NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

The Design, Development, and Evaluation of a Prototype Expert System for Achievement Test Design.

Yan Feng

A Thesis

in

The Department

of

Educational Technology

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Arts at

Concordia University

Montreal, Quebec, Canada

July 1990

ISBN 0-315-59160-9

ABSTRACT

The Design, Development, and Evaluation of a Prototype
Expert System for Achievement Test Design

Feng Yan

One application of Artificial Intelligence (AI) is to use an expert system as an "intelligent" job aid to support on-the-job performance. Many of such expert systems have successfully performed complicated tasks requiring specific expertise, but few are capable of improving their users' overall understanding of the domain. An expert system-Test Authoring Assistant (**TAA**)- was designed, prototyped and tested. The resulting system was designed primarily to help its users to solve open-ended design problems and to assists the users in becoming domain experts themselves. TAA assists teachers, trainers and researchers in developing paper-pencil tests to measure people's achievement in the cognitive domain (Bloom, 1956). The study constructed the knowledge of the achievement test design process, and its underlying concepts and principles into a computable model, using Reigeluth - Merrill's Elaboration Theory of Instruction (Reigeluth & Stein, 1983). Computer experts, subject matter (domain) experts and end-users participated an formative evaluation. The results of the evaluation show that the operating TAA has presented the major features of the intended TAA, but it is still to be considered a prototype. The results of the evaluation also indicate that the intended TAA can be a very useful tool for the targeted users to design achievement tests of quality and to understand "how to" and "why" issues in the design process. Furthermore, the experience drawn from the study can be used in expert system development for open-ended design applications or for the expert system design with a concern for developing users' expertise.

Acknowledgements

I would like to extend my deepest gratitude towards the following people who assisted me in completing this study. To my advisor, Dr. Richard Schmid, for his invaluable guidance, assistance and encouragement. To Dr. Roger Shanghal and Dr. Gary Boyd for their comments and supports.

I would also especially like to thank Dr. Robert Bernard for his much appreciated advice.

I am also indebted to my classmates who gave me suggestions and participated in the evaluation of this study.

Finally, I would like to thank MS. Beverley Boyle for her continuous support during my studies of the past few years.

TABLE OF CONTENTS

		Page
Chapter 1:	Introduction	1
	Context of the Problem	1
	Purpose of Study	2
Chapter 2:	Literature Review	3
	Expert Systems to Solve Design Problems	3
	Consistency Between Job Aid and Expertise Development	5
Chapter 3:	Achievement Test Design As Knowledge Domain	14
	Application of Achievement Tests	14
	Nature of Test Design	15
	Practice in Test Design	17
	Development of Expertise in Test Design	18
Chapter 4:	System Description	19
	Development and Delivering Environment	19
	Description of End User	21
	System Requirements	21
	System Functions	24
Chapter 5.	System Design and Development	27
	Content and Scope of Domain Knowledge of the System	27
	Test Authoring Assistant to Solve Design Problem	32
	Knowledge Organization	33
	Procedural knowledge as epitomizing content	33
	Theoretical knowledge as supporting content	37
	Prerequisite knowledge as supporting content	38
	Knowledge Representation	41
	Architecture of System	43
	Problem Solving Strategies	45
	Communication With User	48
	System Development	53
	Development of individual knowledge bases	56
Chapter 6:	Evaluation	58
	Expert System Evaluation	58
	Evaluation Questions	61
	Evaluation Design	62
	Evaluation With Subject Matter Experts	64
	Evaluation With Computer Experts	64
	Evaluation With End Users	64
Chapter 7:	Results And Discussion	66
	Results	66
	Recommendations	68
	Discussion and Conclusions	73
Reference		77
Appendix		
1.	Task Map of Test Design Process	80
2.	Samples of Knowledge Representation	97
3.	Formative Evaluation Questionnaires	123
4.	Evaluation Results	127
5.	User's Manual	136
6.	Objects and Their Values	142

LIST OF FIGURES

Page

Figure 1. Design Test Instrument As Part of Evaluation Process ---------- 16

Figure 2. Information Dependency of Achievement Test Design Process ---------- 29

Figure 3. Concept Structure and Scope of The System ---------- 30

Figure 4. Integration of Problem Solving and Instruction in Choosing Test Format ---------- 39

Figure 5. Integration of Problem Solving and Instruction in Constructing Test Specification --- 40

Figure 6. System Architecture ---------- 44

Figure 7. How The Assistance Is Given ---------- 50

Figure 8. Matrix of Evaluation Design ---------- 63

CHAPTER 1

Introduction

Context of the Problem

Computer software has been used to reduce the burden of calculation and other repetitive tasks for a long time, but it is only in the last decade that people can use artificial intelligence outside of laboratories, specifically via expert systems, to solve problems which require reasoning and judgement (Walters & Nielsen, 1987). Expert systems, developed as "intelligent" job aids in business and industry, clone and disperse valuable expertise in various types of tasks (e.g., diagnosis, planning, design, monitoring and control). Many expert systems can produce "intelligent" judgement on how to perform the domain tasks, yet, they are often incapable of giving the user overall and heuristic explanations for their advice and actions. Many of such expert systems do claim that they can explain the "why" and "how" of their decisions, however, what they actually do is to track the path of the rules fired and present the path to the user. Often the messages generated in this way are too shallow and too fragmentary to enable the user to gain either a deep or a comprehensive understanding of the domain. Expert systems developed for educational purposes systematically teach the users to gain a deep and comprehensive understanding on a subject, but they usually cannot be used as tools to solve "real life" problems. The reason for this is that the knowledge representations of such systems only weakly reflect the alternative paths and effective control strategies of how the knowledge should be used to solve "real life" problems. More specifically, the knowledge used for educational purposes is constructed and communicated to learners using learning prerequisites in terms of "what must be learnt first", contrasted with "what must be done first" as a major concern in a "real life" problem-solving context.

Compared with the types of problems subjected to an expert system's solution (e.g., diagnostic, planning, monitoring, configuration and control), it is difficult to build real

"intelligence" into an expert system solving a design problem because the design problem is rather intractable. It is intractable since the rules used in a design process are too situationally dependent to be economically recorded into a computable model for expert system development. In other words, it renders the search space too large when design specifics are included in a system, therefore, practically, the possible final product of a design task normally cannot be predetermined and programmed .

Purpose of Study

The purpose of the present study is to examine some of the problems raised by attempting to construct a prototype of an expert "coaching" system which attempts to deal with an open-ended design problem. The system created here is a computable model that primarily helps to solve "real life" problems and at the same time teaches the user the deep and comprehensive knowledge involved.

The topic selected is a very practical one: to provide assistance for instructors and researchers in designing achievement tests. It was chosen because it is time consuming to design and develop a valid and reliable test instrument. Also, a test designer must have good knowledge and experience in the field so that s/he is able to produce an achievement test of high quality. Often, because of time pressure and/or lack of training, instructors develop tests with low levels of validity and reliability. The prototype represents an effort to make the expertise of test design more accessible to the end-users, and expert systems may prove to be good tools to assist them in designing higher quality test instruments.

CHAPTER 2

Literature Review

The purpose of this study is to explore the design strategies of an expert system which is able to assist a user in becoming an expert in the domain while the system is used as a tool to solve a development problem. According to the purpose of the study, Text Authoring Assistant (TAA) should have at least two functions: one is problem-solving and the other is coaching. For the first function,TAA is supposed to assist the user in performing design tasks; therefore, the key issue of this function is how an expert system can be designed to perform the design tasks. For the second function, TAA is supposed to help its user in becoming an expert in achievement test design; therefore, the key issue for this function is what are the content components that should be included in TAA. The two functions are not independent, but they should work coherently; thus, the third issue is that how these two major components, in a form of user-machine communications, should be integrated. These issues are elaborated below.

Expert Systems to Solve Design Problems

TAA is an expert system application to help its users to perform design tasks. Some researchers in the AI field put design and configuration under the same category (Waterman, 1986). Design is similar to configuration in that both are tasks of combining suitable components harmoniously, based on a given set of constraints to satisfy a goal. Usually, in configuration and design problems, the combination of components - a possible final product - cannot be predetermined; on the other hand, the possible solutions of diagnostic problems (e.g., the names of possible diseases and the prescriptions to treat them) can be predetermined. Although they are similar in the general approach used to solve the problems, there is a significant difference in their nature and tactics: a design process relies on very few fixed rules at a concrete level, thus, it is a highly unstructured task; since the knowledge of components and their

compatibilities (e.g., the models of computer parts and their compatibilities) can be expressed explicitly through production rules, configuration is a well structured task. To solve a configuration problem with an expert system, one will select pre-specified components and combine them; however, to solve a design problem with an expert system, one has to create components, and then combine them.

The difficulty in creating components on-line by computer is one of the major reasons behind the fact that expert systems to solve design problems are used mostly in the areas of "hard" sciences or technology, such as digital or microelectronic circuits design: e.g,.PALLADIO (Brown & Tong, 1983), PEACE (Dincbas, 1980), REDISIGN (Mitchell & Steinberg, 1983). Those expert systems have sophisticated simulation and graphic features, so that the user can visualize the process and results of the design. The simulation is used to verify or test design ideas when a user constructs a partial design. Because the features, functions and compatibilities of the components can easily be specified in the microelectronic circuits design, the expert systems do not need to create individual components, rather, they help the user select a good combination of components to serve the purpose of the task. As a matter of fact, most expert systems of this sort are actually just doing configuration, but called it "design". That is why it is said an expert system used to solve a design problem is nothing more than a "compiler", which does very little real "design" (Waterman, 1986).

Compared with the number of expert systems developed to solve design problems in "hard" sciences, fewer expert systems have ever been created to perform design tasks in "soft" sciences. One of the major reasons is that it is far more difficult, in a design task of "soft" science, to describe the nature of the components and their compatibility at a concrete level than it is in a "hard" science. To illustrate this point, it is practically impossible for an expert system to assist in judging the validity and reliability of a test by reading test items created by a user; however, it is a standard practice for an expert

system performing microelectronic circuits design to simulate the consequence of a product designed by a user or partially designed by the system.

For expert system development, the difficulty associated with formulating explicit solutions in the form of production rules is due to the wide gap between the explicit solutions that an expert verbalizes and vague heuristic control approaches that are actually used. In the case of achievement test design, principles are well documented and researched by educators and psychologists, but the domain heuristics at a concrete level are too situational to be transformed into production rules. For instance, if an expert is asked why s/he includes certain tasks/behaviors/performance in a test, s/he may say that the purpose of the test is to see whether or not the learners are ready for instruction. The tasks included are the minimum skills in order to start the instruction. The gap between the conditions (if) and the actions (then) is obvious in the above case, as the condition part of a production rule,"learners' readiness for the instruction" is concrete, but "minimum skills" as a conclusion part of a production rule is very abstract. From that "if-then" rule, one can hardly see any logical connections between learners' readiness and what the expert has decided to be tested, except what has been included in the test as minimum skills. With such a gap between conditions ("if") and actions ("then"), it is unlikely that this type of knowledge can be directly transformed into production rules so that an expert system can give explicit and ready-to-use solutions to its user. Another difficulty in developing an expert system for a design task in "soft" science is the wide knowledge scope a system has to cope with. For instance, a system designated to perform test design tasks should not be restricted to a particular subject matter, e.g., math, social science, language, computer science; nor should it be restricted by characteristics of testees, e.g., university students, first graders in elementary school. Consequently, in expert system design, the coverage of the expertise, the level of abstraction and the possible searching space must be well balanced.

Consistency Between Job Aid and Expertise Development

The designed expert system - Test Authoring Assistant (TAA) - will function as a job aid. The job aid has a long history in the working place (e.g., procedures and manuals). Although traditional job aids are widely used, they often have weaknesses embedded in their construction. For instance, traditional job aids are often too complex to use (Harmon, 1987): i.e., a novice usually is not clear at all about where to look for solutions from an manual, and when s/he is faced with non-standard cases, s/he does not know how to generate new solutions by reading a standard case from a manual. Expert systems used in industry are very similar to traditional job aids in the sense that the purpose of both is to prompt the expected performance by giving the user the guidelines, regulations and information about the tasks. However, expert systems as job aids are superior to traditional job aids in that they can save users' time in searching for solutions and in that they can give solutions according to situations.

