Hierarchical & Information Processing Task Analysis: A Comparison

Paul F. Merrill

Brigham Young University

In a recent article, Merrill (1976) outlined an information processing approach to task analysis. He proposed that many of the concepts and techniques used by information processing theorists could profitably be applied to the problem of task analysis in the design of instructional materials. This approach to task analysis was described as being most appropriate for the analysis of tasks which were algorithmic in nature. An algorithm was defined as a procedure or sequence of operations for solving a problem or performing a task which is guaranteed to produce the correct results. A brief comparison between the information processing approach to task analysis and Gagne's (1962) hierarchical approach was also presented:

"The hierarchical task analysis procedure and the information processing analysis both reveal a structure of skills or operations having an ordered relationship to each other. However, the nature of this ordered relationship is considerably different. A hierarchical task analysis will reveal a structure of 'intellectual skills' such that the learning of one skill is prerequisite to the learning of higher ordered skills. Thus, the purpose of hierarchical analysis is to determine an ordered relationship of learning prerequisites. In contrast, an information processing analysis describes the way an algorithmic task is performed and reveals the information processing relationships between operations where the output of one operation is required as part of the input for succeeding operations. Thus, a hierarchical analysis reveals the *prerequisite learning* stages leading to the terminal behavior while an information processing analysis specifies the *performance sequence* of the suboperations of the terminal behavior [p. 10]."

Merrill further suggests that some tasks, especially mathematical tasks, may have both types of relationships. An instructional designer may find it advantageous to analyze such tasks using both techniques in order to obtain a broader understanding of the actual nature of the task and its subskill relationships. Gagne and Briggs (1974) state that deriving a learning hierarchy is not an easy matter and that mistakes can be made unless one attends to the necessity for thinking out all of the component mental operations used by the learner when he solves a particular problem. However, Gagne and Briggs do not provide any guidelines or procedures for making these component operations explicit. One of the purposes for conducting an information processing task analysis is to explicitly identify the component or sub-operations of a given task. Recently, Gagne (1977) has also recommended that an information processing analysis be conducted prior to deriving a learning hierarchy.

The author has found that many instructional designers in attempting to conduct a hierarchical analysis actually use an information processing approach and end up with an information processing analysis. However, they interpret the resulting analysis as if it were a hierarchy and make certain erroneous assumptions about prerequisite learning requirements. These erroneous conclusions apparently are a result of not being aware of the distinction between prerequisite learning relationships and output-input performance relationships. Conducting both types of analysis on a given task would help avoid this type of erroneous conclusion.

The purpose of this paper is to present a more detailed comparison of the infor-

mation processing and hierarchical task analysis procedures by examining the results of analyzing a specific task using both approaches.

Analysis Diagrams

The specific task which will be analyzed using both task analysis procedures is that of subtracting whole numbers. A diagram showing a hierarchical analysis of this task has been published by Gagne and Briggs (1974, page 114) and is reproduced in Figure 1. A flow chart of an information processing analysis of the procedure or algorithm for subtracting whole numbers is shown in Figure 2. This flow chart reveals the component mental operations involved in performing the task.

The specific steps of the subtraction procedure or algorithm may be divided into operations and decisions. The operations are represented by the rectangular shaped boxes in the flow chart while the decisions are represented by the diamond shaped boxes. The decision points are labeled with capital letters while the operations are labeled with numbers. The seventh operation is broken down into four suboperations which are labeled 7a through 7d. The lines and arrows indicate the order in which the various operations and decisions must be executed.

For some subtraction problems, the same series of steps may be performed several times during the solution of the problem. These iterations are referred to as a loop. A loop is a series of steps that lead you back to where you started from. Examples of loops in the subtraction algorithm are steps A, B, 2, C, and 3, and steps 5, D and 6. All steps of an algorithm are not necessarily used in solving a given problem. For example, if we were to follow the steps of the algorithm to solve the problem 7 minus 5, only steps 1, A, B, 2 and C would be used. For a problem such as 47 minus 25, these steps would be used a second time after executing step 3 and going around the loop. In solving any given problem only a specific set of steps are executed. These specific steps might be considered as a path through the algorithm. If we assume that subsequent iterations around a loop do not create a new path, the number of paths is finite. These paths may be identified; and given problems may be categorized according to the path that is used in solving the problem.

