Humanizing industrial control software

by J. B. NEBLETT and D. J. BREVIK
Honeywell Computer Control Division
Framingham, Massachusetts

INTRODUCTION

Process control by computer is common enough today to justify postulating an industrywide basic software system to aid in the implementation of control systems. There have been many successful industrial control systems; however, the main emphasis of this paper is directed toward the problems that prevented some other systems from attaining really successful operation. The reasons presented are mainly concerned with the software of programming aspects and not with the suitability of hardware or the basic financial justification for the computerized system.

Let's assume that the basic application justifies the installation of a computerized set of hardware, that the hardware itself is matched to the process problem in general and that the basic hardware is operational and can be maintained operational. What then is the underlying problem in getting a system of hardware-software and application knowledge meshed together into an operating process control computer system?

The "meshing" process is, of course, the programming of the system, and it has been common to regard this task as a mechanical one. Functional specs are laid down, then implemented. The resulting program(s) is checked out and put into operation.

The whole system now goes into service and usually, although it meets specifications, falls short of the user's expectations as an automatic controller (or operator) of his plant. Furthermore, he usually finds that the system is surprisingly difficult and expensive to change to meet the requirements spelled out by using it for a while.

This whole experience is analogous to taking a new graduate in engineering, assigning him a major design responsiblility and finding his performance falls short of expectations. The point, of course, is that the new graduate needs a period of TRAINING between leaving college and the assumption of heavy responsibility.

The authors believe the approach to programming a control computer application should recognize this TRAINING phase as being just as desirable for the automatic operator as for the human operator. In the following discussion the reasons for this view are explained.

The supplier's software

The programming language in the past has been very primitive. The basic software approach was:

1. To train the user to speak the language of the machine whether it be basic octal or symbolic assembly language.
2. To cause the user to learn the idiosyncrasies of the particular piece of equipment that he was attempting to apply.
3. To be sure the user had an intimate knowledge of the process to be controlled.

Systems in the past have reflected a notorious lack of a systematic approach to the programming problem. Software systems in the past reflected a definite lack of consideration for the inevitable reprogramming requirement common in process control applications. The poor documentation associated with software systems intensified the problem of the programming and reprogramming as well.

We believe that these problems can be overcome if both the supplier and the user regard the hardware-software system as resembling the characteristics of the human operator to be trained to do the job. This constitutes a large measure of common ground in that both the supplier and the user are familiar with the training of human beings and the terminology and techniques used in such training. The supplier wants to amortize the cost of development of his software over a maximum number of systems while the user wants to apply software

without major modification to that basic software. Few suppliers have the experience of their customers in the customers' applications. They cannot foresee every use, every possibility, without risking either extensive specialization or overgeneralization of software. Few users agree entirely with the suppliers' conclusions that are implicit in the software design. A common meeting ground appears to be in regarding the computer as a human operator.

Software designers of the past have hinted about "humanizing" software. Rarely, if ever, have they admitted a whole-hearted swing toward identifying human operator characteristics with the computer and its software. Our contention is that software should be humanized to the fullest extent possible and that this tactic would be the most common meeting ground of both the user and the supplier. We believe that a remarkable degree of humanization is desirable and possible.

Many times in the past, problem-oriented languages have been designed directly toward one specific type of user, e.g., the civil engineer, the simulation engineer, etc. The field of process control is so diversified that a supplier, without risking overspecialization, cannot afford to concentrate on only one type of process control or to generate problem-oriented languages dedicated and restricted to that type. There must be a common meeting ground where the supplier can make the equipment usable for the great majority and the user can economically make that equipment fulfill his special-purpose requirements.

This approach must include a basis for generating the problem-oriented language in much the same way that a supervisor would train his operator to think, speak, and respond in terms of the problem at hand.

Composite software for the composite user

Who is the User?

