simscript, etc. There are special problem-oriented packages for applications areas, like ICES in Civil Engineering, and various graphics languages. There are special systems for machine-aided mathematics, with which a mathematician is served more comfortably than with general-purpose computermen's languages. There are packages for the automatic design of computers.

There are a variety of processors associated with these many languages, to translate them into machine code, debug them, associate them with higher-level systems, and so on.

There are large and small time-sharing systems, in which any one can interact actively with a computer, one small statement at a time. There are vast operating systems, capable of scheduling the operations of huge hardware/software systems.

Then there are collections of algorithms for computers, available offline in printed form, or online either in source language or as precompiled packages that need only be mentioned within a programming system. There are vast data banks of information for private or public use associated with some systems. There are cathode-ray-tube systems for display of information, together with light pens for entering information directly. There are a a number of technical tools for the design of other languages, including metalanguages, precedence grammars,

decision tables, list processing techniques, and so on.

In summary, the whole range of computer science and computer engineering is available to the designer, to use for either sophisticated or brute force approaches to design. It is making a large impact. Moreover, the field is changing so rapidly that it is no exaggeration to say that here is where most of the action is.

The other principal basic tool of design is mathematics, and particularly those branches of mathematics called numerical analysis and opeations research. Here the action is much slower than in computer technology, but over a twenty-year period the total change is very considerable. Let us examine these tools. For the design of large social systems perhaps a main tool is linear and nonlinear mathematical programming--a subject that did not exist in 1946. Other subjects of operations research are equally new: queuing theory, optimization methods, and so on. Indeed, optimization, which we noted to be a major step in design, has been a subject of intensive research by operations analysts for the past two decades.

The subject of numerical analysis has been known (but under other names) for centuries, and indeed the principal problems now investigated by numerical analysts were known to Runge and researched by him at the beginning of this century. However, it is important to realize that very few of the actual algorithms or methods in current use are older than ten or fifteen years. Reliable finders of polynomial zeros are very recent. The QR method of finding eigenvalues of real matrices was invented less than 10 years ago. The same is true of Romberg integration. Methods of integrating stiff ordinary differential equations are

very new. Splines for the approximation and smoothing of functions have an early history in draftsmen's tools and a paper of Schoenberg (1945) but in fact their use with computers is limited to the 1960's. Computer methods for solving field problems for partial differential equations have been developed in the 1950's and 1960's. The solution of systems of nonlinear algebraic equations has been researched mainly in the 1960's.

We are beginning to witness the application to mathematical algorithms of the concept of the integrated package for problem solving. These have been slow to emerge, perhaps because mathematicians were slow to realize that setting up equations is normally more difficult in practice than solving them, and that the manner of presentation of the solution is more important than just getting it.

At present there is a great deal of action in the use of splines for data-fitting in two or more dimensions, and for the solution of boundary value problems for partial differential equations. There is also a lot of activity in solving stiff differential equations, nonlinear systems of equations, matrix methods for least squares problems, in the invention of interactive mathematical systems, and in many other areas.

DESIGN IN THE FUTURE

It is hard to foresee what will happen in the next ten years of design. The best one can do is suggest one or two things that seem important. First and foremost is the need for complete communication and cooperation among three groups heavily concerned with design: the man with the problem, the computer scientist, and the mathematician. Second, there needs to be a

continuing trend towards a total systems approach to design, in which all parties have an adequate idea of what the other parties need and can furnish.

It is trite to say that a total systems approach is important, and it is equally obvious that it is slow to emerge in new areas. Just recall the types of houses built in California by the heavy stream of immigrants from the midwest in the early part of this century. They were nothing but Iowa farmhouses transplanted to a totally different climate with different solar conditions. Even now I often see houses built at Stanford with far too little attention to the intensity of late summer sunlight.