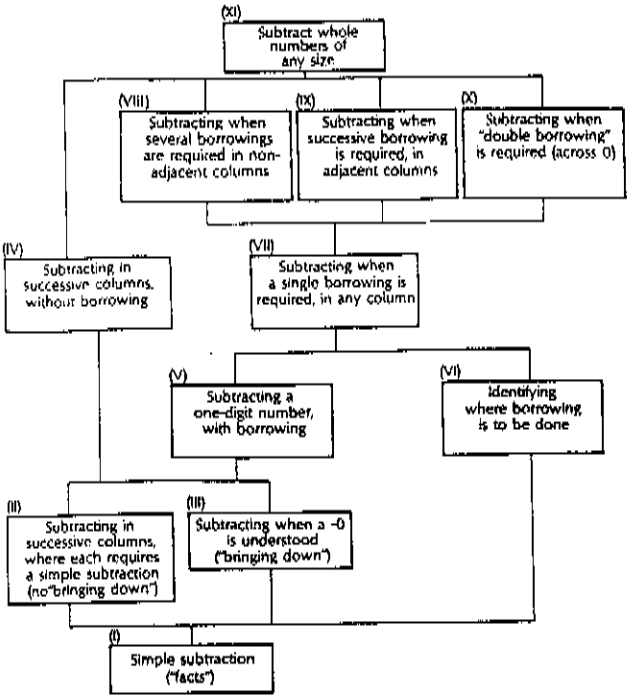

Figure 1. Learning hierarchy for subtracting whole numbers. (From Gagne & Briggs, 1974, p. 114).