The supplier would design the software as a composite of the best operator features known. He would also design it to be used by the best composite user known. This does not assume that the ultimate user is an absolute genius in all aspects of solving the problem but instead assumes that the user is actually many people—the process engineer, the production manager, the plant dispatcher, and perhaps even the accountant. To put it simply, the software is usable by specialists typical in the process control field. The user takes an engineering approach to the problem at hand and recognizes the potential of improvement within the process. While he is not necessarily an expert in computers and instrumentation, he regards the computer and programming as a means to improve his operating process economically.

The user has a general understanding of programming or, at least, is capable of absorbing the principles of programming. He has had experience in personnel training and evaluation. He works with people and is capable of working with new people introduced to his environment. He is not averse to working with the computer if necessary and has no mental blocks against learning how to deal with the computer. He rarely has the time to learn all the intimate idiosyncrasies of a particular computer much as he rarely has the opportunity to learn all the intimate idiosyncrasies of his process. He depends upon others within his sphere to provide him with straightforward answers. He does not want to be an expert in their field. He is likely to regard the computer as an interloper into his particular domain unless the computer is easy to converse with, produces meaningful results, and does not usurp his job.

What is the user's problem?

The user's problem is to cause a digital computer to sample, interpret, and act upon information obtained from the process or from operating or management personnel concerned with the process. Presumably the application of the computer results in an improved economic picture.

His problem should not be to master a new discipline. He should not be required to learn the esoteric information concerned with programming, computer maintenance, and the peculiarities of the particular computer. We suspect he's in enough trouble keeping up with the changing demands of his own discipline.

Instinctive approaches to solutions of problems

An instinctive approach to the solution of the user's problems is based on making possible his communication with the computer as if it were a human person or at least something with many recognizable human characteristics.

While it is clearly not possible in the present state of technology to program the digital computer to act like a human being who is completely conversant in the specialty of the user, it is possible to program it as an "ideal" operator that can:

1. Understand currently known engineering facts of the process,
2. Understand how to apply good operating practices,
3. Make sound engineering decisions, and
4. Make management decisions regarding the economical approach to production.

Traits are what you look for in hiring an operator. These are the characteristics of the individual that indicate how well he can learn new abilities. On the other hand, most abilities are bonuses when you are interviewing an operator. An operator or a potential operator who has the desirable traits and, in addition, has some desirable abilities is a good prospective employee. Similarly an operator with the desirable traits but without the abilities is still a desirable prospective employee. Conversely, a candidate with abilities but without the correct traits is a poor prospective employee. The worst case is an operator who has bad traits and bad abilities (that is, preconceived notions).

Similarly, the software system accompanying an industrial control computer must be judged on its traits rather than the abilities taught to it by the supplier. The desirable traits are not always self-evident. Among them should be included the ease of trainability so that the ideal operator can learn the abilities required for his job. Other traits, such as honesty, integrity, reliability and so forth, are essential to the operator but are frequently overlooked or taken for granted in industrial control software.

With regard to these traits, consider the points of evaluation of a human operator during the pre-employment interview, training, day-by-day operation, and finally, the qualifications for advancement. Important qualifications of an applicant for an operating position would be:

1. Adaptability to the working environment.
2. Basic knowledge of the particular process.
3. Capacity for learning new procedures.
4. Logical rather than emotional judgement in response to process control situations.

The candidate must be reliable and consistent and must exercise discretion when called for. He must have a high degree of retention, and must perform his work accurately. It goes without saying that he must possess a sense of time, and along with this he must have a high sense of urgency in categorizing the relative importance of his functions. In other words, he must recognize that he can't do all things simultaneously but must schedule his activities when many things have to be done almost simultaneously.

While the operator is being trained, he must have a certain rapport with the instructor so that he can indicate lack of understanding when necessary. He should be honest and indicate that he is reaching the limit of his capabilities and is approaching saturation. Along with this, he should be easily retrainable so that when the instructor tells him to forget a procedure which may be erroneous, he will forget and will be able to learn an alternate procedure. He must be capable of learning at a reasonable rate of instruction so that the instructor does not get frustrated and impatient with a slow pupil.

