Computers—From Youth to Manhood

NORMAN H. TAYLOR

YOU are very kind to honor me with the invitation to speak here today, and it is a pleasure to be here. I always appreciate a good excuse for leaving my overcoat and galoshes behind in Boston and coming out to sunny California. Furthermore, we are beginning to realize that although Boston is still of course the hub of the universe, nevertheless there are some pretty good ideas generated out here on the West Coast; and I hope I shall not return to New England without a few dozen of them in my carpet-bag.

At the first Joint Computer Conference in 1951 at Philadelphia, the computer industry was turning a corner. You might say that after a well-protected infancy, it was putting on short pants and getting outdoors and the neighbors were beginning to be aware of it. The machines, like some children I know, could be made to put on their "company manners" when their rich uncles came around, but this was all too often the result of intensive grooming and stern disciplinary measures behind the scenes. Our audience, while sympathetic and hopeful, still could not help feeling skeptical, and some may even have feared that we were raising a little monster who would turn out to be more of a liability than an asset.

Five rapid years of growth have served to vindicate our claims of 1951. I could easily spend an hour giving you chapter and verse on this, but suffice it to say that more than 75 large machines are in full-scale useful operation, and hundreds of smaller machines are doing important jobs for office and industry. The attendance figures at these professional conferences should be some sort of index; there were 880 of you at that first conference in 1951, now we are holding two conferences a year, and the 1955 Eastern Joint Computer Conference alone had approximately 2,000 people in attendance. This Western Joint Computer Conference and its counterpart, the Eastern Joint Computer Conference, will undoubtedly set a new record.

Thus, I think we can say that the computer business is getting out of its childhood, and putting on long pants. Adolescence is upon us, and although some of us, I am sure, look back with a certain nostalgia on the happy childhood days, nevertheless, like good parents we cannot shirk the problems of the present, and must try as best we can to prepare for the future. By and large, we are proud, you and I, of what we've done so far. But no engineer worth his salt is going to devote his time to admiring his past work. What the good engineer lives for, the air he breathes and the meat he eats, is the challenge of new problems. It is not too much to say that without new problems he suffocates and starves, and professionally he might as well be dead. If there were an engineer's Bible, I think this ought to be one of the two great commandments, so, using this text, my little sermon today is going to deal with the problems ahead of us. Please note that I'm presenting problems and asking questions, not trying to give you the answers. Even if I thought I knew a few of them, I'd be spoiling your fun, and that would be a dismal thing to do! I'll save my clairvoyance act for private chats, where I won't be embarrassed by seeing it in print and having to eat it five years later. You fellows are the ones who will have to answer these questions, just as today you have given us answers to some of the questions asked five years ago.

Let me return now and then to that notion of looking at computers as children that we've raised—it isn't really a bad analogy, if I don't squeeze it too hard; you know how children outgrow things, first their shoes, then their pants, then their coats, and so on. Or how a boy's mind in a way outgrows his body, and he imagines himself beating up all the tough guys in the neighborhood and fascinating all the pretty girls. Well, our computers have had and are still having just this sort of growing pain, the parts like arithmetic element, memory, and terminal equipment don't keep pace with each other, and we haven't yet got a mature, harmonious, balanced system. Until recently, our worst shortcoming has been the size, reliability, and speed of central memories; and we have put a lot of effort into improving them. The magnetic-core memory which has come out of this is now fairly well accepted as the central high-speed element of present-day machines, and is probably the largest single change of these past few years. Even so, we haven't quite caught up with arithmetic element speeds; the logical structure of most machines has been aimed at using memory time and capacity to the hilt. Elaborate buffer memories serve the purpose of emptying and filling the high-speed memory, and tricky instructions use other hardware to do jobs that might be done in the central memory if its time weren't so valuable.

At that, we haven't yet exploited magnetic-core memory to the full; the 64 by 64 memory plane is now pretty common, but one naturally asks the question, how much bigger can you build it? Preliminary work leads us to believe that the 64 by 64 plane can be extended to 128 by 128 and probably to 256 by 256 without any serious loss in speed or reliability. If this expectation becomes a reality, we shall have a 256 by 256 by 36 memory, that is to say, 2.5 million bits of storage with 6 to 7 microseconds random access. Surely this will be a handy tool for the future. How shall we use it? It certainly ought

---

Full text of the keynote address presented at the Western Joint Computer Conference, San Francisco, Calif., February 7, 1956.

