FORMALIZATION: PAST, PRESENT AND FUTURE

H. Zemanek

In the Muir Woods near San Francisco, there is a cut through a redwood tree more than 900 years old. Such a cut is an excellent example for the relationship between formal and informal structures. Certainly the growth of a tree is a very natural process, and still the result comes very close to the formal notion of a circle. In this particular case the sequence of growing circles is related to the circular movement of our planet around the sun - two spherical objects, by the way, whose appearance suggests again the formal shape of a circle.

Very generally, formal expression can frequently be achieved by only little correction or manipulation of the informal pattern. The common ground is given by clarity and economy, by simplicity and minimum effort: the informal circle usually is the consequence of a balanced process, and the formal circle is the shortest, the clearest, and the simplest way to describe any closed, somehow round shape - every other description requires additional information.

The close connection of informal and formal description is not only true for shapes, it is equally true for many other structures. And it is particularly true for language in general - our means of communication and our possibility to express our thinking. In the most natural forms of language there is a remarkable co-existence of informal and formal aspects.

The letters, for instance, are suggested by the sounds which the body can produce in a very natural way. A natural word has not been created by a construction or any other artificial procedure; its history is all informal, and still it is composed of sounds or letters in a highly digital principle. Misspelling evokes syntactical correction, even if the receiving individual has had very little training or education; a child can speak correct words which he does not understand.

Some sort of formality, some amount of formality, therefore, appears to be natural and reasonable. The question is the proper balance between informal and formal: how far to push formality. And this question can be split into two components. Namely, how far should one go with formal derivations, that is, formal conclusions from existing formal structures - and, secondly, how far should one go with formal methods into the basis of what we are doing. Obviously, the better the basis is formally defined, the safer we are later in the formal derivations. The ideal case is total axiomatization, the reduction of the structure to atomic elements and their logical interconnection. But the work starts always in the middle: we are born into an environment which is very much like the redwood cut: there are suggested and even self-suggesting formal notions, but at the beginning of any investigation, such notations are not precise enough. Their further development frequently leads to a kind of separation - even of two hostile universes, the informal and the formal universe. In order to avoid frictions, tensions and fights, it is necessary to understand the virtues and the limitations of formalization and how its embedding in the real, essentially informal world can be done without harm. Such an understanding is better derived from a study of history than from a look at the present situation. So I will describe history first, then have a look at the present situation in general terms - many of you know more about formalization technology than I do - and finally I will try to draw a number of conclusions for future development.

The History of Formalization

Formal thinking is as old as technology, and technology is as old as the human mind. The ancient expression tools, however, were crude and did not lead far. The formal methods of the Chaldeans or of the ancient Greeks, consequently, could hardly be made useful for the daily life of the average citizen. And ancient science not only remained of philosophical character, it always remained rather an art than a formal building. Anyway, the basic principles of our present formal science were laid down as early as that.

Two important ideas were those of atomism and axiomatization. Both deserve a closer look, because after more than 2000 years of scientific development, they still seem to be the best our mind can offer for an understanding and a control of nature, and they are basic for information processing. While the idea of deriving a whole field - like geometry - from a few basic sentences by logical rules, and the idea that what ever we see is composed of a small number of indivisible particles combined and interacting on the basis of absolutely logical laws, are extremely formal in essence, they were expressed by the Greek philosophers in natural language, just denoting variables with symbols and writing numbers formally (but not yet decimally).

It was not until 820 A.D. that mathematics began to be formalized. The man who did this important step was an Arab living in the city of Khiva (now Uzbekistan), then called Khorezm (which due to the difficulties in writing vowels in Arabic was spelled as Khwarizm and in this form the city occurs as last part of the name): Abu Dshafar Muhammed Ibn Musa al-Khwarizmi. He had to describe the legal problems occurring, when an Arab with up to four wives of different legal standing died and left a big inheritance for them and many children. The partition was a huge mathematical problem, and it was for such problems that Muhammed invented algebra, in a book with the title "Kitāb al-jabr w'almuqabalah". The term algebra comes from this title and the term algorithm comes from the author's (that is, from the city's) name. The book was translated into Latin in Europe around 1200, but it was not until the 16th century that algebra was really accepted as a formal method. In that century, there was the struggle between the Abacists and the Algorithmists, between the concrete calculation by

means of calculi, little stones or coins, and the abstract calculation on paper and by more and more formal rules. The algorithmists won, because suddenly paper could be produced at a much lower price - which shows once more how much we depend on technology, even in such mental aspects.