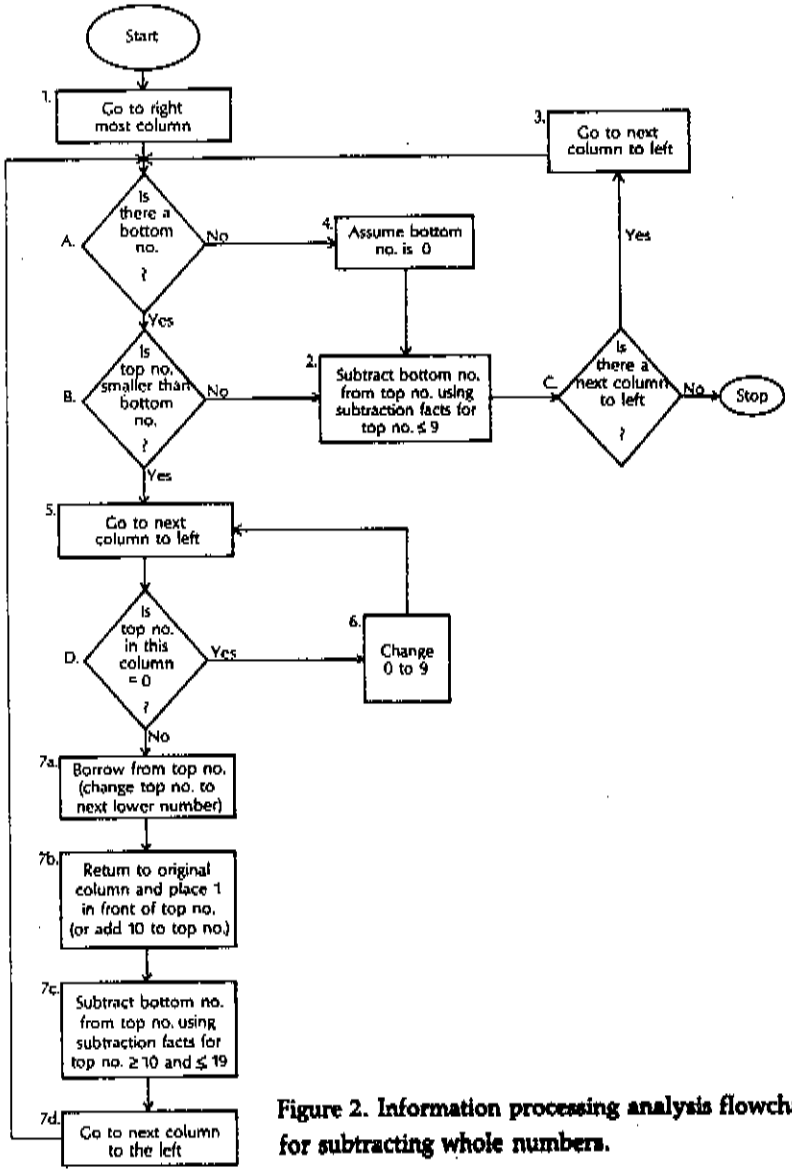

Figure 2. Information processing analysis flowchart for subtracting whole numbers.

The unique paths through the algorithm can be represented in terms of a directed graph where the decision steps are represented by points and the operation steps are represented by arrows or arcs (Scandura, 1973b). Figure 3 shows the directed graph for the subtraction algorithm and identifies 11 unique paths through the algorithm. The numbers on the arrows in the graph correspond to the numbers of the operations in the flow chart while the points in the graph correspond to the decision points and start-stop points in the flow chart. Note that only one point is used for adjacent decision points and only one arrow is used for adjacent operations. Thus decision boxes A and B are represented by one point and operation boxes 7a-d are represented by one arrow. Examples of problems which require the steps of a given path for their solution are shown in the column to the right of their corresponding paths. For example, path P3 would be used in solving the problem 647 minus 25 while path P8 would be used in solving the problem 607 minus 469.

