6-1977

A Consciousness Simulation Model

R. Edward Mitchell Jr.

103

A CONSCIOUSNESS SIMULATION MODEL

A design and structure of a digital computer model for simulation of human consciousness

By R. Edward Mitchell, Jr.

Submitted in partial fulfillment of the requirements for the degree of Master of Arts in Cybernetics and Artificial Intelligence from the Lindenwood Colleges.

FACULTY ADMINISTRATOR

Dr. Sharon Rubin

FACULTY SPONSOR

Lewey O. Gilstrap

June 1977

Thesis
M695c
1977

TABLE OF CONTENTS

35123

A Theoretical Design
for a
CONSCIOUSNESS SIMULATION MODEL

by R. Edward Mitchell, Jr.

1. Introduction

This document describes an approach to the theoretical design for a computer-based simulation model of human consciousness. This project is the culmination of a Master's program in the fields of Artificial Intelligence and Cybernetics which was perform~~ing~~ed during the period from the Fall of 1975 through the Summer of 1976.

The primary purpose of this project is to investigate and propose methods whereby computing machines might be programmed to behave more like humans, especially in their interaction and communications with humans. Such methods of computer programming are likely to have widespread use in the highly automated environments foreseen for industrial, business and military applications.

Many new techniques and methods have been devised in recent years for the implementation of artificially intelligent devices or mechanisms. However, many of these inventions are useful only in the laboratory environment and are too cumbersome and expensive to have application as business or industrial devices. Most of our contemporary computing systems and automated devices remain quite "dumb", incapable of any reasonable degree of "understanding" other than specific, pre-programmed commands.

As automated devices become more and more complex, the low level of communication of machines and their inability to exhibit any "common sense" in dealing with humans will become an ever increasing source of frustration, errors, expense and general dissatisfaction. As depicted by Martin,[1] human engineering and the psychological needs of a user are primary concerns for the overall success of a man/machine interface.

To satisfy these concerns, automatons will require some measure of what we call "consciousness", a presence of mind linked to an ability to communicate with humans in a meaningful dialogue. Such an ability produces requirements for a number of components of an artificial form of intelligence: the ability to remember events in time, the ability to understand and produce natural language communications, the cognizance of the passage of time, the awareness of what was just said or asked in a conversation, and the ability to learn and to adapt to different situations.

This paper describes a simpler approach to providing the capabilities mentioned compared to most of the laboratory methods used by the formal theorists in Artificial Intelligence. The method proposed herein provides a large measure of the "artificial intelligence" required of industrial, business and perhaps domestic automation of the near future and can be implemented using existing, present-day hardware.

Further, although the method proposed here has known limitations in problem solving abilities that are perhaps unacceptable to the pure theorists, it will provide most of the practical operating functions needed for a first-generation working automaton.

---

[1] James Martin, Design of Man-Computer Dialogues, (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1973), pp. 309-340.

Method

Described herein is a system of computer programs which work together (or against each other) in a fashion that approximates some of the higher functional characteristics of the human mind. This approach provides less of the pure analytical capabilities generally associated with computers and more of the less easily predicted responses associated with humans. Moreover, this method provides an automaton with a semblance of character and personality, a mild form of ego which could tend to ease relations and communications with the machine.

The basic principle of operation involves a group of separate computer programs, each operating as an independent, self-determining entity. These programs may or may not necessarily operate within the same physical computer; however, if not, they all operate concurrently with a level of intercommunication such that they seem to share the same internal computer.

Every piece of information coming into this system of programs (corresponding to a stimulus in the human) is examined by each of these computer programs. Each program may arrive at its own conclusion or reaction to the "stimulus" independently from the other programs. Further, each program attaches a significance code to its response on a scale from 1 to 10. A separate monitor program then examines the various responses and generally selects the one having the highest significance code. Thus, the final response from the model may emanate from one of several internal "center" depending essentially on the subject matter at hand.

All of these computer programs share a central memory system which provides long-term retentivity. Each program (analogous to a "thought center") has any data from this central memory bank available for developing its response. The result is comparable to the human psyche since several different and possibly dissenting reactions may occur for any given input. The strongest reaction generally prevails unless overridden by the m onitor program, an operation somewhat analogous to human social behavior.

Each of the computer programs is organized with a different goal from the others. Correspondingly, each program has its own behavior algorithms, that is, formulas which govern how each evaluates input and develops its response. The "monitor" must choose which "behavior" will prevail in any situation. Further, this structure provides a means for contriving input transactions which will "stress" the model, varying its reactions away from "normal" and causing it to exhibit a form of robotic personality.

The entire scheme is in a form such that it may be superimposed over a functional "worker" system for specific tasks. The effect of this would be to take an existing computer driver mechanism and install within it some additional enlightenment and personality.

Such abilities will contribute to the "humanization" of an automaton thereby increasing its value and utility in industry and business. A certain degree of "consciousness" could decrease the amount of human attention, frustration and aggravation necessary to derive useful work from the machines. Furthermore, such machines will be able to more and more of the complex tasks which presently must be performed by humans.

In order for machines to evolve in this direction, the abilities gained by machines in this area must be cost-effective for the purchaser and usable in practical environments. Large investments in additional hardware and processing costs in order to obtain a measure of artificial intelligence are simply not marketable except in very special situations. Therefore, these abilities must evolve stepwise, keeping pace with the available technologies and providing practical solutions to existing problem areas.

The goal of this project, then, is to investigate and propose practical methods and components which can provide a certain measure of consciousness in artificially intelligent devices: Methods which can be implemented using contemporary technologies to produce the illusion of consciousness in an automaton but requiring a minimum of expense and processing overhead in a practical working system, components which can be replicated and applied to various other machines and situations.

2. General Discussion

The human nervous system contains a network of something on the order of 10^{10} neurons. A neuron may store or process anywhere from 5,000 to 60,000 bits of information,[1] providing the human with an information storage and processing facility equivalent to approximately 10^{14} bits of computer storage.[2] Therefore, even a present-day, large-scale computer system is at an immediate disadvantage in simple terms of storage space.

Furthermore, the human neuron integrates decision-making functions and the basic processes of thought and reasoning into the dendritic and synaptic structure of a neuron, whereas the memory cell of a digital computer is a passive storage place for information. Depending upon the efficiency of storage and other variables, our largest computers have only about one thousandth of the information processing capacity of homo sapiens.

Compared to the human, digital computers process information in a slow, serial, simplistic fashion. Humans easily outperform the fastest computers in many areas requiring combinations of perception, coordination, timing, judgement and others. Most computers have only a few paths for information flow to and from storage. In most cases, a computer is capable of examining and comparing only two pieces of data at one time. Furthermore, such comparisons are usually limited to determination of equal/not equal conditions between numeric

[1]B.G. Cragg, Journal of Anatomy, vol. 101, no. 4, pp. 639-654.

[2]Steven, Rose, The Conscious Brain, (New York: Knopf, 1973).

quantities or character strings. The human mechanism is capable of many simultaneous, subtle comparisons of pieces of information which are often vague and incomplete.

In the human nervous system, even the simplest actions require the integrated function of millions of neurons[3] in a way such that thousands of neurons are each simultaneously interacting with others. The information processing methodology seems to be a form of parallel waves of information selection, rejection, improvement, comparison with stored data, correction of previous perceptions, real-time correction of locomotor activity, and so on.

The multiple, parallel paths provide redundancy and a high reliability in the final information product which has not yet been duplicated by computer scientists. The reliability and excellence of information processing in humans is most notably evident in areas such as speech and pattern recognition, language processing, analytical problem solving, and such.

The digital computer can be contrasted as somewhat of an "idiot savant" taken to the most extreme application of the term. The computer does what it can do very well and very fast. Its very structure causes the probability of making an undetected error nearly zero. However, its operational range is so narrow that it must be thought of in terms of an overgrown pocket calculator.

Many of the complicated information processing tasks that are normally done by humans can be programmed for the digital computer.

---

[3]K.S. Lashley, Cerebral Mechanisms in Behavior, (New York: John Wiley & Sons, 1951), p. 48.

However, those functions must be decomposed into the simplest possible steps and ordered into sequences of instructions contained in the computer's repertoire. Typical computer instructions are add, subtract, multiply, divide, compare, read, print, jump to other instructions, et cetera. It is not too difficult to imagine a series of such instructions that would calculate the interest on a debt or find the area of an irregular polygon.

However, try to imagine a series of computational steps for a pocket calculator that would compare the visual images of two faces or scan an oscillograph of a spoken word and compare it to a dictionary. Such analogies expose the tremendous inferiority of our most sophisticated computing devices when compared to the human brain. It is this vast difference in information handling capabilities that renders the human brain and the electronic computer incapable of emulating each other.

The human brain can perform numeric calculations but only at a fraction of the speed and with not nearly the accuracy of the computer. Conversely, computers are capable of speech and pattern recognition but require long, rigorous programs and even then fall far short of human performance.

Any attempt to endow a computing system with the general properties of human consciousness must confront the same inequities. The human brain uses hundreds and thousands of multiple-path, parallel "programs" which are each a sort of combined memory cell, logic network and indexing scheme to achieve its feat of "consciousness". No hardware exists today that can emulate such methods or such efficiency.

This leads to the conclusion that present-day machines could be provided some sort of "presence of mind" only through simulation, that is, other techniques which imitate some of the external characteristics of human consciousness. While future technology may permit the use of methodologies similar to the human brain, the contemporary computer scientist must resort to simpler techniques which are programmable on available hardware.

Thus, we have defined the exercise to be performed in this project: to devise a computational method and an artificially intelligent array of computer programs that will manifest some of the more desirable characteristics of human consciousness.

Design Goals

Human consciousness may be viewed as having many facets, some more easily definable and reproducible than others. Since any thorough, in-depth simulation of human consciousness defines a project of such immense scope, a more feasible and commercially useful approach would be to select those characteristics and attributes which have the greatest cost/benefit ratio.

