AD 713716

Professional Paper 18-70
June 1970

Rational vs. Empirical Approaches to Job/Task Descriptions for COBOL Programmers

by

Felix F. Kopstein

Presentation at the
Computer Personnel Research of the
Association for Computing Machinery
University of Chicago
Chicago, Illinois June 1969

DDC
RECEIVED
NOV 5 1970
B

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
Springfield, Va. 22151

20050405089

This document has been approved for public release and sale; its distribution is unlimited.

HumRRO

HUMAN RESOURCES RESEARCH ORGANIZATION

16

The Human Resources Research Organization (HumRRO) is a nonprofit corporation established in 1969 to conduct research in the field of training and education. It is a continuation of The George Washington University Human Resources Research Office. HumRRO's general purpose is to improve human performance, particularly in organizational settings, through behavioral and social science research, development, and consultation. HumRRO's mission in work performed under contract with the Department of the Army is to conduct research in the fields of training, motivation and leadership.

The contents of this paper are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

ACCESSION for
CFSTI WHITE SECTION ☐
DDC BUFF SECTION ☐
UNANNOUNCED ☐
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
DIST. AVAIL. and/or SPECIAL

Published
June 1970
by
HUMAN RESOURCES RESEARCH ORGANIZATION
300 North Washington Street
Alexandria, Virginia 22314

Prefatory Note

This paper is based on research performed by the Human Resources Research Organization, Division No. 1 (System Operations) at Alexandria, Virginia, under Work Unit IMPACT, Prototypes of Computerized Training for Army Personnel. M. H. Gotterer and R.J. Seidel provided many constructive comments during the preparation of this paper by the author.

The paper was presented at the Seventh Annual Conference of the Special Interest Group, Computer Personnel Research of the Association for Computing Machinery, held at the University of Chicago in June 1969.

RATIONAL VS. EMPIRICAL APPROACHES TO JOB/TASK DESCRIPTIONS FOR COBOL PROGRAMMERS

Felix F. Kopstein

The context and background for this paper involve the development of a computer-administered instruction (CAI) course designed to produce competent COBOL programmers for the U.S. Army. The considerations discussed in the paper obviously transcend COBOL as subject matter and training development as goal, but both are used to provide an illustrative framework. Because the primary concern is with technique in the sense of *practical* (not formal) methodology, only hypothetical (simulated) empirical data will be presented. It seems reasonable and proper to protect the privacy of a client (i.e., the U.S. Army), whenever actual empirical data add nothing that is indispensable.

The terms "rational," in the sense of pure deductive reason, and "empirical," in the sense of pure inductive reason, are contraposed here as they were in 18th-century philosophy. The contraposition is one of formal methodology rather than of current practice. In searching through case histories of job/task analyses, it might be difficult to find a pure example of either approach.

Empirical Approach

Viewed historically, job analysis began mainly in relation to personnel selection and vocational adjustment. This orientation is reflected in earlier writings (e.g., Burtt, 1) where job analysis is described as "closely related to job specification or occupational description." In Burtt's view, "the analysis is the means and the specification the end." Even he, however, recognized other purposes including the improvement of work methods and effective methods of training.

During the 1940s and 50s, concerns seemed to shift progressively toward the two latter purposes. The increasing technical sophistication, especially of military equipment, led to a preoccupation with job/equipment design in terms of the optimal allocation of functions to man and machine. Similarly, there arose a preoccupation with training design, including the design of training equipment, so as to provide a functional context (though not necessarily physical congruence) identical to the job environment (Crawford, 2; Shoemaker, 3). With the advent of programed instruction, the concept of terminal behavioral specification was widely accepted. Providing such specifications prior to an instructional program design required the stipulation of what behavioral capabilities were to be developed. In turn, this implied a requirement for precise specifications of what set of behavioral capabilities was, in fact, needed on the job.

During this period, *job* analysis gradually shifted toward *task* analysis, suggesting a more fine-grained analysis carried on within tasks qua components within a job grouping. In task analysis there is also a shift of emphasis from requisite aptitudes to requisite achievements, and progressively from molar forms of behavior to molecular forms and their successive levels of integration (Gagne, 4).