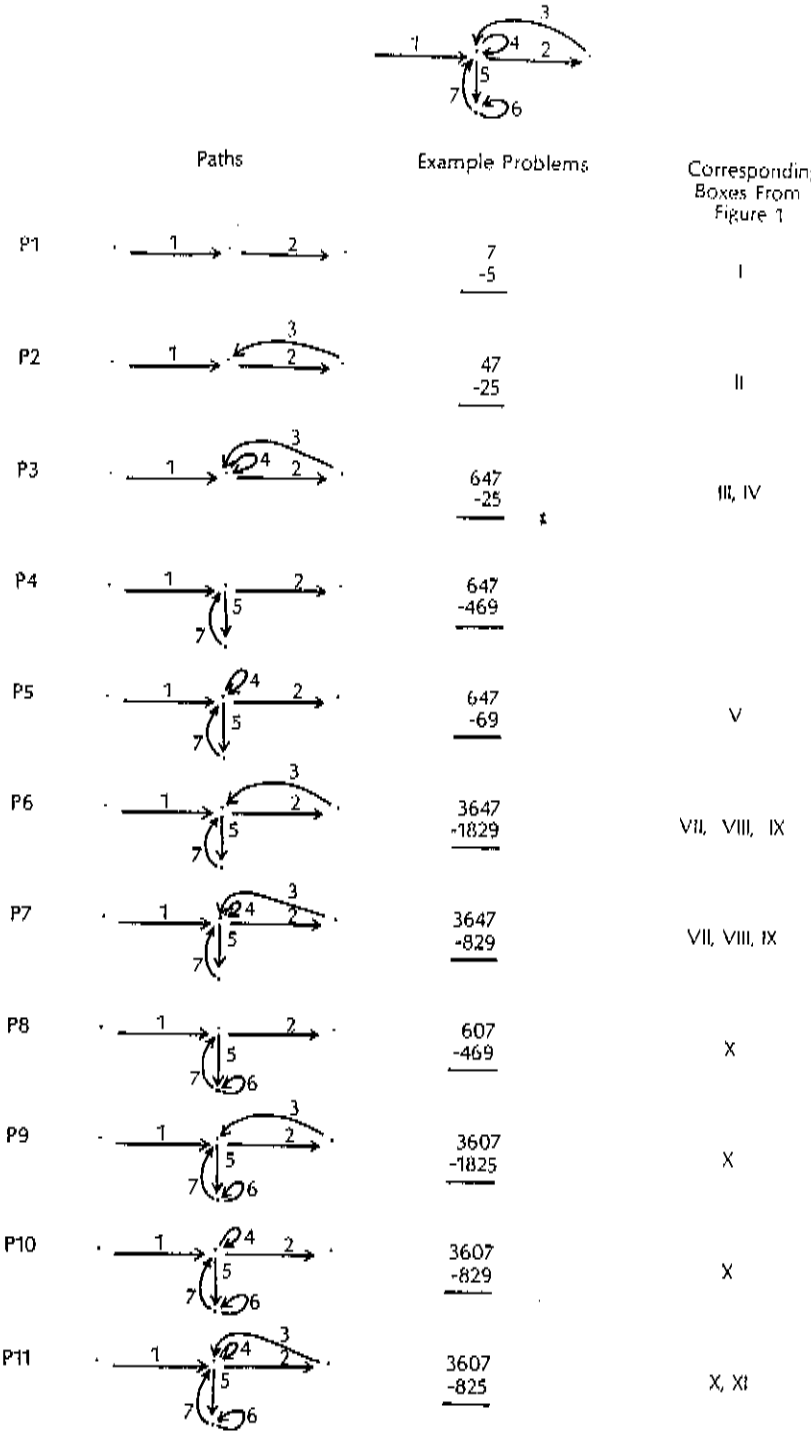

Figure 3. Directed graphs for subtraction algorithm.

Comparison

The boxes in the learning hierarchy shown in Figure 2 can also be related to a specific set of subtraction problems. For example, box 11 represents all of those subtraction problems which involve simple subtraction in successive columns without bringing down. This set of problems would correspond to those which can be solved by following path P2 of the subtraction algorithm (See Figure 3). The righthand column of Figure 3 shows the boxes of the learning hierarchy which correspond to the paths of the subtraction algorithm. The correspondence between the boxes in the learning hierarchy and the paths in the algorithm is not one for one. For example, problems corresponding to boxes VII, VIII and IX in the learning hierarchy can all be solved using path P7 from the subtraction algorithm. The subskills related to boxes VII, VIII, and IX in Figure 1 all involve exactly the same operations and decision points. They only differ in terms of the order in which certain steps of the path are executed or the number of times a loop is executed. For example, the skill "subtracting when a single borrowing is required" (box VII) requires only one execution of the loop which includes operations 5 and 7, while "subtracting when successive borrowing is required in adjacent columns" (box IX) requires several consecutive executions of the same loop. "Subtracting when several borrowings are required in non-adjacent columns" (box VIII) requires that each execution of the same loop be separated by the execution of other steps in the path (operations 2 and 3).

Scandura (1973b) has suggested that an algorithm may be broken down into simple enough steps so that every subject in a given population would be able to perform each step of the rule in an all or none fashion. He has further hypothesized that each path through the algorithm may be performed in an all or none fashion. Each path effectively partitions the domain of problems which may be solved using the algorithm into a set of mutually exclusive equivalence classes. Based on these assumptions, Scandura concludes that success on any item from one equivalence class implies success on all other items in the same class. According to this logic and the analyses of the subtraction algorithm

shown in Figures 2 and 3, it should be possible to combine the subskills of the learning hierarchy shown in boxes VII, VIII and IX into one subskill.

Scandura (1973b) also suggests that the paths of an algorithm can be partially ordered according to difficulty. The most direct path (P1) would be the least difficult while paths which include the steps of another path would be more difficult. This ordering of the paths according to difficulty is not necessarily linear since some paths (eg., P5 and P6; and P9 and P10) may not be ordered as to difficulty. Paths 5 and 6 include the same number of steps but each include one step which is not included in the other. The ordering of the paths in Figure 3 from path P1 through path P11 corresponds very closely to the ordering of the subskills identified in the learning hierarchy in Figure 1 from subskill I to XI. This correspondence should not be too surprising. According to Resnick and Ford (1976) positive transfer is expected to occur from simpler to more complex tasks in a learning hierarchy because the simpler tasks are included in or are components of the more complex tasks. This parallels directly the rationale used to order the paths through the algorithm (Scandura, 1973b).