Another important step of abstraction happened in the construction of buildings: the development of architecture. There also was a struggle between two schools, between the Italian and the British style of architecture, a subject which I intend to study in more detail. I will come back to the notion of architecture. Here, I will restrict myself to simple examples of architectural principles. One is the difference between English and French garden architecture: between the informal philosophy of the English garden and the formal art of the French park. The other is the difference between the Red Square in Moscow and the Place des Vosges in Paris: between the unsystematic combination of different buildings to a picturesque ensemble and the systematic design of a whole city square by one architect. If you add to these two pictures two other ones, say an arbitrarily selected set of blocks in Manhattan and a district like Teesside in England (an irregularly filled unit of a regularly planned city and a part of the country which was destroyed by unplanned, unsystematic and unnatural erection of poor people's houses), you get an idea of how ordered and how wild technology may go. And it is not different in our own field.

A third field of abstraction is mechanical construction, where design is based on blue-prints. We may not realize, but a blue-print is a formal definition of the object to be produced, composed of ideal straight lines, circles and other elements; somewhere in the corner of the blue-print there may even be found an indication which material should be used to construct the object.

The field of formalization, as we can see, is much larger than just mathematics, but there we can see much better how the idea of formalization develops. Mathematics finds out a need of formal definition of itself; after having used formal structures for centuries, in the 19th century it is recognized that the use of formal

structures in itself is not a guarantee for a clean situation - formal definition of the basic logic and of the basic notion is required. This chapter of history of mathematics starts with the name of G. Boole, and the next name to be mentioned is G. Frege. Russell and Whitehead formalize in their "Principia Mathematica" an essential part of mathematics, but this shows also how many problems there are still open. Hilbert gave an account of this in his Programme, but soon after that Gödel proved that the world was not quite as ideal or orderly as the mathematicians had thought and that even a field as numerical mathematics has undecidable spots.

Hardware is much better off than software in this respect, because digital hardware functions on strictly formal grounds. While it is true that in switching algebra the engineers reinvented the propositional calculus, they built their switching circuits for computers - without really knowing - on the universal foundation of mathematics, and in so doing they prepared the universality of the computer from the very beginning. Switching algebra, by the way, originated in Japan, where Hanzawa and Nakasima published their first articles in 1936, before Shannon's paper of 1938.

The formal character of the switching circuits enables the hardware engineers to turn to automatic design and subsequent automatic production with little change in the structure philosophy. Miniaturization amplifies the need of formal methods and strongly forced hardware to extreme discipline. This is how hardware became a model case for software: while programmers work with formal texts, they do so too frequently with highly informal (not to say: irrational) methods producing a situation where automatic design and automatic production become increasingly difficult.

I will come back to these problems later.

Formalization outside Mathematics

Not very many fields were formalized before our computer times, but formalization was not at all restricted to mathematics only. A few examples shall illustrate this statement.

Few people realize that musical notation is a true formal language, and even fewer think of music in binary terms, although almost everybody can see that most popular songs, whether children's songs or hits, are based on a 16 bar structure. Songs like

Hänschen klein, ging allein
Ein Männlein steht im Walde
Weißt Du, wieviel Sternlein stehen
Ich fahr' mit der Post

Jack and Jill went up the hill
Little Jack Horner
Pat a Cake, Bakersman

and hits like

Hüo ho, alter Schimmel, hüo ho

show the 16 bar structure perfectly.

The binary character of music goes, however, much further into what I call abstract architecture of composition. I was really struck by the fact that the first movement of Beethoven's VIth symphony (The Pastorale) had exactly 512 bars. It turned out that this was the only movement of the nine symphonies to show a precise power of two, but many show sums of two powers of two or come close to round binary numbers. Beethoven and many other composers liked building blocks of eight or four bars, and 64 bar blocks seem to be the most frequent bigger blocks in compositions. Of course, an artist never follows - rationally or intuitively - an abstract scheme without deviations or exceptions. But take out for instance the cries "Halleluja" from Händel's "Halleluja" and there remains a perfect binary composition.

The explanation for the binary character of music comes to a good part from the symmetries in the composing principles, each symmetry adding a bit. More results from a binary look at composition and

composition architecture can be expected.

