A MODEL FOR THE CONTROL MODE MAN-COMPUTER INTERFACE DIALOGUE

By Roy L. Chafin

Jet Propulsion Laboratory
California Institute of Technology

SUMMARY

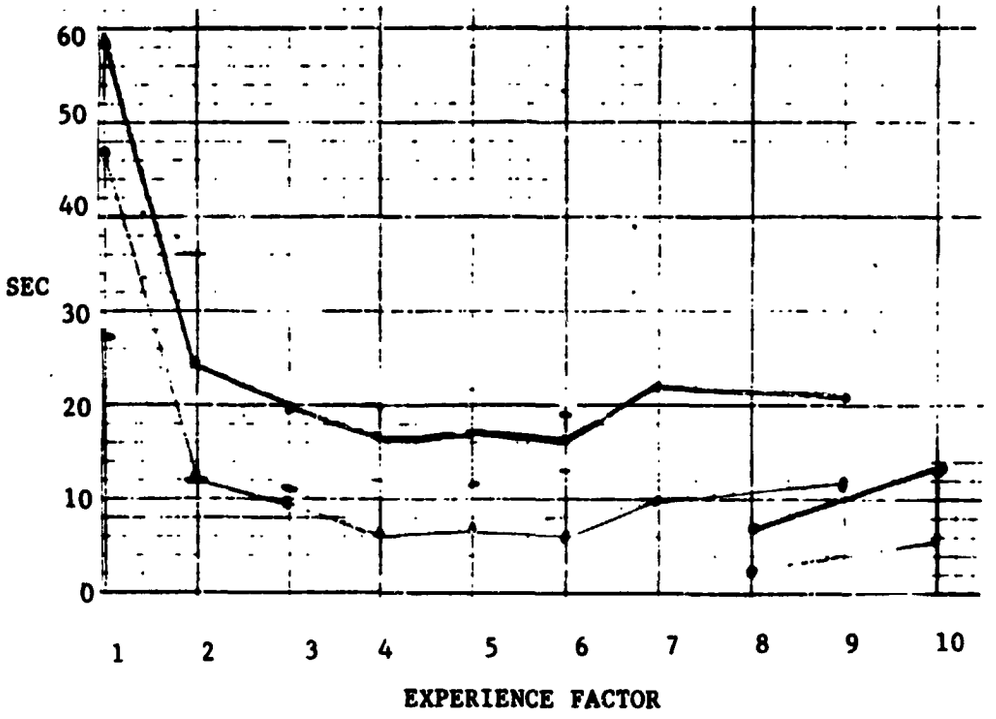
A four stage model is presented for the control mode man-computer interface dialogue. It consists of context development, semantic development, syntactic development, and command execution. Each stage is discussed in terms of the operator skill levels (naive, novice, competent, and expert) and pertinent human factors issues. These issues are human problem solving, human memory, and schemata. The execution stage is discussed in terms of the operators typing skills. This model provides an understanding of the human process in command mode activity for computer systems and a foundation for relating system characateristics to operator characteristics.

INTRODUCTION

Computer systems have two basic modes of operation, the control mode and the data mode. In the control mode, the operator controls the system by commanding it to take specific actions. For a telemetry system, it might be to acquire a specific data stream. For a teleoperator system, it might be to extend the arm and pick up an object. For a text editor system, it might be to delete some portion of the text or to place the text in a specified file. In the data mode, the operator is either entering data into the system or retrieving data from the system. For example, after a text editor has been commanded to accept text for insertion into a specific location, the text to be inserted is entered. That is the data entry mode. Or for a Data Base Management system, a data request is entered in the control mode and the data is presented to the operator in the data retrieval mode. This paper is concerned with only the control mode.

The concepts discussed in this paper are the result of reflections on data taken from a human factors experiment performed in the Deep Space Network (DSN) at the Jet Propulsion Laboratory, a NASA facility (1). The experiment was a man-computer interface test with approx. 100 operators from the DSN. The subjects were given a series of tasks on a CRT display of a simulation computer. They had been randomly assigned one of four command formats, single argument mnemonic, multiple argument mnemonic, prompt, or menu. They entered the command format into the keyboard to accomplish the task. Their solution (the command) was displayed on the CRT for feedback. It was also timed and recorded on disc for subsequent data reduction. Fig. 1 illustrates the performance time of one of the formats for a specific type of task.