It should be possible to reduce the total number of equivalence classes for a given algorithm if we assume that paths are equivalent which contain the same new steps but differ only by the inclusion of steps which have been included in a previous less difficult path. For example, paths P4, P5 and P6 could all be collapsed into the equivalence class defined by path P7. Paths P4, P5, P6 and P7 all include the same new steps (5 and 7) but only differ in that steps 2, 3 and 4, which have previously been included in path P3, are added one at a time in paths P4, P5, and P6. Similarly, paths 8, 9 and 10 could be combined into one equivalence class represented by path 11. These combinations would reduce the number of equivalence classes from 11 to 5 (Paths 1, 2, 3, 7 and 11).

Although the information processing or algorithmic analysis yields similar results to the hierarchical analysis, the algorithmic analysis requires greater precision and forces the analyst to specify in an explicit manner what is meant by a particular skill. For example, the actual steps involved in double borrowing specified by subskill X in the learning hierarchy are explicitly identified in path P11 of the algorithm. The identification of various paths through the algorithm also shows explicitly how one skill may be subordinate to another. The algorithmic analysis also reveals that certain skills are part of the same equivalence class and therefore may not need to be separately identified.

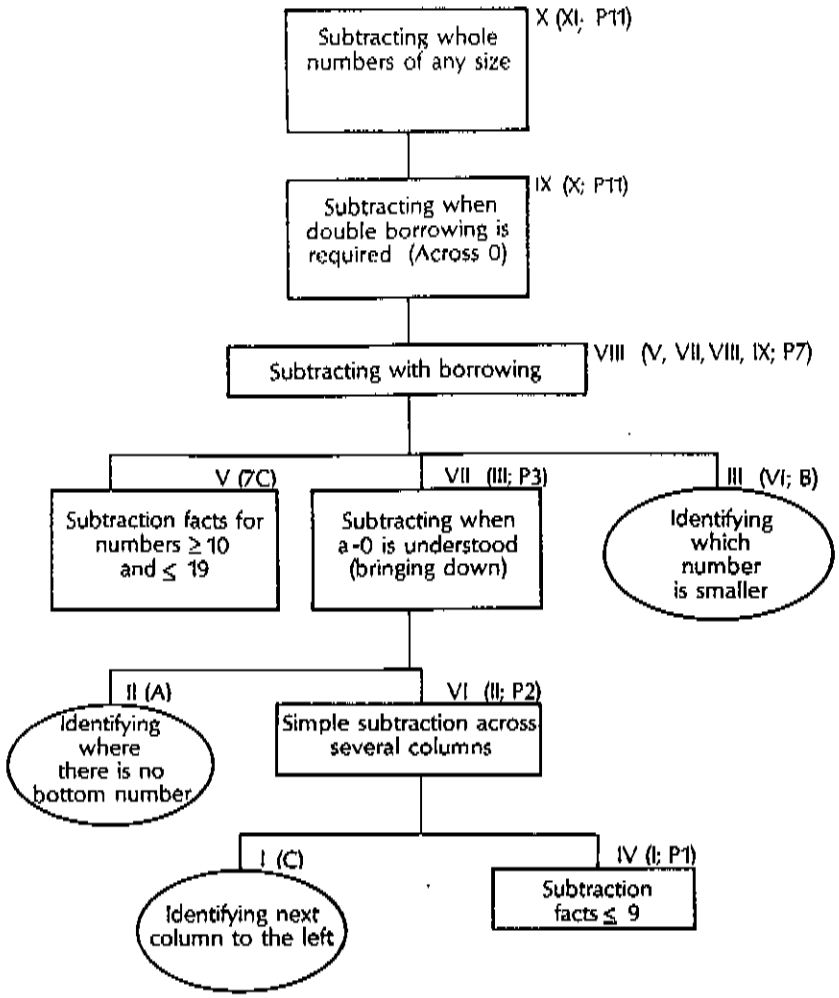

Figure 4. Revised subtraction hierarchy based on information processing analysis.