Although expert systems are superior to traditional job aids in terms of making expertise much more accessible to the inexperienced practitioner, there is concern about the de-skilling of the end-user and also about the danger of incorrect performance due to blind acceptance of the advice from the system (Elmore, 1986; Woolf, 1988). If expert system technology is expected to improve on-the-job performance in a long run, it must have certain features to affect its user's overall understanding of a domain. Research studies on the development of human expertise may give some lights on this issue.

One way of helping novices to become experts is to enhance their reasoning processes through representing the experts' reasoning processes to them. In order to reveal the experts' reasoning processes to the user, it has recently become a trend to establish the psychological validity of the knowledge representation in the expert system design. Psychological validity refers to the consistency between the reasoning process of an expert and the program (Clancy, 1987). The concept of the psychological validity might be used in a broader sense: it can refer to the consistency between the communication of a system with its users and the expertise development in that field. The

issue of psychological validity manifests itself whenever one creates an interface between the communication design of an expert system and the development of human expertise.

In terms of human expertise development, researchers in the field agree that it is acquired in two complementary ways: schooling, and practice in the field (Keravnou & Johnson, 1986; Harmon, 1987). Another way of expressing these two components is that theoretical knowledge is gained in the educational process and that heuristic knowledge is developed from working experience. Of them, the latter can be further categorized into rules of thumb and global problem-solving strategies for a domain.

Although there is general agreement that extensive practice in the field is essential for expertise development, there are different views on how this process occurs. Keravnou and Johnson (1986) state that knowledge represented in education has two features: one is that it is static and that it is others' knowledge instead of learners'. It is common that the knowledge represented in text books or other educational media for schooling are structured by learning prerequisites concerning "what should be learnt first". In schools, practice assigned to students is created according to the students' knowledge level: they are common problems and ideal cases to illustrate how a newly learnt piece of knowledge should be used. Because the knowledge representation in formal education is usually very weak in illustrating alternative paths, key patterns and strategies in "real life" problem-solving, Keravnou and Johnson (1986) consider that in order to become an expert, one has to deeply internalize and restructure basic concepts and theories derived from others. This process requires a person to have a solid knowledge structure to do intensive cross-referencing. S/he also needs consistent feedback during "real life" problem-solving; therefore, basic and theoretical knowledge gained from education, and heuristics compiled through practice in the field are two interdependent instruments which affect the development of the human expertise.

In contrast with Keravnou and Johnson's point of view, Harmon (1987) feels that expertise developed in education and in the field represents a division of cognitive

psychology. That means one can develop expertise without formal education or without extensive practice in the field. Examples cited by him are skilled technicians and master crafts persons who start as apprentices, observe mentors and become experts within their domain without benefit of formal training; on the other hand, some academic specialists, like logicians or theoretical mathematicians, compile general theories without benefit of experience. Therefore, he considers knowledge structure in an education setting to be irrelevant to the design of an expert system as a job aid. Knowledge communicated by schooling is primarily declarative in nature and knowledge in practice is more procedural. The formal knowledge of schooling usually fails to indicated exactly how one should proceed when faced with a specific problem because it is too large to search. The expertise built upon experiential knowledge shows the practitioner to look for the key patterns or important relationships that indicate how to approach the problem which becomes the base for designing a job aid; therefore Harmon (1987) promotes a type of intelligent job aid which prompts performance rather than asking employees to memorize procedures or learn the theory, or deep structure, underlying a procedure.

Harmon's (1987) assumptions are reflected in common practice of expert system applications for schools and for business in a sense that expert systems in education and business are very distinctive. Most of the expert systems used in business and industry incorporate rules of thumb, and their output messages are ready-to-use solutions. Some expert systems are able to communicate with its user about partial problem solving strategies. Usually, the expert systems used for on-the-job performance include little theoretical knowledge structured according to a logical classification scheme; and very few of them have an overlay model to monitor the users' understanding of the task. On the other hand, the expert systems used for educational purposes have the theoretical knowledge structured according to a standard or a popular classification, and the knowledge is communicated to the user according to the learning prerequisite of a topic. So far expert systems for educational purposes have been mainly concerned with how to

teach basic concepts and principles and how to apply them in standard or common cases. It is very common that an expert system for educational purposes has an overlay student model and the expert's model to monitor the users' progress. Also it has the component of teaching strategies to give feedback to the students' actions. The features of expert systems created in these two settings reveal that neither of the standard models fit with the purpose of TAA: the model of the expert systems created for the educational purposes are very weak to solve "real life" problems, and the model of the expert systems created for improving on-the-job performance hardly do anything to help their users to become experts.

If the studies of the development of human expertise are used as guidance to design an expert system, Keravnou and Johnsons' model of expertise development makes more sense than Harmon's model. Formal education and field practice are proportionally different at each learning stage of various expertise development. Patterns of Keravnou and Jonhsons' model of internalization of theoretical knowledge can be evident in most occupations, such as medical doctors, engineers. To become an expert in such an occupation, one must go to formal schooling to get the basic and general education, and then, practice extensively in the field. The internalization of the external and static knowledge is also evident even in the examples cited by Harmon (1987), e.g., craftsmen, locksmith: although this type of expertise (psychomotor skills) development requires mainly practice in the field, there are principles and concepts (cognitive components in psychomotor skills) which can be expressed verbally. On the other hand, academic expertise seems always developed in a formal education setting, however, it does not imply that no experience in the "real-life" problem solving is required. For an academic specialist, like a logician or a theoretical mathematician, the formal education and field practice are intertwined and overlapped, because real world problems can be created and treated in a laboratory.

Although aims of expert system applications in industry and education are very different, the underlying principles of expertise structure are consistent: that expertise has both heuristics and theories, and it has structures for both factual knowledge and problem solving. It might be very difficult to verbalize the heuristics precisely or state underling theories in one or the other case, but it should be seen as proportionally different rather than capability specific to one environment. Therefore, a good expert system either as an educational instruction or as a job aid should have three well balanced components . The three components refer to theoretical knowledge structured as logical chains of the domain knowledge, problem solving strategies and tactics structured according to key patterns of real world problems and a user's model (in an instructional expert system program it is often called as student model). "To be well balanced" means the weight of each of the above components is consistent with the purpose of an expert system program. With the purpose of improving users' on-the-job performance, an "intelligent" job aid should be problem-solving oriented and backed up by underlying theories and concepts. Also it should reflect the experts' problem solving at a more abstract level in order to communicate with the user how to produce optimal solutions in a non-standard situation. The value of this approach extends beyond appropriate solutions for the user, but improves the users' competence in that domain. An expert system as a job aid should have a user's model, but it certainly will be very different with that of most expert systems used for educational purpose. In other words, an expert system design should reflect the features of its purpose.

To summarize, expert systems applications used for formal educational instruction and job performance assistance should not be viewed as two distinct classes if the ultimate aim of the applications is to improve users' or students' competence. Therefore, expert systems used in educational and business settings should have three basic components:

1. theoretical knowledge including concepts and principles;

2. problem solving strategies; and

3. user's model/student model. Furthermore, the content and weight of these components should be selected and balanced to serve the prioritized purposes of a system.

Based upon the above beliefs, a theoretical model of design is useful as a tool in creating such an expert system. Such model helps in organizing and sequencing the logic and concept development of the system. An ideal model should be flexible enough to serve different kinds of purposes, and simple enough to be technically feasible to implement.

A model which has these features is Reigeluth-Merrill's **Elaboration Theory of Instruction** (Reigeluth & Stein, 1983). The theory extends and integrates Ausubel's *subsumptive sequencing*, Bruner's *spiral curriculum*, Norman's *web learning* and Merrill's *component display theory*. It deals with **organizational strategies at the macro level**: selection, sequencing, synthesization and summarizing of subject-matter content.

One of the advantages of Reigeluth-Merrill's Elaboration Theory of Instruction (Reigeluth & Stein, 1983) is it has a great flexibility in organizing program content and constructing operation sequences. According to the emphasis of an application, one may use one of the three types of knowledge as the epitomizing content: theoretical, conceptual and procedural. For example, if the purpose of a system is to improve users' understanding of theoretical knowledge, like in economics, then theoretical knowledge can be chosen as the main structure of the program, and conceptual and procedural knowledge will be incorporated as supporting components. If the problems a system deals with are procedural-decision intensive which are common in expert syste. :s as job aids, the procedural knowledge can be chosen as its epitomizing content, and related concepts and principles will be structured in the forms of explanations and instructions as supporting components. Obviously, using Reigeluth-Merrill's Elaboration Theory of Instruction (Reigeluth & Stein, 1983) as the guideline will make an expert system design

more rational: in the process of choosing epitomizing content as the main structure of the program and in the process of balancing the weights among the components, a system designer has to match the prioritized purposes of the system and the design of the system. Also, defining the epitomizing content and its supporting content will enhance its coherence and consistence where various types of activities have to be integrated in a program.

Another advantages of the Elaboration Theory of Instruction is that it has a powerful capability to aid the selection and sequencing communication with the user. This is controlled by an **elaborative sequence**, which is a simple-to-complex sequence where:
a) The epitomizing is done on the basis of only one single type of content, and it is used throughout a learning unit. Either concept or procedure or principle can be used as the epitomizing content;
b) The ideas are epitomized rather than summarized at the beginning of each learning unit. The epitomizing allows the user to get started at a concrete level with a manageable amount information. The simple-to-complex sequence leads a user to zoom into details as s/he wishes and go back to the main track at any time. Very different from a popular instructional format, only at the end of each learning unit, the epitomizing content is synthesized in a larger knowledge context. The "zoom into" and "zoom out" methods are very suitable for a computer program because the screen of a computer monitor can deal with only a small amount of information at a time. Background information has to be stored in the computer memory and to be represented to the user when it is required. The "zoom into" and "zoom out" methods can help a user to make effective and structured cross-referencing without an overload of too much information.

One thing which needs to be mentioned is that Reigeluth-Merrill's Elaboration Theory of Instruction (Reigeluth & Stein, 1983) was developed originally for instruction in formal education environments. Seldom has this model been used in expert system design which helps its users to solve "real life" problems; therefore, the structure of

procedural knowledge defined in the elaboration theory of instruction (e.g., "state the proper steps of a chemistry experiment") is too sketchy to be used as the epitomizing content when the purpose of a program is to solve "real life" problems. Furthermore, the model is not meant to assist expert system design; therefore, the communication of TAA with the end-user has to be structured more specifically than the model described. (Refer to Chapter 5, Design and Development to see how Reigeluth-Merrill's Elaboration Theory of Instruction was adapted in this study.)

To summarize, in this chapter we have discussed two major issues: development of expert systems to solve design problems and implications of expertise development in designing an "intelligent" job-aid. We also described Reigeluth-Merrill's Elaboration Theory of Instruction model which was adapted for this system design. In the next chapter we will review the nature of achievement tests, the practice in the field and the expertise development in achievement test design.

Chapter 3

Achievement Test Design As Knowledge Domain

Achievement test design, which measures knowledge and skill acquisition in the cognitive domain (Bloom,1957), is chosen as the knowledge domain for TAA. The application and nature of it suggests the coverage and the scope of TAA; the circumstances of field practice and the expertise development of the profession rationalizes why this expertise is chosen as the knowledge domain.

Application of Achievement Tests

As the target of a test instrument, performance is defined as human behavior and accomplishment, both overt and covert. Overt behavior is often used as an indicator of covert behavior. Based on Meherns & Lehmann's taxonomy (1978), which overlaps with Bloom's (1956) performance category for structuring educational objectives, six components can be classified in human performance for measuring purposes:

1. Knowledge acquisition & understanding in the cognitive domain
2. Psychomotor skills
3. Communication skills
4. Attitude, values and preference
5. Behavior patterns
6. Aptitude, potentials and competence.

These six dimensions of human behavior are often measured in the educational field and in the work environment. Different types of test instruments are used to measure different targeted performance, although they may all be constructed on paper requiring that testees use pencils to complete them. For instance, according to the standard definition, an achievement test measures testees' acquisition of knowledge and skills in the cognitive domain (Bloom,1957) while an aptitude test measures a person's

ability in terms of his or her potentials. An achievement test is different also from attitude survey which investigates a person's opinion to infer that how s/he will act in a given situation.