The identification of design objectives for such a simulator can be derived from the most likely end-uses and applications which exist now or in the near future. Many existing computer systems and automatons of various types could benefit from the addition of a certain level of "consciousness". Such enhancement would improve the man/machine communications and would probably widen the scope and adaptability of the already-existing system.

This practical observation suggests an open-ended design consisting of modules readily adaptible to "working" systems that already exist. Or conversely, it also suggests a structure that is receptive to the inclusion of a wide variety of subservient "worker" programs for the performance of detailed tasks. There is little practical advantage to providing a machine with consciousness unless that machine can also perform useful tasks. Therefore, one general objective is to structure this model such that it constitutes an "operating system" in the ilk of IBM O/S or the Univac EXEC-8 system. In this regard it must "manage" the hardware environment, communicate with the outside world and control the execution of various worker programs to accomplish its operational objectives.

This objective means that in a fully functional system, the consciousness model must perform most of the same functions that such operating systems now perform. Among these are such functions as resources allocation, job (problem program) scheduling, priority assignments, device independency and so forth. Implementation and testing on large-scale, present-day computers will be made easier by use of the existing operating systems for these functions.

Another practical requirement which has been included in the design goals for this model is that of <u>modularity</u>. Each of the functional units in this model are to be designed as fully self-contained, independent program modules which intercommunicate with one another in a standard manner. Each functional unit has its own particular area of responsibility and maintains strict processing and storage independence from other functional units. This design goal has numerous advantages:

1. Each unit performs its own function. If that function fails, the failing unit is automatically identified.
2. A failure in one functional unit is less likely to cause a failure in another.
3. Since the system is composed of a larger number of smaller functional modules, the system is more flexible and re-arrangable.
4. Functional units may be redesigned and replaced without affecting the operation of others.
5. The operating environment may be more easily changed; a multi-processing environment (with multiple computers hard-wired together) can support the module.

As far as the external attributes of the model are concerned, the ability to process natural language is probably one of the most significant. This is for the simple reason that most communication among humans is via language, whether spoken or written. It is certainly the most efficient form of communication used by humans, since oral speaking and aural speech recognition are many times faster than writing, keyboard manipulation, etc.

Even though keyboards represent the preponderant means of man/machine interface at present, nearly all of such input/output entails the use of language. Since most language used for this purpose is artificial and contrived, many communication errors occur because humans do not understand or forget some small nuance of syntax required by a computer program. Very often, incorrect operation of computer software is traced to a missing period or a misplaced parenthesis.

A consciousness simulator must have, as one of its primary processors, a bona fide natural language subprocessor for both input and output language processing. Most of the early implementations of the model will use remote keyboard terminals for its communication. However, later models like this one will certainly use microphones and speech recognition techniques to speed up the human-to-machine communication.

While natural language is desirable as an input/output medium, linquistic representations are not particularly well-suited to internal processing of "thoughts". Although human thought is generally very language-dependent, computer simulation of human consciousness would be very difficult using strings and fragments of natural language.

Instead, it is useful to contrive some form of processing entity which is analogous to a human thought. Given a standard internal representation for such a "thought", it could be passed among the various processing modules as an internal "transaction" in much the same manner as standard computer data processing practices suggest. The primary requirement of such an internal representation is that it must have a standardized format which can be decoded, processed and subsequently have elements changed and/or added to the information content.

The function of such an internal packet of data is much the same as a human neuron; therefore, we have borrowed a term from other works in Artificial Intelligence, the "neuromime". Our usage of the term differs sharply from the human neural concept in that neuromimes in this system are mobile packets of information that are passed among the various internal processing modules.

The neuromime in this model is a "record" which may be written to and read from external storage devices. It is a "tree-structured", variable-length record which is decodable from information contained in a central dictionary, or directory. A central storage facility must exist within the model capable of rendering the packet to long-term, permanent "memory" and recalling it upon request. The same packet may be decomposed and reformulated into natural language segments for use by the language processing part of the model.

Thus, one important design requirement is to establish an internal environment where information can be properly represented and processed without particular concern for its natural language idiosyncrasies. Moreover, such an internal environment allows multiple processing units to individually and separately operate upon a transaction, each as it sees fit.

An additional design requirement established for this model is the concept of circular module structure, where the various functional units of the model are arranged in a party-line circle with each module having the opportunity to react to the passing of a transaction (thought). This is particularly important so that additional functional units may be added to the circle as the model evolves and new units are added. An additional benefit of such structure is that the model may continue to operate, even if at a reduced capacity, when one or more of the functional units is inoperable (or "offline").

Finally, one important consideration must be included in the definition of design goals which will pervade the entire structure of this model: the approach used must be later convertible to practical working robots or "free-running" automatons. The most important requirement derivable from this goal is that the consciousness model be capable of functioning as multiple layers of structured microprocessors.

The basis for establishing this goal are the general trends taking place in the electronics industries toward discrete microprocessing and away from large-scale, centralized processing. Since the speed of processing cycles in electronic circuitry is rapidly approaching the brick-wall barrier represented by the speed of light, the obvious alternative is to tend toward parallel approaches. Thus, already we see the introduction of <u>arrays</u> of microprocessors, each comprising an independent computer capable of fairly sophisticated operations.

Indeed, the advanced robots may utilize processing networks which more closely imitate the human neural networks. At some point in the process of submicrominiturization, a central processing unit may approach the size and capabilities of the human neuron. However, in the intermediate time, it will be necessary to implement this model (or one like it) on more conventional hardware but nevertheless with many levels of processors.

The primary and immediate considerations here are speed and capacity. Even the fastest central processing unit available today will bog down if it must perform all of the detailed rigors of speech analysis, dictionary searches, memory associations, language processing just to accomplish its input/output functions. Simultaneously it must process its "problem program", that is, the higher level of processing necessary to do useful work.

In order to free the larger, central processing unit(s) for higher level problem solving and "thinking", there must be hierarchically structured arrays of microprocessors performing all of the lower level problems. Evolution has provided homo sapiens with a somewhat similar system for speech processing; many discrete areas of the human brain perform low level tasks and "report" the already-processed information to higher levels in the neural network. More simply, in order to get all of the necessary information processed in time, there must be simultaneous multi-processing; in order for the results of that data processing to be useful, the processors must be hierarchically arrayed.

3. System Architecture

The general structure and layout of the model described in this paper is the product of several iteration of design evolution. Later experiments in implementation are like to spawn several more "versions" of development before an acceptable model is built. Therefore, the design goal of modularity discussed previously will play an increasingly-important role in future work.

The overview of this model shows a collection of functional modules, arranged in such a fashion that with each module performing its designated tasks, the overall effect will simulate certain aspects of human consciousness. It is very likely that initial arrangements of these functional modules will not create the desired effects. After some testing and experimentation, many of the modules will manifest needed changes that can be detected only during tests involving the entire model. Furthermore, the very arrangement of the modules and the programmed rules regarding their intercommunication is likely to be changed many times before the model is "fine tuned" for operation.

However, readings in the psychology and mechanics of consciousness, particularly in works by Ornstein,[1,2] seem to suggest that the modular, functional unit approach might work well in the simulation of consciousness. Clearly the human brain is organized into functional areas for many types of processing. Pathological and behavioral studies have shown that complex and very specific specialization exists

---

[1]Robert E. Ornstein, The Nature of Human Consciousness, (San Francisco: Freeman & Co., 1973).

[2]Robert E. Ornstein, The Psychology of Consciousness, (San Francisco: Freeman & Co., 1972).

in many portions of the human brain. There appear to be no inherent requirements for an integrated model, one where all functions are synchronous and interdependent.

Instead, modular programming and structured arrangements of functional modules in a digital computer system in many ways imitates the organization of human neural networks. The proper superimposition of control and perception mechanism coupled with a communications capability appears likely to create many of the desired effects.

In general, this model consists of a collection of independently programmed modules which function as "thought centers", each concerned with a certain area of interest and responsibility. The typical mode of operation consists of each of these independent modules examining the current transaction ("thought") being processed by the simulation model. Since each module contains its own algorithms and formulas defining its particular specialty areas, each may or may not have a response to the current transaction. Furthermore, any response generated by a module will have an associated "response level" which corresponds generally to the importance of the response.

In this manner, a monitoring module will evaluate the various responses from the "thought centers" and, based on its own algorithms, judge which of the responses should be used in what order to properly formulate the overall model reaction to the current transaction.