There may be many teachers, many instructors training this operator, so he must be able to recognize gross conflicts of information and facts fed to him. Furthermore, there may be a hierarchy of teachers. After all, we don't want him listening to the janitor in preference to the production supervisor. He must exercise discretion about the information he is given; certain facts may be proprietary, not for everybody to know. Therefore, he must have a sense of secrecy. In short, he must have a keen sense of the meaning of an organization chart of authority.

If an operator becomes ill, he should notify his superior and perhaps provide some degree of self-diagnosis. This, of course, allows the supervisor to take alternative action, perhaps even replacing him temporarily. An absolutely ideal operator would be able to notify his co-workers that he was going berserk.

If an unnatural operating situation has arisen, one that was unforeseen, he should be able to tell his supervisor what actions he had taken up to and during the emergency situation. This enables an analysis or a post-mortem of the event in order to improve future operating procedures.

He must be able to accept responsibility as it is handed to him. While it is not reasonable to expect him to assume complete responsibility at the very beginning, it should be possible to gradually increase his responsibility as his skills evolve. As a corollary, if he has not been able to absorb a new responsibility—perhaps because of poor training—he should be perfectly willing and able to surrender that responsibility and to be retrained in that function.

Training and developing the operator

Given an operator with these traits, the user must train and develop his abilities methodically. There is a step-by-step procedure by which the instructor sets the pace according to the reactions of the student while learning, with a continual, informal evaluation of his performance on the job at each step of the way.

How might this be accomplished?

First, the new operator might be shown the control room and the basic nomenclature and jargon used in the plant explained to him. Certain basic procedures are described—such as how to read the instrument panels, how to contact the laboratory for analysis results pertaining to his job, whom to contact for production goals. At each of these steps, he is tested to determine whether he has learned properly—which may be a reflection upon either the operator or the teacher.

After being told how to read the instruments, he is tested and asked to read a few instruments while the instructor notes his performance. Likewise, as each step of data acquisition is explained to him, he is tested informally to determine whether he is truly following the correct procedure. The first step of this teaching process is to make him aware of his surroundings, the sources of basic information available to him and their uses. Perhaps he has not yet been told what to do with this information but only how to get it.

After this, he may be asked to record periodically or log the information he receives. It may be necessary at this point to explain to him the meaning of the word "log." This is possibly the first piece of responsibility he must assume. After he has done this for a period of time, his performance is evaluated: Is he performing the procedure that was explained to him? Is he extemporizing? Is he doing precisely what was expected of him? If not, why? Eventually he masters that skill, perhaps after a short period of retraining if some deficiencies have been found.

Having mastered one skill, the operator starts to learn the next one and his responsibilities will be increased. The next step may be one of accessing the information he has been taught to get, and comparing it with some rudimentary limits or constraints of operation and reporting violations to his immediate supervisor. Note that he is exercising nothing more than mechanical judgment rather than control or engineering judgment.

The skill he was taught in warning his supervisor when constraints were violated depended upon his previous mastery of an old skill, that of accessing the data needed. Throughout the learning process, new skills are usually dependent upon mastery of old skills. At every point, when a new skill—or procedure—is being taught to this operator, he is tested to make sure he is absorbing the instructions properly and is performing according to the standards set for him.

At any time, new data may be introduced to the operator such as a newly installed recorder, but the skill associated with the data remains unchanged. At any time, the operator should be capable of understanding that new readings or constraints or data are being introduced to replace those he once knew. As a general rule, the procedures he uses, once learned are quite inflexible. This does not preclude the possibility that an operator must be retrained to new procedures.

As the ideal operator progresses, and learns the basic skills of his trade, it is desirable to go beyond the mere mechanical manipulation of the process into a more detailed explanation of the phenomena taking place. We want him to become a practical engineer. This training would be accomplished by a senior operator who has had years of experience in knowing the idiosyncrasies of the process, and who can give a good layman's explanation to the operator. This senior operator knows the control characteristics and can anticipate the reaction to any change he imposes upon the system. If the trainee were truly the ideal operator with an engineering background, an engineer could train him in engineering terms.