Among test instruments (e.g., aptitude test, personality assessment and attitude survey), achievement tests are the most widely used human performance measurement tools and they touch all aspects of our lives. For instance, in schools educators and teachers use performance testing to give grades on students' achievement, classify students for placement or modify learning materials and activities; in business, management uses performance measurement to make decisions on training, promotion and compensation. Classroom testing, performance appraisal and formative or summative evaluation of learning are but a few examples of using achievement tests for performance measurement.

Nature of Test Design

The design and construction of a test instrument are critical tasks in an evaluation process (see Figure 1). Like any problem solving process, the test design process consists of identifying the goals, searching for alternatives and making decisions within constraints and/or altering the constraints. Constructing a test instrument belongs to the design category. Similar to the nature of other design problems, the test design process includes choosing levels of measurement, test type, test format, deciding its length and constructing the test items. All these decisions are made with consideration for the testing purpose, subject content, testees' characteristics and resource availability.

In a test design and development process, an expert will clarify or make decisions for the following questions:

1. What type of decision do we want to make based on the information generated by the measurement?
2. What are the performances or accomplishments that can be used as the indicators to what we want to know?

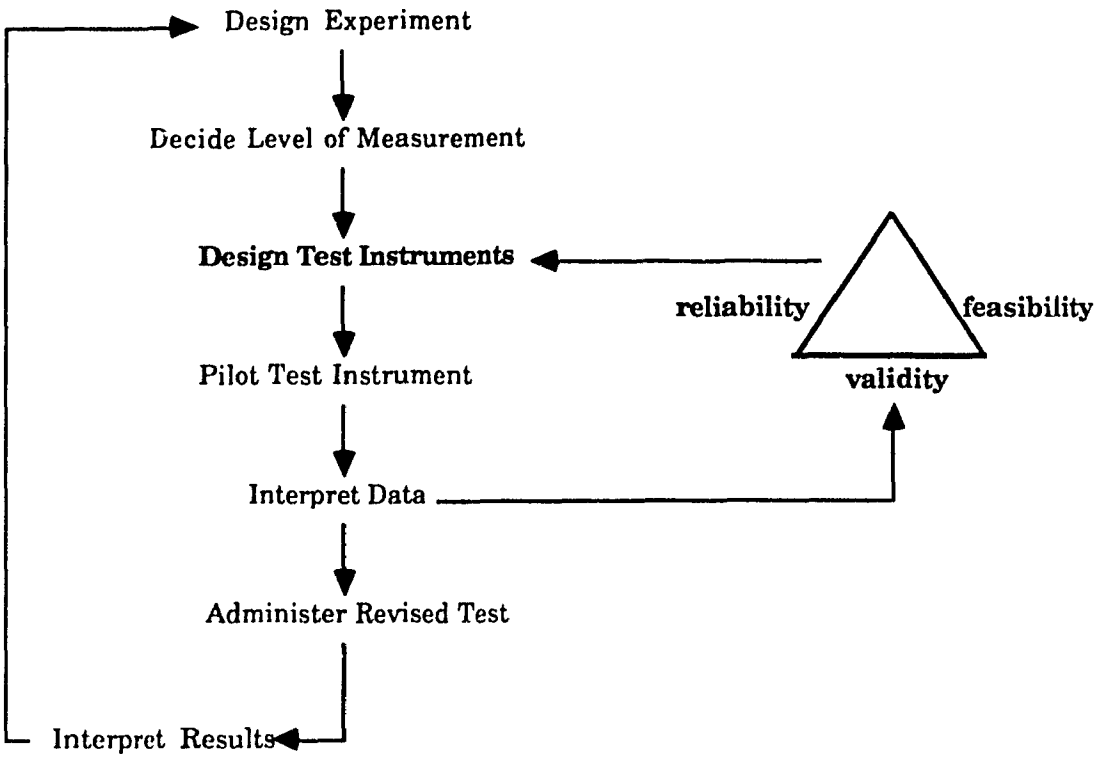


Figure 1. Design Test Instrument As Part of Evaluation Process

3. What are the constraints and opportunities in using the different methods or formats?
4. What are the methods and formats we should use to measure the performance or accomplishments we are interested in?
5. In what manner should test items be constructed.

All these decisions in designing a test are governed by the principles of validity, reliability and feasibility.

Although these three factors are overall concerns in test design, the relationship among them is complex. For instance, reliability provides the consistency; however, a test instrument with a high degree of reliability does not guarantee a high degree of validity of the test (Gronlund, 1981). Thus, the spirit and also the difficulty in test design is to ensure an optimal balance of the three. For instance, the degree of reliability

will be increased if the number of items for an objective is increased; however, because of limited test time, only a certain number of items can be included in any given test. Considering this, a test designer has to decide how to distribute the number of items among tested objectives so that the reliability of an objective and the reliability of a test as a whole will be balanced.

Even though the above three principles applies to any test design situations, with different purposes of using test instruments, people will have different emphasis or concerns regarding validity, reliability and feasibility. For instance, the reliability of a test instrument for a research purpose will carry more weight than that of a test instrument used for grading learners' achievement at the end of a term. In Chapter 5, System Design and Development, how the domain knowledge being structured into a computable model is explained in detail.

Practice In Test Design

The concepts and principles for designing a valid and reliable test instrument are not difficult to understand, but tests are often poorly designed. For instance, according to the 1975 study supported by the U.S. Office of Education, 20 percent of adult Americans are functionally illiterate. This figure contrasts sharply with earlier traditional statistics in which the illiteracy rate was described as being only 1 per cent. The sudden leap was explained as follows: up to 1969, literacy was measured by reading behavior; but in the 1970s, literacy was measured by reading accomplishment (Gilbert, 1978). The discrepancy between the purpose of evaluation and the performance to be measured is one of many serious problems existing in achievement test design.

It is time consuming to design a good instrument to measure human performance, because a test designer has to make extensive cross-referencing from a great deal of information and continually balance many factors throughout the design process. It is said that an experienced person cannot write more than ten test items a day (Mehrens & Lehmann, 1978). During this process it is almost unavoidable to overlook some

important elements that should be taken into consideration. Because of the time required to design a high quality test instrument, an instructor made test (vs. a standard test) is often produced by sacrificing quality to the development efficiency.

Development of Expertise In Test Design

Expertise in test design belongs to the cognitive domain. Like any skills in the cognitive domain, expertise development requires a good understanding of the subject. This understanding includes both theoretical knowledge and heuristics. Also, the development of expertise requires sufficient practice in the field, but practice will not have much effect unless it is supported by timely and clear feedback on the performance.

Although there is a body of knowledge and methodology on human behavior measurement which is well researched and documented, this knowledge is largely restricted to psychologists, professional testing writers and evaluators. In either a business or school setting, most tests are designed by practitioners, and few of them have adequate training in designing high quality performance measurement instruments.

Even for those who have previous training, the opportunities to practice are insufficient to develop expertise unless the individual is a professional test developer. For instructors and trainers, the design of tests and appraisal instruments is only a small proportion of their daily tasks.

Furthermore, it seems that years of experience in the field dose not necessarily lead to better expertise, as practitioners in the field seldom get clear feedback externally from peers and supervisors or internally from the consequences of the designed test. Often, feedback emerging as a consequence of inappropriate measurement is too subtle to have an effect on the designers' future performance, because it is very difficult for people who do not have special knowledge in the area of test design to recognize that an inappropriate decision has made on the basis of the information generated by an inappropriate measurement.

Chapter 4

System Description

At present, the designed system (Test Authoring Assistant) has not reached its final stage of development. To varying degrees, TAA has met some of its requirements defined below. Please refer to Chapter 5 "Design and Development" for a detailed explanation on the present development stage and Chapter 6 "Results" for recommendations for further development.

Development and Delivering Environment

TAA was developed to assist teachers, trainers and researchers in achievement test design. Achievement test design is a mixture of algorithmic (at procedural level) and heuristic (at decision making level) tasks. Expert system technology was chosen because it can accommodate both types of tasks. The conventional computer technology may be efficient do perform algorithmic tasks, but it is not optimal to do pattern matching for heuristic reasoning and inference.

TAA was built by using Intelligence Compiler version 2.1, an expert system shell for the IBM PC. The prototype of TAA was developed on an IBM PC with one megabyte of memory. Because a monochrome monitor could not show the difference when a predicate was defined, a color monitor was used for the development. TAA, as a prototype, can be delivered on an IBM PC with one megabyte memory and on either a monochrome monitor or a color monitor.

The programing tool for TAA's development was chosen among three expert system development shells available in Concordia University: M.I, Exsys and Intelligence Compiler. In choosing a development tool for prototyping TAA, some authoring language shells available in Concordia University were also considered, namely, Hypercard and Course of Action. The decision to use Intelligence Compiler as the programming tool of TAA was based upon the compatibility between the requirement

of TAA's development and the capabilities of each expert system development shell. Based upon the following facts Intelligence Compiler was chosen as the development tool for TAA:

a) Intelligence Compiler has a greater variety of knowledge representation schemes, (e.g., frame, list, backward and forward chaining rule, table and static text) than the other two shells do. For instance, the other two shells do not have frame and table representation schemes which are essential for the targeted test design problem: three tables are needed when TTA is constructing test specifications.

b) Intelligence Compiler has a tracing function which reveals how the inference process is proceeding, at a chosen speed and a level of detail. Neither of the other authoring languages has a tracing function which allows a programmer to see how the rules are fired. The tracing function of Intelligence Compiler is superior to that of the other two shells in terms of its convenience and flexibility. Without such a tracing function, the efficiency of the system development is compromised.

c) In order to read users' input accurately, TAA needs various types of menus from which a user can make his or her choice(s). Intelligence Compiler has sets of built-in commands which allow such menus to be constructed easily, but the other two shells do not have similar built-in commands.

d) When it helps users to develop an achievement test, TAA needs to reorganize and refine a set of data iteratively; therefore, an appropriate development shell must have corresponding built-in commands (e.g., no-backtracking, recursion, variable substitution) to perform such tasks. Among the three expert system development shells, Intelligence Compiler has the richest variety of such commands.

e) The operation of TAA requires efficient variable substitutions because many inference procedures and the structures of screen displays will be used several times. Neither Hypercard nor Course of Action support efficient variable substitution.

Because of the above technical weaknesses with these two authoring languages neither of them was chosen as the development tool for TAA although they enable a programmer to build a high quality user interface easily.

Description of End User

TAA is designed for teachers, trainers and researchers who will design and develop test instruments to measure achievement in the cognitive domain. TAA is supposed to meet the different needs of different people: from those who have a minimum amount of knowledge in teaching and evaluation to those who are experts in test design. TAA also accommodates the user who has little knowledge about computer programming.

System Requirements

The primary purpose of TAA is to assist the user in completing the test design task successfully. Furthermore TAA should help the user gain a deeper understanding of the design process. In order to fulfil these two purposes, TAA should be able to:

1. lead the user through a process that an expert will go through in his/her design process;
2. give easy access to alternatives in designing a test;
3. be used partially for a particular task;
4. assist the user in making systemic decisions in choosing alternatives along the design process;
5. offer the user the background knowledge in the domain of designing a test instrument;
6. give the user its justification and explanation about its advice. The justification should reflect the overall concern or strategies in the professional design process;
7. communicate with the user in a natural way;
8. give assistance to the user tailored to the users' needs;
9. substantially reduce the design time required if they are done manually and properly.

TAA has two major limitations: one is that it automates only the procedural and technical, but touches little of the ethical (philosophy) and aesthetic (artistic) aspects of test design, and second, it cannot help its users to produce graphics as a part of a test item.

Although the tasks in different phases (e.g., design an experiment or evaluation, design testing instrument and interpret results) of an experiment or an evaluation are governed by a single set of principles (validity, reliability and feasibility), TAA deals only with the test design stage of the overall evaluation process: it does not touch the tasks and its related issues in experimental design or in the analysis of test results.

TAA helps the user to complete six major sequential tasks (See Appendix 1, Figure 1). The flow charts in Appendix 1 show how the procedural tasks are performed by TAA. Please refer to Appendix 6 to see the objects and their values used by TAA to perform these tasks. The following discussion explains the task sequence and the overall strategies of TAA operation. If you want to see how the decisions and advice are made, please refer to the coding of the backward rules on one of two diskettes attached to the thesis.

