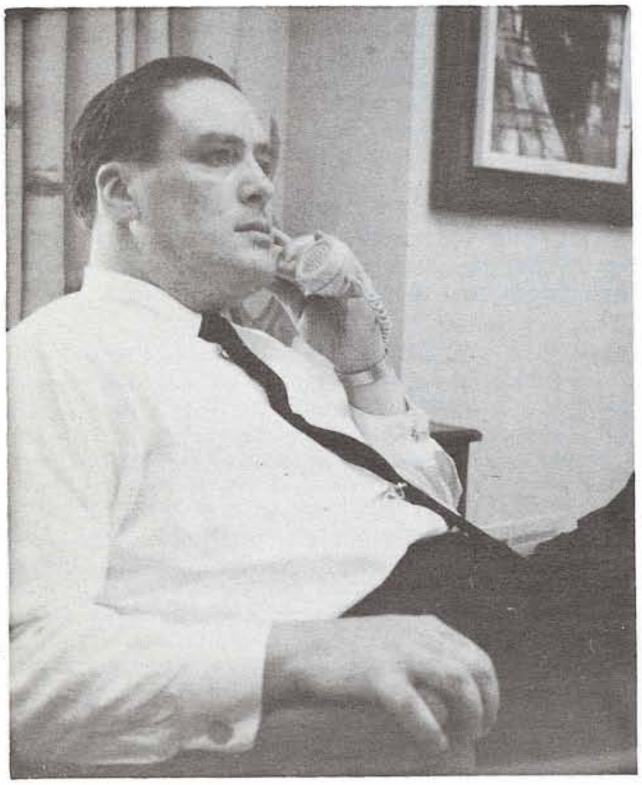

AN INTERVIEW WITH ELMER C. KUBIE

IN A RECENT INTERVIEW, ELMER C. KUBIE, PRESIDENT OF COMPUTER USAGE COMPANY, INC., GAVE HIS OPINIONS ABOUT DIFFERENT FACETS OF THE DATA PROCESSING INDUSTRY AND ITS FUTURE DIRECTION. THE FOLLOWING ARTICLE IS A RECAPITULATION OF HIS REMARKS.

The problem most discussed in the computer field recently is the need for qualified people. How do you feel about this problem?

This is a problem of qualifications rather than numbers. The field is so glamorous and full of opportunities that if we must triple the personnel in it over the next five years, this will be easily accomplished because the attraction is so great. The problem really is one of selection and training in order not to just fill vacancies, but in order to have sufficient numbers of qualified professionals so that the field can advance.

What do you think is behind this need?

In a sense we have found that it's easier to build machines than to use them. It's often convenient to use the field of music as an analogy. Supposing we had a factory which could turn out excellent musical instruments at a very fast rate. The result being that there just weren't sufficient musicians around to create the end product, namely music. This is precisely our situation today. The manufacturers construct magnificent instruments, but unfortunately there are too few accomplished musicians.

Why is it that most of the ads for programmers seem to play up the exotic applications in the field?

It's sometimes disturbing that the work in our field which is "furthest out" gets all the play. This happens often to the detriment of much solid " breadand butter" work that needs to be done and is being done. For example, the programmer who is doing the work on the exploration of the moon gains a certain amount of status compared to the programmer who is working on an inventory control and commission analysis application for a used car dealer. Now, it's possible that the fellow working for the used car dealer has a problem as complex logically, or perhaps even more complex than the fellow associated with the lunar project. Unfortunately, however, his wife or girl friend won't understand this and, in fact, very few people will, so somehow the fellow working on the moon project is a near genius, while the fellow working on the used car application is pretty ordinary. On top of this, the fellow at the moon project has a multimillion dollar system with which to work, while the fellow in the used car business must solve his problems with relatively modest equipment and solve his problems at relatively modest costs. The result of this paradox is that from the standpoint of intellectual complexity it's sometimes possible that the chap working on a "mundane" application has the more difficult task although this will seldom be recognized.