A revised learning hierarchy (Figure 4) has been developed based on the algorithmic analysis shown in Figures 2 and 3. In order to facilitate comparison, the Roman numerals which correspond to the boxes in the learning hierarchy in Figure 1 and the path numbers which correspond to the paths in the algorithm have been included in parentheses following the corresponding box numbers in Figure 4. Thus, Box VII in Figure 4 corresponds to the box labeled III in Figure 1 and path P3 in Figure 3. The oval shaped boxes in Figure 4 correspond to the decision points in the algorithm shown in Figure 2. Thus, box III in Figure 4 corresponds to box VI in Figure I and decision point B in Figure 2. Box V in Figure 4 corresponds to step 7C in Figure 2. This step is shown explicitly as a box in the diagram of Figure 4 since it probably should not be considered as an entering behavior for certain students. The skills represented by oval boxes in Figure 4 could be deleted if it is assumed that the target population has these concept identification skills as entering behaviors. They are included here to show their relationship to the other analysis diagrams. Although the hierarchy in Figure 4 is based on Figures 2 and 3, it is a very useful diagram because it synthesizes the information gained from the algorithmic and path analyses and shows the learning prerequisite relationships.

Additional Algorithms

One of the implicit assumptions of hierarchical task analysis is that there is only one procedure for accomplishing any given task. In contrast, information processing task analysis implicitly assumes that there may be many procedures for

performing the task. It is very possible that students who are able to perform a given higher level task without being able to perform all of the subskills considered prerequisite, as revealed by a particular hierarchical analysis, may actually be using an alternate algorithm which does not require those particular subskills. For example, a slightly different algorithm for subtracting whole numbers than the algorithm shown in Figure 2 may be found in Scandura (1973b, page 200). A radically different algorithm for subtracting whole numbers based on an "equal additions" procedure is shown in Figure 5. The algorithm in Figure 5 will provide the correct solution equally as well as that in Figure 2. However, the algorithm in Figure 5 does not include the traditional borrowing procedure nor does it require the loop for double borrowing across zero. Rather than borrowing one from the top number in the column to the left (as in Figure 2) the algorithm in Figure 5 adds one to the bottom number in the next column. Readers who are unfamiliar with the algorithm in Figure 5 should walk through both algorithms using the same problem such as 3,607 minus 825.

It should be obvious that a hierarchical analysis based on the algorithm in Figure 5 would be considerably different than the one shown in Figure 1 or Figure 4. A learning hierarchy for subtracting whole numbers based on the algorithm in Figure 5 is shown in Figure 6. There are three significant differences between the hierarchies shown in Figures 4 and 6. These differences are directly related to the differences between the algorithms in Figures 2 and 5. Note that the subskills represented by boxes VII and IX in Figure 4 are not found in the hierarchy shown in Figure 6. The subskill represented by box IX in Figure 4 is not required in the new algorithm. The subskill represented by box VII in Figure 4 was not included in the algorithm in Figure 5. Apparently Scandura did not feel it was necessary to make this subskill explicit and assumed that it was part of the subskill related to subtraction facts (Box III in Figure 6). Although the subskills identified by box VIII in Figure 4 and box VI in Figure 6 correspond to the same task, the procedures involved in performing the task are very different as can be seen by comparing steps 7a-d in the algorithm found in Figure 2 and steps 4a-c in Figure 5.