NORMAN H. TAYLOR is with the Lincoln Laboratory of the Massachusetts Institute of Technology, Lexington, Mass.

The author wishes to acknowledge the helpful suggestions of W. A. Hosier in the preparation of this address.

to affect our present concept of providing large buffers feeding a small high-speed memory; it has about the capacity of eight conventional magnetic drums. Does this mean that drums are obsolescent?

But can the memory designer relax? Is he going to be free of demands for more speed and capacity from the direction of the arithmetic element? Vacuum tubes have been for several years the principal limitation on arithmetic element speed: we have pushed adders up to 1 or 2 megacycles, and with multipliers it is commonly felt that a law of diminishing returns sets in somewhere in the region of 1/2 to 1 microsecond per bit. I think the memory people, with the 6-microsecond memories, have provided, potentially at least, a good match for such circuits in speed and reliability.

But what about the future? Surface barrier transistors already give promise of operating at 5 megacycles in arithmetic and switching operations. What will come with the new gaseous diffused or drift transistors? I'm afraid the answer is inevitable: Memory will lag behind and the race will be on again. Why all this speed? Do we really need it? We are already being pressed to design larger and faster machines to tackle problems that are bulky and complex and have to be solved quickly. One way or another, I'm sure we'll try it. But if we had speed an order of magnitude over what we've got now, couldn't we do more time-sharing and substitute speed for equipment? Surely if something like this can be done, machines will be simpler, smaller, less expensive, and more reliable. This is a problem for the logical designer.

I have dwelt here on a few aspects of the internal consti ution of computers. In our analogy to the child, these would be his health, growth, and internal development. These are going well, with good momentum. I don't think we have any cause to fear that these children of ours are sickly or stunted. But as we all know, this is, if anything, the small end of the problem of rearing children: as they grow up, they leave more and more their old sheltered environment; they must be educated, they must (usually) do useful work, they must gradually accept more responsibility. I'm sure this isn't all, but these three requirements give me a convenient handle for the remaining things I have to say. Let's talk about this subject of useful work.

You can divide work roughly into two kinds: the kind that deals with symbols which you might call "white-collar work," and the kind that deals with matter and energy, which I suppose you would call manual labor or something like that. The kind of work that deals with people I don't feel has got into the picture yet. When we ask a computer to do the first kind of work, we call it data-processing; when it does the second kind, the current terminology would seem to be "automation." I am aware that the last word is often used more loosely, and that the dividing line between the two kinds of work is fuzzy in spots, nevertheless the division is a help in thinking about the problems, because the problems in the two cases are essentially different, and since there is no better word for the second kind, call it automation.

Data-processing deals with symbols, and symbols for the most part are discontinuous, discrete things: they come in chunks, and are grist for the digital computer mill. We are getting high-speed printers to grind them out. If we can get high-speed readers of print to grind them in without going through the clumsy media of punched cards or paper tape or the volatile and intangible medium of magnetic tape, we shall have gone a long way toward making the "white-collar" boys happy. So much for data-processing. Let's consider automation.

This field sometimes is referred to as real-time control, and it is quite a different story from data-processing. It seems to me to be the most challenging area of the future, heavy with problems, but also glittering with promised rewards The newspapers say it's here already, quoting from Mr. Kenney in the *Christian Science Monitor*:[1]

"In this contemporary fantastic era of research and automation, people today are better fed, clothed, housed, heated, cooled, propelled, entertained, and defended than at any time in all history.

The Alice in Wonderland, razzle-dazzle of new methods and processes are zooming living conditions and industrial activities into a never-never land undreamed of a decade ago and even today are leaving the most dramatic prognosticators almost baffled in attempts to describe what is coming up in the future.

Automation is in its infancy. Yet the pattern is taking form. Transportation demands magic controls. Airplanes both commercial and military are speeding through the skies so fast that human reaction is not keen enough for proper operation. Electronic, radar, and automatic controls are guiding the giant ships through storm and fog, over land and sea ...."