The six major tasks that TAA performs are :

1. gather background information about the test;
2. formulate performance objectives;
3. choose test format;
4. develop test specifications;
5. decide the length of the test;
6. write up test items.

In the first phase of the design process, a user (test designer) defines the purpose and preconditions of the test. The purpose of the test refers to what types of decisions will be made from the results of the test. The preconditions of the test will be the external constraints of the test. Some of the constraints cannot be changed for the test (e.g.,

testees' education level); while others can (e.g., length of test time or method of scoring the test). Factors which will influence the decisions in the test design process will be gathered and recorded in this stage (See Appendix 1, Figures 1.1).

In the second phase, the user (test designer) will consider what types of performance (Bloom, 1956) should be measured for the purpose of the test. From there, the test designer will formulate the behavioral objectives as targets of the test. In this phase, the importance of different types of performance defined according to the purpose of the test and benefit for the testee will be clarified (See Appendix 1, Figures 2.1, 2.2 and 2.3).

In the third phase, TAA and the user (test designer) will jointly decide which of the six test formats will be used for a particular behavioral objective. The analysis is based upon the compatibility between the type of performance that an item format can measure and a particular type of performance to be measured. Having decided what type of test item will be used for a particular behavioral objective, TAA will decide the time needed for a testee to complete the test item (See Appendix 1, Figures 3.0, 3.1 and 3.2).

In the fourth phase, by using a matrix, TAA will help the user to develop a test specification as a blueprint for the test. The blueprint is drawn based upon the intentions of the test. By defining a score distribution among the tested topics with the corresponding levels of performance, a test specification should express the comparative importance of levels of performance corresponding to the topics. In the process of constructing a test specification, TAA will present the matrix of a test specification on the screen with topics displayed in the first column and performance levels on the first line. Based upon the purpose and the type of the test, TAA will advise the user about the appropriated focus, difficulty level and coverage for the test, and then TAA will require that the user puts in the subtotal of scores distributed at each level of performance and with each topic. TAA will calculate appropriate cell values and assign them to each cell which represents the score value for each topic at a particular performance level. The

calculated cell values are the closest ones to the user's desires, which is based upon the score distribution subtotals put in by the user on the performance levels and the topics. While calculating the score distribution, TAA adjusts the inconsistency of the subtotals that the user has put in. Please refer to Appendix 1, Figures 4.1, 4.2, 4.3 and 4.4 for detailed calculation formulas.

In the fifth phase, TAA will help the user to decide which objectives and how many test items for each objective should actually be included in a test. The decision is based upon the test time available and the test time required for objectives (testing time required is calculated by TAA). The actual score distribution among levels of performance with topics are balance according to the test specifications developed in the fourth phase. The discrepancy will be identified by comparing the planned value and the actual score value at each level of performance within each topic. The discrepancy occurs when too many or insufficient objectives are included in a test according to the test specifications. The difference among them would be balanced by adjusting the number of objectives or the test items of a particular objective or the test specifications (See Appendix 1, Figures 5.1 and 5.2).

In the sixth phase, TAA will assist the user in writing individual test items according to the types of test items (See Appendix 1, Figure 6.1).

These six major tasks are further decomposed into more concrete subtasks. At each stage TAA is able to give advice on how to perform the task and on how to use the computer. The related concepts and principles are ready for the user to review along the test design process.

System Functions

To assist the user in completing the six major tasks mentioned above, TAA plays three roles. The user can choose any combination of them. The roles it plays are:

1. an adviser
2. an instructor

3. a compiler.

As an adviser, TAA draws conclusions and makes decisions as other expert systems do. Since the design of achievement tests is an open-ended process, TAA sometimes tells the user upon what factors a decision has been made, and then it asks the user's opinion about the decision. That means the user has the final say on the decision, provided that the principles and examples of making a decision on a particular task are presented to the user. Besides the explicit conclusions that it will give (e.g., decisions on which item format should be used), TAA also gives the user advice in a rather directive manner. For instance, TAA will look at the purpose of a test and then suggest difficulty level, coverage and focus of a test be used. Such suggestions are subjected to interpretations of the user (e.g., focus in a test to measure prerequisite skills should be the minimum skills that a learner needs in order to start instruction). More specifically, it is the user's (test designer's) responsibility to decide what are the minimum skills to be tested.

As a compiler, TAA handles the basic administrative tasks in test design. The tasks of book-keeping and administration for an expert system dealing with a design problem is heavier than that with a diagnostic problem, because a test design program, like any other design application, is not only advisory but also constructive in nature (Freksa, 1986). That means, in its process of continuous refinement, a great proportion of data and information has to be processed and reconstructed several times before the test is completely developed. For instance, after general objectives are formulated, all of them will be retrieved and refined into specific ones. Also, when a test designer writes test items, it will be easier for him or her to write them according to the topics; however, the developed test items of a test should be rearranged according to their test types (e.g., starting with true-false type, followed by multiple-choice items, and then essay type questions), and only after that can the test be implemented. This order reflects how most tests are arranged (Mehrens & Lehmann,1978). To reduce this burden of trivial, but important and time consuming, tasks for the test developer in writing test items, TAA

presents the objectives to the user by topics and under the heading of a topic the test items are developed. TAA will later reorganize the test items according to the format as a ready-to-use test. Besides retrieving and reorganizing data, TAA does many calculations in the test design, such as calculate score values and develop test specifications.

As an instructor, TAA gives the user instructions on the concepts and principles which a test designer should understand in order to develop a validate and reliable test. The instruction also puts the immediately related concepts and principles into a larger context in terms of how these concepts and principles are related to the others. Please refer to Chapter 5. "System Design and Development" for a detailed explanation of the TAA's instruction function.

CHAPTER 5

System Design And Development

Content and Scope of Domain Knowledge of TAA

The most general way to describe expert system design is to state the scope of the domain knowledge in which the system operates and then to decide how a knowledge representation should best be structured for easy understanding, and for effective and efficient operation. Both decisions (the scope and the structure of knowledge representation) depend upon a good understanding of the domain knowledge.

The sources of the domain knowledge are mainly Bloom's (1956), Mehrens & Lehmann's (1975) and Gronlund (1981)'s works; as well as the subject matter experts (domain experts) associated with the project at Concordia University.

Basically, the knowledge of any domain can be modeled in two ways: either by the information dependency for the problem-solving process or by the formal structure of the principles and concepts. The model of information dependency concerns "what task should be completed first" and "how decisions are made in each task" while the model of theoretical knowledge concerns "how the concepts and principles of a domain can be logically classified". With an expert system solution for improving on-the-job performance purpose, domain knowledge is mostly modeled only by information dependency for decision making. With an expert system for educational purposes, domain knowledge is modeled by the formal classification of concepts and principles.

If an expert system purely gives advice on how to perform the task, it will be sufficient and effective to model the domain knowledge only according to the information dependency of the decision process. Since TAA has a second function, that is, to teach the user the deep and overall knowledge underlying the task, it is necessary that the theoretical and background knowledge is incorporated into the information dependency model and structured by the learning prerequisite .

Based upon the above consideration, for TAA's operation, the domain knowledge is modeled from two angles: one according to the information dependency for the decision making, the other according to the formal concept and principle structure in achievement test design. Because the primary purpose of TAA is on-the-job problem-solving, the theoretical knowledge is incorporated within the structure of the information dependency module. The most important principles of achievement test design (i.e., validity, reliability and feasibility) are used as the overall control strategy, and they are modeled within the theoretical knowledge structure as a learning content. Figure 2 illustrates the information dependency of the test design process. Figure 3 illustrates the logical structure of the theoretical knowledge and how it is integrated into the decision making model of what to measure and how to measure; and its overall control.

From the decision-making point of view, a test developer will specify "what to measure" and eventually arrive at the decisions regarding "how to measure". "What to measure" refers to what type(s) of performance will be chosen as the target of a test and this decision should be derived from an analysis of the purpose of the test and characteristics of the testees. Concerning the compatibility of the two, the decisions on "how to measure" depend upon the "what to measure", and the external constraints of a particular test. In terms of "how to measure," the test designer must decide:

1. what combination of test item types (e.g., multiple choice, restricted response), should be employed;
2. the length of the test (e.g. how many items are needed for each performance measured) and
3. the wording of the items (which refers to the actual construction of individual test items).

The decisions on "how to measure" questions are not isolated ones but rather are interdependent. For instance, the decision of choosing an item format depends on the