The digital character of music could also be derived from the fact that digital automata have been used for more than 600 years to produce music automatically; from the chimes to the nickel-piano, there is a long history of formally performed music. Edison's phonograph opened a century of analog-stored music and its reproduction, but I am convinced that digital systems will take over not far in the future.

Another example of a binary craft and art is weaving - and again this is a field where automation started. The punch-card invented for the weaver's loom is only a systematic improvement of gadgets used in weaving for thousands of years, to be found in any folklore. One of my colleagues recently discovered an intermediate step of very early days in a local museum in Upper Austria: a programming unit built in 1740 consisting of a closed-loop piece of linen on which little wooden bars control the steering of the loom - an invention which may go back before 1690.

The explanation of the binary character of weaving is that each crossing of two threads marks a point in a digital weaving pattern. Mathematical thinking is, as is all human thinking, sequential. Neither the computer nor the loom are bound to the time sequence - maybe computer programming could learn from the loom and from weaving programming how to get rid of excessive sequentialization.

A third example of formalization is book-keeping, again with a formal tradition of centuries. The book-keeper restricts himself to the correctness of his books which are essentially extended forms. He insists on having the same final sum in both the "Credit" and "Debit" column, but he leaves the verification of the semantics of his data outside the books - precisely what the programmer does at the computer.

A similar reduction of a complex reality to simple forms and subsequent formal processing is the population register, the census - another origin of automatic computing.

In 1889, Hermann Hollerith filed his final patents for the punch-card system designed for census processing, and the US census of 1890 was the first one to be elaborated automatically - one of the two first rather, because the Austrian census of 1890 was also run on Hollerith machines, imported and improved by Otto Schäffler, who got the first patent for 'computer' programming in 1896. It took me three years to reconstruct the life of this Austrian pioneer.

I have mentioned already the blue-print. There was one attempt to formally define the blue-print. In 1907, a Spanish pioneer, Leonardo Torres y Quevedo, submitted to a meeting of representatives of Academies of Sciences in Vienna a paper containing the proposal for a notation for the formal description of mechanical constructions. As an example, he formally defined a small machine the description of which he had published a year before, a gadget for the automatic computation of the product of two complex numbers. The formal definition looks almost like APT and the formal description in the proposed language consists of 31 equations, including even the box of the gadget. Torres applies in his paper arguments in favour of formal definition which are as valid today as they were sixty years ago. (The Academies did not accept this paper.) As a final example for formalization, I want to take science in general - including philosophy of science.

For 200 or 300 years, science and technology have worked on what could be called a formal model of the universe. The method is basically the ancient Greek concept of Atomisms and Axiomatization. In the first two decades of this century this method had reached a peak of success. Russell and Whitehead had formally defined Mathematics. In Physics and Chemistry, atoms had become an established reality and laws of nature had been used to master not only all kinds of material but also all kinds of energy which must have been proven to equally have atomic character. In Psychology, a direction called Associationism was most powerful. It teaches that all we sense, remember and think is based on atomic sensory inputs (elementary sensations) and consists of their combination by the laws of association. The situation of science around 1910 called for somebody to generalize it to a philosophy of science of atomic and logic character.

This man in fact appeared. It was the young Viennese engineer Ludwig Wittgenstein, born in the year in which Hollerith filled his patents. At the end of the First World War, Wittgenstein had finished a manuscript entitled "Tractatus Logico-Philosphicus", which was nothing else but an algorithm for a formal definition of the universe in terms of logic and logical atoms. These atoms he called elementary sentences (the Vienna Circle later used the term protocol sentences). Try all possible logical combinations of all elementary sentences, says his general algorithm, check by means of verification whether they are factually true or false; throw away the factually false ones and collect the true ones, and you will have a perfect and complete formal description of the universe. From this algorithm, Wittgenstein correctly concluded the end of philosophy. For if that principle is made to work, there is nothing more to consider. And the last sentence of the Tractatus consequently reads: What we cannot speak about, we must pass over in silence.

Wittgenstein's derivations were not wrong, but some of his assumptions were. One could mention Gödel here, namely that not each logical decision process must come to a decision, and one could speak about the difficulties of verification. But much more basically, there is the fact that outside logic there are no atoms. We know this from physics and chemistry, where we have finally got a class of small particles which do not just promise a perfect description of the physical universe. The same fact could be established for psychology. First of all it is true for language. Even in the smallest unit of language (say: a gesture) one can find the pack a full story. Wittgenstein recognized this fact around 1933, and he started a philosophy II in which for instance the meaning of a word depends on the language game within which it is used.