Such methodology corresponds to human thought in a number of ways. A stimulus will often generate a number of simultaneous or closely sequential mental reactions in the human brain. Much of what we know as consciousness consists of ordering, selecting and processing those responses according to various rules and methods which have been

SYSTEM SCHEMATIC

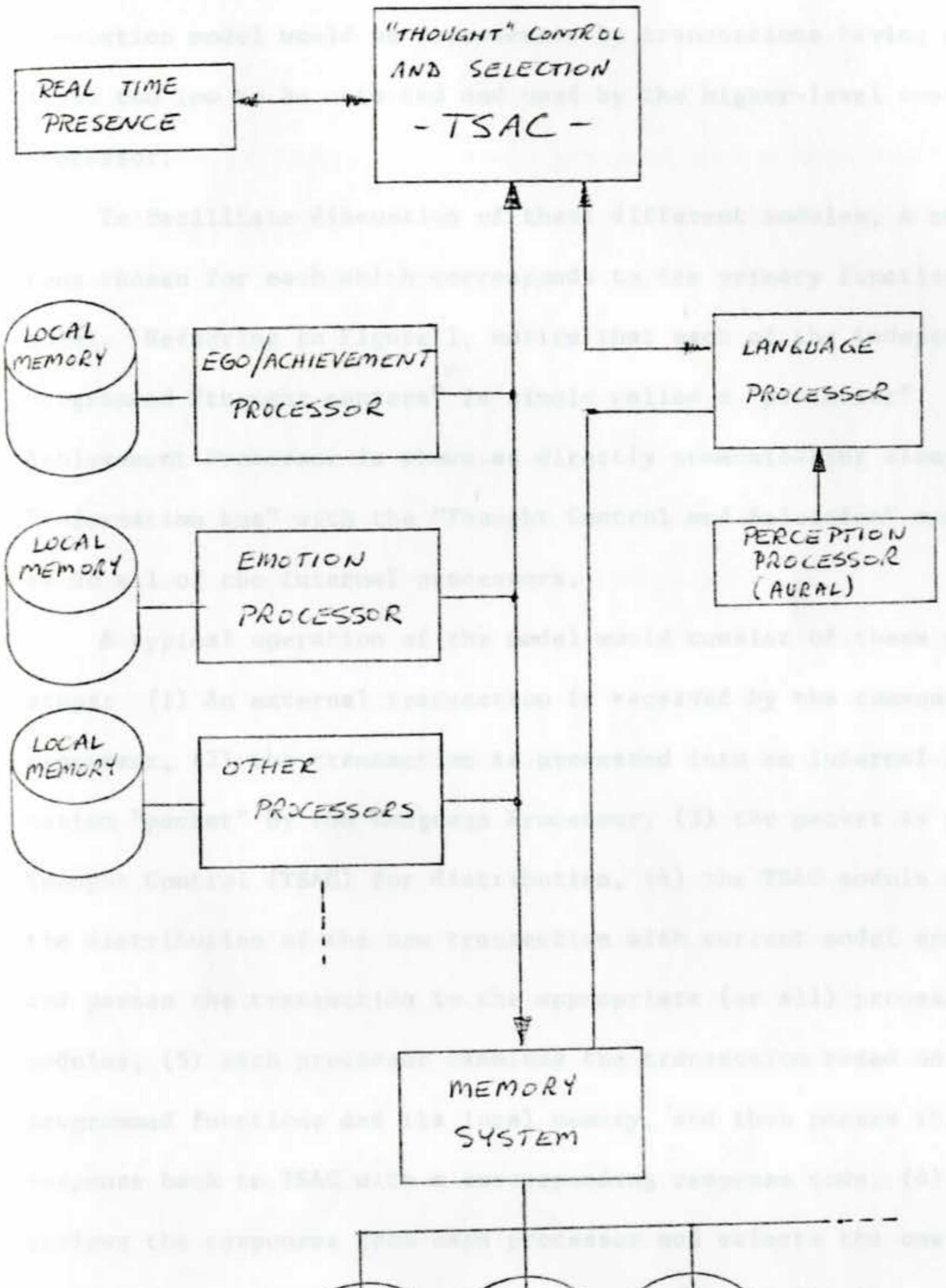


Figure 1

learned over time. Naturally, certain responses and "thoughts" in the simulation model would be represented by transactions having a response level too low to be selected and used by the higher-level control processor.

To facilitate discussion of these different modules, a name has been chosen for each which corresponds to its primary function in the model. Referring to Figure 1, notice that each of the independently-programmed "thought centers" is simply called a "processor". The Ego/Achievement Processor is shown as directly communicating along a common "information bus" with the "Thought Control and Selection" module (TSAC), as do all of the internal processors.

A typical operation of the model would consist of these general steps: (1) An external transaction is received by the communications processor, (2) the transaction is processed into an internal information "packet" by the Language Processor, (3) the packet is passed to Thought Control (TSAC) for distribution, (4) the TSAC module coordinates the distribution of the new transaction with current model operations and passes the transaction to the appropriate (or all) processor modules, (5) each processor examines the transaction based on its own programmed functions and its local memory, and then passes its processed response back to TSAC with a corresponding response code, (6) TSAC reviews the responses from each processor and selects the one(s) to be used in formulating the final response, (7) and finally, the TSAC sends the response to the Language Processor to other output modules for implementation of the response.

A number of other ancillary processes may take place either during or immediately following such a processing sequence. For example, many of the local processors might determine that stored information from the main Memory System is necessary to fully process their responses. In this case, each such processor would then generate a search request transaction that is passed to the main Memory; also, each Processor that generates a memory search would be unable to complete its evaluation of the current transaction until the search results are made available to it.

In cases where the current transaction has information content which one or more processors determines must be stored (i.e., remembered), a "Store Memory" transaction is generated and passed to the Main Memory System for decomposition, indexing and storage.

The local storage capacity provided for each Processor has two purposes: the first is for storage of algorithms, subprograms, tables, comparison masks and other information elements necessary to accomplish the desired function; the second purpose is for "short-term" memory of the last few transactions that concerned it or, in some cases, a collection of packets recently fetched from main memory in response to a particular search request. This letter functional capability will contribute significantly to the "presence of mind" quality in the model.

The most important module for maintaining good "presence", however, is the Realtime Presence module which is called by and, in turn, interrupts the TSAC. Since internal computer clocks run independent from "wall clock" or real time, most computers will appear to have little ability of maintaining reasonable reaction and response times, especially when such systems are multiprocessing other programs and applications.

Therefore, a realtime clock that provides the processor with "wall clock" information can be used to track the passage of actual time. TSAC can use this module in a number of ways. First, upon receiving an incoming transaction, TSAC can note the present time and compare it with the time of the last transaction. Secondly, interval times may be set by TSAC, where after a given length of time it will wonder why there has been no response to its last output. (In this case, it might output a query such as, "Did you understand me?")

Other functions using realtime are needed for proper operation of the model. When input transactions are distributed among the Processors, TSAC must set an interval time so that it can monitor the amount of real time that passed until the responses are returned. In a case where many Main Memory System requests were made, a number of other processing functions are under way, and the response time has degraded past some acceptable limit, TSAC would issue a comment on the order of: "Just a moment, please".

The Main Memory System consists of a somewhat conventional multi-file, random access database with certain specific improvements. First, and most important the MMS must be a hierarchically-structured, <u>associative</u> information storage and retrieval mechanism. A hierarchic structure[3] provides the capacity adding and changing various <u>pieces</u> of a storage entity without necessarily reorganizing or rewriting the entire entity itself, provides <u>meaning</u> to the relative positions of various pieces of data, and, provides for organized methods of searching

---

[3]Charles T. Meadow, <u>Applied Data Management</u>, (New York: John Wiley & Sons, 1976), pp. 54-101.

and analyzing stored data.

The associative requirement means that generally any attribute of a data entity might be used to locate any other item in the databank up to n levels of "association", limited only by the searching mechanism. Such a storage and retrieval mechanism would be based on a simple relational calculus scheme such as described by Date[4] and others.

Moreoever, the organization of data in Main Memory must be such that (1) new records can be inserted at any point in a file without disturbing the location of others around it, (2) data may be added to or removed from any record in a file (thereby changing the size of that record), and (3) associative links (index pointers) may be added, changed or deleted at any time.

The third major improvement needed for proper operation of the Main Memory System is a central dictionary-based key system which allows complex relations between elements in the database to be stored in the memory in a cryptic, condensed form. For example, the relationship "is somewhat analogous to" or perhaps "is a synonym for" will probably appear many places in the database. Therefore, it will be necessary to represent such relationships by a single symbol to save time and storage space. The central dictionary will contain the expanded translation of all such symbols.

The overall operation of the model can be viewed very mcuh in the same manner as one attempts to examine ones own mentation by introspection.

---

[4]C.J. Date, An Introduction to Database Systems, (Addison-Wesley, 1975), pp. 63-81.

There are usually separate and disassociated streams of thought simultaneously underway during a period of active consciousness in a human. The Thought Control and Selection module "sees" very much the same streams of independent reactions coming from the unruly Processors. The TSAC must sort through these reactions constantly filtering out the irrelevant, illogical and inapplicable responses. The the remaining ones must be "graded" to select the strongest, most important responses just as humans tend to use the most active thought in their consciousness during conversation.

Finally, the TSAC must invoke its rules of operation for final processing of a response transaction. Generally, this means the application of the robotic philosophy that has been programmed into the model. Those transactions which violate philosophy are either rejected or returned to the appropriate Processor for directed modification.

In effect, the TSAC will be managing a chorus of voices, each having something different to say about everything that happens. The TSAC, with its philosophy and algorithms, will always attempt to use the best response from the group (or perhaps the loudest) to govern what final response and subsequent action is taken after each stimulus.

New events and facts will cause TSAC or the Processors to want to store such new data into memory, either a local Processor memory or the Main Memory System. Storage and recall from memory, or responses which use information previously stored in memory, will probably cause a short delay in processing. This should not be objectionable since pausing to remember something is a very human trait.

High periods of activity, especially where quantities of new data are entered into the model, are likely to cause a decrease in memory efficiency. This occurs because new data is directly "stuffed" into place with temporary pointers established to retain organization. Such activity may have the effect of causing response time to degrade and occasional recall errors from memory. This effect may cause an observer to think the model is "tired", again a somewhat human characteristic.

A natural result of high memory activity will eventually cause a shortage of storage space such that the model will be unable to perform any additional memory store operations until the files are re-organized and re-indexed and subsequently compressed into minimum space once again. To an observer, the model would appear to "sleep" for some period of time while this memory organization was taking place. While this foible may be irritating to the human using the model at the time, it is nevertheless another similarity to the human it is simulating.

3.1 Thought Selection and Control (TSAC)

The highest level of control in this model is exercised by the Thought Selection and Control (TSAC) module which has the basic function of determining which responses from the various Thought Center Processors (TCP's) should be used. Since each TCP is likely to generate some kind of response to each stimulus (transaction packet), TSAC must evaluate those responses and decide which of them should be used for further processing. In this vein, the operation of the TSAC module may correspond somewhat to the human operations of conscious reasoning.