Throughout this training process a methodology is developed. The operator is told procedure, is tested and if found satisfactory is given more responsibility which requires exercising the newest skill as well as other skills previously learned. This cycle is repeated until the operator is fully capable of performing what was originally expected of him. And also, at any point in his training or even after he is fully trained, he must be retrainable with a minimum of difficulty.

Traits inherent to the digital computer

There are certain traits desirable in the operator which are inherent to the digital computer. These are:

1. Reliability.—Computer technology has advanced to a state where extremely high reliability should be expected in any solid-state computer and peripheral real-time equipment.
2. Accuracy.—The inherent accuracy of a digital computer is generally superior to the sources of information from instrumentation and the typical accuracy attainable by normal operating personnel.
3. Retention of Information.—The computer is superior in its capability of retaining details associated with information. Typically, however, a human being has more overall capacities for information retention.
4. Speed.—The computer will typically be able to react and solve a problem more rapidly than a human operator.
5. Diligence.—A computer system is inherently more diligent than an operator and is unlikely to be found sneaking off to a smoking area at a critical time in the operating cycle.
6. Consistency.—A computer reflects no quirks or aberrations, but consistently executes what it has been told to accomplish. A human operator may be inclined to experiment without authorization; a computer system has no such ego and will consistently do what it has been told to do.
7. Flexibility.—A computer system which operates with a stored program has a flexibility somewhat

similar to that of the human memory and mind.

8. Emotional Stability.—The computer has no family problems to interfere with its control of a process system. The computer bears no grudge against the boss or ambitious fellow-operators.
9. Raw Senses.—Synonymous with the five senses of a human operator, the real-time computer has a capability of sensing external stimuli such as time, interrupts, raw instrument readings, and indicators from the process.

There are certain desirable and valuable traits of a human operator which are very weak or lacking in a computer system. They do not preclude the utilization of a computer as a substitute for the operator.

1. Curiosity.—A good operator is typically curious and probing to find out more about what is going on in a surrounding situation. A computer system is curious only when it is told to be curious and about those things that it has been instructed to probe further. Curiosity is valuable in an operator because questions associated with the operation of a process may be the first reflection of a possible abnormality within the process operation.
2. Heuristics.—The human being has the latent capability of learning which a computer system does not have. Although there are currently very active investigations into heuristic type software, it has not yet reached the state where we feel it is practical to include it in the discussion of a computerized process control. The heuristic capability of a human operator provides a capability of self-training which is not inherent in a computer system. A computer learns only what it is specifically taught and told.
3. Limited Communications.—Over the years, human beings learned to communicate not only by words or tone of voice but by the facial expressions and gestures that are typical in a conversation. The computer system is somewhat stilted in its communication and consequently cannot fully match the dialogue attainable by two human beings.
4. Self-Diagnosis.—A human being can usually detect when he is getting sick whereas a computer often reacts in an instantaneous and catastrophic fashion to a malfunction similar to the effects of a heart attach on a human being. Because of the nature of the malfunctions associated with a computer, it is often meaningless to advise a supervisor of pending "immediate" diaster, but is more logical to provide redundancy and/or external holding circuitry which can buffer instantaneous malfunction from the process. This allows a more leisurely take-over of control by a human operator.
5. Mobility.—In case of disaster, such as a fire within the process, an operator can run out and escape; but a computer is immovable and possibly will be destroyed in the blaze.

The above discussion is concerned with the basic inherent traits of a computer, both strong and weak. Given these traits and a basic command repertoire, it is possible to program certain *abilities* into a computer hardware-software system. These abilities should be analogous to the abilities we have described for a human operator.

We are about to suggest a computer hardware-software system with traits and abilities of an ideal operator. To distinguish between a human operator and the thing that is the computer software system we have chosen the term "android" for the latter. Strictly speaking, what we are describing is closer to a robot, but the term "robot" has certain overtones of sensationalism in the yellow journals. On the other hand, the term android is somewhat of a misnomer since the dictionary definition is that an android is an "automaton of human form." An automaton is a terribly clumsy form of human activity; a robot is a more sophisticated form, while in the science fiction literature magazines the android is the highest form. We have no pretentions of describing the highest form of android, but wish to avoid the visualization of a mechanical monster from Mars clanging down the aisles. We wish to avoid the typical connotation of a robot with mobility that is able to walk from place to place and having the manual dexterity typical of human beings. Our term, android, covers only certain mental characteristics or traits common to a human operator.