The consequence of this phenomenon is that there tends to be a drift in the field toward the exotic and computing professionals seem to take substantial pride in their work being "far out" rather than taking pride in quality craftsmanship of high utilitarian value.

This effect, unfortunately, is promulgated by the press, both popular press and trade press. The tendency is naturally to write from the standpoint of readership interest, therefore the more bizarre an area is in computer use, the more coverage it gets. Therefore the general public often gets the impression that this constitutes the bulk of our efforts in the field and therefore the computing professional gets the idea that if he's worth his salt, he should be in an exotic area also.

Third generation machines have brought the user many new problems. What do you think about the state of present hardware?

It is virtually impossible to apply present day hardware without first developing certain support software. Years ago one could plug in a machine and immediately turn to the task of applying it to user applications. Today, unless one has received from the manufacturer a package of support software programs this is not possible. In other words, if the equipment arrived in "virgin" form, without manufacturer supplied support software, the user would have to devote substantial initial effort to developing his own support software. Now this phenomenon has gone too far, for it seems unreasonable that a user cannot efficiently employ the "virgin" machine. It's like peach dumplings where the peach is the machine and the dough is the support software. This support software in a sense insulates the user from the machine and in effect he must eat peach dumplings although he only desires peaches.

Secondly, we have a situation where the computer designers are being heavily influenced by the support software specialists. Present day hardware is being designed substantially to satisfy the desires and wishes of the support software specialist rather than the needs of the ultimate user.

What about hardware in the future?

Future hardware should be designed with greater consideration being given to the ultimate user of the system instead of insulating the equipment from the user with ponderous support software. The machines should be designed to incorporate in hardware more features which are directly available to and directly in support of the user. For example, the machines of the future might contain built-in SORT instructions so that the user could employ this compound instruction for the purpose of performing standard sorts rather than depending upon support software for this purpose. Many years ago this was the trend, but it has not been for some time. For example, the earliest machines could not subtract, but one had to use complemented addition for this purpose. Early machines could not multiply, and a program of iterative addition had to be applied for this purpose. Early machines could not divide and a division algorithm had to be programmed for this purpose. About ten years ago built-in floating point was added to hardware for the first time. All of these things obviously improved the effectiveness of equipment substantially, both in terms of speed and ease of use. Unfortunately this mode of progress has seemed to be more or less abandoned in favor of doing everything by sopport software. Perhaps future generations of equipment will discontinue this trend.

We've heard many different opinions about the current and future use of COBOL, FORTRAN, ALGOL, PL-1 and other, more specialized programming languages. Do you think they offer the solution to the so called software crisis?

For years the advocates of programming languages have prophesised an imminent doomsday for programmers. Many languages were going to be so powerful, so universal, so compatible, so efficient, and so easily understood that the programming profession would die in its infancy. How can we reconcile this with what we today call a crisis for qualified personnel in this field? The answer is that higher level problem-oriented languages are, indeed, not any panacea. In fact, some of them might add to the problem.

There seems to be a love for the English language in this field, which doesn't make much sense. It has the questionable advantage of making it appear to untrained people that they are able to read computer programs with understanding. Unfortunately, however, the English language is full of subjective connotation and therefore it must be in some way restricted in order for it to control anything as objective as a computer. Furthermore, it seems to me that every technological advancement man has achieved has been with the aid of some cryptic notation that is, indeed, only understood by those expert in the field. Where would the chemist be without his form of chemical notations? Where would the physicist be if he could not write such meaningful unambiguous and concise statements as f = ma? Indeed, what progress would all of science have made without mathematical notations?