What is of importance for computer science is the fact that the digital computer perfectly realizes the world of the Tractatus. Whatever happens in a computer system is a combination and sequence of true atoms of information - of bits - and about what cannot be said in bits the computer indeed keeps silent. It is only we who talk about it. To make it clear in terms of Wittgenstein II: the meaning of any program depends on the computer game within which it is used. There

will be for ever a gap between the formal universe in our systems and the informal reality, between the domain of the programs and algorithms and the life of people and communities - a gap which remains to be bridged by the human being, before and outside the mechanical and formal tools.

The Present

The computer has accelerated formalization in certain mathematical fields, and outside mathematics, has made it a practical need - simply by being itself formal.

The instruction set of the processing unit, the machine language, is an absolutely formal system with both syntax and semantics defined by the switching circuits provided by the electronics engineer who in turn was programmed by the mathematician who wanted the tool to be able to solve practical problems. The universality of the computer resulted from totally pragmatic motivations - the best case that could occur.

The way from the instruction set to the programming language is marked by steps which again were essentially practical. The idea of the flow diagram helped to pin down the general lines of a program - to any desired detail. What has to be mentioned next in this context is Zuse's "Plankalkül", a perfectly formal language going far beyond mathematics, which Zuse proved by writing a game of chess in his formalism. It is a pity that he could not continue this work and that not more attention was paid to it: it could have accelerated considerably the development of programming languages. It was Rutishauser in Europe with his "Rechenplanfertigung" who triggered algorithmic languages independent of the American efforts to automatize formula translation. Fortran and Algol were the results.

At this point of the paper, before proceeding, I should insert a detailed chapter on the theory of languages. But since I can assume that my readers know this theory fairly well, I can restrict myself to a few keywords, just to lay out a framework for the continuation of my line of thoughts.

The theory of scientific language as developed by Peirce, Morris and the Vienna Circle, distinguishes three levels of investigation, to which I want to add a level 0 which historically has always been the first item in language description: the alphabet. Here, we see language starting from an atomic concept, which however does not help for its essential content.

We find the only true atom, the bit, which can be used to construct a code for any alphabet, so that the character appears as a molecule: a combination of bits, a different one for each character. On the next level, an intermediate level which I will not number, the combination of characters yields the word - a unit of language as mysterious as the human mind, well supporting the particular meaning which the bible attaches to the word Verbum. As easy as it is to code a word, as difficult is it to catch its meaning, to define formally what a traditional word means. Only the constructed meaning offers itself to formal definition.

The proper levels of investigation are syntax, semantics and pragmatics. The purest definition of syntax makes it a set of rules to define all well-formed sentences; it describes the combination of characters or words absolutely independent of any meaning. The perfect syntax of a well-constructed language permits mechanical checking, whether the sentence is well-formed, and the rejection of syntactical errors.

Semantic studies the meaning of the (well-formed) sentence. The ideal principle would be to verify the factual truth of each sentence. The Tractatus (and with the Tractatus the Vienna Circle and Logical Positivism) assumed that verification was possible for a sentence (at least in principle), but time has shown that many difficulties inhibited any successful programme to do so. The Austrian-British philosopher Sir Karl Popper concluded that since verification was not possible, we are obliged to try to falsify sentences as best we can. Only those sentences which resist falsification are allowed to remain in our basket of temporary knowledge, and there is no final one except the tautological truth within formal systems.

The failure of Automatic Translation of natural language is certainly connected to those difficulties. Only in the constructed world of logic and mathematics and the computer are we on safe ground. There we have the chance of formally defining the meaning and to prove semantical correctness.

The third level of the theory of languages is pragmatics - everything not covered by syntax and semantics, like the use of the language, its user, the fashions and dialects, the history of the language (if applicable). What to achieve, in other words, with a designed language is pragmatics; this shows again that pragmatics is the begin and the end of language investigation, and syntax and semantics are only middle sections of it; and secondly, since it is obvious that there is no hope to formalize pragmatics, it makes clear once more that syntactic and semantic structures are essentially embedded in real life. We will never attain a total formalization of anything.