COMPUTER SETS PACE

"One of the most dramatic and important developments in the automation field is the computer. It is often called the electronic 'brain.' The computer can do just that at enormous speeds. It can add, subtract and multiply, etc. ..... "

This puts me in mind of an experience we had with radar during the war, which some of you may have heard of: The Admiral on one of our battleships, flagship of a task force in 1942, didn't put much stock in the reports of his radar man. But one night the radar man called up to flag pilot, saying, "Carrier Saratoga 2,000 yards off the starboard bow." The Saratoga in some aspects gave a characteristic double-humped blip on an *A*-scope that an experienced operator could easily identify. But the Admiral knew different. "You're crazy as Hell," he said, "The Sara's been off the port bow all night." But the radar man stuck to his guns, so the Admiral finally had a blinker message flashed, asking the ship to identify herself, and, sure enough, damned if it wasn't the Saratoga off the starboard bow. Well, that really converted the Admiral to electronics. So much so, that when the radar man a half hour later said, "Destroyer 4,000 yards dead ahead," the Admiral asked, "What's its number?"

I'm sure you can see the moral for us computer engineers in this little tale. If we're not well aware of our limitations and can't explain them intelligibly to others, we're likely from time to time to be asked to bite off more than we can chew. And believe me, in this automation field we have plenty of limitations at the present time.

For one thing, the computer is no longer operating in a vacuum. It's got to operate in somebody else's system, and if you just try to plop the computer into the middle of the system the odds are about 99 to 1 you can't make it work. You've got to study the whole problem and probably re-engineer a large fraction of the system before you reach any sort of harmonious solution. And this process has to be detailed and careful, not general or hasty. For the engineer on this sort of job, a fast, reliable central computer is not enough: he must take off the blinders and look at a whole new group of problems; in short, we need systems engineers.

For another thing, the sort of information that the system feeds to the computer is not usually a nice chopped-up bunch of easily-digitized symbols. It is on the other hand usually a nasty continuous rope of information from something like a thermocouple, a strain gauge, or a radar set; these analogue inputs are apt to be as embarrassing as a long continuous piece of spaghetti if we can't devise ways of handling them. What I'm asking you is this, have we put enough effort into analogue-to-digital conversion devices?

Not only does our young computer have to learn how to simplify information so that he can comprehend it, he also has to learn to be discriminating, to tell the truth from lies, to tell the important from the trivial. This is the noise problem. It's almost negligible when you're dealing with symbols, but when sensitive analogue measuring instruments and transmission lines from remote places come into the picture, it can become pretty bothersome. Communications engineers over the years have devised means, such as statistical processes, for coping with noise. We are likely to need closer liaison with them, and doubtless will have to help to extend their work.

Finally, of course, this computer out in the world of work not only requires eyes and ears and a central nervous system, as it were; it also must have muscles and tools. The azimuth and elevation of a gun, the force of a hammer blow, the degree of opening of a valve, the current in a welding arc—here we are again with physical variables, continuous functions, analogue devices. So here we require digital-to-analogue converters and servomechanisms. Are we attacking these problems as hard as we ought to be?

Return for a moment to the problems of the adolescent child, in particular, education. For a digital computer, this means one thing: programming. This area of programming is a complex problem. I would like to be able to make a few sage comments and dispose of it quickly, but I cannot. I have talked with several program people in an attempt to define just what basic problems they have which might be relieved by machine organization or electronic aid. I have been unable to recognize any single great weakness but here are some of the problems that could be relieved.

In an automatic control system the efficiency desired in a program is high. Many parts of programs run over and over again. The program must be written to accomplish its results in a period of time compatible with the demands of the system which is under control.

The program must be versatile; it must take into account the human monitor when he acts, yet proceed according to the established routines when he does not. It must be flexible. As we learn about new variables pertinent to a given problem we advertise that our general-purpose computer can take these into account by a simple change in program. The program must be written to allow such program changes without a complete rewrite for the problem.

The cost of such programs is high, and time to get them is long. It may surprise you to learn that, in general, the cost of programming a computer for an automation job is roughly the same as the cost of the machine. The cost is justifiable and defendable, but I feel sure that I should ask the question: What are we going to do about simplifying the process? Some advances have been made by using the computer itself to do the bookkeeping tasks and program assembly, register assignment, and the like. What else can it be used for?

In the strictly mathematical area the language needed to express procedure to the machine is fairly well understood and can be generalized and automatized. This is not true in all areas. Can we learn enough about the problems in automatic control to express the variables so that the machine can understand without being given every detail?

Our childhood analogy here becomes tantalizing. As the child grows he learns. After having solved one problem he uses the previous knowledge to attack a new problem. Can we design a machine to learn? Enough work has gone into this to show that the road is hard. The present crop of computers are fast and frisky, but they have the intelligence of an earthworm. I shall not pursue this further, lest I be quoted to my sorrow.