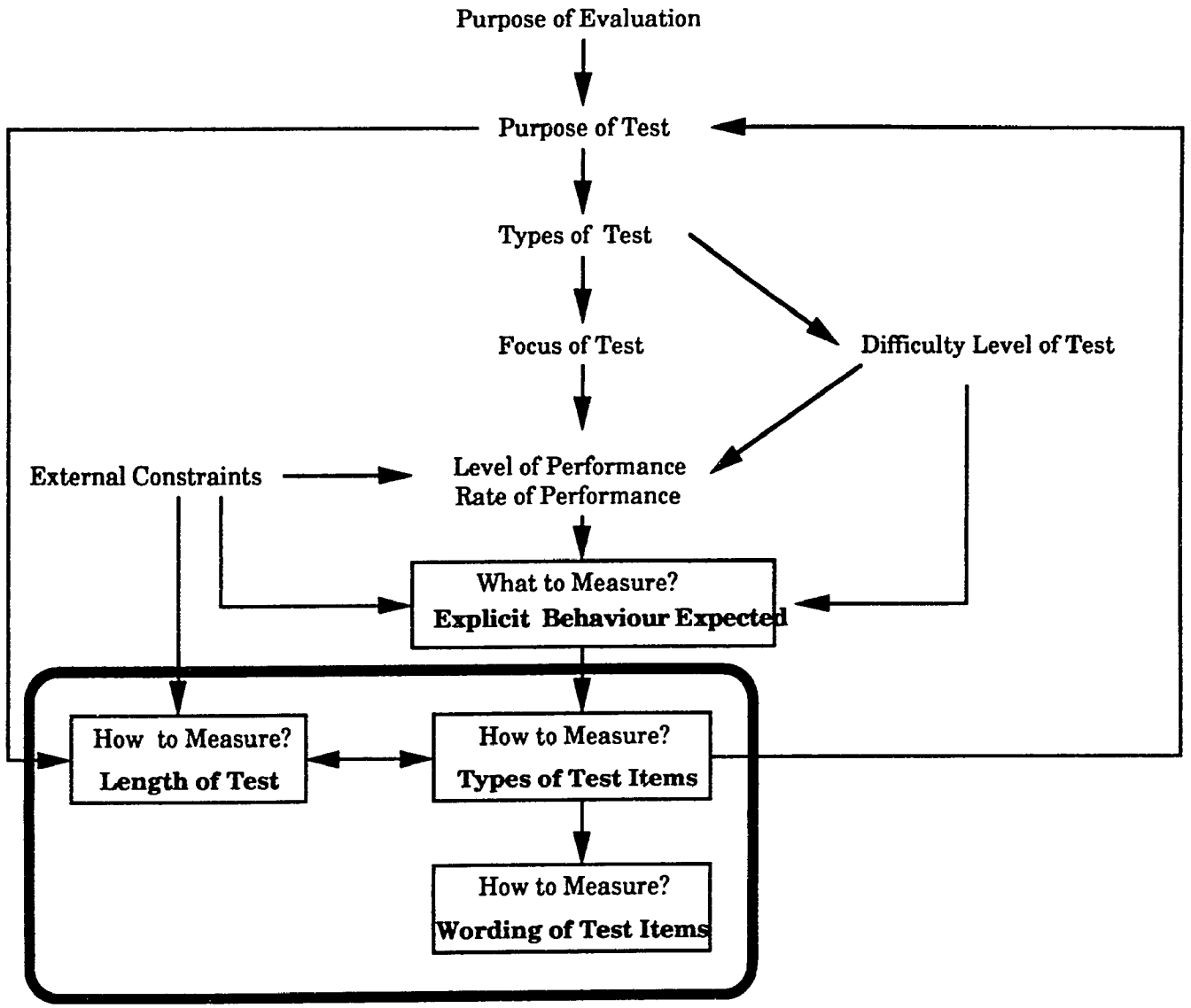


Figure 2. Information Dependency of Achivement Test Design Process

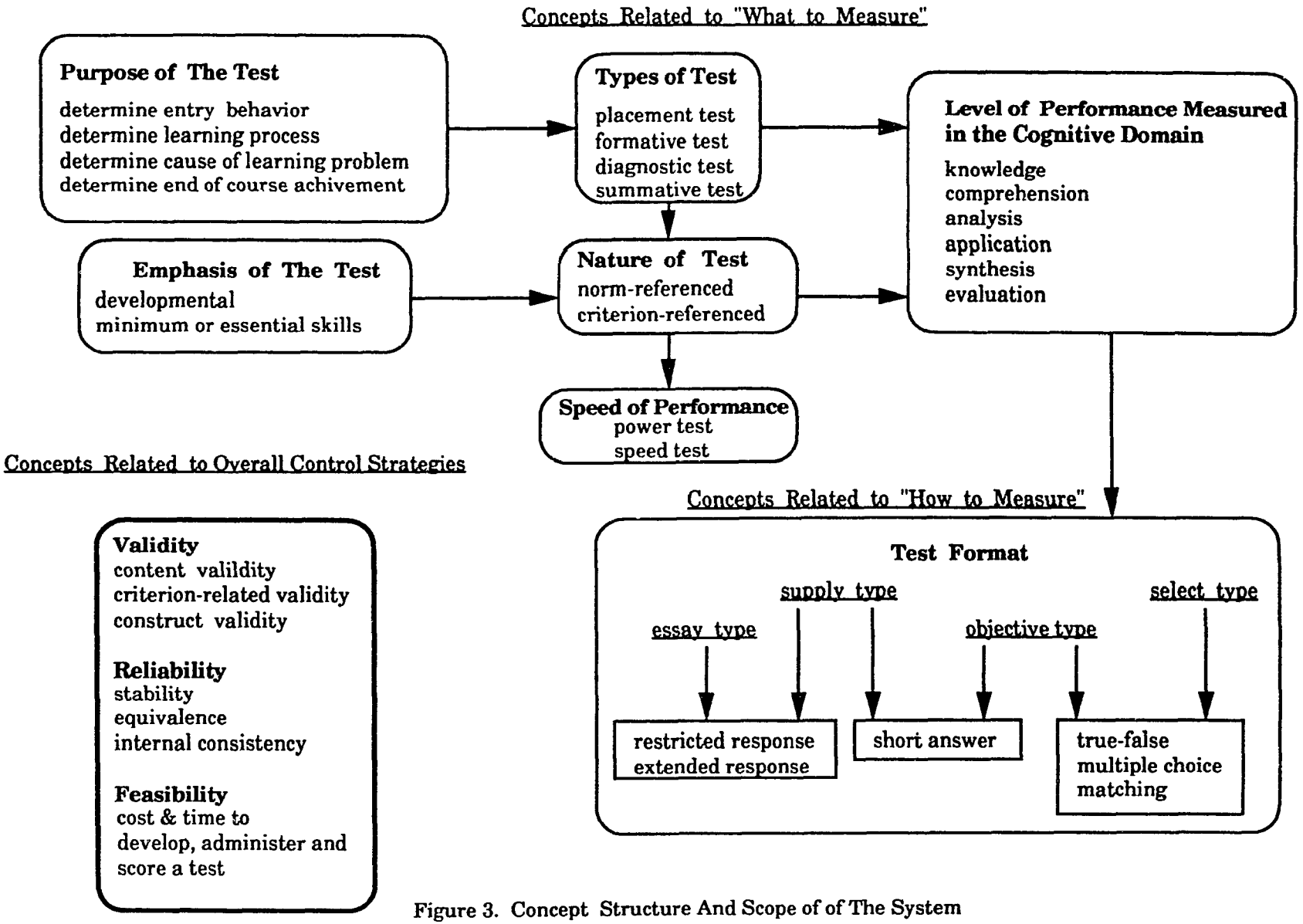


Figure 3. Concept Structure And Scope of of The System

performance it measures; and the decision on the length of the test is based upon the degree of reliability required, formats the test employed and the test time available.

From the classification of theoretical knowledge point of view, the concepts and principles of the achievement test design are modeled under three titles corresponding to the structure of the decision making in the achievement test design :

a) what to measure;

b) how to measure; and

c) overall control strategy.

Regarding the concepts and principles under the "what to measure" category, Bloom's (1956) taxonomy is adopted as the scheme to represent levels of performance to be measured. The taxonomy consists of six levels:

1. Knowledge
2. Comprehension
3. Analysis
4. Application
5. Synthesis
6. Evaluation

The knowledge level is the lowest level and the evaluation is the highest. A higher level encompasses lower levels of performance.

The six most popular test items formats are classified under the "how to measure" category. Choosing an item format, like true-false or multiple choice, is based upon a particular performance in focus and what level of performance the tester is interested in. Figure 3 shows the classification of the test item format. Also "difficulty level", "focus" and "coverage" are the concepts and principles in the "how to measure" category.

The most important principles which affect the test design process as a whole are those of validity, reliability and feasibility. They are the overall control strategy in the test design. Validity refers to the extent that the results serve the particular uses for which

they are intended. Reliability refers to the consistency of measurement (Gronlund, 1981). Feasibility is a practical consideration in the design of a test, such as test time available and resources available to construct and score the test. Validity and reliability are governed by the purpose of an evaluation; they are both constrained by the feasibility in the situation of a particular test and affected by how the test items are structured and worded.

The above discussions describe how the practice and the static knowledge of achievement test design are conceptualized for the TAA's operation. In the following sections, the discussions focus upon the implementation issues of TAA.

Test Authoring Assistant to Solve Design Problems

Text Authoring Assistant (TAA) is an expert system application to help its users to perform design tasks. Because of the disadvantages (the impossibility to pre-form explicit solutions and the wide coverage of the subject matter) embedded in the nature of design tasks, the major challenge of TAA's design is to balance the coverage of the expertise, the abstract level it should operate at and the search space the operation requires.

To reduce TAA's huge searching space required to deal with the wide coverage of the subject matter of a test, TAA is designed to operate at a more abstract level of principles when concrete solutions are practically impossible to be formulated. For instance, TAA does not always deal with the subject matter specifics (e.g., math, language, social science) of a test, while the type of subject matter will affect the global approach in the test design. Considering the relationship between the subject matter of a test and the performance to be tested, TAA may advise its user not to overemphasize the knowledge of names, dates, places, and the like, if the subject matter of a test in social science.

To overcome the impossibility of formulating explicit solutions on-line, TAA is designed to communicate with its user about the abstract principles which are illustrated by examples. For instance, TAA does not always deal with test specifics, such as word

test items; however, when a user/test designer is writing an test item, TAA will explain the to her/him the components of a particular item type (e.g., the meanings of the answer and the distracters in a multiple choice item) and how to construct a test item of quality (e.g., avoid using double negatives in a question).

TAA, which does not produce explicit solutions at every point but guides the user to make informed decisions, is very beneficial to its user, because such a system forces its user to use his or her own judgement. In a joint decision making process, the users' understanding of the task will be improved if the program has appropriate coaching capabilities. As a result, a user may detect errors embedded in the system rather than blindly accepting the system decisions. With the above approaches (communicating the principles of achievement test design with its user and by guiding its user to make informed decisions instead of feeding her or him the ready-to-use solutions), hopefully, TAA will accelerate the process of that user becoming an expert.

Knowledge Organization

In order to accommodate various needs of the end-users, TAA is designed to allow the user to use it as an advisor only, or as an advisor with instructions on theoretical knowledge, or with a proportional combination of different types of assistance. To meet the above various users' needs, the domain knowledge is organized adopting the instructional model of Reigeluth-Merrill's Elaboration Theory of Instruction (Reigeluth & Stein, 1983).

Procedural knowledge as epitomizing conten. The tasks which TAA performs are procedural-decision oriented. Thus, according to the Elaboration Theory of Instruction, the procedural knowledge of the achievement test is chosen for the operation structure (epitomizing content) of TAA because it appears more natural to the human expert(s) and TAA designer if the theoretical and background knowledges are used in the supporting components to help users reach optimal solutions. The essence of this approach is to treat

the decision process and theoretical knowledge as a unified whole with the decision process as the primary structure.

The model of procedural knowledge represents the order in which experts work through different levels of abstraction. The concept of this top-down analysis is simple: reduce the problem down to manageable subproblems and deal with them locally. Appendix 1 represents the primary structure, or more precisely, an optimal task sequence in the achievement test design. The six major tasks are further decomposed into subtasks until no further subdivision and specification is necessary. Although there are alternative paths among subtasks, the relationship among these six major tasks is sequential: it is appropriate to complete all the subtasks of a major task before proceeding to the next major task, even though the decision may not be valid or optimal. In other words, a test designer may go back to revise the decision made before, a normal practice even with professional test developers, but s/he will try to complete certain task(s) before s/he move on to another.

Although technically there are not very many alternative solutions for each individual task, it is complicated to choose among alternatives consistently and coherently with the whole test in mind. For instance, there are a finite number of test types (e.g., norm-referenced test, criterion-referenced test, power test, skill test ... etc.) and item formats (true-false, multiple choice ... etc.) that the test designer can choose from; however, it is a subtle matter how these can be arranged in good combinations considering validity, reliability and feasibility. Unlike computer configuration tasks and architectural designs, compatibility among the components in designing an achievement test is not clear cut, but falls in an approximation range of balance. To reduce the difficulty of the system design and operation, one should use procedural knowledge as a primary structure with inference knowledge embedded in it.

Although Elaboration Theory of Instruction fits TAA's purpose as the primary structure of TAA design, the procedural knowledge as defined in Elaboration Theory

deals with only elementary steps in performing a task (e.g., to know the steps of a chemistry experiment); thus, it is too simple and too general for the design of an expert system to perform achievement design tasks. In TAA's design, therefore. the structure of procedural knowledge is re-defined in order to be used in the application. As a result of this re-definition process, the procedural knowledge of TAA contains the following components:

a) Task - gives a definition of a task;

b) Purpose - refers to why a task has to be completed in test design;

c) Procedure - contains steps required to complete a task;

d) Advice - explains in what manner a task should be completed;

e) Example - illustrate ideas that TAA expresses

f) Conclusions - output results of TAA inference.

Messages in the "task" category give a user/test designer the name and its definition of a task. For instance, the definition of a table of the test specifications states that it is a blue print of a test design which indicates the relative emphasis given to each aspect of evaluation (Gronlund, 1981).

Messages in the "purpose" category are very similar to "cognitive-strategy" as defined in Elaboration Theory. The messages in this category are supposed to bring the users beyond an operational level and help them to understand the advice given by TAA. To achieve this, TAA tells its user why a task to be performed is necessary in achievement test design. For instance, before s/he constructis test specifications, TAA will tell a user/test designer the purpose of constructing a test specification: "The table of test specifications are used as a tool to obtain a representative sample, because a test, no matter how extensive, is almost always a sample of the many possible test items that could be included. That means that our limited samples must be selected in such a way that they provide as representative a sample as possible in each of the various areas for which the test is being developed".

The messages in the "procedure" category tell its user the necessary steps to complete a task. Usually, it can be expressed by step 1, step 2 ... . For instance. the steps in building the table of a test specification are:

Step 1. obtaining the list of performance objectives;

Step 2. outline course content;

Step 3. preparing a matrix specifying the nature of the desired test sample.

The messages in the "advice" category will tell its user the strategy(s) or the method(s) that should be employed in a task. A piece of advice can be very abstract and concrete examples are impossible to create. In this situation, TAA will direct its user to consider certain factors and draw their own conclusions. For instance, when a user/test designer is constructing a test specification or s/he is choosing what items should tested, TAA will tell her or him what difficult level and the coverage should be. Because it is impossible to give concrete examples to illustrate what a test with a high difficulty level will look like, TAA will advise its user to include both easy and difficult test items to permit scores to spread out over the full range of the scale (provided that the test is a norm-referenced test). To make this advice as specific as possible, TAA will tell its user to eliminate those items that all testees are likely to answer correctly or wrong, and include the items that half of the testees may answer right. Sometimes, when a piece of advice is abstract, a concrete example can be cited. For instance, when a user/test designer is writing a short- answer type test item, TAA will advise her or him not to just copy a statement from a text book, because it is usually too general and ambiguous to serve as a good short-answer item (Gronlund, 1981). For instance, the example (Gronlund, 1981) that TAA cites to illustrate the point of the above advice is the comparison of a statement taken from a textbook and a revised form for a short-answer test item:

Poor: Chlorine is a (halogen).