In a training development context (e.g., a CAI course in COBOL) empirical job/task analysis is normally applied in the way described here. In order to decide what ehavioral capabilities the training must develop in the trainees, the capabilities required in the job context need to be ascertained. Thus, surveys of the jobs are conducted. Sampling and survey techniques themselves, though important, are not of immediate relevance to the issues of this paper. In essence, surveys show the outcomes (products) that the incumbent's on-the-job performance is to engender, the resources from which he is to produce them, and the circumstances under which all of this is to take place.

Next, this information is converted into behavioral terms. Every identifiable descriptive statement concerning on-the-job functions is analyzed with respect to the behaviors requisite for them. Classification of behaviors can be according to any of several schemes (Cotterman, 5; Stolurow, 6; Gagne, 4) so long as consistency is preserved. Component statements in the specification of terminal behaviors can now be grouped, and classified as to requisite performance level, and frequencies of occurrence can be tallied. Usually there is at least a tacit assumption that the higher the frequency the greater the criticality. Ultimately, all of this information guides the design and development of the training content.

Before turning to the rational approach, two points that are usually true can be noted. First, the empirical approach samples given jobs as they are, without questioning such matters as manpower resources for filling them, optimality of job structures, job-function allocations, and so forth, or considering their interdependence in a job-family model. Second, job samples are taken, for all practical purposes, at a given fixed time; there is no sampling over time. Thus, potential changes in equipment, manpower supply, available training time, and so forth, are not assessed in the samplings and are unlikely to enter into further consideration. In short, the pure empirical approach cannot assure that all possibilities have been covered.

Rational Approach

Task analysis produced an interesting variant, that is, task/equipment analysis (e.g., Demaree *et al.*, 7). In task/equipment analysis there is an attempt to deduce the behavioral requirements for its operators and maintainers from the functional characteristics of equipment—often from specifications and before the first prototype has been built (e.g., Haggard, 8).

The significant point is the fact that there is no body of actual experience that can be examined to establish what most incumbents do

on the job, what capabilities seem to be essential, and the frequency with which certain tasks or task components occur. For example, if the equipment in question were a digital computer, it would not be known in advance that a design flaw would frequently produce conditions under which resistor "x" on card "y" will fail. Consequently, it could not be pre-established that recognizing a given malfunction syndrome and relating it to resistor "x" on card "y" would be a critical job or task requirement. What could be pre-established, however, and with perfect assurance would be the *possibility* of resistor "x" failing and producing certain symptoms. What can only be established *a posteriori* is the *probability* with which resistor "x" or card "y" will, in fact, fail. Both, incidentally, are independent of criticality, or to what degree the continued functioning of the equipment under consideration is contingent on the malfunction.

The cybernetic view is an alternate base from which to develop rational approaches to job/task descriptions. To do so it is necessary to view the setting within which any given job exists as a process of energy/information transformation. Within that total process there is some subprocess, or subsystem, that is wholly or partially regulated and controlled by the incumbent's functioning on the job. What regulatory and control functions must be exercised to maintain the subprocess or subsystem in question within specified constraints? The question can be viewed as a design problem (e.g., Inaba, Wulff, and Kopstein, 9; Simon, 10). It is necessary to design the set of control functions which a control subsystem—that is, the incumbent on the job—must exercise to meet the design criteria. In other words, what behavioral capabilities must the job-incumbent qua control subsystem possess? If he does not already have them, these are the capabilities that a training process must instill.

Again, let me note that a pure *a priori*, rational derivation of behavioral capabilities requisite to a given job must treat them all as equiprobable. It cannot discriminate a capability whose exercise will be required once in a million times or less from one that will be, practically speaking, always required. Of course, experts might estimate about these matters, but that amounts to an approximation of empirical data. On the other hand, a scale of *criticality* can be developed quite easily.

Rational vs. Empirical Approach

To clarify the contrast between the purely rational and the purely empirical approach, the difference might be put thus: The purely rational approach will develop an *exhaustive* set of the behavioral capabilities requisite for a certain job or task constellation. This is illustrated in Figure 1. It shows the capabilities as discrete elements (ε) in the total set, that is, some hypothetical job or task. These elements are subscripted merely for identification; no order is connoted, since it would have to be based on some relation (e.g., precedence, inclusivity, criticality) defined on the set. If such a relation were defined, the dashed line in Figure 1 might be regarded as an abscissa.