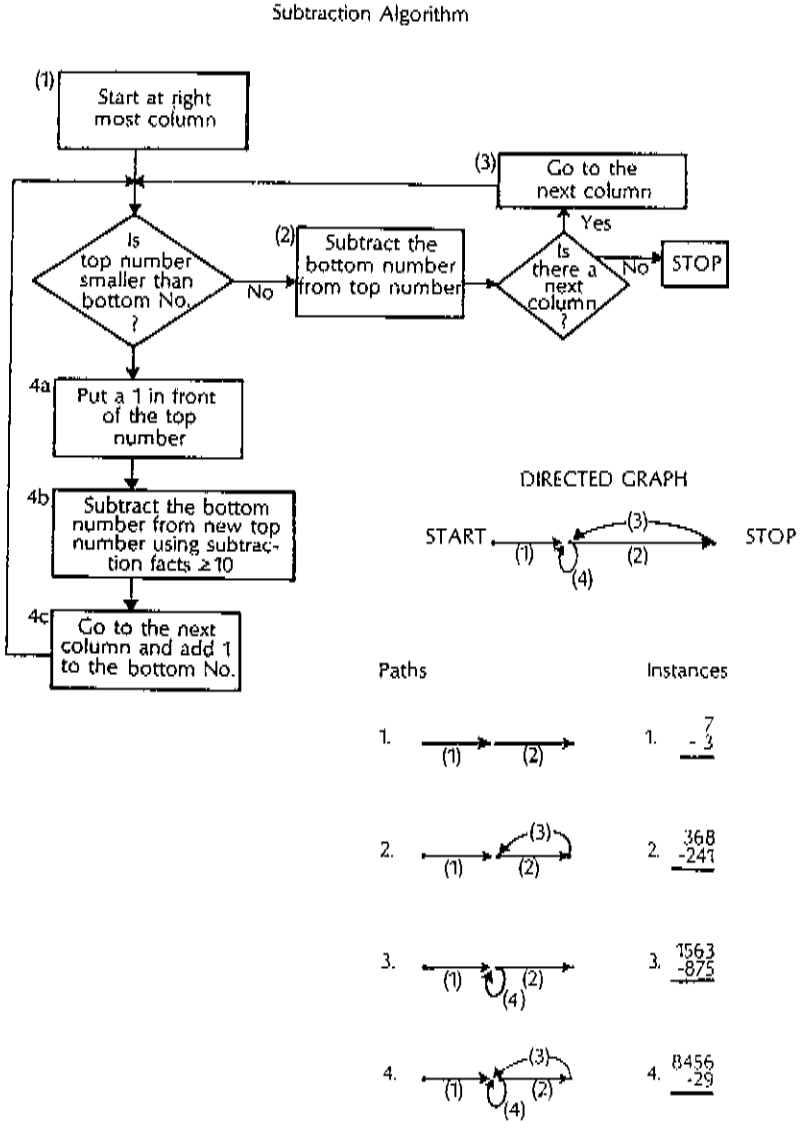

Figure 5. Flowchart and directed graphs for "equal additions" subtraction algorithm. (From Scandura, 1973a, p. 11).

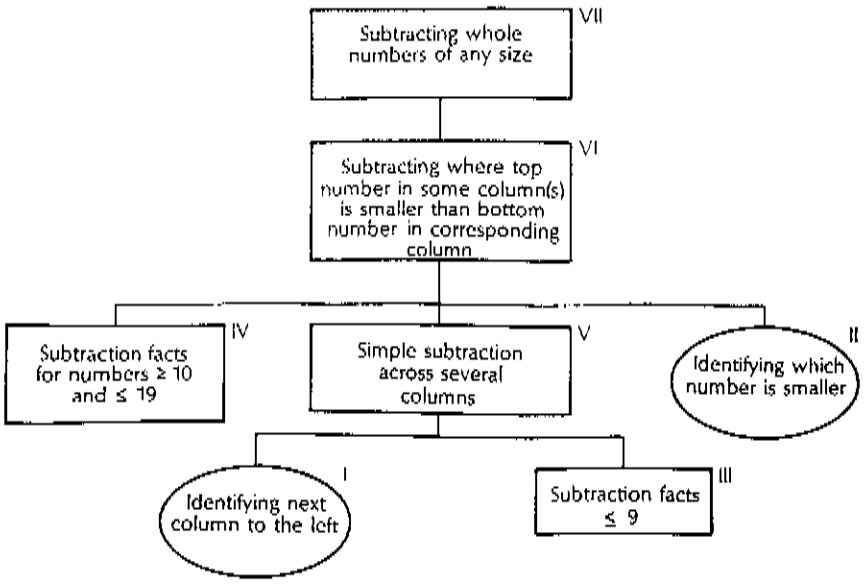

Figure 6. Hierarchy for subtracting whole numbers based on equal additions procedure.