Let me shortly mention the distinction between natural and constructed languages, a distinction I have used already; natural languages can be alive and dead. There are mixtures between natural and formal languages. An example for this case is the medical language of which certain formal parts, the indication of the body temperature, for instance, like any other value of a physical measurement, can be the subject of formal handling and, therefore, of computer processing. The many, sometimes very advanced mathematical structures of medical knowledge are as formal as any technical construction. But there are other parts, for instance in a medical record, which are highly informal; it would be extremely hazardous to submit them to automatic processing other than storage and reproduction. I am sure that nobody would entrust himself today to a totally formalized and computerized medical treatment. This gives an idea of the heavy problems of medical information processing.

Finally, I have to mention the distinction between language meta-language, meta-meta language, and so on (meta^{n} language). It is clear that a natural language can be used as meta-language of any level. The open question is whether we shall see one day a formal language which is recursive to the degree that it can be described in itself, so

that we can do away with a natural language as the ultimate meta-language. So far, I have not yet seen any promising attempt to achieve this.

Now I will turn to the discussion of three levels of formalization which have gained importance around the computer; these three levels are

(1) formal notation
(2) formal definition
(3) correctness proofs.

The first level has been in use for a long time: it is the introduction of a formal language which allows mechanical derivations, so that it is possible to achieve general agreement on the result: if the rules have been properly observed, the outcome must be accepted by everybody. What can be challenged are the assumptions or, under certain conditions, the used notions. These notions will be challenged, if it is possible to derive contradictions, paradoxa. Then, a next level must be introduced, a formal definition of the notions, which is possible - I repeat - only in contructed languages. Constructed languages have the further advantage that their knowledge can be acquired in the mother language, so that in the formal field communication becomes independent of the command of a foreign language (of English, in the computer field) and of excessive language finesse of an author (which usually is identical with obscurantism anyway - not always, I admit).

The second level of formalization, the formal definition of the applied formal language, is practically always achieved by means of an abstract machine - the Turing machine is the best known example.

The abstract machine can assume a set of states and moves from one state to the next on the basis of a well-defined transition function. Since there is no reason to exchange all parameters during one transition - rather the transition will in most cases concern only a small subpart of the structure - it is convenient to organize the state as a complex system of substates and to introduce a full set of

transition functions, conditioned by state (sub-state) and input (or next elaborated part of text). A special control mechanism, which can be a part of the abstract machine, ensures the proper sequencing of the steps.

The abstract machine can define the syntax and the semantics of a formal language or a formal system, based on operational or axiomatic principles. It is certainly known to all of you that the Vienna IBM Laboratory has designed a notation, a method of definition and finally the formal definition of the semantics of PL/I, so that I am not just speaking of possibilities, but of real results (and a result of non-trivial size).

But formal definition is not the end of the formalization game. The next level is characterized by correctness proofs for the solutions. Since no human mind watches the processes running in nano-second steps through the computer, there is never an instance in which a wrong assumption can be observed out of the process details. Of course, we flatter ourselves that in our programs every possibility has been preconceived and that, consequently, there is nothing to be observed which is not already known. But are we always sure that this is really so? Even in a purely human operation there is much trust in offered solutions, and in many cases it would be difficult to object to a solution, because the whole situation has informal ingredients which remain dark. (Sometimes the solution is ready, the problem to which it applies is unknown. There is some logic in such a case: there are so many more questions than answers that it is economic to work out answers independently of the questions...)

A correctness proof establishes a link between a formal question and a formal answer: it makes sure that the answer indeed fits the question, that the solution indeed resolves the problem. For this purpose, however, formal notation and definition are a prerequisite, but they are not sufficient. A formal environment for the solution process is to be established. If the correctness proof, then, is achieved, the user enjoys all the reliability of a formal system.

This highest level of formalization requires an extremely clean field of operation, and definitely not every author in computer science is fighting for it. The tragedy is that it is possible to fight for formality on very obscure paths.

On the other hand, it is possible to fight for informality with apparently sound arguments which are not easy to refute. Programming in natural language sounds so attractive, so useful, so familiar. Why should it not be possible? And would it not be the solution for the average user? Natural language is what he knows, what he needs to learn (like a programming language). And yet, programming in natural language - except in trivial applications - is a naive error, committed and even cultivated, however, by the most sophisticated computer specialists.

If the problem exceeds the brains of the user, then only the programming specialist can help - if at all, because even the most intelligent specialist can only program what has been made clear to him.