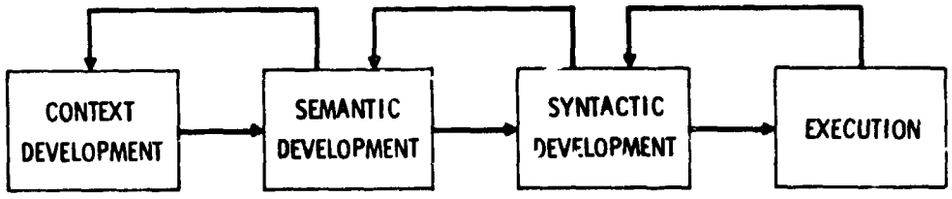
668



Experimental Performance Time

Fig. 1

The vertical dimension is performance time, the time required to complete the task. The horizontal dimension is the number of times this specific type of task has been attempted, it is an experience factor for the specific command format and the specific type of task. The first time this type of task was attempted required an average of 59 sec to complete the task. For the sixth attempt, the average task completion time was 16 sec. The first six attempts were consecutive. The seventh and nineth attempts were separated from tasks of the same type by a number of different type tasks. The eight and tenth attempts were partial tasks and are not of interest in this discussion. What is interesting is the increased time required to complete a task when it has been separated from previous tasks of the same type (ie. 6th at 16 sec and 7th at 22 sec). At least two explanations can be offered for this performance differential. One is that the subjects have forgotten because of intervening tasks. The other is that each task requires a context to be developed and subsequent same task allows the subject to keep the preceding context. Intervening tasks require that the subject change the context and that requires some time. The lower curve is the time required to enter the first character of the command. It represents the time required to compose the command, that is the think time. It produces the principle variance in the overall performance time. The time required to actually execute or type in the command is the time between the two curves. A four stage model (Fig. 2) is proposed to represent the total time required to generate or compose a command in the control mode. It can be used to explain the experimental performance time as typified in Fig. 1.



Control Mode Operator Model

Fig. 2

The first stage is the context development, the operator's definition of the domain of relevant information for the specific tasks being addressed. The second stage is the semantic development, the understanding of the factors and relationships which apply to the command generation. The third stage is the syntactic development of the command, the actual codes and symbols which make up the command. The fourth and last stage is the execution of the command, typing it into the keyboard and verifying its operation.

HUMAN FACTORS ISSUES

Several human factors issues interact in the control mode model. A very important issue is operator skill level. At some skill levels the command development is basically a problem solving exercise. At other levels it is basically a memory exercise. At some skill levels, the command development is a cognitive process and at other levels it is an automatic process(schema). This section presents an operator taxonomy based on skill levels and discusses human problem solving, memory, and schemata.

Operator Taxonomy

We intuitively understand that operators do not all have the same capabilities and skills, however much of the literature and most applications do not take operator variablility into account.

Operators vary over many dimensions. Eason (2) uses a "kind of user" taxonomy of clerical, manager, and specialists. Clerical users are principally data entry operators. Managers are principally data retrievers. And specialists use computer systems as a tool to accomplish some specific job. Bennett (3) divides users into those who are committed by their jobs to using the computer system and those whose computer use is discretionary. Similarly, Codd (4) divides users into those casual users who infrequently use the system and those dedicated users who frequently use the system. We would expect the manner in which they most effectively use the system to be different.

These taxonomies are related to how the user makes use of the system. Another interesting dimension is skill levels. Eason (5) also considers naive users who use the system as a tool but that do not have a deep knowledge of

the system. He implies that they are relatively unskilled. Hiltz and Turoff (6) suggest a four phase user skill development from their experience with computer conferencing systems. The user initially approaches a system with uncertainty. He progresses to a stage of insight when he understands the general concepts of the system. The incorporation phase is when the mechanics of the system interaction become second nature, a part of the users normal environment. And saturation occurs at some point in their experience.

The Hiltz and Turoff four phase skill development taxonomy can be generalized by considering that these four phases or stages can be static as well as dynamic. If a user is a casual operator, he may never develop beyond the insight stage. The skill level may be a function of the kind of system and the application tasks as well as a transitory development phase. To provide a generalized operator skill taxonomy, skill levels will be defined for naive, novice, competent, and expert operators.

Naive operators are those who have essentially no understanding of the system. They must rely on external assistance (either other users, trainers, or documentation) in order to use the system.