The abilities of the computer-operator

Traits alone are not enough. Traits are a measure of the character of the individual as well as the character of the computer. Abilities, on the other hand, are training patterns that take advantage of the traits. As an example, a typical trait of the human being is either honesty or dishonesty. An honest man can be taught how to operate the cash register and become a productive employee. Teaching the same ability to a dishonest man is a step toward financial disaster. Previously, we have enumerated the basic traits which are looked for during the interview of a candidate for an operator's position.

If we puruse the idea of identifying human characteristics with the digital computer in order to more

effectively simulate the human operator, then we must consider the abilities that are prerequisites for the basic operator.

Communicating with the computer system

The language of communication to the computer system should be that of a high-level compiler. A compiler is specified because it is possible to slant language to the user rather than to force the user to learn the particular language of the machine. While machine language has its place, we do not consider this the natural language used in industrial process control by an engineer or a chief operator talking to a human operator. As with any compiler, the user must have the ability to declare a form of data. Data takes the form of real, integer, Boolean, and logical, plus additional forms of analog, digital, BCD and the unique forms found in process operations. The data declaration capabilities in a compiler provide the *nouns* which can then be utilized to train the operator in the process. Future reference to previously defined data may then be made in a nomenclature familiar to all associated with the process.

Other nomenclature peculiar to a process where verbs can best identify the procedures of the process, the supplier should apply a basic set of verbs common to all users; however, the capability of defining additional verbs must be a latent part of the compiler system. Procedures consist of nouns and verbs and may be defined by another verb. Once defined as a procedure, future reference to the defined procedure can be made in terms of the new verb.

As the new operator learns the names and procedures associated with a process, so the computer system has this ability to learn such details required to operate the system. This simplifies future communication and once learned, permits much simpler reference to these previously taught nouns and verbs. Certain verbs are basic to the system, such as DO, PRINT, CONTINUE, IF, and others found in a language such as FORTRAN. New verbs imply a sense of urgency, the ability to wait or pause and proceed, and the ability to suspend at any point until the completion of a response to an action. In addition, the language must provide the ability to evolve toward a problem-oriented system. We do not claim that the language modifies *itself* toward a problem-oriented language but that it enables the user to think in a problem-oriented fashion.

There must be a means of indicating the relationship of urgency between all procedures. The computer—much as the operator—cannot be expected to sort out the relative emergencies within the system, but must be given indicators by the teacher during the learning process. These should not be rigid parameters of urgency, but relative instead. As new procedures are taught, the urgencies of all procedures must be sorted out by the computer system rather than have the user go through all the old procedures and redefine the urgency relationship of all.

It is natural to assume the operator has a strong sense of time and is capable of looking at a clock. Similarly, the language should implicitly assume a knowledge of time. Statements such as, "at 1100 hours do the following," etc., should be allowable. Statements similar to "every hour on the hour" should also be allowable. And statements such as "delay ten seconds" should be permissible. The sense of time implicit in the system should be the same sense of time a human being feels. The user should not have to supply long and tedious calculations to represent a next call time for the procedure.

A human operator is quite capable of understanding the context of a sentence such as "do this and, in the meantime while it is completing, go off and do something of less urgency but keep your eye on the completion of that event that you started because when it's dome go back to the more urgent activity." It is a natural way of expressing what to do when you reach an impasse. An analogy might be "start the pump motor and when the tank is full turn off the pump; but meanwhile take care of your other duties." Here the operator is assumed to have a sense of completion of an event. He does not have to sit there and stare at the tank to determine when it is full. He may have to glance periodically at the level indicator and, when it begins to approach fullness, turn complete attention to the tank; but while his full attention is not needed as the tank is beginning to fill up, he can temporarily divert himself to other duties.