Formal languages are a help for the clumsy style of a weaker intelligence, but in no case is the clumsy informal style a help against a lack of command of a formal language.

Whoever has the ability to think clearly and to express himself clearly, can learn a programming language in order to define the flow of data, energy and material to the precision required by automatic function. Lack of this ability will produce descriptions or instructions which the most advanced computer program will not be able to follow; the computer will do the right thing by mere (and small) chance.

The only other way out is what may be called "invisible programming": the design of an automatic system in which the processes - simple or complicated - are triggered by trivial actions (insertion of a magnetic card or a coin, deposit of a load), while the programs remain totally within the boxes.

The computer is an intelligence amplifier, if there is intelligence the input. The computer is an unintelligence amplifier, if ructed by ignorance.

It is a good goal, it probably is a necessity to protect the computer user from unnecessary learning (technology in general and computer technology in particular overburden the user with learning, in large but more so in small quantities); but there is some psychological economy in dealing with automatic equipment from which one cannot deviate too far. Quasi-intelligent appearance of the computer can only lead to deception.

Arguments for and against Formalization

A few keywords for and against formalization are

FOR	AGAINST
clarity	clarity
economy	learning
security	risk
freedom	freedom
generality, elegance	costly, impractical .

Clarity and unambiguity are the most important virtues of formalization. The abstract notions of the formal system are different from the concrete notions used in natural language, they are free from connotations and tacit assumptions - it is possible to express no more and no less than what should be said.

But this is precisely a reason to be against formalization: too much clarity binds all concerned to an amount which is frequently very bad or at least cumbersome. There are, of course, formal methods to leave out parts which are to be specified later, and no precision is lost - but such methods require a knowledge, a command of formalization which is not easily available.

There is a more important argument, namely that informal expression carries more information than formal - which is true. Think of the medical example given earlier: medical descriptions, medical knowledge can be analyzed and transformed into a formal system, but there is no doubt that some information is lost in this process. Clarity can indeed be a negative argument.

Economy, reflected by the shortness of expression, mechanical simplification and repetition, is obviously a strong argument. Repetition, in particular, yields the basis for savings and is the precondition for automitization, for

automatic programming
automatic generation
automatic simulation and
automatic documentation.

The counterargument against economy is to say yes, that is all true, but it requires an amount of education on the producing and at the applying side which destroys the advantages. This is in fact an important point which keen formalizers often overlook.

A similar double argument can be presented for security and reliability gained by formal treatment. Structures become exception-free, mechanic evaluation, perfect syntax - everything is very good - the interpretation goes outside the formal structures and becomes independent of the personal command of any living language, in which things otherwise have to be handled.

The counterargument again says that there is risk in the new principles; nobody can guarantee that they really will be successful, so that the overall gain is questionable.

Another fine property of formalization is generality and elegance; problem situations can be preconceived at an early stage of development, there are no arbitrary entries to the set of concepts and each of them is accepted only in the most general form; whatever one is working on, it can be applied to any other field where it applies.

The counterargument here is that generality is costly - not only in dollars, but also for example in compilation time. Elegance is often impractical and, in fact, some very practical things are highly inelegant. The specific solution by many people is to prefer the general, not the latter for economical reasons. If you want to make big business, rather you try it with a special patent button which is different from those on the market than with a generalized button including the common features of all others on the market.

Freedom, finally, is opened by formalization for the subsequent steps, because the system is ideally transparent, all possibilities can be seen without long trials. But the argument can be turned around stating that formalization removes the freedom of reinterpreting the early description, the possibility to say: you have misunderstood me, what I really meant to say was the following...

Of course, nobody will sell such a reinterpretation as a design principle, but intuitively it might frequently be a power in the fight between formal and informal.

All those arguments together lead to a basic principle of design which carries the name of architecture - abstract architecture, I should say, in order to distinguish it from what the term designates now - namely the same sloppy way of glueing parts together, which was formerly called logical design.

One can make an interesting exercise: looking up the Encyclopedia Britannica and reading what building architects understand by the term. It is defined as

The art and technology of building, fulfilling the practical and expressive needs of civilized people. The main goals are

suitability to use by human beings
stability and permanence of its construction
communication of experience and ideas through form.

Are not all of those thoughts perfectly applicable to what we are doing? Or to what we should do?

The chapters of the Encyclopedia keyword are

use - types
- planning
techniques - materials
- methods
expression - content
- form

and "Economy prevents work without (or only potential) demand".