When I compared our situation with these fledgling machines of ours to that of parents with children about to be thrust out to take their place in the adult social fabric, I mentioned one last aspect of the situation; the need for gradual increase of responsibility. No one expects a callow youth to be running much of a show on his own; for a long time he inevitably relies on the supervision and judgment of more experienced hands. In the computer's case, the experienced hands can only be human beings, and the early automation systems are going to require extensive human monitoring. To be sure, in the fullness of time we may hope to build such sound judgment into some of these systems that they can tick along unattended all by themselves, but that day, the day of the truly automatic "no hands" system, is not yet. The need to know what is going on is more than just curiosity; it is part of the evolution towards complete automation and provides the means of correcting and improving our understanding of present concepts. Human judgment is needed in the more complex 1956 central systems for several reasons. First, in order to be completely automatic one has to understand a process with all possible contingencies well enough to tell a computer how to respond in any combination of events. Second, many people have to be confident that this situation does,

in fact, exist. There are other problems too, but these two alone demand an adequate monitoring system and my question to this conference is: How do we provide adequate monitoring systems to help our computer through adolescence?

How can the important Charactron and Typotron display tubes which are now available best be used on this problem? More important: What information must be presented to the human monitor, and in what form? Have you looked in a modern airplane cockpit lately? This is a good example of the monitoring problem with literally hundreds of meters, scopes, lights and knobs. We have so much information to give to the monitor that we ask: How can he absorb it all? Once he has received intelligent and accurate information and made a decision he must make this known to the system. What techniques can be used for this? The famed pushbutton certainly is effective, but has limitations of speed and accuracy. Certain electronic aids, such as the photoelectric pickup, have shown promise. Here is an area where new ideas and ingenuity are welcome.

Could we possibly consider using human voice to talk back to a computer to tell it what to do? How difficult would this be? The telephone people are considering voice-operated dialing. Why not talk to the computer in some similar way?

Another suggestion, less ambitious, proposes that a manual typewriter be used and the computer be programmed to decode the English language for its instructions.

Conclusion

I have posed some difficult problems to accomplish the education, and to increase the usefulness, and responsibility of our adolescent computers.

In 1951, the coming childhood of these infant computers seemed fraught with problems, but we have conquered most of them with determination and enthusiasm. Now in 1956 we have added to that determination and enthusiasm the confidence that the child is healthy and growing and fairly well accepted in society.

If you step back and look at it with a prospective of a few hundred years it seems to me that this being in on the childhood and adolescence of the digital computer art is a rare privilege which most of us here are sharing. Haven't you ever wished that you'd been living back in the days when some of the great sweeping syntheses were made that changed man's viewpoint and ways of thinking? I mean, for example, Faraday's discovery of magnetic induction, Maxwell's mathematical handling of radiation, Descartes' analytical geometry or Newton and Leibniz's early calculus, or, in our own time, relativity, quantum theory and all that has proceeded from them. There was always a time, but it had to be just one time and for rather few interested people, when these concepts were hot off the griddle and were the meat of red-blooded arguments, then, for all time to come, they got taken for granted and salted away in textbooks. Now I'm not flattering myself into believing that digital techniques are as far-reaching as Maxwell's equations (or as neat either), but I do believe they will have many unforeseen consequences that none of us dreamed of when we started playing with them, and for digital techniques the time is now, the people are you, and who knows, Joe Doakes' principle of binary substitution may be something your grandchildren will sweat over in first-year graduate courses!

And thus it is, gentlemen, that I am sure these precocious adolescents of ours are here to stay and to make their presence increasingly felt, it is up to you and me to make first-class citizens out of them.

Reference

1. Automation Widens Vistas of Research Potentialities, H. C. Kenney. *The Christian Science Monitor*, Boston, Mass., Jan. 4, 1956, p. 16.

Gestalt Programming: A New Concept in Automatic Programming

DOUGLAS T. ROSS

Synopsis: In any human endeavor there are three major phases: conception, expression, and execution. Gestalt programming is an attempt to make these three phases as nearly identical to each other as possible with respect to computer programming. In this paper the word Gestalt is used to mean a concept of a task to be performed by a computer. In a Gestalt system of programming, the Gestalt, or idea, is expressed simply and unambiguously in a special language, rather than through the laborious assembling of machine codes, pseudocodes, subroutines, etc. Using a Gestalt system, the expression itself in effect ties together integrated units of computer behavior, which function singly or in interrelation, to achieve the desired effect. The purpose of a Gestalt system is to facilitate the transmission of general ideas as in a conversation, between a human and a computer, so that the maximum use of their respective capabilities can be made.