Novice operators are those who have a general but not a specific understanding of the system. They know what the system does but typically not how to operate it. They still need external assistance but of a different kind. They need a demonstration of how to operate the system.

Competent operators are those who understand the system and can use it effectively. Their knowledge of the system is sufficient for them to determine the actions required to control the system, primarily a cognitive process. They do not require external assistance beyond possibly an occasional reference to the user manual to refresh their memory.

Expert operators are those who know the system so well that they do not have to think about the control actions, their actions are automatic.

Problem Solving

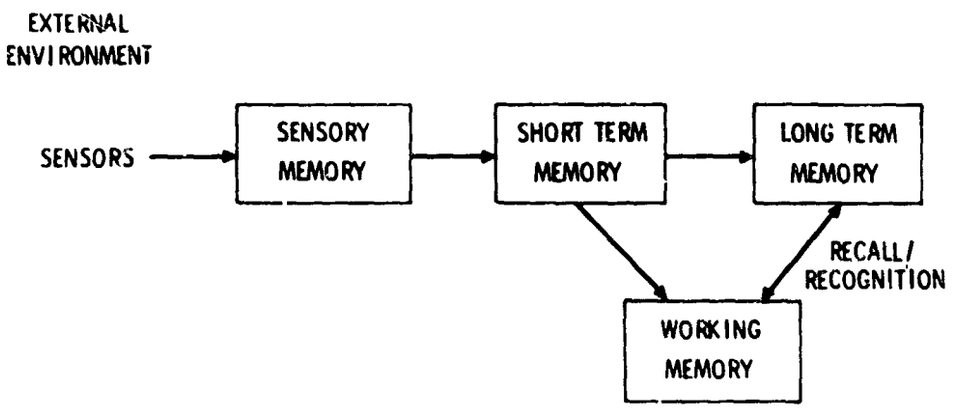
Problem solving is the process of creating a solution to a given problem. Over past years there have been many problem solving models presented (7). They tend to represent the originators perception of the process and the specific types of problems being solved. The problem solving models range from 4 to 9 stages. Osborn (8) suggests a 7 stage generalized model from which we will select a 5 stage problem solving model that is appropriate for the development of the various stages of the control mode model.

* Preparation - gathering the pertinent data.
* Analysis - breaking down the relevant material.
* Hypothesis - piling up alternatives by ways of ideas.
* Synthesis - putting the pieces together.
* Verification - judging the resultant ideas.

The pertinent data is gathered from the problem statement, the documentation, and from the operator's previous experience on similar problems. The analysis identifies the relevant elements so that the relationships between them can be evaluated. A number of the most reasonable alternative relationships are selected for further consideration. Synthesis puts the elements together within the relationships hypothesized, and selects the one best solution. That solution is evaluated against the problem statement (or task) and the documentation (or the remembered items from the documentation). Any discrepancy causes iteration back to previous stages in the problem solving process.

Memory

Shniederman (9) refers to a 4 element human memory model. He used the model in terms of defining the programming process but it is also very useful in the development or generation of commands in the man-computer dialogue. The model is presented in Fig. 3.



Human Memory Model

Fig. 3

The external environment is seen, heard, felt, etc., through sensors into the sensory memory. Information is stored in the sensory memory for a very short time, a matter of only fractions of a second. For example, an image of a printed page would be stored in the sensory memory. A part of the information in the sensory memory is passed onto the short term memory where it is held for a few seconds. For example, a set of characters will be selected from the page image in the sensory memory, interpreted, and stored in the short term memory. Short term memory seems to fade significantly after some 5 to 20 seconds, unless it is rebuilt by a process called rehearsal. To hold information in short term memory for longer periods of time, the individual concentrates or reiterates that information, that is, rehearses it. In addition to time limitations, it is also capacity limited. Short term

memory is limited to about seven units of information (10). Many of the operator errors that are experienced in system operations are due to system demands which exceed the individual's short term memory capability. Examples of this problem are the commands that require 8 to 12 digit frequency numbers (example, 2202786.012 Hz). A method of handling the bigger numbers or groups of characters is available by grouping the characters into meaningful combinations, each of which is easy to accomodate. This is called "chunking". It is illustrated by the common representation of telephone numbers. Even though there may be up to 15 digits in a telephone number (area code,number, & extension) it is grouped into 3 or 4 digit chunks, each with a specific meaning (area code, exchange, line or instrument, and extension number).