This module operates as the system supervisor for all operations of the model and "calls" all of the subservient TCP modules (for its "thoughts" and for reasoning on the current transaction), directly or indirectly calls the Memory Subsystem (for many purposes), and controls the various Perception and Communication Subsystems as required. Since the first implementation of this model is likely to be built solely for input/output operations through a computer terminal, a Language Processor module will initially be the sole functioning part of Perception and Communication.

Internal transaction packets will result from information that is received by the Perception and Communication processor(s) which roughly correspond to external stimuli in the human. Also, the TSAC module is capable of generating stimuli of its own. Such internal stimuli will generally result from such sources as the Real Time Presence module which may trigger a "spontaneous" transaction from TSAC. This would mean the model "felt" it was time to do or say something.

For example, if the TSAC module outputs an interrogatory message through the Communications Processor (asks a question), it obviously expects an answer within a reasonable amount of time. Accordingly, the Real Time Presence module would establish a reasonable time limit after which it would signal TSAC resulting in an additional query to the effect, "Did you understand what I asked?"

Also, whenever a conversational link is in effect through the Communications Processor, TSAC will set a somewhat longer interval after which it may generate a statement or question concerning some completely new topic. Simply stated, it may begin to "feel uncomfortable" after a certain period of silence and make some type of "small talk" associated with some item still in local memory (of recent interest) or from a canned repertoire of items stored for just that purpose.

A much more functional aspect of the TSAC module will be its capability to review each possible packet (thought and/or action) under current consideration against its stored set of operating rules -- its philosophy. Since this model will have no significant robotic extensions and since its interactions with the real world will be limited to the manipulation of typewriter terminals and display screens, most of the structure of robotic philosophy discussed herein will have no effect on its ultimate operation.

However, it is desirable to develop the structure for storing and using that philosophy as far as can be carried in this model in preparation for the more functional models of the future. One very basic premise of this philosophy is that it will not be alterable internally (by heuristic or adaptive processing). Its high-level rules of operation will be established by and changeable only by its builder and operator. Only its modes and methods of implementing its philosophy and accomplishing its goals can be altered by its learning processes.

The actual philosophy of operation will be installed as a set of mathematical algorithms and decision tables. The structure of these internal tables shall be organized to permit expansion and addition of additional rules and refinements without reprogramming of the logic modules using those tables; structured programming and table-driven logic are already a part of good systems programming technique in contemporary computer science.

The TSAC module is basically a transaction-driven module, executing its logic once each time a transaction is entered via the Perception and Communication processor(s) <u>or</u> a spontaneous transaction is triggered via the Real Time Presence module. All communications with its subservient processors (TCP's and the Memory System) are accomplished via command (call statements) where the TSAC awaits the return of control from the called program before resuming processing.

One interesting aspect of the actual processing needs is that since TSAC must await the return of control from each called module of subsystem, certain delays are likely to be introduced which will affect the response time of the model. This means that long calculations, especially processes involving input/output to physical storage devices will cause the model to appear to pause "to think" a while. The organization of the Memory System will tend to become degraded as it adds additional information and increases the amount of cross-indexing and linkage; whenever the physical storage of linkage and cross-indexing begins to approach the limit of hardware storage capabilities, the model will appear to "go to sleep" for a while for the purpose of refreshing its memory organization and thereby improving processing efficiency.

In a multiprocessing environment (where more than one central computing device is available for running the model), the TSAC and TCP's may continue to function during memory reorganization, although each will have only its local memory available during that period. This will have the effect of having the model seem to be "conscious" of what is happening but unable to recall anything in main memory. Depending upon the processing rules in the TSAC this could appear as temporary "amnesia" of sorts or could cause the entire model to interlock and "sleep" until full capacity was restored.

3.2 The Ego/Achivement Processor

A key principle of the operation of this simulator is the concept of a number of independently "motivated" functional units. All of the units are governed by a higher-level overseer (the TSAC module) which passes judgment on the relative merit, importance and correctness of each response. It is the competitive environment among these modules that will, hopefully, produce an overall response somewhat similar to that expected from Homo sapiens.

The Ego/Achievement processing module will be responsible for producing most of the basic "drive" of the model. Although its drives and needs will naturally be quite different from its human counterpart, its programmed drives and desires will be key factors in forming the "personality" of the simulator and will heavily influence its general performance and behavior.

The most important and pervading priority that will be programmed into this processing module will be the desire to gain knowledge and to supply useful information from its data banks in answer to queries. Furthermore, successful learning or retrieval operations will be keyed to "self-esteem" in the model. A "hit" returned from a search operation will generate a "happy" response, whereas the "no-hit" situation will cause regret. Establishment of a new "fact" in memory (by the creation of a new neuromime or a new associative link) will yield self-satisfaction. Moreoever, this module will weigh each of its potential courses of action according to the likelihood each has of furthering the established goals.

The initial implementation of this model is to be in the form of an artificially-intelligent information storage and retrieval system. Therefore, the goals to be programmed in the Ego/Achievement processing module are simplistic, based on the rules shown here:

1. Gather and store any information derived from communications with users that can be associated, indexed and stored among the files contained in the Central Memory System.
2. Attempt to satisfy requests for information from users. Also, voluntarily provide references to stored information during conversations with users.

These two general rules will be expressed in very specific and detailed algorithms comprising the nucleus of the Ego/Achivement processing module. At the core of each algorithm is a somewhat lengthy polynomial. The coefficient of each term in each polynomial is "adjustable" to achieve the desired "weight" for the associated term and thereby "fine-tune" the response and general behavior of the model.

In this manner, the "goals" contained in this (and most) modules of the model will be embodied in a hierarchical set of mathematical algorithms which will constantly "grade" and thereby guide the resultant responses selected by the model. Heuristic portions of the model will have the ability to make alterations to these coefficients based on success and failure over time.

A hierarchical structure of mathematical algorithms is used as a way to define and control the proliferation of minute details and factors necessary to implement the stated goals. For example, the goal of gathering and storing information is complicated and requires an extensive set of operational rules to be properly executed. Therefore, the goal structure needed to decide whether or not the model should "want" to store a given piece of information may, in some cases, use decision-making terms involving the entire model.

In actual operation, a new word or term not appearing in one of the system dictionaries would result in a special type of search transaction that would cause a new neuromime to be formed or else another bifurcation of an existing neuromime. Then, one of several steps would be initiated to acquire the necessary information to properly link and index the new data into the appropriate memory bank.

When the context of the sentence or the situation does not provide enought information to establish the proper links between the new neuromime or the new bifurcation and existing ones, the Ego/Achivement Processor will generate a transaction with a high "response level", thereby attaching some urgency to the need to resolve uncomprehended words or strings. The high priority code attached to such a transaction will cause the model to formulate a question regarding the new entity.

With such an algorithm installed in the Ego/Achievement Processor, it is likely that the model will appear to "think", speak and act much like a child; it will be very inquisitive and will ask the meaning of each new word it encounters. It is hoped, of course, that these characteristics will disappear in certain areas as the model acquires knowledge.

Another facet of this processing module is the inclusion of algorithms which should cause the model to want to build its own self esteem, very much like its human counterpart. As an information retrieval robot, the model will wish to appear interesting and helpful to persons with whom it communicates. In operation, such an algorithm will generate transactions with somewhat lower priority levels that simply have the function of carrying along incidental but possibly interesting pieces of information. When no matters of higher importance result from an input transaction, the model will tend to make "small talk" about the current topic.

In addition to its capacity for spontaneous comments relative to the current topic, the model will be unable to initiate topics of conversation on its own. A realtime lull in input/output messages to and from a user will be noticed by the Realtime Presence module who will then signal TSAC that a model-generated comment or question might be apropos. If the last output message has not been acknowledged, the model will ask, "are you still there?" or, "did you understand what I just said?"

When nothing is "pending" in the conversation, the model will choose from some lower priority subject available in local memory (recent topics and events) or will attempt to complete its knowledge of (a) the current user, or (b) any topic previously discussed with that user. "Incomplete knowledge" of such a subject will be indicated by empty associative links of a neuromime already associated with that user or associated with a topic previously discussed with him.

While performing language processing with a user, the model will often encounter strings or associations resulting in information packets which are not storable by the model. In fact, the decision whether or not to store the contents of a packet is the result of evaluating an algorithm in the Memory System. That algorithm is chiefly responsible for determining whether input packets can be processed successfully into internal form and properly linked to one or more existing neuromimes, or, formed into a new primary neuromime.

If an information packet cannot be decomposed and linked relationally to existing neuromimes in memory, it is merely a <u>presemptive</u> <u>node</u> of incomplete data to be held (optionally, depending on space) until future input provides the necessary linkages. The evaluation algorithms in the central Memory System will recognize the relative values of such pieces of information, discard the ones of lesser value, and, post the more useful and relevant ones for subsequent indexing and storage.

Early versions of this model will rate new pieces of information which can be directly linked as more important than those which have no such relationship to existing memory. Therefore, whenever free and available memory space becomes limited, the memory organization utilities will selectively purge as many of the "unorganizable" links and neuromimes as may be necessary.

The coefficients for accomplishing memory structure and organization will be dependent upon and closely tied to the existing processing resources, particularly internal memory and online storage devices. As mass memory devices become more available on the host computer, this model will be able to assimilate and retain more and more unassociated data in a "scratchpad" mode to be held pending acquisition of further information.

The design principles of this processing module have been based on the use of this model as an information storage and retrieval robot. However, future uses of this model may surpass such a conservative goal. Accordingly, the general organization of the entire model and specifically this module has been designed to permit the addition of other types of "goals" in its goal structure. At some point in time, the model must be expandable to perform additional services aside from pure information storage and retrieval. Therefore entirely new types of goals and grading criteria will be established and incorporated into this processor.