Implied abilities

The computer system should be self-documenting. An operator can be called upon to repeat what he has been told. The computer should provide the same ability. A computer should be able at any time in the future to retell what it has been told. Obviously, it should retell in the language of the user and not in the natural language of the computer, that is, machine language. Everything that has been told the computer to date should be readily accessible in the user's language as well as in any internal form the computer may choose to use. However, a first offering might be hard copy produced at source time and consisting of a listing of the source program along with the date and the name of the user. This places some burden on the user to classify and collate the papers.

The ultimate offering, though, is complete self-containment in the computer system.

A human operator is self-organizing, he does not have to be told in which part of his brain to store information. Such an instruction can be as meaningless with computers as with human operators these days. A self-organizing computer system is required if the computer is to give any indication of potential saturation. Otherwise, the users are left on their own to update memory maps, and make intuitive judgments as to the degree of saturation of the system.

The ideal operator can be trained on the job and does not have to be pulled off of productive work to go to a classroom. It is the same with the computer. What is known now as background/foreground processing is a requirement.

Background is the place where new skills or procedures are taught to the computer system, while foreground is where previously learned skills and previously defined responsibilities are executed. The operator is trained methodically and is given increasing degrees of responsibility as confidence in him in gained. A new procedure introduced to a computer is debugged methodically. A self-documenting system which states unequivocally the exact procedure explained to it must also be able to state the sequence of execution that actually occurs. This documentation must also be in the user's language.

When an operator is given an explanation of a new procedure, he is generally stepped through it by the teacher to see if he really understands. He may be asked to repeat each step as he does it verbally so that the teacher may see whether the operator really understands what he is supposed to do. The first few times an operator attempts a procedure he is not allowed to actually influence the process. He goes through the motions and simulates what he is supposed to do. This simulation is expressed by some verbal comment such as "and now I twist this valve." This is simulation of the operation, and this software system should be capable of such simulation.

The teacher may wish to present a test case or test inputs to the operator to determine if he is doing the proper thing. He may give a typical problem, such as "suppose the temperature of the vessel reaches 500° what do you do?" The operator then is expected to go through an explanation of what he would do including how he would adjust control points in the process. Eventually, he would be allowed to act on real input data, but still would be restricted to going through the motions and explaining what he would do with that real data.

As the teacher gains confidence in the operator's ability to master the procedure, he increases the operator's responsibility. He is permitted to throw a switch, or turn a valve, or influence the process in some way. He may not be allowed to do everything at once but perhaps is permitted to throw the switch but not to affect the process in any other way. The simulation is graduated and the teacher may choose to what degree the operator may affect the process. The user must also have the capability in the android to choose the degree of simulation to be used during debug. Eventually the teacher is satisfied that the procedure has been mastered by the operator—or by the android—and assigns full responsibility.

The android must be capable of assuming responsibility methodically. A responsible procedure is placed in the foreground and becomes part of the real-time system. At that point, no interaction should be permissible between the background and the foreground. The new procedures that will presumably be taught in the future should not be able to influence the proven procedures introduced into the real-time system unless allowed to by the teacher.

The android originally must have been taught a form of organization chart. Not everyone should be permitted access to the background program development features in the system. Just because responsibility can be assigned to the computer does not mean that anybody can assign it. Each potential teacher permitted access to the background features of the system should be assigned his own secret recognition word known only to him and the android. The computer is responsible only to the individual that introduced a particular procedure and who taught the android that procedure. Conflicts of usage can be minimized in this fashion.

There may come a time when something unforeseen occurs and the operator will be asked to account for his actions. The android should retain a diary of what has happened recently for a possible post-mortem call. This diary should give a gross account of the procedures which have been called over the past few moments, the order in which they were called and a rough account of the outputs to the process they made. This permits a post-mortem to be made so that improvement of the procedures can be accomplished.

CONCLUSIONS

Certain features of the android are currently available. Background/foreground processing is offered by a great many suppliers. While it is not as sophisticated as proposed in this paper it is a start in the right direction. Current background processing usually means the ability to compile while foreground programs run real-time. Little emphasis has been placed on the ability to *methodically* link a new

background-produced program into the foreground.