Again, some part of the information in the short term memory is passed to the long term memory where it is held permanently. There seems to be no limit on the amount of information stored in long term memory. Two retrieval methods from long term memory have been suggested (11). One is recall, in which information is recalled directly from memory. The other is recognition, in which the information that cannot be obtained from direct recall can be recognized when matched with some external sensing. For example, an operator can recall a command mnemonic for a command that is used frequently, but cannot recall (or remember) the mnemonic for a command which is used infrequently. However, he can scan a list of mnemonics and readily recognize the correct one when he sees it.

Working memory is the area used in processing the task or problem. In the context of generating commands, it is the area where the problem solving activity occurs. The working memory receives information from the external environment through the sensory and the short term memories and from the long term memory directly. Parts of the command development process is stored back into long term memory to be used in subsequent command generation activities.

Schemata

Another important concept is the automatic actions of an expert operator doing a repetitive routine task. It is called a schema. As Zipf (14) has suggested in his studies on the use of language, people tend to use the minimum effort in accomplishing tasks. When a task has become repetitive and routine an individual develops a scenario or schema which he can use to accomplish the task without thinking about it (15). Once the task has been identified and the proper schema triggered, the individual goes through the actions automatically. He doesn't have to consider and think through each action in the schema, and his decision requirements are minimized. An example that is familiar to all of us is driving to work. After driving the same route for several years, we can drive it without thinking. We do not have to decide where to turn, how fast to go, where to slow down, how fast to take the curves, etc., we do all these things automatically.

When an expert operator on a system uses schemata, his work load is decreased and his performance is increased. Of course, we have to be careful that the basic action sequence does not change without our realizing it. That

would make the schema inappropriate and would lead to errors and incorrect results. A common cause of operator errors with "He is one of our best operators" is due to the operator triggering a schema without realizing that the task has subtly changed. System operating procedures can be error prone when the system does not provide adequate clues to the operator that the task is different from his routine task.

OPERATOR MODEL

The command mode operator model consists of the context development stage, the semantic development stage, the syntactic development stage, and the execution stage.

Context Development

Context is an essential element in human discourse. Without a mutual understanding of some context our speech would be hopelessly long, involved and difficult to follow. As human communicators, we typically assume a context based on our understanding of the other person, the situation, and past history. This usually works, however, a more interesting situation is when we do not have these clues and we have to use our skills to develop the context under which we will carry the conversation. Our conversational success then depends upon these skills. An advantage in context development between humans over man-computer context development is the flexibility that both sides of the human conversation can bring to the process. The typical computer system is very constrained in this issue. Although there are some exotic programs coming out of the artificial intelligence field in which the program participates in the context development (16), the programs which are developed for most applications demand that the operator develop the context.

Grosz (17) defines a domain of discourse. Without the ability to focus on the subset of knowledge relevant to a particular situation, the amount of knowledge overwhelms even the simplest system. The process of defining this domain of the discourse and to limit the knowledge base needed for a particular application is what we will define as context development.

Context development is the successive narrowing of focus from the general to the specific. Context development then is a selection process. It uses either problem solving or memory depending on the skill level of the operator. Naive operators have no experience to provide memory capability, so context development is very much a problem solving process. The system characteristics which aid this process are a well structured man-computer interface (MCI) design and knowledge aids. The MCI structure is most effective when it requires a minimum of selection at any hierarchical level, and each selection is well identified. Understanding must be developed at each selection, therefore, the system documentation becomes extremely important. As the

operator's skill increases, he operates more on understanding and memory. At the novice level we would not expect the memory to be extensive, it would most likely give clues to the problem solving process. Because the understanding is somewhat greater, the external documentation is of less value. The more important MCI characteristic is cognitive simplicity (18). Cognitive simplicity is the use of internal MCI aids to help the novice operator's understanding.

The competent operator has the experience that allows him to use his memory for the context selection process rather than going through a problem solving process. The expert operator has developed schemata to accomplish the context development. All he requires is to identify the situation and he will trigger the appropriate schema. The competent and expert operators require an environment which is very straight forward. They would prefer to go right to the context rather than going through a series of levels or stages. This requires a different MCI organization than for naive or novice operators.