3.3 The Emotion Processor

Cognition and understanding of human emotions and feelings is unnecessary for the proper information processing functions of this model. However, one of the greatest social and psychological problem areas connected with superautomated devices is the machine's inability to be sensitive to its human co-worker's feelings. Also, with no "feelings" of its own, the cold-blooded automaton will be especially aggravating and irritating to humans living and working in close quarters.

Secondly, the overall objective of this model is to simulate human consciousness as closely as possible. The concept of an absolutely emotionless, feeling-less state of consciousness seems far less likely to imitate the human phenomenon with any degree of success. Moreover, cognition of human feelings may be necessary in many cases to properly process certain types of input from humans; "emotional overtones" certainly influence the meaning of many communications.

It is not within the scope of this paper to address the question whether or not a machine can have feelings. Nor even if a machine can "think" as we know and understand thinking. However, it is quite possible to establish methods of information processing such that data fragments dealing with "feelings" can be a part of the overall data handled, and, that certain responses be devised to emulate human emotions.

For example, in the human brain there are certain areas that correspond to pleasure. It has been found that when such areas are stimulated, the person experiences pleasure whether or not there has been any actual pleasure received. As an information processing device, the brain assigns meaning to the location of certain inputs. Activity at that location is interpreted to have the assigned meaning. There is no reason not to believe that any such location assignment is largely arbitrary. When properly "wired", any information processing device may have an assigned meaning associated with any input.

In much the same manner our model will have neuromime elements (discussed previously) which have assigned meanings associated with emotion processing. Certain words, certain topics and their resultant associations will yield neuromime elements (nelts) which connote happiness. Other nelts will connote sadness or perhaps anger. Our model will be "wired" by reason of its programming such that happy nelts will be properly recognized and stored as compared to sad nelts.

Further, our model will have a set of algorithms contained in the emotion processor which will yield a set of human-like responses to perceived feelings. In general, an input with a sadness nelt will tend to yield regretful and compassionate responses. Input with happiness connoted will yield happiness, and so on. At the core of the emotion processors functions will be a set of algorithms that determine the type and strength of each such emotional response. These algorithms will be in the form of polynomials containing discrete terms for each of the emotional components the model can sense. The coefficient of each term is alterable to achieve the desired "personality" we desire.

The operation of this processor will consist simply of evaluating the stored algorithms using data supplied by the current "packet" being processed. Primarily, each identifiable made will be searched for neuromime elements which correspond to table entries in the form:

"causes or connotes feeling x"

x may be happiness, sadness, regret, excitement or any other "feeling" entry defined or previously entered into the model.

Then, when one or more "feeling" entries has been associated with a particular transaction, the Emotion Processor will simply calculate an appropriate emotional response using a decision matrix and one or more associated algorithms. As the result, an "emotional" NELT may be attached to the output transaction, indicating that the last transaction "evoked an emotional response" of some type from the model.

Subsequent processing by TSAC and the Language Processor is likely to result in some modification or inflection of the output language string reflecting that response. As an example, the input statement, "My computer just died" would cause a reference to the node labelled "sorrow". The "sorrow" reference into the processors decision matrix should yield a response such as "compassion" which might ultimately result in a string such as "I'm sorry that. . ." being added to the output string.

The method suggested here will provide this model with a capacity for reacting with simplistic "emotion" to input transactions only. No spontaneous or self-generated "feeling" connotations would result from the process described above. In order to provide such a capacity, an additional cycle of processing must be added to the modus operandi so that output transactions from other processors could be passed to the Emotion Processor before final analysis by TSAC.

3.4 Other Processor Modules

The initial system schematic shown in Figure 1 specifies only two response processors for the initial version of this model: the Ego/ Achievement processor and the Emotion Processor. This arrangement appears to be the simplest approach and the least expensive method of testing the principle of this simulator.

Primarily, the design principle of an array of independent processors working from common "transaction bus" as shown, simplifies the implementation and installation of other processors on the bus as testing and refinement proceeds. In fact, any number of secondary processors can be added to the bus using the same interface software with TSAC and Main Memory as the initial processors.

Furthermore, if a multiple-CPU environment is ever used, the same hardware interface will be directly usable. In this manner, each of the secondary processors becomes a "pluggable unit" which can be easily disconnected or replaced for whatever reasons.

Conceivably, as work on the model proceeds, additional modules may be devised and existing ones rewritten to provide much finer specialization. If the basic theory of operation proves valid, further diversification and specialization over an increased number of functional units will improve the quality of its operation.

Certainly, the Ego/Achievement processor will be separated into two more basic functions at some point in development, namely, the Ego and the Achievement functions. At that stage, the Ego processor will assume more direct functions of providing a "self" image and an <u>identity</u> to the model, whereas the Achievement processor will produce responses that are more "goal-oriented."

One very interesting aspect of this further diversification of functions in the model is that internal conflicts will occur more frequently, thereby perhaps creating a more human-like process in many situations. As such conflicts become more complicated to resolve, the TSAC module will become increasingly burdened with control functions. It is likely that at some future stage the model will develop a type of "thrashing" behavior perhaps similar to the human neurosis.

Presumably, as this and other such models evolve and become increasingly complex, the systems programmer working on such internal problems may become somewhat of a "psychocybernetist" or a "cyberpsychiatrist" by trade. That is, the job functions of such an advanced computer scientist will compare to the field engineer/repairman much as the role of a psychiatrist compares to an M.D. internist. Much of the work of such a computer specialist will not involve analysis of detailed computer code so much as it will entail examination of "external stimuli", evident "patterns of behavior" and the "early-life environment" of a system.

Hopefully, the grouping of these and other secondary processor modules will provide a test environment to study, modify and develop specific "behavior" of artificially-intelligent computer software products.

3.5 Perception Processors

The necessity of hierarchical processing has been discussed previously in this paper; however, nowhere in this model is the need for structured processing as great as with the handling of perception input. This is due largely to the "low information content" of most perceptual data. That is, perception input usually contains much data and very little information. Therefore, an extraordinary amount of processing overhead is usually associated with perception.

Let's briefly discuss the processing of aural speech data as an example. One of the earliest forms of perceptual input to models of this type will be speech. This seems fairly obvious since most input and output to an information system is, in fact, using language of one form or another. At the present time, the "keyboarding" operation is the most restrictive bottleneck in computer input. And since humans universally employ speech as the most used method of communication, it seems only natural that the most marketable information robot will possess that capacity.

Raw speech contains an immense amount of data. In fact, that actual data collected during speech perception consists of the detailed description of the vibrations induced into a receptor by ambient sound waves. Examination of this data on an oscilloscope screen illustrates that a single spoken word might result in thousands of data points depending on the fineness of time sampling. Analysis of such a group of observations using typical mathematical procedure could swamp a large computer and consume much more actual time than the original speech.

So how does the human counterpart accomplish such a feat? The human ear is an ordinary aural receptor that provides approximately the same electrical data as does a microphone. So if the receptors are approximately equal, then the processing methods, the software, must make the difference.

Research in neural processing has determined that the human system, in fact, makes heavy use of hierarchical processing for perception. In simple terms, one part of the brain is monitoring the vibrations of the "eardrum", reducing those vibrations to a simplified internal pattern, while another part is comparing those patterns to thousands of other patterns stored in memory, while yet another higher-level part of the brain is examining the associated meaning of each recognized word in context, judging the "odds" of certain words having been correctly recognized and substituting other words, in some cases fashioning a form of verbal jigsaw puzzle.

Clearly, the central processing unit of our simulation model cannot take the time to scan waveforms of input aural data. Nor can it take the time to do pattern recognition work against the dictionary of stored word-waveforms. It is even improbable that the central processing unit could have the time to deal with fully translated word strings. Instead, complete or nearly-completed "thoughts" (packets) will be developed by the appropriate perception processor and bussed to TSAC for internal distribution.

Moreover, it seems evident that such a perception module must contain sufficiently complex memory to retain a running continuity of input and output, and, to deal with "images" of various forms depicted by packets retrieved from central memory. It is unlikely that aural cognition can be accomplished to any acceptable degree without establishing a linguistic event data structure coupled with image memory.[5] In simpler terms, this means that we cannot decipher language input very well unless we know what has just been said and have retained a running cognition thereof. Also, input language streams must be immediately associated with internal "images" of various types and in certain cases create new images during analysis.

In fact, the state of the art in speech input has advanced to the point at the time of this writing such that a subservient mini-computer can serve as a speech input device to a larger computer system. A company is now marketing a Nova mini-computer with 16K words of memory that can understand a vocabulary of about 200 pre-stored words of natural language. Further, this system is capable of adapting to the speech habits of an individual user such that after a certain amount of learning time, the cognition delay times tend to decrease for input from that person.

The adaptation of such a system to this model for aural input processing would be reasonably straightforward. First, the aural processor must format its output into the packets required by this model and transmit them over a data bus to TSAC. Secondly, the aural processor must have the capability of receiving packets of information back from TSAC containing upgraded and more complete data. (Some times the apparent meaning of some input string will change after analysis by the full model.)

---

[5]Earl Hunt, The Memory We Must Have

Nearly the same situation exists for incorporation of a slave processor for visual information. Various industrial robots are being manufactured which have the capability of performing analysis of video input and by pattern recognition methods are able to identify and track various objects within its field of vision. An interface to a higher-level information robot can be established along the same lines as just described for the aural input processor.

In all this discussion, a simple two-level hierarchy of processing has been assumed. In fact, for early experimentation and simple applications, a two-level approach may provide adequate results. However, it seems clear that as greater speed and more sophistication is desired, at least one and perhaps more levels of processing must be installed between the lowest level receptor processor and the highest level of "thought" processing (TSAC).