I have often thought of writing a brief article for our field which would consist of one simple example. I would take a relatively simple piece of music written for the flute and print it in the normal form of musical notation. I then would endeavor to duplicate this cryptic, concise and unambiguous manuscript with its detailed equivalent written in the English language. it would read something like "We will play a G at normal amplitude for .10 seconds. Then we will rest for .05 seconds and then slur into B flat slightly increasing the amplitude while holding the note for .65 seconds – – – – – –, etc." Not only is this obviously inefficient, but I don't believe that the musician could follow the English equivalent of the composer's piece while playing, unless he committed the piece to memory. In otherwords, the English equivalent of the musical notation would be barely intelligible. So what I am saying is that we would not have music as we know it today without the highly developed form of notation that exists. Just as we would not have today's chemistry, physics, medicine or any of our technological advancements if we had not developed appropriate forms of notation in support of these fields. I believe that one day we shall develop comparable notation for data processing. I believe that the field will remain in difficulty unless, and until, we do, and I believe that this notation cannot and should not be in the form of the English language.

Time Sharing has been a subject of a great deal of discussion lately. What are your views?

There are certain applications involving real time, or involving dynamic common data bases for which time sharing provides the only meaningful means of implementation. I have serious doubts, however, in respect to the expansion of this concept to include construction of what one may call data processing utilities. This is the idea where people at remote locations could tie in to a central computer through communications channels of various sorts, with those people solving problems of wide variety, and, in fact, doing most if not all, of their data processing through that means. I think there are two reasons which, in a sense, handicap this idea:

(1) The very universality of this concept would require extensive supervisory hardware and/or software. The more universal such a thing is, the more ponderous it becomes and the more inefficient it becomes. The system devotes significant time to administering and keeping track of itself rather than providing service to the user. I call this activity introspection. Therefore, such universality will be costly both in terms of capital investment and efficiency of the system.

(2) Time sharing is easy to demonstrate and difficult to make work. This is because a demonstration is performed with virtually no load on the system. It, therefore, avoids most, if not all, of the problems of queuing and interference. It is like the service one would experience with our telephone system if everyone else in the country would hang up. As a result the time sharing system is built and demonstrated, and seems to operate well, but this does not in any way allow simple projections of the behavior of the system as it becomes loaded. This is due to various reasons including the simple fact that queues do not develop on a linear basis. If one has a system which is lightly loaded to which one adds an additional slight load, the effect is one of little consequence. However, if one has a system which is already amply loaded, to which one adds an additional slight load, the effect may be catastrophic in terms of the behavior of that system.

What is questionable is not the ability to create such systems, but rather the ability to create such systems economically. By economically I mean that the systems provide services to the user which are profitable to him relative to his cost. At the same time the system must be profitably operated by the supplier who furnishes the service. To my knowledge there is no such system or service existent today that fulfills this criteria. In fact, in the past few weeks one such service went bankrupt in New York and another, which has been getting substantial play in the press, only survives through numerous injections of new capital.

What about current problems in support software?

Almost in every case where delivery or performance of support software is disappointing, we have a case of desire exceeding capacity, or imagination exceeding resources. It seems almost universal that support software specialists, when designing their system, try to make it as universal as possible and try to have it encompass everything that lies within the furthest stretch of their imaginations and indeed, the furthest stretch of the imaginations of everyone else who influences the design. Why should this be consistent with the resources in time, money and manpower available for the implementation of such systems? The answer is that it usually is not, and that such systems designed only with conceptual limitations will not have commensurate resources available for their implementation. Indeed, if a support software designer today said "I believe that with this time, with this money and with this manpower, I can produce this support software system," he would probably be chastised and told that he lacked imagination, for that system would not represent the limit of his conceptual thought. As a result he is under pressure to propose the most advanced and sophisticated support software that he is capable of conceiving without giving sufficient weight to whatever limits in resources exist.

What do you think of the overall contribution of EDP?

In spite of the many problems we have faced, and will face in the future, EDP is not in any way a hoax. It is real and it has contributed tremendously to our society and to our development and it will in the future do so to an ever-growing extent. There is not much technological advancement possible in any field tod without significant reliance upon data processing quipment. Government, business and commerce is also becoming increasingly dependent upon effective use of this equipment, and the overall social, economic and technical effect will be, and has been, beneficial.