Normal system operation requires a change of context or focus as the system sequences through its tasks. As the task which the system has to accomplish changes, the context of the MCI also has to change. This situation is very similar to the starting operation context development with two additional steps involved. The first step is recognition, the operator has to recognize that the task has changed and that the context must then change. The second is evaluation, he must evaluate where he is and where he has to go. For naive and novice operators with highly structured MCI's the process must work in reverse until they navigate back to a level which allow them to go forward again. An MCI design which caters to naive and novice operators must take particular care to provide for this need. MCI's designed for competent and expert operators do not have this problem, they can and prefer to go directly to the new context.

Semantic Development

After the context has been established, the next stage is semantic development. Command semantics is the knowledge of how the command relates to the task that it is supposed to accomplish. A command consists of a function select and possible arguments to satisfy the required parameters for that function. Some examples:

1. PUMP
2. PUMP/1,ON

The semantic knowledge associated with "PUMP" in #1 is that it controls the pump and turns it on. The "PUMP" in #2 refers to more than one pump and an argument is required to select the desired pump (ie. 1). Also, the function select "PUMP" in #2 can turn the selected pump on or off, so an argument is required to determine whether the pump is to be turned on or turned off. This semantic knowledge is independent of the command style, that is, whether it is mnemonic, prompt, or menu style. This is the semantic development required of the operator, he must understand that part of the system that he is attempting to control.

The naive operator must use a problem solving process for semantic development. Without experience, he has nothing to go on. Three things will help the naive operator to develop the semantic understanding. One is good documentation. That is, documentation or user manuals that allows the operator to rapidly find the function (context) and explictly defines the function select and the arguments. This would not likely be the Theory of Operations section of the manual because it tends to be too general and too bulky. It would most likely be the Operator Instructions section of the manual because it tends to be more direct. Second, the system can be designed to have compatibility with the operator's previous experience. Compatibility is a technical term in human factors which means that a process is what a person expects it to be. His expectations may be due to past experience with similar systems or to a more natural connection such as a car steering wheel turning right for a right turn. Compatibility is a powerful way to help the operator's semantic development. Third, the MCI can be designed to be "User Friendly", which seems to be a buzz word for a menu driven system. Menu systems tend to help the operator in context development because it leads him through the choices. It is helpful in the semantic development if it is sufficiently explicit, however, this tends to produce menus which are very wordy. Another characteristic of a menu system that is helpful to a naive operator is that, at any level, all the choices are available to the operator. He may not have to understand the function completely if he is able to correctly differeniate between the choices (like guessing on a multiple choice test). But again, the menus must be explict or a high error rate will be experienced.

A novice operator has more experience to draw on. His semantic development will most likely be a combination of memory recall and documentation referal. Even when using the documentation, he will very likely be using the recognition memory mode, he will scan the manual and recognize the command when he sees it. Prompt and menu MCI formats are appropriate at this skill level. The menus can be less explict and less wordy at this level. In fact, they should be less wordy or they will become unactractive.

The competent operator tends to work primarily from the recall memory mode. And the expert tends to operate from his schemata. For both skill levels, the so called "User Friendly" MCI's are not really friendly. They tend to be too long and involved to be comfortable. These operators tend to lose patience with prompts and menus because their own pacing is faster than the pacing of the MCI.

Syntactic Development

Shneiderman (9) points out that syntactic knowledge is the second kind of information stored in long term memory. He also points out that it is more precise, detailed, and arbitrary, he also suggests that it is more easily forgotten. Sachs (19) supports this suggestion from work in recognition memory for syntactic and semantic aspects of sentences. The meaning of sentences is much easier to remember than the exact syntax. Of course, in human discussion the meaning is important not the exact syntax. Although, philo-

sophically, the same comment applies to man-computer communications, ie. it is the meaning that is important, practically the limitations on the "understanding" capability of the computer increases the inportance of exact syntax considerably.

In an MCI, the syntax is the specific codes and symbols used to specify the command and its arguments, the punctuations or delimiters, and the structure that ties these elements together. The naive operator must determine the syntax from the documentation or from the display for prompts and menus. His task is basically problem solving. The problem is to determine the proper syntactic elements to implement the semantic development. Aids to the novice's syntactic development are things that tie into the semantic content or are compatible with prior experience. This is the attraction for so called natural language MCIs, they are supposedly compatible with human communication syntax. Menus are appropriate for naive operators because they minimize the syntactic elements that he has to create, he only has to choose between the elements presented to him.