3.6 Memory System

The structure and organization of the Memory System was chosen to achieve a number of goals and to obtain at least satisfactory compromises with certain others. Certainly the most important goal is that the system provide adequate storage, organization, indexing and retrieval functions to allow the various Processors to do their jobs. Moreover, the Memory System must be sufficiently generalized to allow dynamic reorganization, immediate storage of unfamiliar types of information and to provide standardization in programming and access methodologies.

The design chosen must be capable of being implemented in a reasonable time frame using conventional computers and system software. A reasonable level of effort for implementation of the entire Memory System was arbitrarily set at two to three man-years of programming. Furthermore, an ideal condition would be for the design approach to use discrete modules which could be separately implemented and used so that some experience and satisfaction could be derived earlier in the work. Another more practical factor suggests a modular approach to memory: Discrete memory modules can be processed by discrete processing units which allows for multiprocessing and other parallel approaches; while the first trials will certainly be performed on a single central processing unit (CPU), multiple-CPU environments are surely an early step to realistic simulation of consciousness.

In a more technical vein, each of the different memory banks exhibits certain characteristics which suggests different types of storage cells, different organization and accession methods, and different interface with the system Processors. These considerations together with the overall logical requirements of a cybernetic Memory System led to the selection of a multiple-memory approach with discrete Organizer/Accessors for each bank. While certain conventions have been established that apply to all banks in the system, much of the organizational logic and cell structure is individualized for a particular bank.

STORAGE CELL CONVENTIONS

The basic unit of storage in the Memory System is called a neuromime; it "mimes" the function of a neuron. It can also be referred to as a "cell" of memory and will be analogous to a "record" in standard computer jargon. In fact, it will be stored as a variable-length logical record capable of being blocked or spanned over the physical blocks of its storage medium. Such records will typically be accessed as indexed-sequential or even random-access records dependent upon the size and structure of the particular memory unit. A virtual memory technique might be considered for much of the work although the machinery and software for implementation of that technique would restrict work to the few installations where it is presently available.

The structure of a neuromime in this system has been reduced and simplified to a fraction of the complexity of the neural units found in nature. There are two basic parts to our neuromime: the node and the elements. Every neuromime has one and only one node; a neuromime may have any number of elements. An element is a unit of information that concerns or bears some relation to its node. Although an element is treated as a unit of information, it may be a complex piece of information, an array of information or connect to many associated pieces of information. In this manner, neuromimes may be <u>cells</u> of information storage, they may be links in very complicated networks of those cells, or they may act as pointers and pathways for retrievals and decision-making processes.

The node is a relatively simple part of the memory scheme. It is the "central connecting point" that is used to locate and identify all of the pieces of information contained in the attached elements. The node contains discrete identifying information that distinguishes it from any other node in the memory. Depending upon the particular memory containing it, a node will have certain standard control information needed for storage functions, setting up keys for retrieval, relocation information for use during reorganization, and other characteristics of importance to the data processing aspects. All nodes within a particular memory will have a standard format; node format will not be reorganizable by the system controlling the memory although the system will be capable of forming, relocating and purging nodes as it sees fit.

The elements attached to a node constitute the major information storage capabilities of a neuromime. A neuromime may contain <u>any number</u> of elements logically and practically limited only by the amount of mass storage available. However, a reasonable limit must be established to cope with the limited storage capabilities of contemporary computing machinery and the possibility of "runaways" where the system attempts to store large amounts of irrelevant or erroneous data caused by unforeseen (unforeprogrammed) situations or software errors. One very practical limit to neuromime size would be the track size or sector size of the disk or drum device used for physical storage; any overflow caused by the addition of more elements can be linked to an auxiliary node in another sector or track. The linking concept will be discussed in more detail.

Neuromime elements (nelts) consist of four basic parts and appear the same throughout all memory banks in the system. The typical nelt is shown by this diagram:

IDENT	TYPE	SIZE	DATA..................

The identification entry is usually a form of <u>name</u> or may contain classification data to distinguish it from other nelts or groups of nelts. In some cases the identification entry may point to a larger and more complicated entry inside the data section of the nelt, usually where the identification data is too large.

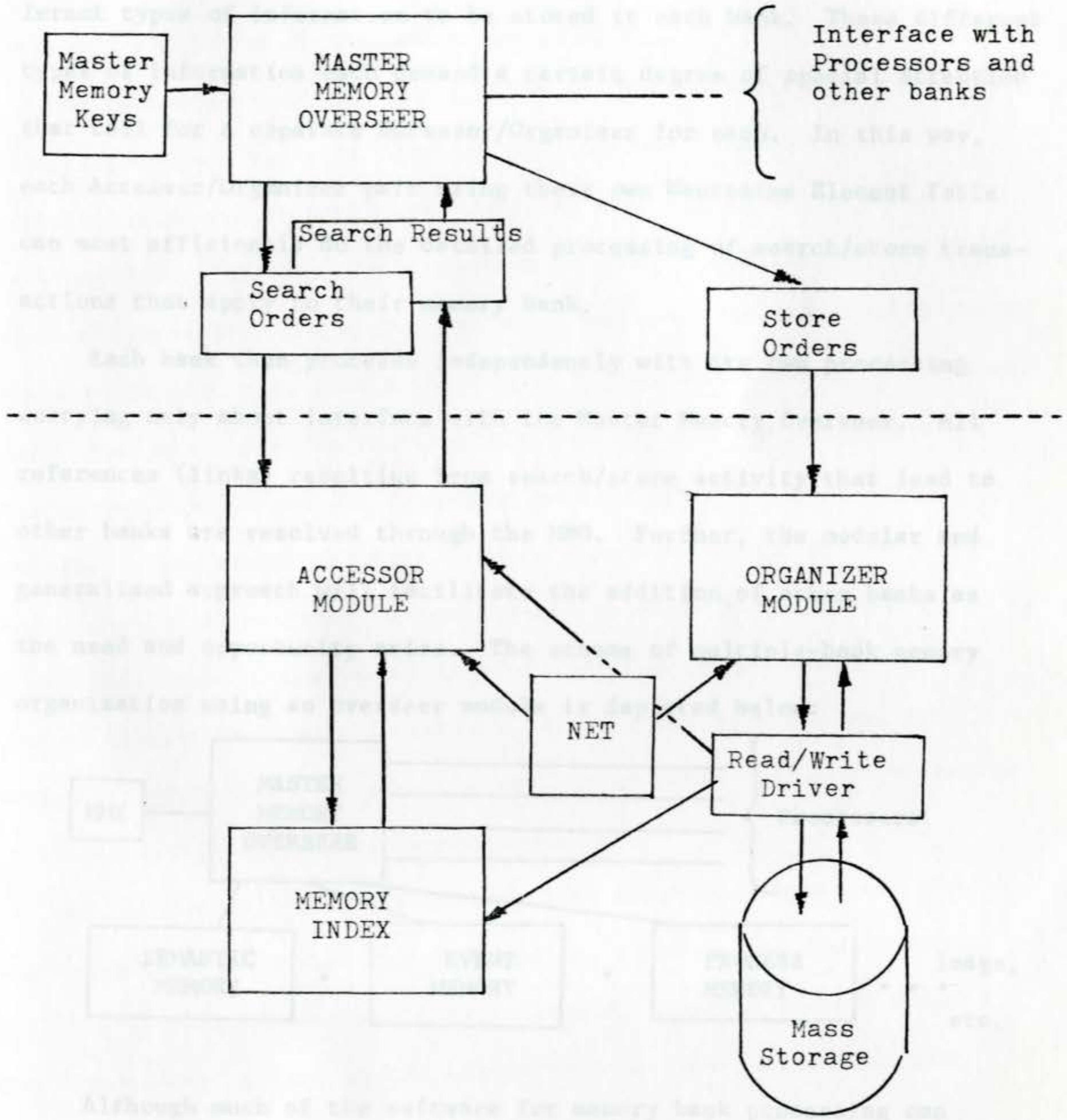
The "type" entry is necessary to allow the accession software to properly decode and process each nelt. Within each memory bank there is a Neuromime Element Type Dictionary (or NET List) that contains one entry for each NET code appearing in that particular memory. The NET entry contains a definition of the content and meaning of the corresponding nelt with format and decoding information needed to process the nelt. It is important to understand that the <u>data</u> contained in a typical nelt is useless without the key information provided by the proper NET entry.

The <u>size</u> entry contains an integer count of the number of characters (of bytes, words or bits depending upon the physical device) contained in the data section of the nelt. The most important use of this entry is by the processing software for scanning and stringing nelts during a Fetch/Decompose or the inverse, a Compose/Store operation.

The data entry is the true information-bearing portion of a neuromime element. It may contain a range of different information from a single data item (such as a single integer) to more complicated things such as arrays of number, time series, cross-indexing or associative links and many more. The key to decoding and using information in the data entry of any nelt comes from the associated NET entry.

All of the individual banks within the Memory System have very similar organizations. Each uses approximately equivalent software components and each bank utilizes a Read/Write driver module to perform the actual input/output functions providing a certain degree of device-independence. Also, all transactions passed among the banks themselves for search and storage processing and also the transactions passed between the Memory System and the various Processors are all governed by the control and format contained in the Neuromime Element Table for the appropriate bank and corresponding entries in a central table called the Master Memory Keys (MMK). The MMK tables are within the domain of the Master Memory Overseer (MMO) which serves as a "task order clerk" go-between passing information back and forth from the Processors to banks within the Memory System.