Better: Chlorine belongs to a group of elements that combine with metals to form salt. It is therefore called a (halogen).

The textbook statement version will more likely solicit the answer "gas". The revised version measures important knowledge which does not depend on the specific phraseology of any particular text book.

The examples, such as the one mentioned above, belong to the fifth component of the TAA's procedural knowledge. Examples are used not only in performing the design tasks, but also they are used in the instructions.

The sixth component of the procedural knowledge is the concrete solutions that TAA formulates, such as deciding an item format of an test item. The conclusion is suggestive, which means that a user can revise it. The revision of TAA's solution should have some restrictions, e.g., a user may be advised not to revise TAA's solution in a particular way under certain circumstances. At this stage, such restrictions have not been defined.

The above described six types of TAA's output messages belong to the procedural knowledge which is the epitomizing content (operation structure) of TAA; the next two sections describe the supporting content of TAA. The "communication with user" section of this chapter will describe how the procedural knowledge is related to the other types knowledge.

Theoretical knowledge as supporting content. In terms of "epitomizing", at the beginning of each of the six major tasks TAA presents to the user the definition of the task, its purpose and procedure. The advice on how to complete a task is inserted whenever it is necessary. Above mentioned are the components of the procedural knowledge. The theoretical knowledge is the supporting content to the procedural knowledge, which consists of the concepts and principles of achievement test design.

The theoretical knowledge is communicated with a user/test designer as instructions. Several characteristics of the instructions should be mentioned: first, the

messages as instructions do not summarize or overview all the content involved at the beginning of a particular task, but rather, at the beginning of a task, the content of an instruction is restricted to a concrete, meaningful, application level. Only at the end of a task, does the instruction systematically review what has been learned in a major task and the messages of this type are classified under the title of "summarizer". Only the immediately relevant content of the task is included in "summarizer".

In order to increase the retention of the users, it is important that the instruction help the user to create additional links among the parts of the knowledge, and between the new knowledge and a user's relevant prior knowledge. Based upon the above consideration, important concepts and principles of more complexity and further distance from the core knowledge are organized under the title of "synthesizer". At the end of each task, the knowledge in a "synthesizer" will be presented to a user showing how the newly learned ideas are related to other concepts and principles. Actually, a "synthesizer" is an extension of a "summarizer" since it places what has been learned in a larger context.

Summing up, the theoretical knowledge is organized as supporting content to the procedural knowledge. By demonstrating to the user the relationships among the ideas and principles, the "summarizer" and "synthesizer" improve retention of learning. Except for the "synthesizer" in the instruction, which build links of wider scope, all of the instructional messages are given locally. In the other words, all of the instructional messages, except messages of the "synthesizer", concern the problems at hand to satisfy the users' immediate needs.

Figure 4 and Figure 5 show how the theoretical knowledge is embedded in and supports the procedural-decision making tasks of "3. Choosing test item types" and "4. Constructing test specifications" in the test design process respectively.

Prerequisite knowledge as supporting content. Because TAA is supposed to support end-users with varying knowledge and experience, in addition to the theoretical knowledge, the prerequisite knowledge is defined as the second supporting content to the

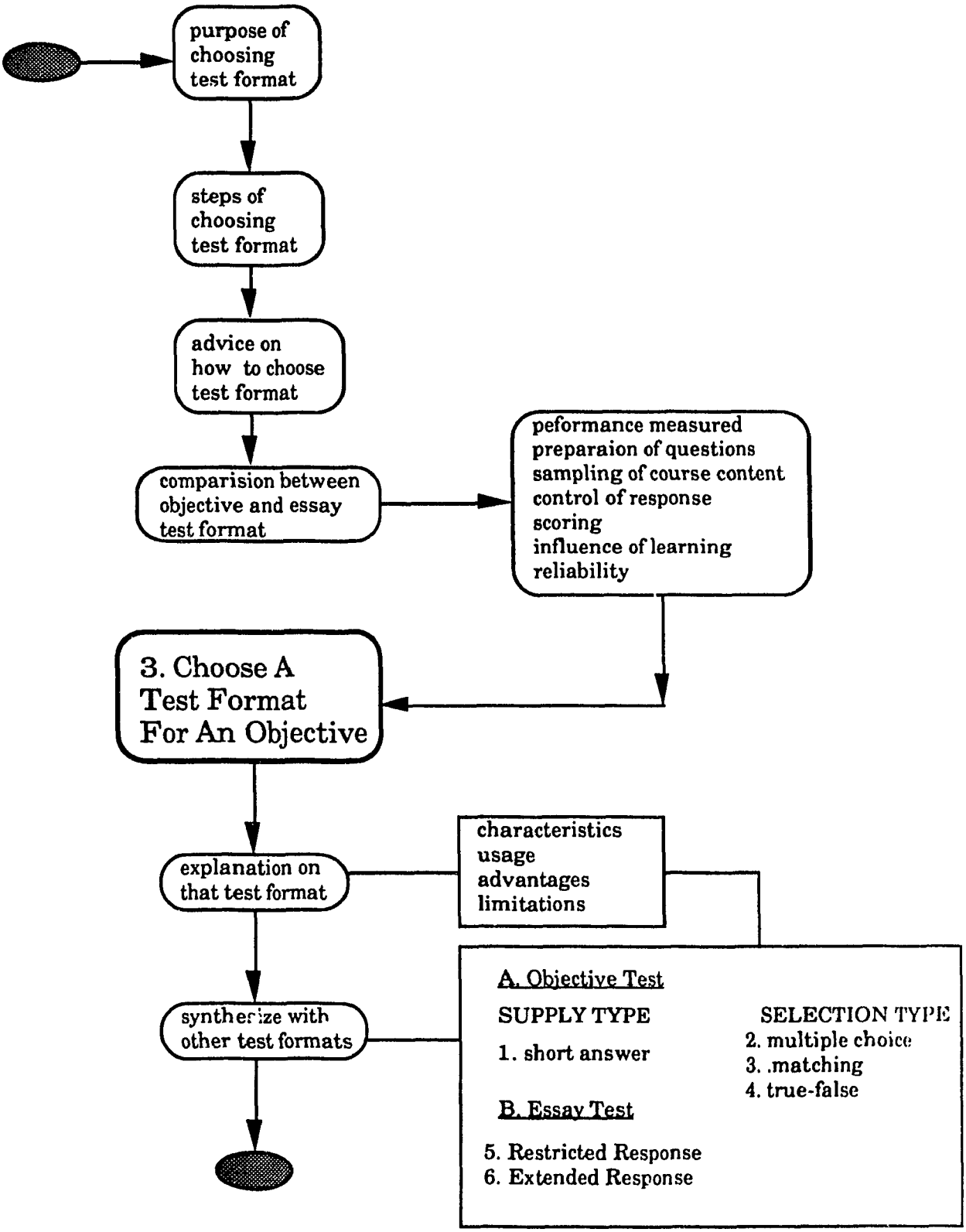


Figure 4. Assistance Given In Task 3. Choose Test Format

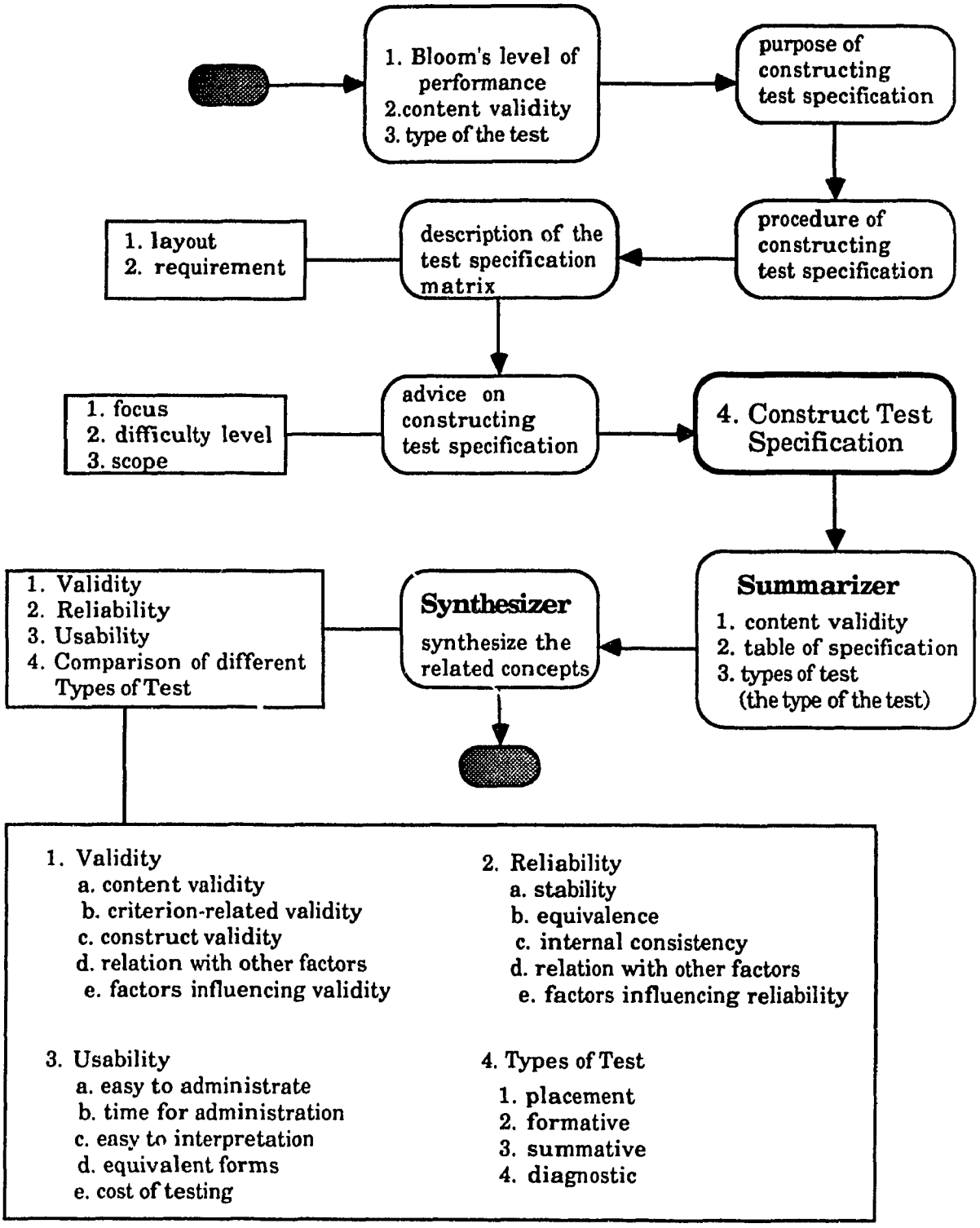


Figure 5. Assistance Given in Task 4. Construct Test Specification

procedural knowledge. The prerequisite knowledge is related to the knowledge that user will learn during the process of completing the task: it refers to the minimum knowledge that a user must possess before s/he is able to proceed with the task and its related instructions. The prerequisite knowledge and learning hierarchy are specified and structured within the boundary of each of the six major tasks of achievement test design defined for TAA. Only the minimum essentials are given to the user in order to reduce the load on the user to complete the task and at same time learn background knowledge. Figure 4 and Figure 5 show how the prerequisite knowledge is used as a supporting component for "3. Choosing test item types" and "4. Constructing a test specification".

Knowledge Representation

Representation of knowledge requires that relevant objects in the knowledge domain be named, described, and organized, and that relationships between objects be expressed, including constraining relationships that govern the storage and retrieval of object properties (Parsaye & Chignell, 1988). The knowledge representation design of TAA is based upon the following requirement: the knowledge representation should ensure that the system functions are carried out in an efficient and effective manner (Parsaye & Chignell, 1988; Walters & Nielsen, 1987); it should reduce the difficulty and efforts in TAA's development and maintenance; and it should support TAA in giving the users justifications of its advice and explanations of its actions. To meet these requirement, the knowledge representation of TAA is seeking a psychologically validity with the knowledge structure of a domain expert(s) (Clancy, 1988), and also sufficient pragmatic utility to allow for manipulating the knowledge (Hartley, 1985).