By definition, a novice operator has been exposed to the syntax of an MCI. He would most likely operate in a problem solving mode for some syntactic elements, from recognition memory for others, and he may be able to recall other elements. And as he progresses in experience he develops the capability to recall more of the syntactic domain. The aids which are important to the naive operator are still useful to the novice operator. He will most likely refer to the documentation and would respond favorablely to menus but he is more able to operate independent of these aids. The documentation that he uses would most likely be the Operator Instructions rather than the Theory of Operations sections and as he becomes more proficient he would prefer quick-look MCI tables. Cognitive simplicity (18) becomes more important for the novice operator because he is often operating from the recognition memory mode.

The competent operator will be working from the recall memory mode and as such doesn't need the recognition memory aids or the problem solving aids which are so useful to the naive and novice operators. He will use the documentation infrequently. He would prefer process simplicity (18) because it is easier to execute. Process simplicity is the concept of minimizing the execution effort. Prompting and menus may be tolerated but they are inappropriate and he would prefer the more straight forward mnemonic command style. He would prefer the increased control which he has with the menemonic command formats.

The expert operator will be working from schemata. The problem solving and memory aids are inappropriate for him, in fact they are undesirable because they interfere with schema development. They tend to make the schema longer, more involved, and harder to execute. The expert operator is likely to actively dislike a prompt or menu command format because they get in his way (3). They would prefer extreme process simplicity.

A complicating factor is that very often an individual operator's skill varies greatly over the entire syntactic domain of a system. He may be an expert in the commands and functions that he uses often, and naive in the others that he doesn't use so often. The challenge for the system designer on an extensive system is to gracefully handle this extreme skill range.

Execution

The execution phase of the model is the actual execution of the command. Card's (20) keystroke model was for expert users performing routine tasks. This model is concered only with the command entry and does not include any data or text entry. Card provided an excellent discussion of the elements of the command execution and this paper will not discuss these items further. A generalized command mode model must include a broad range of tasks and operator skills. Once the operator has developed the syntactic representation of the command, the skill differential is mainly in typing capability. Operators can be skilled typists or unskilled typists. A task which contains considerable data or text entry usually demands a skilled typist but one which is primarily command entry only can use either skilled or unskilled typists. The principle difference between skilled and unskilled typists is the use of the touch system for the skilled versus a hunt and peck system for the unskilled typists. Another difference is in the need to look at the keyboard when typing. The skilled touch typist can maintain his attention on the task while entering the command, he doesn't have to divert his attention from the task to look at the keyboard. The unskilled typist must take his attention away from the task and apply it to selecting the keys on the keyboard. Changing attention requires the use of short term memory to hold the information. The unskilled typist operator's performance is influenced by his short term memory limitations whereas the skilled typists are not operating under this limitation. The short term memory limitation causes the operator to have to chunk information into small groups, a process which can be error prone if the MCI was not designed to accomodate it.

Task and typing skill interact to influence the operator performance. A task which demands the operators continous attention will suffer with an unskilled typist and should have a skilled typists. A task which allows the operator to compose the command syntax in his head and does not require continous attention would be appropriate for either a skilled or unskilled typist operator.

Some observations from the typing tests of the DSN Human Factors experiment (1):

1. The performance difference between trained and untrained typists is approximately 1:2.

2. Random characters are more difficult to type than English text. This supports other findings in the literature (21).

3. The trained typist's speed for a different number of random characters in a group is quite constant. This supports the position that they do not have to take their attention away from the task.

4. The untrained typists speed decreased for increased number of random characters in a group (over the range 3 to 9 characters per group). This supports the short term memory influence on untrained typing performance.

5. The above comments apply to alphabetic characters only. When the full range of ASCII characters were allowed in the random character groups, whether the operators were trained or untrained was immaterial. Apparently typing training doesn't handle numbers, punctuation symbols, or other special symbols very well.

This execution model which includes a variable for trained/untrained typists suggests that consideration for operators typing skill should be included in the MCI design. If the expected user population contains a high proportion of untrained typists, short term memory limitations should be seriously considered in the MCI design. It also suggests that, regardless of the typing skill, the MCI should be designed with commands that are familiar to the operator rather than what might seem like a collection of random characters.