The notion and the term of computer architecture were introduced in a book on the computer "STRETCH" in 1962, and there the definition given by F.P. Brooks is quite clear and alright. It was applied in the hardware design of the IBM System/360, but soon the term and the idea disappeared from literature. It was one of the /360 fathers who published since then on the notion of architecture, and he points out what makes good architecture. He distinguishes three steps in design:

Architecture, implementation and realization. Architecture is the functional appearance of the system to the user (what happens). Implementation is the logical structure - in detail - which performs the architecture (how it happens). And realization is the hardware construct (where it happens).

The key idea is consistency: if the architecture is consistent, a partial knowledge of the system allows the remainder (this is why I like to define abstract architecture as creative redundancy - repetition and symmetry are building principles not only in technology but also in the arts and in music particularly).

Good architecture is also characterized by orthogonality, propriety and generality. Orthogonality means no unnecessary coupling of concepts or functions which are proper to the essential requirements. Generality means that each concept, property or function which must be introduced will be introduced in its most general form.

Good architecture will show openendedness and completeness; openendedness means that there are no unnecessary restrictions and there is ample space for later and further development; completeness means that there is no arbitrary selection - if a part of a set, then the full set. Further keywords are symmetry, transparency, compatibility, reliability; good architecture is stimulating and self-teaching.

Of course, there is the danger that a perfect architecture is designed, but that it finally shows the property of poor performance - but such a bad result will not occur, if the whole design was based on the advantages of clean formal methods. System architecture requires, in other words, thorough formalization; it cannot be left to "natural evolvement". And this remark opens an interesting

relation to "General Systems Theory" - into which I will not look any further, however.

The Future of Formalization

Formalization is not a solitary, esoteric amusement of some extreme specialists, a theory for the sake of theory. It will become in very short time:

A production tool,
a management tool and
an application tool.

The computer is much too powerful to produce, to organize and to apply it by intuition or by casual methods. Information processing requires more precision in every layout, the faster, the bigger and the cheaper the equipment becomes - only thorough formalization offers the tight control which our powerful tool needs, if it should not go astray organizationally and financially in its growth and its infinite applications.

Formalization, therefore, will extend to the environment of the proper information processing equipment, a formal definition of the total system within which the computer is used will be necessary, an abstract machine modeling the total enterprise, institution or any other structure, not only its mechanical parts but also its human, to its natural structures.

This in turn will sharpen the problems of formalization. Many questions will be raised: what is the relation between a formal model and an organism, the relation between all those constructed languages and the natural ones? I have mentioned medical structures as examples of complex nature - legal structures are another instance of a complex situation. Although legal language and legal thinking have a certain formal character, the legal philosophy of formalization is essentially different from the logical and technical formalization; to put legal structures on the computer is highly necessary, but extremely difficult because of the gap between logical and legal thinking. Many other examples of problem areas for formalization and computer application could be given, a lot of work is ahead of us.

Formalization will grow in importance and extension, and it will have many feedbacks to the structures on which it is applied. Formalization will influence

The products resulting from formalized processes.

The enterprises and institutions applying formalized processes.

The professions which will have to go into formal treatment of their work.

The country in which substantial computer applications will produce a tight network of formalized systems.

Formalization will develop - forcefully pushed or happening as a side-effect of computerization - the art of abstract engineering, the type of engineer very different from the classic mechanical engineer, but based on the same virtues of compromising between needs and possibilities, between quality, cost and punctual delivery, between mathematical precision and illogic intuition.

Formalization will influence society. The life of the increased population of our planet can only be maintained by means of advanced technology, and good mechanisms, concrete as well as abstract ones, will be absolutely needed. But man is more than a machine, and human behaviour must be protected from totally submerging in a formalized universe. Also, society is more than a machine which runs perfectly if serviced by keen social engineers. There are fields which must be kept away from formalization and mechanical derivations, and the abstract engineer should warn certain other professions not to fall into admiration and simulation of engineering, where other principles of life and work should be pursued.

Formalization will have a feedback on the human mind which will dispose with the advance of formal methods of extremely powerful abstract mechanisms, which will make people like the rules and hate the exceptions - but this would be the contrary to the humanization of our world, where the exception is more important than the rule. Liking and understanding the exception will always remain the most human task for the human being, and in all our work in the formalization of information processing we should never forget this side of our human existence.