Much of the currently available abilities programmed into a industrial process control computer have been based upon the supplier's experience in specific problems. His background has been in a particular marketing area and he has learned some general approach which seems to have satisfied his previous commitments but which may inherently contain what becomes a preconceived notion when applied to other industrial control areas.

An example is the "generalized scan program" which is intended to gather information from the process and lay it down in core memory. Typically, such a scan program is based upon experience in the petro-chemical field. How useful this approach can be to other industrial processes is an open question. Speaking bluntly, it is an opinionated approach, biased by the experience of the supplier in his early marketing ventures. Much the same can be said about the generalized logging routines which seem to assume the presence of a 30-inch typer. It is very rare when so-called generalized software of the nature of scan and logging programs is accepted readily by the operating personnel who actually have to contend with the process. How many suppliers have thrown up their hands with despair when the software is "modified" by the users! The choice has been, in the past, to modify the software or extensively retrain the operators or ignore the software and start over. But within all these software efforts has been the assumption that there is something basic to all systems having to do with scanning or logging or other operator functions.

Executive routines have been furnished which have a sense of urgency inherent in them. These too are based on prior applications. Most executive systems require the full understanding of all problems to be solved before the first problem can be tackled. The typical industrial process changes with time as the process engineer improves the production equipment or the objectives of the process. Such changes sometimes require a "re-education" of the executive routine.

There are instances in the past of the process control computer being abandoned because the re-education process was too difficult. Perhaps the original programmers had drifted away—as programmers are prone to do—or the original documentation was incomplete, or one of a myriad of problems occurred. Rather than contend with retraining the beast, the computer was downgraded to merely gathering data and printing a log. This is an example of the process evolving without having easy and economical reprogramming features.

Android abilities currently attainable

An extension of the background/foreground scheme would allow for methodical insertion of a debugged program into the real-time system. Some currently available compilers allow for a source language trace of the execution. This can be provided in a compiler intended for industrial control use. Self-documentation is also readily attainable but has normally been limited to computer systems with dedicated peripheral I/O equipment. This feature can be implemented in an industrial control android if the user is willing to dedicate certain I/O devices to the programmer. The self-organization and documentation characteristics desirable in the android require a certain amount of dedicated hardware capability for such a function. Directories of active "programs" in the system, drum mapping and core mapping programs, and other important documentation services require memory and time within the system to maintain up-to-date documentation.

It is important that the android system have an up-to-date library of all source information within the operating system. It must be able to have on call any selected program in order that the user can be assured of what is in the system and understand what the latest version of any program is. One of the weakest links within any user and computer system is the mismatch between the content of the program the user thinks is in the computer system and what the computer programs actually contain. If the computer system can give back only machine language information—a foreign language to the user—it is very difficult for the user to interpret this in order to understand the current situation.

The organization chart for recognition and identification of authority can be readily programmed into a computerized android system. Such a feature requires that the user identify himself prior to any attempt to modify information. Along with the identification capability, the computer system can easily be programmed to incorporate the corresponding sense of discretion required.

It is not difficult to program the sense of time within the computer system. Such statements as delay, or start at 11 o'clock may be incorporated easily within the android system. Simulation is attainable in graduated steps. The first step would be user-supplied inputs to see if the android is solving the problem correctly, then actual field inputs but trapped outputs to determine if the model was realistic. If all input/output calls are channeled to a central monitor, trapping of the actual inputs or outputs and substituted simulation is feasible.

Giving to the android the ability to remember the last few things it did, so that a post-mortem call can be made, is not difficult. Admittedly, it does occupy core memory but with a central monitor structure all activity can be retained up to a certain point. This, of course, is especially useful in the early debugging stages.

A limited degree of self-diagnosis of android problems is also possible. Potential saturation of core or auxiliary memory is not difficult to program and to announce to the user. The android can even detect when it is reaching a saturation of computational time. The android can keep rough track of how much time is spent in useful programs and how much time is spent in training and idle time. Compilers are available in current industrial control computers and eliminate the conversational problem of the user talking to a machine in assembly language. Few compiler offerings provide the ultimate in communication but are a step in the right direction. FORTRAN-based compilers offer the engineer the opportunity to speak in a mathematical language somewhat akin to his; however, none offer the syntax devoted to training the human operator. But it can be done if the syntax used in training can be defined.