CONCLUSION

A four stage model of the control mode command generation process has been presented. It consists of the subelements of the cognitive process of composing the command and physical element of the execution of the composed command. The composition subelements are the context development, the semantic development, and the syntactic development.

The value of this model is in the understanding of the human process that it gives to the system designer when designing the system's man-computer interface. It provides a foundation for relating MCI characteristics to different operator skill levels. We would expect that matching these characteristics (man and machine) would provide systems that are "easy to use", have few errors, and have better user satisfaction.

REFERENCES

1. Chafin, R.L.; and Martin, T.H.:DSN Human Factors Project Final Report, Jet Propulsion Laboratory, Calif. Inst. of Technology, 4800 OakGrove Drive, Pasadena, Calif.,Contract No. 955013, Task Order No. RD-142, 1 Nov. 1980.

2. Eason, K.D.; Damodaran, L.; and Stewart, T.F.M.:Interface Problems in Man-Computer Interaction, Human Choice and Computers eds E. Mumford and H. Sackman, North-Holland Publishing Company, 1975.

3. Bennett, J.L.: Incorporating usability into System Design: the Opportunity for interactive Computer Graphics, Proceedings of the International Conference on Cybernetics and Society, Tokyo, Japan, Nov. 1978.

4. Codd, E.F.: Seven Steps to Rendezvous with the Casual User, Data Base Management eds J.W. Klimbe and K.L. Koffeman, North-Holland Publishing Company, 1974.

5. Eason, K.D.: Understanding the Naive Computer User, The Computer Journal, Vol. 19, No. 1, pp 3-7, Jan. 1976.

6. Hiltz, S.R.; and Turoff, M.: The Network Nation, Human Communication via Computer, Addison-Wesley Publishing Company, Inc., Advanced Book Program, Reading, Massachusetts, 1978.

7. Sackman, H.: Preliminary Investigation of Real-World Problem Solving with and without Computers, Volume II: Complete Results, R-1205/1-NSF, Rand Corp., Santa Monica, Calif. 90406, April 1973.

8. Osborn, A.: Applied Imagination, Charles Scribner's Sons, New York, 1954.

9. Shneiderman, B.: Software Psychology Human Factors in Computer and Information Systems, Winthrop Publishers, Inc., Cambridge, Massachusetts, 1980.

10. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information, Psychological Review, Vol. 63, pp 81-97, 1956.

11. Loftus, G.R.; and Loftus E.F.: Human Memory: The Processing of Information, New Jersey, Lawrence Erlbaum Associates, Publishers, 1976.

12. Cheriton, D.R.: Man-Machine Interface Design for Timeshare Systems, Proceedings of the Annual Conference, Association for Computing Machinery, New York, 1976.

13. Watson, R.W.: User Interface Design Issues for a Large Interactive System, Proceedings of the National Computer Conference, Vol.45, pp 357-364, 1976.

14. Zipf, G.K.: The Principle of Least Effort, Addison-Wesley Press, Inc., Cambridge, Massachusetts, 1949.

15. Oldfield, R.C.:Memory Mechanisms and the Theory of Schemata, The Cognitive Process Readings, Harper,R.J.C., et al. eds., Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1964.

16. Hendrix, G.G.: Human Engineering for Applied Natural Language Processing, Proceedings of the 5th International Joint Conference on Artificial Intelligence, pp 183-191, 1977.

17. Grosz, B.J.: The Representation and Use of Focus in a System for Understanding Dialog, Proceedings of the 5th International Joint Conference on Artificial Intelligence, pp 67-76, 1977.

18. Chafin, R.L.: Simplicity in Command and Control Systems: A Human Factors Considersion, Proceedings of the 1980 International Telemetering Conference, 14-16 Oct. 1980, San Diego, Calif., 1980.

19. Sachs, J.S.: Recognition Memory for Syntactic and Semantic aspects of Connected discourse, Perception and Psychophysics, Vol. 2, No. 9, pp 437-442, 1967.

20. Card, S.K.; Moran, T.P.; and Newell, A.: The Keystroke-Level Model for User Performance Time with Interactive Systems, Communications of the ACM, Vol. 23, No. 7, pp 396-410, July 1980.

21. Schoonard, J.W.; and Bores, S.J.: Short-Type: A Behavioral Analysis of Typing and Text Entry, Report RC 4434, July 16, 1973, IBM Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, New York 10598.