Schematic Representation of Behavioral Capabilities Within Some Job/Task Set

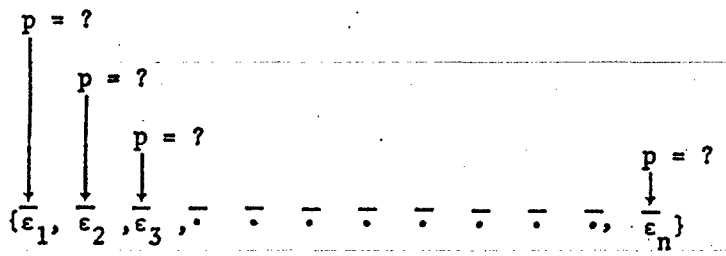


Figure 1

However, there exists no information from which an ordinate reflecting either frequency or probability of occurrence might be constructed.

The purely empirical approach will develop a set of behavioral capabilities together with associated frequencies of occurrence (probabilities), as illustrated in Figure 2. It may be immediately apparent that elements (behavioral capabilities) in the set are *not* exhaustively enumerated. They represent a subset that includes some (possibly all) of the elements in the total set. It is readily possible to order the elements in this subset by frequency. However, such an ordering must not be taken as equivalent to relative "difficulty" in any sense of that ambiguous term.

Schematic Representation of Frequency of Occurrence of a Subset of Behavioral Capabilities

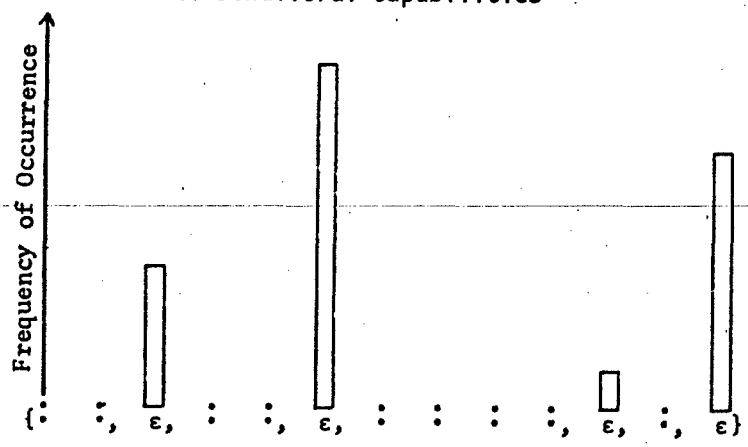


Figure 2

Combined Approach

So far, empirical and rational approaches have been characterized briefly and abstractly. To illustrate the methodology combining

both approaches, reference will be made to the COBOL programmer's job.

The basic task of the COBOL programmer, or any programmer, is to control the flow (i.e., sequences of transformations) of certain sets of data by means of a program. The course of that flow links the data input with the data output, and so determines the transformation of the machine input state to the machine output state. The question now is the range of control schemes that the COBOL programmer might be called upon to devise. In other words, what sort of input may be converted to what sort of output and in what ways? We shall return to the last point later and consider, for the moment, the I/O transformation.

"Transformed" means a change in form and signifies that the input data form is changed to the output data form. Change, of course, can affect only that which is changeable or variable. Hence, the variable characteristics of input and output had to be established. This was done intuitively and reflectively by the author (cf., Schaeffer *et al.*, 11) while consulting with several knowledgeable individuals. Gradually, a list emerged; it was reviewed by two experts in computer science and the final version appears in Table 1.

A clarification of the entries in the margin of this matrix follows. To begin with, information input (data) may be thought of as having a location in space and/or time. For example, the information to be processed may be stored on punched cards, or magnetic tapes, or on disks. In the same sense, information may be stored on a document that can be scanned by a magnetic ink character reader or an optical character reader. Possibly it may be generated in real time on a keyboard. It could pre-exist in a core area or it might be a timing pulse from a light pen.

Physical spacing is intended to mean the spatial distribution of data in their location. Thus, they may be so tightly packed that for practical purposes no gaps in time or space separate them. If such gaps do exist, they may be uniform, or they may be of varied size. The concept of "blocking" may suggest what is intended here.

Content refers to the symbol set within which information is encoded. Under this heading there may be an inconsistency since blanks may have to be thought of as "non-data." "Garbage" simply means such irrelevant information as might be left accidentally in a storage location that has not been cleared. Data grouping refers to the external meaning of the content. In other words, the file might hold a set of records containing the same information (e.g., name) on different individuals. This would be an example of the homogeneous grouping. By contrast, a file might contain records of various types possibly grouped by individuals. For example, for a given name successive records might refer to payroll data, production records, property charged out to the individual, and so forth. This would be an example of the heterogeneous data grouping. Under "Data Organization" a hierarchical or nested structure suggests that records of individuals might be subsumed under their department, under the plant, and so forth. A non-hierarchical structure simply means that this is not the case.