Wade Cole used to tell a story about a program for computing eigenvalues of matrices that he had. Engineers wanted to use it. So they instructed their desk calculating teams to evaluate all the complicated expressions necessary to generate the coefficients of a matrix for each case, and then brought each numerical matrix over to Cole to get its eigenvalues. Then, when they got the eigenvalues, they went back to their desk calculators to compute some more things, and so on. They had not the slightest appreciation that the automatic computers could handle the whole calculational problem, and thus furnish a problem-solving package for the total system. I am sure we have passed that stage in the solution of common engineering problems. But I suspect that many groups of mathematicians, engineers, and computer specialists are behaving in analogous ways about their total problem.

One notable characteristic of mathematicians is their desire to get theoretical in the narrow sense. That is, they are

accustomed to modifying the presented problem so that they can solve it elegantly. They would sometimes rather do this than directly attack a complex problem, or take it to a computer specialist, who might be able to simulate its solution.

Computer scientists can also be theoretical in the narrow sense. One of their failings is a desire to set up a complex operating system, a new language, and various processors that employ a hardware/software system more efficiently in a sense understood only by systems programmers. In their own way, computer systems people can avoid the actual problem quite as effectively as mathematicians! For example, have you ever tried to invert a large matrix while operating within the environment of demand paging? If so, you know that you must rewrite all of your matrix routines, to fit into the procrustean confines of the fixed page size.

Thus people with actual design problems need the cooperation of mathematicians and computer scientists who really put solving these problems ahead of creating either mathematics or computer systems. And, in particular, these practitioners of the tool professions must learn enough about each others' fields to be able to integrate arithmetic, algebra, analysis, engineering, file manipulation, user packages, programming languages, etc. with the subject matter of the problem into a truly powerful problem-solving capability.

In order to achieve this cooperation, each specialist will have to know more about what the other specialists can do. The computer systems man will have to know what problems are actually being solved on computers. The mathematician will need to realize the main facts about computer use, including how computer-executed arithmetic differs from the arithmetic of real

numbers. The engineers and others with problems will have to know something about what both mathematicians and computer scientists can do. For it is a priori obvious that the proper melding of differing capabilities can achieve better solutions to problems than any one specialist can find on his own.

These ideas have implications about the organization of our universities, where the next generation should be learning about them. How can universities, notoriously conservative about changing, keep reorganizing themselves so that students are not faced with barriers to learning the different materials that are important to their future work? With the enormously accelerated changes in our technology, no one has shown how universities can change at corresponding speeds. Students, left to their own devices, are ever ready to specialize within a limited area. Moreover, they have a liking for the abstract. Both tendencies interfere with their attention to problem solving, and with learning a broad enough base of methods to solve problems well.

I don't know the answer for the students. But I suspect that the leaders in universities must show the way to students by being quite ready to alter department structures, and to help some departments grow, and others to shrink, in close coupling with the changing needs of mankind.

Not only universities, but also Government agencies and professional societies can help focus attention on interdisciplinary activities necessary to problem solving. This conference itself, sponsored both both SIAM and both ACM, is an example of this desirable activity. Let us hope for many more such instances of cooperative activity.

And now let us get on with the important matters, the technical contributions to the conference!

BLANK PAGE

Unclassified

Security Classification

DOCUMENT CONTROL DATA · R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author) Computer Science Department Stanford University Stanford, California 94305	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP --

3 REPORT TITLE

DESIGN -- THEN AND NOW

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

Manuscript for Publication (Technical Report)

5 AUTHOR(S) (First name, middle initial, last name)

George E. Forsythe

6 REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
August, 1969	15	9

8a. CONTRACT OR GRANT NO N00014-67-A-0112-0029 b. PROJECT NO NR 044-211 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) none

10 DISTRIBUTION STATEMENT

Releasable without limitations on dissemination.

11 SUPPLEMENTARY NOTES	12 SPONSORING MILITARY ACTIVITY
---	Office of Naval Research

13 ABSTRACT

The author defines "design" and its subprocesses of analysis, synthesis, and optimization. He then compares design in 1946 with its scope today, both in regard to things designed and the design tools. This expository paper is concluded with an exhortation to designers, mathematicians, and computer scientists to meld their capabilities for optimum design in the future.

DD FORM 1 NOV 68 1473 (PAGE 1)

S/N 0101-807-6801

Unclassified

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Design						
mathematics						
computer science						
history						