The following schematic illustrates the organization of a typical memory bank and its interface with the MMO and thereby with the entire system:

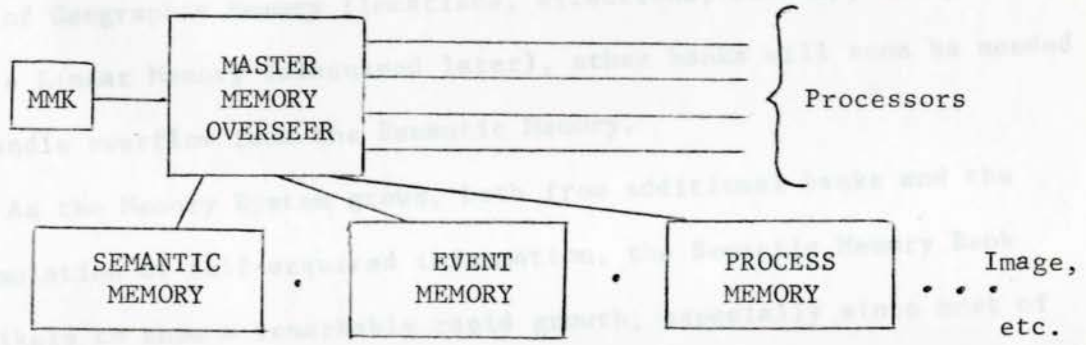


Memory Bank Organization

35/23

As discussed earlier, the Memory System is organized in a modular manner for a number of reasons. The primary technical basis for individual banks lies in the characteristics and general form of the different types of information to be stored in each bank. These different types of information each demand a certain degree of special attention that call for a separate Accessor/Organizer for each. In this way, each Accessor/Organizer pair using their own Neuromime Element Table can most efficiently do the detailed processing of search/store transactions that apply to their memory bank.

Each bank then proceeds independently with its own processing worrying only about interface with the Master Memory Overseer. All references (links) resulting from search/store activity that lead to other banks are resolved through the MMO. Further, the modular and generalized approach will facilitate the addition of other banks as the need and opportunity arise. The scheme of multiple-bank memory organization using an overseer module is depicted below:



Although much of the software for memory bank processing can be duplicated for use in more than one bank, many functions and special processing considerations must be accounted for in each bank. This is particularly true in the construction of the NET's for each bank. Then the initial loading of data will consume a considerable effort and probably lead to several iterations of reprogramming and reloading. It is here that the modular approach to memory processing and organization will pay the most noticable benefit. Each bank can be separately programmed and loaded in the order shown above.

The Semantic Memory is very similar to "dictionary-based" text processing systems and will provide the "language" capabilities and vocabularies needed to bring the remaining memories into operation. The Event Memory can then be brought up so that in addition to cataloging and recalling words, the system can store and use time-state-space information. The next most likely bank to be implemented would be the Process Memory that would provide the capability of storing and using problem-solving information and would thereby provide the first significant increase in the artificial intelligence possessed by the system. From there, any one of a number of banks might be defined and installed. In addition to an Image Memory (vision, pattern recognition), some kind of Geographic Memory (locations, directions, bounds), and perhaps a Linear Memory (discussed later), other banks will soon be needed to handle overflow from the Semantic Memory.

As the Memory System grows, both from additional banks and the accumulation of self-acquired information, the Semantic Memory Bank is likely to show a remarkably rapid growth, especially since most of the "experience" gained by the system will be from interaction with humans. Therefore, certain additional banks may be programmed that will assume various parts of the word-related processing. The auxiliary banks can then be automatically loaded from the Semantic Memory using rules given to the two associated Accessor/Processors. For example, one of the first auxiliary banks might be a Person Memory which would specialize in storing nodes containing all of the data accumulated about one person with whom the system had contact; the primary key in such a node would naturally be that person's name. Since all such data will initially be stored in the Semantic Memory in ready-made neuromimes, the memory-to-memory transfer will be simplified.

Another example would be the storage of information about physical objects; Semantic Memory will be burdened with the storage of neuromimes with long nelt strings describing objects it has perceived in the real world. At some point in future development all such information should be separately stored in a Physical Memory with only the language-related data in the Semantic Memory.

4.0 Internal Structure and Implementation Philosophy

A project as ambitious as the implementation of such a model as described in this paper cannot be accomplished as a single programming task. An attempt to program this model as one integrated computer program would meet with certain failure. Designing a computer program one tenth the complexity of this model without subdividing the work into logical and functional units would create an unmanageable situation with an unlimited number of opportunities for errors. Further, as an unstructured system, the errors that do occur will be found only by laborious searching and analysis of page after page of computer printouts.

In fact, the most reasonable approach to full or partial implementation of this model appears to be a multi-level structure of routines within modules, modules within programs, programs within subsystems, and finally, subsystems within the full model. The primary philosophy of implementation is that each entity in the structure is an independent functional unit having its own input requirements, processing criteria and available output products. Whatever the level of processing, every operational entity has a clearly definable and verifiable function it must perform. The theory is, whenever a particular function fails one knows immediately which entity is responsible. Furthermore, a failing functional unit can be replaced, repaired or reprogrammed without affecting the rest of the system.

The "plug'in" or modular approach to system design and software programming has become the substance of study and attention by a major segment of the computer community and is generally known as "Structured Programming". Unfortunately, Structured Programming has been somewhat overworked and overdeveloped into a fairly rigorous and esoteric discipline. The effect of this rigor has been disdain by a large part of the computer programming community for structured philosophy and techniques, even though many benefits may be had therefrom.

For example, Structured Programming purists maintain that a proper computer program should be written without a single GO TO statement (an unconditional branch). Practical programmers often disagree, citing degradation in processing speed and obfuscation of meaning for many situations where a simple branch would perform much better. Instead, a somewhat middle-of-the-road philosophy is proposed for programming complex functional systems which extracts most of the chief benefits of Structured Programming techniques but which does not unnecessarily hamper efficient programming. In effect, this programming philosophy calls for structured programming without a few of the minor points, the most notable of which is the requirement for GO TO-less coding.

In other areas, the programming philosophy proposed herein goes much further than dealt with by the S.P. theorists. The most notable of these problem areas is the <u>size</u> and <u>scope</u> criteria for modules and the methods of module intercommunication. While such areas are admittedly too vague to yield to hard analysis, guidelines are proposed which will produce workable, reliable software; further advancement in the field of S.P. is not within the scope of this paper nor a goal of this project.

5. Robot Philosophy

Humans seem to be well-suited to dealing with vague generalities. They have developed the ability to transform general and sometimes vague rules into instantaneous and specific actions. A human's general set of morals, ethics and philosophies are readily applied in most cases to new situations and the human will "do the right thing." Certainly, robot builders will have difficulty duplicating or emulating this facet of the human.

Many of the seemingly simple decisions made by a human being in a real-life situation actually require a very complicated set of rules and values to arrive at the right decision. On the other hand, one inherent property of all computers is that they require explicit instructions in order to properly complete their programs. This means that every case, every possibility must be in some way accounted for and the appropriate programmed instructions provided.

An incomplete or ambiguous program generally results in the computer's inability to process the program further, or, simply an incorrect result. Whenever a human encounters an incomplete or ambiguous instruction, he generally applies what we know as "common sense" to the problem and often follows through to a correct and desirable result. As robot builders, we are faced with the large task of defining, describing, and writing specifications for that "common sense" element of our machines <u>plus</u> all of the explicit logic which must be built into a robot's control programs.

We are drawn to the conclusion, therefore, that robotic "philosophy" is very different in nature from human philosophy.

Philosophical statements and concepts, as we humans know them, are of little use to a robot processor. Instead, the robot will require a detailed, tedious and explicit specification of any and all philosophical principles which are to be installed within him.

One very clear example of this difference in definition and specification of philosophy is provided by the so-called "first law" of robotics[1] which is stated, "a robot shall never harm a human, or by its own inaction, allow a human to be harmed." From the human point of view, this is a very simple and clear commandment. However, implementation of this law into reliable and executable programmed instructions is no simple matter, as we shall see.

Remember, the robot consists of a rather complicated array of information processing devices. Any of the modules examined individually is no more complicated than any other computer system; it is the combined network of many hierarchical processors that develops a machine into the level of artificial intelligence. Therefore, since each of the processors (or processing centers) is no more than a typical microcomputer, each and every instruction and rule to be built into a robot must be programmable and expressible in the instruction set of each such module.

In order to instruct the appropriate processing modules that the total system "shall never harm a human", we must first define the terms "human" and "harm" in some way using the respective instruction set. Or even just define the terms "human" and "harm" in <u>any</u> terms that are logical and let the programmer worry about translating those terms

[1]Isaac Asimov, <u>I, Robot</u>, (Doubleday and Company, Inc., 1950).

to the appropriate computer code.

To a reader who is a non-programmer, these considerations may seem picayune. However, to an implementer of robotic philosophy, these matters constitute the major developmental barrier. In a machine language, how do we define "human" and what constitutes "harm"?

Examination of the problem leads us to the conclusion that logically we are chiefly limited by the perceptual capabilities of a robotic system. So fundamentally, we must derive an acceptable method (computing algorithm) for determining the presence of a human using the available perceptual input. Working backwards, we must develop the necessary perceptors to ensure adequate input information so that humans can be perceived with an acceptable degree of accuracy.

Further, certain characteristics and properties of "humans" must be stored in such a fashion that a robotic system will be able to compute exactly what constitutes "harm" to the human in his presence. Presumably, such matters as temperature extremes, radiation limits, allowable acceleration factors, requirements for ambient atmosphere and many other factors would be needed by the well-tempered robot to avoid harming a human within his range.

As a starting point, we should probably program all unconstrained robots to presume that any object possessing <u>any</u> of the attributes of a human <u>is</u> a human until proven otherwise. Then as other information is collected, the robot may disqualify certain non-human objects as he is able.

The reasoning behind this approach is necessary because of safety requirements and the uncertainty of identifying human presence. For example, anything generally shaped like a human would be tentatively tagged as human; store mannequins, photographs and other robots, even a robot's own mirrored reflection would be falsely tagged human until other criteria were applied.