Table 1

Matrix of Relation T Defined in ‹I, O› and Hypothetical Frequencies of Occurrence[1]

Variable Properties of Input (Data) / Variable Properties of Output

Input \ Output: A. Location	1. Printer-paper	2. Typewriter paper	3. Cards	4. Tape	5. Disks	6. Drum	7. Data-Cells	8. Paper Tape	9. CRT	10. Core area	11. Magnetic Ink Character Document
A. Location											
1. Cards	T		T	T	T	T	T			T	
2. Tapes	T		T	T 15	T	T	T	T	T	T	
3. Disks	T		T	T	T	T	T	T	T	T	
4. Paper Tape	T		T	T	T	T	T	T		T	
5. Drum	T		T	T	T	T				T	
6. Data Cells	T		T	T	T	T	T			T	
7. Magnetic Ink Character Document				T	T	T	T			T	
8. Documents capable of being scanned				T	T	T	T			T	
9. Keyboard	T		T	T	T	T	T		T	T	
10. Core Area	T	T	T	T	T	T	T		T		
11. Light Pen									T	T	

Input \ Output: B. Physical Spacing (Spatial Distribution)	1. None (no gaps)	2. Uniform interstices	3. Varied interstices
B. Physical Spacing (Spatial Distribution)			
1. None (no gaps)	T	T	T
2. Uniform interstices	T	T	T
3. Varied interstices	T	T	T

Input \ Output: C. Content (Symbol Set)	1. Data (meaningful sets)	a. A-Z	b. a-z	c. Mixed upper & lower case	d. 0-9	e. Special Characters	2. Blanks (non-data)	a. In numeric context	b. In alphabetic context	3. Garbage (random or arbitrary sets)	a. A-Z	b. a-z	c. Mixed upper & lower case	d. 0-9	e. Special Characters
C. Content (Symbol Set)															
1. Data (meaningful sets)															
a. A-Z		T		T							T				
b. a-z			T	T											
c. Mixed upper & lower case				T											
d. 0-9					T										
e. Special Characters						T									
2. Blanks (non-data)															
a. In numeric context					T			T							
b. In alphabetic context		T	T	T		T			T						
3. Garbage (random or arbitrary sets)															
a. A-Z		T									T				
b. a-z			T									T			
c. Mixed upper & lower case				T									T		
d. 0-9					T									T	
e. Special Characters						T									T

Input \ Output: D. Data Grouping	1. Homogeneous	2. Heterogeneous
D. Data Grouping		
1. Homogeneous	T	T
2. Heterogeneous	T	T

Input \ Output: E. Data Organization	1. Structure	a. Hierarchical (nested)	b. Non-hierarchical	2. Ordering	a. Random	b. Sequential	(1) Forward	(2) Backward	(3) Mixed	3. Length (0 ≤ N ≤ ∞)	4. Partitions	a. Files	b. Subfiles (Filesections)	c. Records	d. Data fields	e. Sub-datafields	f. Characterspaces
E. Data Organization																	
1. Structure																	
a. Hierarchical (nested)		T	T														
b. Non-hierarchical		T	T														
2. Ordering																	
a. Random					T	T	T	T	T								
b. Sequential					T	T	T	T	T								
(1) Forward					T	T	T	T	T								
(2) Backward					T	T	T	T	T								
(3) Mixed					T	T	T	T	T								
3. Length (0 ≤ N ≤ ∞)										T							
4. Partitions																	
a. Files												T	T	T	T	T	T
b. Subfiles (Filesections)												T	T	T	T	T	T
c. Records												T	T	T	T	T	T
d. Data fields												T	T	T	T	T	T
e. Sub-datafields												T	T	T	T	T	T
f. Characterspaces												T	T	T	T	T	T

Input \ Output: F. Identification	1. Not Reference Tagged	2. Reference Tagged	a. Direct	b. Relative
F. Identification				
1. Not Reference Tagged	T	T	T	T
2. Reference Tagged	T	T	T	T
a. Direct	T	T	T	T
b. Relative	T	T	T	T

[1]Presence of T in cells denotes [illegible] ability of I/O transformation. Numerical values indicate hypothetical frequencies of occurrence.