TAA uses backward chaining rules, lists, frames, and canned text as the knowledge representation methods. The backward chaining rules are used to carry the inference process to accomplish the tasks and subtasks in the test design. These tasks and subtasks (represented in Appendix 1) can be viewed as nodes in a decision tree or as goals which have to be reached. Backward chaining rules are best for the focused inferences. The goal

of a group of rules reflects the purpose and direction of a particular task. Using backward chaining rules clearly reveals the decision-making process in the test design and allows easier modification of the program. Figure 1 in Appendix 2 shows how the decision process is structured and inference is carried out through backward chaining production rules.

The knowledge representation also uses meta rules in the form of backward chaining rules. The meta rules do not make actual inferences but rather index production rules for inference (See Figure 2 in Appendix 2). This approach increases the efficiency of the implementation because it allows the development and revision done locally without interfering with other parts of the coding.

In order to reduce the memory space required by TAA's operation, the system uses the variable substitution method to code those procedure structures which will be used more than once by TAA's operation. This type of coding is also represented through backward chaining rules (See Figure 3 in Appendix 2).

Frames are used to store the information in which the facts can be grouped by a certain category, e.g., the data about the testee will be stored under the frame named Testee (See Figure 4 in Appendix 2), or to store the information which has an inheritable nature (See Figure 5 in Appendix 2). For instance, both true-false and short answer share their parent's nature of objective test item format, but they are distinctive in terms of that true-false is select type and short answer is supply type; therefore, in storing this type of information of "true-false" and the "short answer" are coded into two frames and they share the same parent of "supply type". Using frames in the above situation will require less programming effort than that of other information storing methods.

Lists are used to store the items which will be presented to users in menus. From a menu a user may chose one or more items (See Figure 6 in Appendix 2).

Canned texts are used to store the static knowledge which may used as messages displayed to the user. They include information, advice, solutions and instructions (See Figure 7 in Appendix 2). Variable substitutions are used in many canned messages.

Generally speaking, the inference knowledge is represented by backward chaining rules. Three types of backward chaining rules are coded in TAA's inference engine: the productions rules are used to express the actual reasoning of TAA; meta rules are used to index the productions rules; and a production rule with a variable substitution(s) is used wherever the structure of a inference procedure will be used repeatedly. The factual knowledge, which is the material for the inference of TAA, is represented in the form of frames, lists and canned messages. If a group of concepts can be categorized or they share a inheritable nature, then, they are stored in frames. Messages for the advice and the instructions are canned messages. Lists of items presented to the user to choose from are recorded. Please refer to Appendix 6 to see the objects and their values used in TAA.

Architecture of TAA

Figure 6 illustrates the architecture of TAA. At the most abstract level TAA has two parts:

1. knowledge base
2. inference engine

In its knowledge base, four types of factual knowledge are kept in separate locations in the form of lists, rules or frames. The first type is the facts of the domain knowledge (e.g., taxonomy of purposes of achievement tests, types of item formats, classification of verbs corresponding to levels of performance). The second type of factual knowledge is the contextual information about a particular test which is either put in by the user, (e.g., purpose of the test, educational level of the testees), or generated by the machine, (e.g., test type, item format and number of items for each objective and score distribution among levels of performance with topics). The third type of factual knowledge is assistance messages to the user, including both instructions on theoretical knowledge and

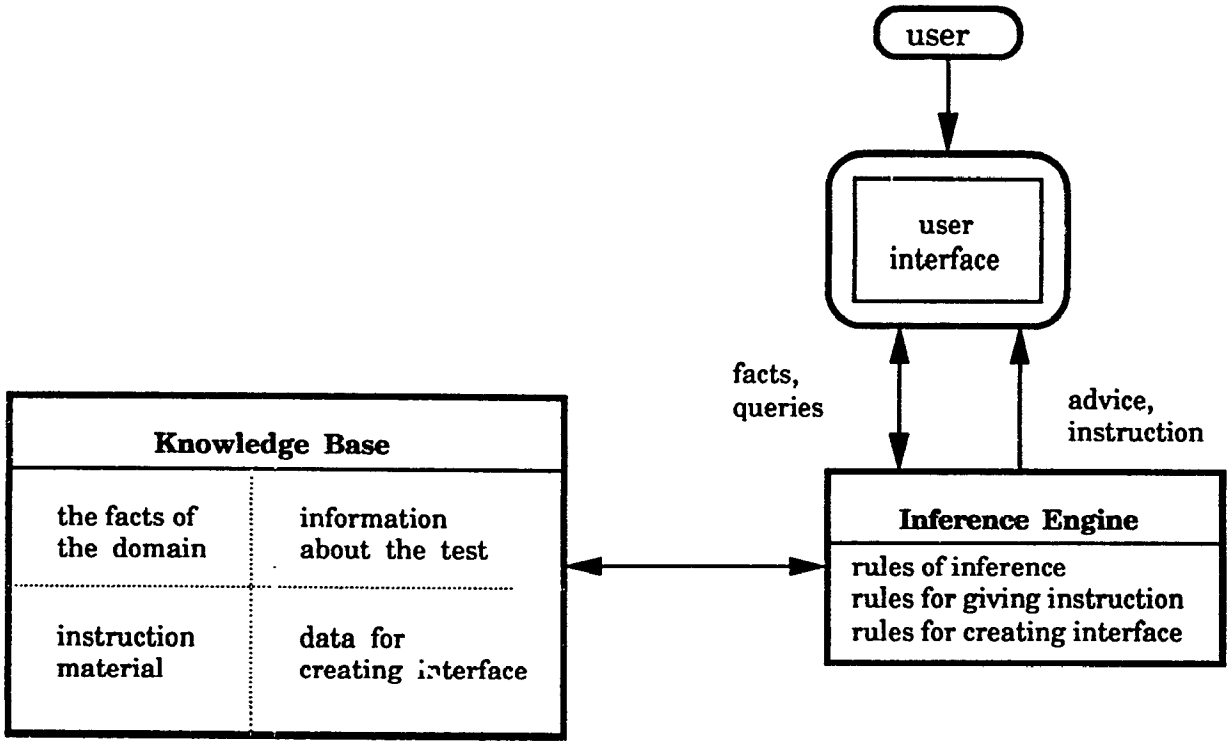


Figure 6. System Architecture

suggestions for practical problem solving purposes. The fourth type of factual knowledge concerns interface design and it is of interest only to the programmer.

The inference knowledge is the knowledge on how the factual knowledge are to be used. The inference knowledge is separated from the four above mentioned types of factual knowledge and is recorded in the inference engine of TAA. Three types of inferences defined for TAA's operation are in the form of backward chaining rules. The first type is to make decisions about the tasks in designing a test. The second type concerns when and what instructions should be given to the user. The third type relates to the process of constructing the machine-user interface. These three types of inference knowledge are indexed by the meta rules according to the tasks and their sequence of the achievement test design. The goals of the meta rules reveal the nature of a particular inference.

Problem Solving Strategies

According to Wolfgram (1987) problem solving strategies refers to the methods in three categories: 1. search model, 2. control mechanism, 3. reasoning strategies. The problem solving strategies should reflect the nature of the expertise in the field to enable effective communication among the user, the expert and the program designer.

The nature of designing a test is basically a process of iterative refinement: the previous decisions will be used as the input for the next one. In the iterative refinement process, some of the inferences are procedurally or sequentially oriented reflecting an optimal sequencing in designing an achievement test that experts in this field can commonly agree upon. While there is an optimal sequence in the supertasks of test design, the design process is not linear. The true meaning of an optimal sequence is at a very abstract level, (e.g., developing performance objectives first and then choosing what item format will be used for a particular objective). Appendix 1 shows an optimal task sequence in achievement test design at an abstract level. Although those flow charts reveal some of different possible paths and also parallel paths in the test design process, it

does not show how a decision is made in detail (e.g., it shows the conditions of the branching in choosing an item format, but it does not show the conditions under which a particular test item format will be finally chosen). To review the actual reasoning, refer to the coding on two diskettes attached to this thesis.

According to the above mentioned two natures of solutions in a test design process, the search model is composed of two aspects:one is the searching of procedure oriented task and the other is the searching of situation dependent task. TAA mostly uses a depth-first search strategy for the procedure oriented case in which no contextual information is needed to decide which path (node) to search next. However, when situational information is needed in an infernece, a thorough search down a single path will sacrifice the efficiency of the operation if it finds the solution is not valid only at the very end and has to backtrack to the beginning. To improve the efficiency of the operation, when the search path is too long or there are too many possible solutions or branches of the paths, the heuristics used by experts are built into the searching model which reflect the experts' working style in their decision making. The search is very similar to the 'best-first' notion where path selection is not arbitrarily sequential. During a 'best-first' search the conditions of the possible paths are evaluated and the most promising one will be selected for the next level of evaluation. The searching strategy actually backtracks up the decision tree to a higher level and then repeats the process if any constraint is met. For instance, when TAA chooses an item format for a particular objective to be tested, it may first look at the level of performance that the tested behavior belongs to. From there it will have a general idea of whether the essay form will be used or not. If an essay form of item(s) should be used, then TAA will choose the item type from extended response or restricted response. But if the performance level of an objective is at the knowledge level which is not suitable to be tested by an essay item, TAA will not go further along the essay type alternatives; rather it will check the

possibilities in other types of item format. Figure 7 in Appendix 2 shows how the heuristics are build into the depth-first search in "4. Choosing Test Item Format".

The control mechanism is designed according to the nature of how the inference is controlled in reality. From the technical and abstract point of view, the alternatives (nodes) in each task of the test design are limited; therefore, the inference of TAA uses only backward chaining rules as a control mechanism, even though it is a design problem rather than a diagnostic problem. When the inference is sequence dependent, its rules are grouped together under a goal according to the design algorithm which reflects the expert search strategy. By isolating rules that are most relevant to the task at the moment, the control strategy limits the search space: the operation proceeds at a particular level of abstraction without the need to cope with the details at another level or branch.

To reflect the nature of test design, the procedurally oriented inference should be enforced by a control mechanism. Equally important, the control mechanism should ensure flexibility for an efficient operation. That means it should be give the flexibility when alternatives must be made situationally. Where there are parallel alternatives in a decision making process, the clauses of that set of rules are related by "or". For instance, the user will be asked the question of "has quota?" (see task 1.1 in Figure 1 in Appendix 1) only when one of three conditions is true:

If

purpose of test = Job Assignment

or

purpose of test = Grading

or

purpose of test = Certifying.

Many expert systems use certainty factors, semi-exact rules or confidence factors as their reasoning strategies to improve robustness. Robustness is a measure of an application's ability to continue to produce correct judgments or outputs in the face of

deteriorating input (Summerville, 1989). Because the domain knowledge is reliable and static, TAA use only a straight forward monotonic reasoning strategy, that means the hypothesis of a rule will be returned a value of either true or false after an inference without certainty factors, semi-exact rules or confidence factors being employed. The reasoning strategy of TAA is consistent with that of the experts' in the field. In order to make a definite and optimal decision in an achievement test design process, an expert needs two sets of data: one is the information concerning preconditions about the test and the other is the information concerning the subject matter to be tested. Both types information required to design a good test instrument are easy to collect, i.e., few values in basic parameters for achievement test design are unknown or difficult to obtain. On the other hand, we have little knowledge about how a decision at one stage affects the quality of decisions at subsequent stages in an expression of mathematical measurement: Are the effects compounded? Are they additive or equal to the weakest? Therefore, more in-depth studies are needed before certainty factors or inexact factors can appropriately be used in computer assisted achievement test design.

Communication With User

Ideally, a comprehensive user's model should be built to monitor the user's understanding of the domain, and then TAA will choose what should be communicated with the user accordingly. However, no user's model is built for TAA's communication with its users. This is because it is practically impossible to build a machine which can accurately map users' intellectual activities. Many researchers have questioned both the need for detailed student/user models and the practical possibility of building them (Sanberg, 1987; Self, 1988). They considered that the student modeling problems expand from computational questions, to representation issues, through plan recognition, mental models, episodic memory to individual differences - to encompass almost all of cognitive science.