After presenting the abstract theory of Gestalt programming this paper discusses several Gestalt systems in use today at the Massachusetts Institute of Technology (MIT) and describes briefly the types of computer hardware which are best suited to this application.

AS computer techniques have developed over the last few years, there has been a growing trend toward more sophisticated methods for connecting the human, who states the problem, to the computer, which is to solve the problem. Great strides in automatic coding schemes and algebraic coding schemes have been made, and the feasibility and value of these techniques are now well established.

Out of this trend has come, as a natural consequence of the maturing technology, a desire to use computers for solving problems which cannot be completely specified in terms which the computer can handle. This type of problem is united with automatic problem stating, referred to in the foregoing, in the general problem of using humans and computers together to solve problems. In the one case, the goal is to state the problem so that the computer can execute the solution, and in the other case, the goal is not only to state the problem to the computer, but also to assist the computer in obtaining the solution. In both cases, the human and the computer do only those parts for which they are best suited.

If the human and computer are to work together to solve a problem, there must be some means provided for the transmission of ideas or results between the two, since the contributions of each will depend upon the actions of the other. There is no known way in which ideas can be transmitted directly, so that an intermediate stage of expressing the idea in some language is always required. A language consists of two parts; a vocabulary and a set of syntactical rules. An idea is then transmitted by transmitting the expression of the idea; i.e., a sequence of words from the vocabulary. The final stage in the transmission is recognition by the receiver.

A major problem, then, in using humans and computers together is to choose an appropriate language for the interchange of ideas. This language must bridge the gap between the fundamentally incompatible characteristics of the two parties. The human is quick-witted but slow, while the computer is slow-witted but extremely fast.

Most people connected with the computer business seem to be superbly equipped for voluble discussion on any topic. It would therefore appear at first that the language should be chosen for the convenience of the slow-witted computer. Such is definitely not the case, however, because once the computer has been given a language, it becomes a very formidable associate, firing questions and answers at a rate which very quickly becomes alarming to the human. For this reason the first rule in establishing a language is that it must be as natural and convenient as possible for the human to use, not only in the interest of reliability, but for psychotherapeutic reasons as well. Programmers with persecution complexes are already far too numerous.

Since the language is to be used for the transmission of ideas, the most natural way to obtain convenience for the human is to have the language operate entirely at the idea or concept level. In other words, the language should be designed so that general statements can be made easily by the human, with the computer itself filling in the necessary details. This concept should work in the other direction too, i.e., the statements made by the computer to the human should be pertinent digests at the idea level, and not detailed reports.

In order to use the human and computer together efficiently, a statement in the language must lead to direct and immediate recognition and reaction. This may be accomplished by designing the language so that when a statement expressing an idea is made, the receiving party, human or computer, is able to recognize immediately the elemental concepts which are to be united to give the desired idea.

A word already exists which carries all of the connotations of simultaneity and sudden bringing-together of basic units into a single entity or pattern, and that word is "Gestalt" as it is used in the Gestalt theory of psychology. Since there is no single word in the existing computer terminology which works both ways between human and computer, and includes the connotations of being at a high level of communication and implicitly including active execution, the word Gestalt will be borrowed from psychology, and will be used in this paper with very nearly the same meaning in connection with computer programming.

The decision to introduce this new word is not capricious in any way, but is made to facilitate the presentation, and to assist in the establishment of a new emphasis and point of view with respect to the general problem of the interconnections between humans and computers. The actual material discussed in this paper is, for the most part, not new, but the way in which it is discussed is new. This new approach has been found to be very fruitful and clarifying, and is the primary motivation for this paper.

Although the idea of using humans and computers together to solve problems is relatively new, enough examples have been developed by various groups throughout the United States to demonstrate that these techniques show considerable promise. After mentioning a number of applications, (some of which have not yet been tried), to motivate the discussion, this paper considers in some detail the various stages involved in designing computer systems of this type by solving a hypothetical example. The abstract structure of such systems is then outlined, using the example for illustration. Finally several systems in

DOUGLAS T. ROSS is with the Massachusetts Institute of Technology, Cambridge, Mass.