Which of the many possible algorithms for performing a given task is the best? Merrill (1976) has suggested that algorithms may be compared or evaluated in terms of the relationship between the difficulty level of the operations of the algorithm and the entry behavior of the target population which is expected to perform the algorithm. The length of time and number of operations required to perform the different algorithms given several initial inputs may also serve as a basis for comparison. The algorithms may also be compared in terms of Bruner's (1966) concepts of economy and power. Economy relates to the information storage requirements of the algorithm while power relates to what effect the algorithm may have on the student's ability to generate new hypotheses and combinations. Resnick and Ford (1976) have suggested that an algorithm which demonstrates how skilled individuals perform a task may not be the best algorithm to use to teach a novice to learn the task. They suggest that an algorithm to help novices learn a task should meet the following three criteria:

1. It must adequately display the underlying structure of the subject matter.

2. It must be easy to demonstrate or teach.

3. It must be capable of transformation into an efficient performance routine.

For example, one of the supposed purposes of the new math movement was to teach students algorithms which more adequately displayed the underlying structure of the subject matter. However, many parents seem to be complaining that their children have not been capable of transforming these algorithms into more efficient performance routines.

Conclusion

The purpose of this paper was to compare the hierarchical and information processing approaches to task analysis through taking a specific example and analyzing it using both approaches. It was argued that for many tasks both types of analysis should be used in order to ascertain the various types of relationships between the subskills or suboperations of the task. In order to accurately perform a hierarchical analysis, the instructional designer must have in mind a specific algorithm for performing the task. However, many mistakes can be made unless this algorithm is made explicit through an information processing or algorithmic analysis. Once the operations and decision points of the algorithm have been identified, they can be diagramed in flow chart form. This flow chart will show the sequencing and output/input relationships between the operations of the task. The various paths through the algorithm can then be identified and thereby partition the domain of problems which can be solved by the algorithm into mutually exclusive equivalence classes. These equivalence classes can subsequently be ordered according to difficulty. The information obtained from these analyses can then be used as a basis for conducting a hierarchical analysis.

The instructional designer should be aware that there are many possible algorithms for performing any given task. The available algorithms should be evaluated and the most appropriate one selected according to the criteria specified in the previous section.

Instructional designers should also be aware that neither the hierarchical nor information processing task analysis procedures are appropriate for all types of tasks. Many tasks are neither algorithmic nor hierarchical in nature. Efforts to force such tasks into a hierarchical or algorithmic mold are futile. An analysis procedure which reveals other types of relationships such as complexity, chronology, or spaciality (Evans, Homme and Glaser, 1962) may be more appropriate.

References

Bruner, J.S. *Toward a theory of instruction.* Cambridge: Harvard University Press, 1966.

Evans, J.E., Homme, L.E., and Glaser, R. The ruleg system for the construction of programmed verbal learning sequences. *The Journal of Educational Research*, 1962, *55*, 513-518.

Gagne, R.M. The acquisition of knowledge. *Psychological Review*, 1962, *69*, 355-365.

Gagne, R.M. Analysis of objectives. In L.J. Briggs (Ed.) *Instructional design: Principles and applications.* Englewood Cliffs, NJ: Educational Technology Publications, 1977.

Gagne, R.M. and Briggs, L.J. *Principles of instructional design.* New York: Holt, Rinehart and Winston, 1975.

Merrill, P.F. Task Analysis—on information processing approach. *NSPI Journal*, 1976, *15*(2), 7-11.

Resnick, L.B. and Ford, W.W. The analysis of tasks for instruction: an information-processing approach. In T.A. Brigham and A.C. Catania (Eds.) *Social and instructional processes: foundations and applications of a behavioral analysis.* New York: Irvington Publishers, in press.

Scandura, J.M. Structural learning and the design of educational materials. *Educational Technology*, 1973 *13*(8), 7-13. (a)

Scandura, J.M. *Structural learning: I. Theory and research.* New York: Gordon and Breach, 1973. (b)