Clearly the data in a set (or the records in the file) may be randomly ordered (e.g., as encountered). If the ordering is sequential, it may be either forward or backward (or in ascending or in descending order). A mixed ascending and descending ordering is also conceivable. Data sets may range in length from null to infinity. Sets of data of any length may be partitioned in numerous ways. For practical purposes, we might think of the total set as the file that can be further partitioned into major sections or sub-files presumably containing some records that are ultimately composed of character spaces.

Identification means that any item of data or set of data may be indexed or not indexed. For example, a record of cash register entries specifies only the amount and makes no reference to the item purchased nor the cash register used. Thus, there is no reference tag (index) attached to the data item. In other situations such reference tags are attached. For example, the data item may belong to a specific "named" individual, or it might be tagged to an ordinal position—the "third customer." Except for some three items under "location," the list of variable properties of output is identical with that of input. Thus, all explanations pertain to it as well.

The next step is to establish the range of *possibilities*. We may do so by considering first that we have listed in Table 1 two sets that we may call I and O. Taking the elements comprising these sets in ordered pairs, we may inquire whether a relation T holds between them. The relation T is *practically* defined as "the element in I can be transformed to the element in O." The combined and reconciled results of expert judgments[1] are shown in Table 1.

Wherever a "T" appears in one of the cells, the relation "I can be transformed to O" is thought to apply. Close inspection will establish rapidly that T has not been entered in many cells where the logical possibility, at least, cannot be excluded. This reflects the expert judgment that the use of COBOL for such purposes would not be warranted. Of course, disagreements with judgments—no matter how expert—are always possible. Possibly the diagonal entries reflecting transformation from "Garbage" to "Meaningful Sets of Data" may seem surprising. It reflects the consideration that edit programs can accomplish this.

The next step is to establish the distribution of *probabilities* across T or the range of possibilities. This is done by examining a sample of problem specifications obtained as part of a standard survey of jobs held by COBOL programmers. In other words, managers of COBOL programming groups are asked to furnish representative sets of problems handled by their installations. The problems are examined with respect to which ordered pairs of I/O elements are specified, and a frequency

[1] Inevitably, "experts" maintain varying and often contrasting points of view. Reconciliation amounts to the author's best judgment after discussions with his consultants.

tally is readily made. A set of hypothetical results[1] is shown in Table 1. They reflect a small set of representative COBOL programming problems from each of several surveyed installations.

Discussion

Two prefatory comments may be required—first, this paper is addressed to a *practical* methodology with no pretense of its being epistemologically pure. Second, the methodology unquestionably has numerous ramifications; in the time available it is possible to deal with only some of them illustratively.

The first matter for discussion might be labeled the "complexity-reality" issue. The lists in Table 1 attest to the practicality of specifying the functional variables of equipment, or, to put it another way, the variables of the job qua process to be controlled and regulated by an incumbent. Could the same sort of feasibility be demonstrated for the computer repair and maintenance job suggested in the earlier illustration? Is it realistic to list every physical component in a large computer—each wire, resistor, capacitor, transistor, and so forth, and so forth—singly and in combination, and to relate them systematically to a set of to-be-corrected malfunction syndromes? If the problem is viewed in this precise way, it seems likely that it could not be solved in finite time. However, if a more coarse-grained approach is taken, the picture may be very different. For example, feasibility may be reasonably self-evident, if the smallest malfunctioning component is considered to be a subsystem, module, or perhaps even a circuit card. Similarly, some "logical" possibilities can be discussed *a priori* as having negligible probabilities attached to them. For example, a wire not subject to physical stress or current overload—for practical purposes—will never fail. In short, complexity can be whittled down to realistic dimensions.

The next point relates to the use or uses that can be made of the combined rational and empirical job/task analysis data. There are, of course, a great many, but since training provided the context for this paper, it will further serve as an illustratory context. An immediate problem for any training program is to verify the existence of a specified terminal behavior capability. A criterion achievement test must be devised. How can those responsible for devising it be sure that it is a true random sample of the task population within the job? Obviously, by sampling across the *range of possibilities* where, however, some constant has been added so that no possibility has a probability of zero, as illustrated in Figure 3.

The addition of a constant to assure some probability for each possibility immediately suggests a generalization. The combination data amount to a statistical model of a job family, for example, COBOL programmers. Not only is it possible to add constants, but each of

[1]Consistency with obtained results has been preserved.