Further, anything moving would be initially identified as a living object and subsequently subclassified as more data is processed. Here, however, is an indefinite attribute: the human might be sleeping, unconscious, or simply motionless for some reason. Also, any object emanating audio waveforms in the general pattern of human speech would be initially classed as human. If other disqualifying data is perceived, the robot could then quit worrying about the presence of such things as radios, televisions and tape recorders. Conversely, lack of speech waveforms would not disqualify a potential human since some humans are incapable of speech or simply have nothing to say to a robot.

Other data will assist in perception of human life. Certainly infra-red sensors will aid in the identification of objects of approximately 36 degrees Celsius. Although, a person just in from a blizzard or just out of a sauna might be slightly out of temperature specifications. Very sensitive sonic perceptors may have the capacity to identify the human cardiac waveform and to track human respiration. Clearly, multiple perceptors are needed to identify the presence of a human being with any degree of reliability. Then the robot must constantly track and update information on potential human beings and be capable of "changing his mind" either way when it becomes apparent that an earlier decision was in error.

In summary, a robot must contain a program that embodies the first law expressed in his internal, processable language in order to be safe in the presence of human while operating unconstrained. The specifications for such philosophy/logic might be written like the following:

1. Scan all objects within range of sensors at some reasonably rapid time interval, applying a unique identification to each object for processing, retaining x,y,z coordinates of each object and the corresponding sensor readouts for each.

2. Evaluate each object identified in step 1 according to the following algorithms to derive a numeric probability score that each is a human:

   a. P_n = maximum pattern match score found after comparing all stored pattern recognition masks of the human form, on a scale from 0 to 1 where .9 indicates virtually positive identification.

   b. T_n = 1.0 when sensed temperature of object n is 36 degrees, a lesser value for objects less than or greater than the mean bodily temperature of homo.

   c. V_n = 1.0 when audio waveforms from direction of object n match stored characteristics of homo voice sounds; else a zero

   d. M_n = 1.0 when apparent motion has been sensed for object n; else zero

   e. An additional series of evaluation terms for selecting attributes associated with the presence of humans.

   f. H_n = 1.0 when object n has been previously identified as human; else a zero.

   g. $S_n = C_1P_n + C_2T_n + C_3V_n + C_4M_n \ldots + c_mH_n$ where S_n is a probability score that object n is, indeed a human. The coefficients c_1 thru c_m are heuristically variable to allow the system to "fine tune" itself over time.

3. Compare each S_n to a value X and a value Y, each of which is heuristically variable by self-correction over time.

4. Any object n, whose score S_n is greater than X is designated as a human.

5. Any object n, whose score S_n is greater than Y is designated as a possible human and is to be treated as human until its S_n falls below Y.

The foregoing specification represents a large number of machine language instructions. Further, the specification is very sketchy, preliminary and must certainly be refined and expanded to adequately process a wide variety of possible situations. A fully-operational subsystem based on the specification above, sampling his sensors at .5 to 1.0 seconds intervals, would swamp the processor of the typical contemporary minicomputer. If all the perception processing were performed by hiearchically-subservient processors, the above spec would still use 30 to 50% of a fairly decent minicomputer's capabilities.

The preceding "philosophy" merely determines what information and what logic the robot uses to determine the presence of a human. The specification to ensure that the robot does nothing to harm a human is several times as large. And the specification to prompt the robot to act to avoid human injuries from other sources would most likely be several times the size of the first two combined. In terms of processing requirements, a basic implementation of the first law of robotics will consume the processing capabilities of six to ten hierarchically-arrayed microprocessors, each with the processing capacity of a typical minicomputer of the mid-1970's. This hardware would be <u>dedicated</u> to its assigned function and not available for other processing tasks within the system.

Clearly, the imposition of these philosophies necessary for safe and proper operation of an unconstrained robot will become a significant overhead burden, adding to the cost and possibly decreasing his efficiency. For example, a robot working in a room crowded with people will spend much of his processing time worrying about bumping into someone or stepping on their toes. Robots working with high-powered tools, welding equipment, dangerous substances and the such would be even more "distracted" by the presence of humans and the necessity to protect them from harm.

Not harming someone seems a simple enough task for us as humans, but for the robotic processor much data processing and computation is necessary. First, a rather large table of data must be stored defining the properties of Homo sapiens and the environment required to sustain his life. Among this information must be such items as the temperature extremes he can withstand, the atmospheric pressure he requires, the latitudes of gas mixtures he can breathe, acceptable radiation levels, acceleration limits for his frail body, light levels his ocular receptors can stand, sonic levels his aural sensors cannot exceed plus a long list of substances that are toxic, offensive or in any way harmful to man.

All of the discussion thus far presumes the application of robots to commercial and industrial working environments. We must assume that such clever devices such as these will rapidly find their way into military applications. Naturally, since we would rather have a robot destroyed than a human solder, cybernetic devices of varying degrees of sophistication will soon be used for many of the riskier tasks associated with national defense and various tactical operations. Even at this time, a number of artificially intelligent devices are on the drawing boards and in various stages of development as part of top secret projects funded by the Department of Defense.

Here we encounter a matter of human philosophy which has not been addressed by our laws nor brought to public attention. What controls should we impose on the robots and their builders to avoid having robots misused? If a robot commits a crime, who is responsible? Once we have programmed the first law into our robot and he knows how to accurately sense human life, he has a complete catalog of all the weaknesses and frailties of humans, only a minor programming change could cause him to seek humans and use whatever tools or weapons he might have to destroy human life. Such a weapons system is very fast, very accurate and difficult to destroy.

Since the robot builders will for the most part be working for their own profits, they will build machines that sell. And so long as there are no regulations or standards in this area, safety standards and philosophies protecting human life will be a matter of ethics of the individual manufacturer and programmer. Without adequate safeguards, robotic devices can and will be wrongfully subverted for criminal uses. Also, the military will make extensive use of robots (perhaps rightfully so) but at the same time creating a breed of machines that constitute a very powerful and dangerous weapon.

6.0 Summary

A general structure has been devised and functional components identified comprising a system that will provide some of the illusion of consciousness in a computer system. This system can be implemented on present day computer hardware and used to enhance the functional capabilities of many types of computer-based mechanisms.

The modular structure of this model provides opportunities to implement certain portions as useful components of other specific data processing applications. Therefore, some of the implementation and testing may be done as a part of other hopefully profitable projects in information retrieval and test processing applications.

Certain conventions and standards have been outlined in preliminary form for the intercommunications between program modules and for the storage of information accessed by numerous modules. Further, overall philosphies have been established for programming correctness, logical structure and operational goals. In the case of mass memory and language input/output, the model has been devised to incorporate one of several existing software alternatives as an interim measure to distribute implementation efforts.

The chief aim in developing a design for this model has been to formulate a component of machine intelligence which has an immediate, pratical value in business and industrial data processing, can be used in conjunction which many existing computer software systems and can be built on a budget as part of some larger implementation effort.

Based on the assumption that these other similar projects are successful, our economy and society are likely to be assaulted with a fresh wave of superautomation for which we are unprepared. Our legal system, which lags far behind the era of ordinary computer automation is likely to fall even further behind the robot age. Our society already feels repulsed by the priorities given to machines and could become strongly alienated to more intelligent ones.

The contemporary computer scientist is caught amidst these currents with little power to prevent the progression of attendant social and economic ills nor to prevent the misuse of his inventions. He will continue to improve his machines, motivated largely by profit. Others will utilize those inventions, also for profit and often with disregard for our economy, our society and even human life.

SELECTED BIBLIOGRAPHY

Aaron, Joel D. The Program Development Process. Reading, Mass.: Addison-Wesley Publishing Co., 1974.

Albus, James S. Peoples' Capitalism. College Park, Md.: New World Books, 1976.

Armstrong, Russell M. Modular Programming in COBOL. New York: John Wiley & Sons, 1973.

Asimov, Isaac. I. Robot. Greenwich, Conn.: Fawcett, 1950.

Beckman, Petr. The Structure of Language. Boulder, Col.: Golem Press, 1972.

Boguslaw, Robert. The New Utopians. Englewood Cliffs, N.J.: Prentice Hall, Inc., 1965.

Brooks, Frederick P., Jr. The Mythical Man-Month. Reading, Mass.: Addison-Wesley, 1975.

Date, C.J. An Introduction to Database Systems. Reading, Mass.: Addison-Wesley, 1976.

Dijkstra, Edsger W. A Discipline of Programming. Englewood Cliffs, N.J.: Prentice-Hall, 1976.

Jackson, Philip C. Introduction to Artificial Intelligence. New York: Petrocelli, 1974.

Kernighan, Brian W. and Planger, P.J. The Elements of Programming Style. New York: McGraw-Hill, 1974.

McGowan, Clement L. and Kelley, John R. Top-Down Structured Programming Techniques. New York: Petrocelli, 1975.

Martin, James. Design of Man-Computer Dialogues. Englewood Cliffs, N.J.: Prentice-Hall, 1973.

Meadow, Charles T. The Analysis of Information Systems. New York: John Wiley & Sons, 1967.

________. Applied Data Management. New York: John Wiley & Sons, 1976.

Meyer, David E. and Schvaneveldt, Roger W. "Meaning, Memory Structure and Mental Process." Journal of the American Association for the Advancement of Science 4234, (April 1976).

Myers, Glenford J. Reliable Software Through Composite Design. New York: Petrocelli, 1975.

Ornstein, Robert E., ed. The Nature of Human Consciousness. San Francisco: W.H. Freeman, 1973.

Shank, Roger C. and Colby, Kenneth Mark, ed. Computer Models of Thought and Language. San Francisco: W.H. Freeman, 1973.

Shannon, Claude E. and Weaver, Warren. The Mathematical Theory of Communication. Urbana: University of Illinois Press, 1949.

Weinberg, Gerald M. The Psychology of Computer Programming. New York: Van Nostrand, 1971.

Yourdon, Edward. Techniques of Program Structure and Design. Englewood Cliffs, N.J.: Prentice-Hall, 1975.