Systematic detection of responsibilities can also be programmed now. It is a more difficult thing than merely adding responsibilities because the android may be simultaneously executing that responsibility when the deletion is requested. A deletion of a responsibility can be permitted only when it does not upset current real-time programs. It is not a simple matter of erasing a program previously defined.

Traits that cannot be economically programmed now

Heuristic judgment is currently being programmed in large computers, but is not economically attainable in small computers such as found in industrial control. Self-learning ability would be an ideal trait of the android but unfortunately must await future technological development of the state-of-the-art of programming.

An android that can express lack of understanding is also somewhat in the future. While it is possible and presently attainable for a compiler system to issue diagnostics about obvious misuse of the syntax and misrepresentation of data declaration, anything beyond that is currently unattainable.

The trait of curiosity is also a heuristic ability. The android is only curious about what it has been told to be curious about, and that is not true curiosity; it is only an expression of the programmers will. It may be possible to program the android to express "curiosity" about everything that occurs by causing it to print many messages, but this converts the android into a gabby operator to whom no one listens.

There are some aspects of self-diagnosis which are beyond programming today. Ideally the android would be able to predict that it was going to collapse in the near future or that some small aspect of it was in trouble now or that it was currently having localized problems. Although vendors produce test programs for manufacturing personnel to locate problems during hardware checkout, these programs have rarely been successfully incorporated into a real-time system.

Some projects are currently investigating a form of mobility for the computer. They have attached TV cameras as eyes and pseudo limbs as arms and perhaps some time in the future an economical version of a completely mobile android will occur, but not now. This android must sense the process from a fixed position and must have every input brought to it and every output taken from it by electrical means. It cannot stroll over to a recorder or to a manual station and expect to influence the process.

The android as a student

Programming the digital computer requires a methodical approach. First, the source program is introduced and the android as a student must interpret what has been told to it and digest it in its own time. The android must be able to announce basic misunderstandings about syntax and data declaration. This too can be done by current compilers. Once the source program has been accepted by the android, a debugging procedure is entered. The teacher does not allow the android to assume total responsibility and execute the program in real-time affecting the process. Rather, the teacher approaches debugging in a very cautious way. Artificial inputs are introduced to see what the android's program will do about them. All outputs are trapped, no output is allowed to go through the process. Instead, the output generated by the programs are simulated onto a typewriter so that the teacher may evaluate them. When the teacher is satisfied that synthetic inputs are being satisfactorily handled by the android's program it may permit the android to accept real inputs but still trap the outputs and type them for evaluation. This process proceeds at a pace set by the teacher (the programmer) who evaluates the programs according to a methodical, established debugging pattern. Eventually the teacher will allow the android full responsibility toward this program. The android then assumes responsibility.

Background compilation is quite common these days and, to some degree, introduction of a background-produced program to a real-time foreground program is permitted. However, a methodical debugging of limited yet expanding responsibility has not yet been done. It requires simulation of the inputs and output to the process. This can be done with today's technology of programming. Programs to accomplish that simulation would probably take a considerable amount of memory.

The compilers currently being utilized have the ability to accept data declaration and predefined verbs as a mechanisfor communication. The language requirements of the industrial process user dictate the necessity for the user to define new verbs to the compiler within a limited syntax. He must have the ability to teach the nouns and verbs commonly associated with his process in order that future communications are in a domain familiar to his environment. The associated processing procedure for such verbs as alarm, read, analog point, compare against limit, and set control point must be definable in order that future reference is understandable. This capability is not impossible because a procedural language such as FORTRAN can be used to obtain problem-oriented capability. (DYSTAL and SNOBAL are typical of such language.)

The android approach can be to industrial control computer programming what FORTRAN was to scientific computer programming—a basic system usable across the full breadth of the field while still permitting adjustment by the user to particular applications.