Hypothetical Distribution of Requisite Behavioral Capabilities

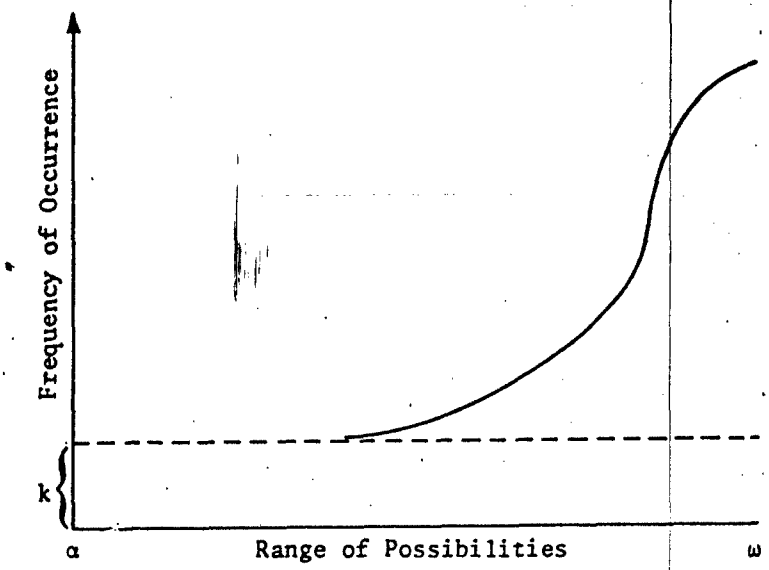


Figure 3

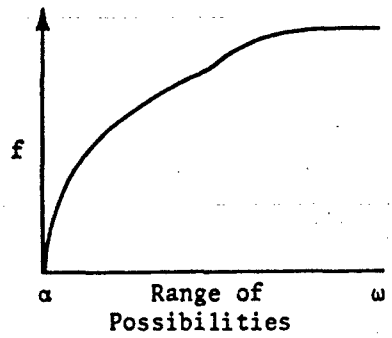
the model's parameters can be altered to study a variety of effects and interactions (Figure 4). For example, one might weight low frequency task components in training so as to favor practice of them (Figure 4a). The effect would be graduates with a high performance reliability in *any* encountered task. Alternatively, one might wish to examine optimal task allocations to job-levels so as to minimize training time and costs (Figure 4b). Here the most numerous "apprentice" level would be trained on (i.e., practice) only the most frequent tasks (albeit to high proficiency), the less numerous "journeyman" level would practice a broader range of tasks, and least numerous "master" level would practice all tasks to high proficiency.

Undoubtedly, many other ways of using this model will suggest themselves. At least one way also provides a return link to the training context. Constructing criterion tests is only one facet of the problem, the other is the construction of a program of training items (or frames, displays, units) that will produce criterion proficiency as quickly as possible. Again, a sampling of the distribution of problem tasks is required. Of course, beyond such a sampling is the issue of the sequential and/or hierarchical ordering of these problem tasks within the course presentation (the training item sequence), but that is outside the scope of this paper.

What is *not* outside our scope is the option of judiciously re-shaping the probability distribution of the job/task content, especially with reference to criticality. This might assure a supply of more sophisticated manpower than available hitherto and engender all of the consequences

Modifications of Job Model Parameters

(a) Weighting Lower Tasks



(b) Segmenting Tasks by Job Levels

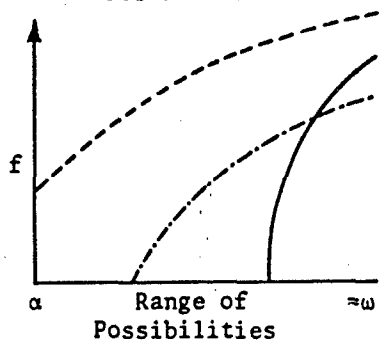


Figure 4

attendant on such a change. At the very least, it could mean that a hypothetical COBOL programmer encountering a most unusual programming problem on the job may retain enough proficiency from his training to cope with it.

LITERATURE CITED

1. Burtt, Harold E. *Principles of Employment Psychology*, Harper and Row Publishers, Inc., New York, 1942.

2. Crawford, Meredith P. "Concepts of Training," in *Psychological Principles in System Development*, Robert M. Gagne (ed.), Holt, Rinehart and Winston, Inc., New York, 1962.

3. Shoemaker, Harry A. *The Functional Context Method of Instruction*, HumRRO Professional Paper 35-67, July 1967.

4. Gagne, Robert M. *The Conditions of Learning*, Holt, Rinehart and Winston, Inc., New York, 1965.

5. Cotterman, Theodore E. *Task Classification: An Approach to Partially Ordering Information on Human Learning, Technical Note 58-374*, Behavioral Sciences Laboratory, Wright-Patterson AFB, Ohio, 1959.

6. Stolurow, Lawrence M. *A Taxonomy of Learning Task Characteristics*, Technical Documentary Report No. AMRL-TDR-64-2, Behavioral Sciences Laboratory, Wright-Patterson AFB, Ohio, 1964.

7. Demaree, Robert G., Marks, Melvin R., Smith, Walter L., and Snyder, Melvin T. *Development of Qualitative and Quantitative Personnel Requirements Information*, Technical Documentary Report No. AMRL-TDR-62-4, Behavioral Sciences Laboratory, Wright-Patterson AFB, Ohio, 1962.

8. Haggard, Donald F. "Training Guidelines for the US/FRG Main Battle Tank," in *Use of Job and Task Analysis in Training*, HumRRO Professional Paper 1-69, January 1969.

9. Inaba, Kay, Wulff, J. Jepson, and Kopstein, Felix F. "A Rational Method of Applying Behavioral Technology to Man-Machine System Design," in *Air Force Human Engineering, Personnel, and Training Research*, Glen Finch (ed.), National Research Council Publication 783, Washington, 1960.

10. Simon, Herbert A. *The Sciences of the Artificial*, MIT Press, Cambridge, Mass., 1969.

11. Schaeffer, K.H., Fink, J.B., Rappaport, M., Wainstein, L., and Erickson, C.J. "The Knowledgeable Analyst: An Approach to Structuring Man-Machine Systems," Air Force Technical Report AFOSR 4490, Stanford Research Institute, Menlo Park, Calif., 1963.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Human Resources Research Organization (HumRRO) 300 North Washington Street Alexandria, Virginia 22314	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

RATIONAL VS. EMPIRICAL APPROACHES TO JOB/TASK DESCRIPTIONS FOR COBOL PROGRAMMERS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Professional Paper

5. AUTHOR(S) *(First name, middle initial, last name)*

Felix F. Kopstein

6. REPORT DATE June 1970	7a. TOTAL NO. OF PAGES 11	7b. NO. OF REFS 11
8a. CONTRACT OR GRANT NO. DAHC 19-70-C-0012 b. PROJECT NO. c. 2Q063101D734 d.	9a. ORIGINATOR'S REPORT NUMBER(S) Professional Paper 18-70 9b. OTHER REPORT NO.(S) *(Any other numbers that may be assigned this report)*	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES Paper for Association for Computing Machinery, University of Chicago, Chicago, Illinois, June 1969 (IMPACT)	12. SPONSORING MILITARY ACTIVITY Office, Chief of Research and Development Department of the Army Washington, D.C. 20310

13. ABSTRACT

The purpose of the paper is to contrast *empirical* approaches deriving from job analysis and *rational* approaches deriving from task/equipment analysis, and to suggest the differences in the information to be gained from each. Job analysis establishes what exactly a representative sample of incumbents do on the job. Task/equipment analysis deduces the behavioral requirements for its operators and maintainers from the functional characteristics of equipment, or from task situations that do not yet exist in actuality. The purely empirical approach develops a set of behavioral capabilities together with associated frequencies of occurrence, but cannot guarantee that the required set of behavioral capabilities will be exhaustively enumerated. The purely rational approach will develop an exhaustive set of behavioral capabilities requisite for certain job or task constellations, but will provide no good way of establishing their probabilities of occurrence. Therefore, a combined approach seems desirable. It is illustrated in the context of a COBOL programmer's job. The use of data from combined rational and empirical job/task analyses for statistical models of job families is discussed. The uses of these models in training design is also discussed.

DD FORM 1 NOV 65 1473

Unclassified

Security Classification

Unclassified

Security Classification

14. KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
COBOL Programmers						
Computers & Computer Programing						
Empirical Approach						
Job/Task Descriptions						
Rational Approach						

Unclassified

Security Classification