As an alternative to building a learner's/user's model to detect his or her misunderstanding or inappropriate decisions s/he made, TAA gives a user the opportunity to make an informed decision concerning what concepts and principles s/he interested in or needs. More specifically, using the simple-to-complex sequencing described by Reigeluth-Merrill's Elaboration Theory of Instruction, TAA assists its users in forming a proportional combination of advisory assistance needed to performing a particular task. Figure 7 shows how the assistance will be provided based upon the user's requirements.

The type(s) and the amount of assistance are based upon three factors:

1. his/her familiarity with TAA;
2. the amount of training in test design that the user has;
3. whether s/he wants assistance or not.

To evaluate the users' needs, the information on these three questions will be used throughout the process of designing the test.

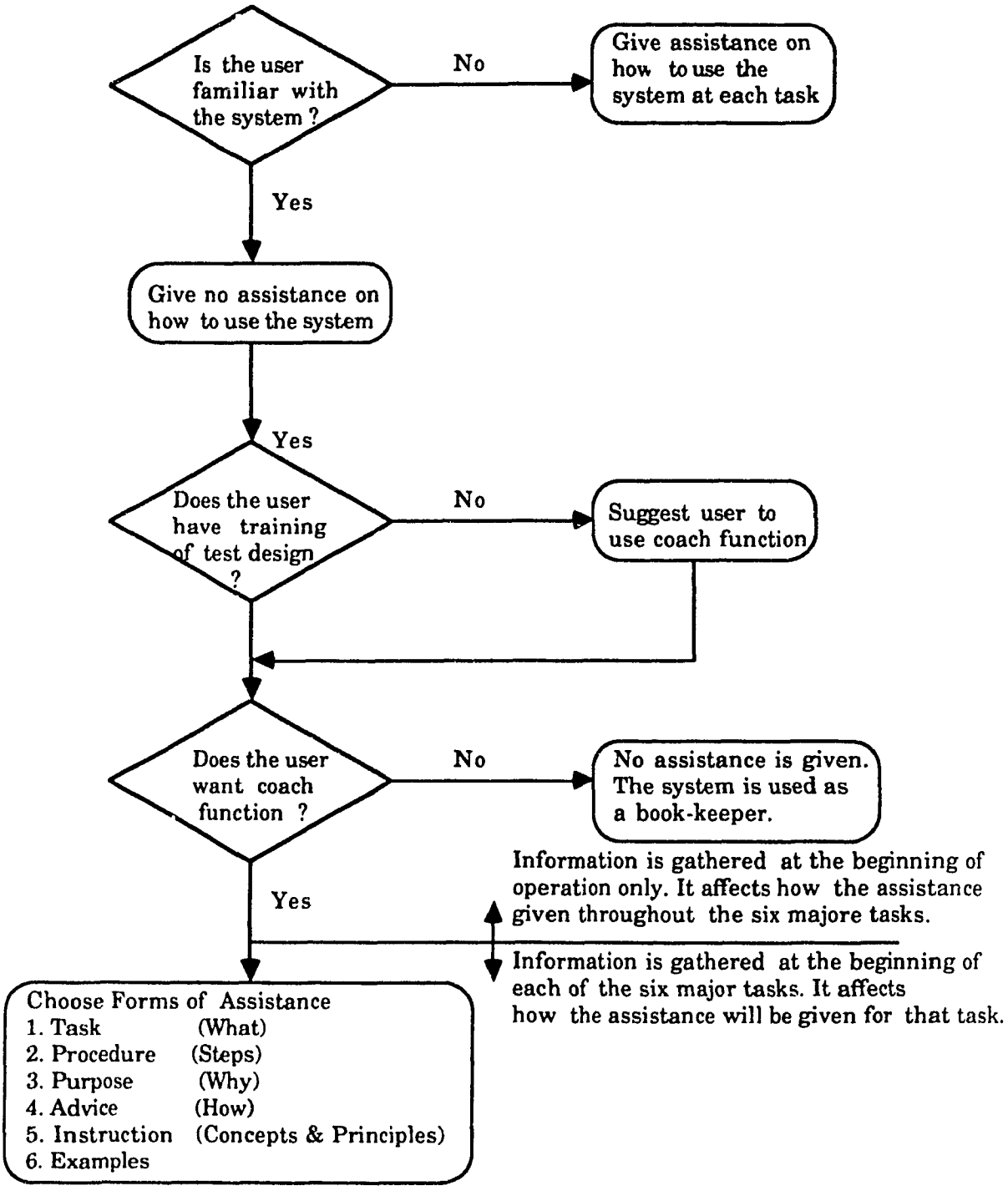
However, the information on the above three factors is not sufficient to give appropriate assistance to the user. To know explicitly what a user needs with a particular task that s/he is performing, additional information is collected from the user at the beginning of each of the six major tasks. For instance, if the user indicates that s/he wants to use the coaching function of TAA, then at the beginning of each of six decomposed major tasks the user will be asked to choose the types of help from the assistance menu . More specifically, the assistance is decomposed into the following categories:

1. Procedure - (Steps)

   describes the procedures and steps to complete a particular task, which is given before the task is performed.

2. Purpose - (Why)

   tells the user the purpose of performing the task.



Figur 7. How the assisstance is given

3. Advice - (How)

gives guidance or suggestions on how the task should be performed.

4. Instruction - (Concept and Principle)

explains the major concepts and principles in the test design.

5. Task - (What)

tells the user what TAA is doing at this moment and how to respond to it.

(This type of message is more relevant to the user who is not familiar with TAA.)

6. Examples

Among them the messages of Procedure, Task and Advice types assist the user in performing the tasks at hand and messages of Purpose and Instruction are to enhance a deeper understanding of test design; Examples back up both the Instruction and Coaching parts of TAA.

TAA accommodates end users with various levels of experience. For instance, a user may have experience in designing the test, but possesses little knowledge about how to use this system. In this situation, the user only needs an explanation of particulars of TAA, e.g., which key to press at this moment. In these circumstances, an experienced test designer who is unfamiliar with TAA may choose only the assistance on how to use TAA without assistance on test design. In an other circumstance, it may be that a user is an inexperienced test designer who has no time to go through all of the instructions on the related concepts and principles, but who needs advice on how to complete certain tasks in an appropriate way. In this situation, the user may choose advice on the tasks at hand without being instructed on the theoretical knowledge.

The messages on how to operate TAA at each particular stage are divided into two types. One is easy-to-remember key strokes that the user will get familiar with very soon, such as "press return" to proceed. Another type may be more confusing to the user, (e.g., how to choose items from a menu). At the very beginning of an operation, if the user tells the computer that s/he is not familiar with TAA, TAA will give him or her both

sets of assistance in how to use TAA, messages like "Press return to ..." will appear on the occasions when the user may not to know second type of assistance is given during the whole operation.

TAA not only can give its users the different types of assistance they need, but also it is also flexible in its instructional quantity and branching. That means the theoretical knowledge in the instructional part can be reviewed by a user in different degrees, from everything to only certain areas, based upon the user's choices after TAA explains to her/him how these concepts and principles are related to the task at hand. what to do next. If it is indicated that the user is familiar with TAA, then only the While the user can select the content s/he wants to learn, one restriction is that the user can see the instruction of a certain level only after s/he has entered one level above it. It is rather like the fact that one has to enter Canada before s/he can visit either Ottawa or Montreal. To avoid trapping the user in instruction that s/he may really want to discontinue, TAA allows the user to skip chunks of content or discontinue the instruction on a certain topic before it is completed and go back to continue her or his test design tasks. This approach is like a scope of coverage on a big map where the center of the attention is the task at hand and the user will choose how much s/he wants to be included in the converge for his or her review.

The user controls when the next message will replace the present one, so the user controls the pace. TAA also gives the user the freedom to choose the instruction with or without examples.

TAA's user interface regarding data collecting is designed to minimize difficulties for the user in proceeding with the task and also the difficulties for TAA to interpret users' input. There are two formats inputing data: one is by giving the user a list of items (a menu) so that s/he will make choice(s) from it and the other is by letting the user type in whatever s/he wants to enter. TAA uses the menu type interface as much as possible and during the TAA's operation the user can make her/his selections from a menu via the arrow keys. The type-in format is used as little as possible, to reduce the burden of

typing on the user and the difficulties of interpreting the input by TAA. If a type-in input is really necessary, the user will be given a message such as "type in ....". The user input should be monitored and the user should be given a chance(s) to reenter the data after feedback is given to an illegitimate input (not completed).

System Development

In choosing a development strategy, efficiency and effectiveness of TAA's prototyping are two major considerations. If an application is large, as is the case with TAA, and if it is developed as a whole entity from the very beginning, the efficiency of its implementation will be hindered greatly resulting from system testing and revising. Because it is not efficient to build it in one chunk, TAA is developed modularly in terms of six knowledge bases that are built corresponding to the six major tasks of the test design. Each knowledge base has its own inference engine and canned messages. To increase the efficiency of system modification, the production rules within each inference engine are indexed as rule sets by the tasks they perform so that the modification can be done locally.

When a system is developed via separate modules, the major problem is that the data structure may not support all individual sections. In order to ensure the adequacy of overall knowledge representation structure, the six knowledge bases are developed in parallel and started with their skeletons. For the same purpose, frames and rules which contain overall test design variables (e.g., the education level of the testees, content area of a test and rules of coaching strategy) are coded in the files shared by the six knowledge bases. As a result of sharing a single knowledge representation structure, the structures of the individual knowledge bases are identical. Developing six knowledge bases in parallel and using one shared data structure permits all six knowledge basis to be integrated smoothly into the framework as a whole application, thus enhancing the effectiveness of TAA development.

In terms of system operation capability, TAA development can be defined in three stages. TAA, at the end of its first stage of development, in terms of performing test design tasks (vs. coaching and explaining test design tasks), should be able to perform the basic required tasks and show all of designated features of TAA. During the first stage of the development the procedure-decision knowledge in the test design is formalized and structured into a computable model. In the second stage, a coaching function should be built into TAA. The coaching function consists of two types knowledge: one is well researched and documented theoretical knowledge, and the other is heuristics and rules of thumb. The theoretical knowledge is structured within the instruction component and the heuristics are recorded under the advice component. As result of second stage of development, TAA should be able to explain "How" and "Why to the user and exhibit the basic capabilities and features of the final user interface. Also, in this stage, the requirements specification of TAA should be finalized. In the third stage, TAA should be integrated, validated and verified as a final product. Also all documentation should be completed, including the user's manual.

At the completion of this thesis, procedure-decision making knowledge is formalized into a computable model; and instructional and advisory models have been constructed completely and can be applied to all six knowledge bases. The TAA can operate well as a tool to build a test, yet it is unable to give high quality and comprehensive advice and explanations on the actions it takes. Also TAA cannot be used by a novice user because its interface is too crude. Therefore, TAA is half way through the second development stage and needs to be revised in the areas of advice content and user-interface format.

Regarding the advice contents and its inference capabilities, TAA has incorporated a great proportion of the theoretical knowledge but few domain heuristics. In order to be able to give deep and comprehensive explanation of the actions it takes, TAA needs more input from the experts, especially heuristic knowledge, rules of thumb and short cuts in

test design. In terms of its coaching capability, the advice is generally relevant to the task, but often it does not hit the precise point, since the conditions for giving a particular message are too general. For its instruction capability, TAA is not developed evenly among the six knowledge bases. The theoretical knowledge is structured for the tasks of "2. Formulating Performance Objectives", "3. Choose Test Item Format" and "4. Constructing Test Specifications".

Another aspect of TAA's advice contents and its inference capabilities is that most of knowledge incorporated into the system is more relevant to test design in an educational setting; there should be more knowledge identified for test design for business and research purposes.

The user interface is adequate to perform the basic tasks if the user is very familiar with TAA. The development of the user interface can be viewed in two aspects: one is interface structure and the other is assistance on how to use TAA. The design of the interface structure for individual tasks is completed, but it is developed unevenly among the tasks. At a higher level than that of the test design task control, the design of the interface structure for system control has not completed, (e.g., exiting the system operation, going back to the previous point and printing file to a printer), because the Intelligent Compiler shell does not directly support the development of such system control.

Summing up, in order to maximizing the efficiency and effectiveness of TAA development, several development strategies were employed:

1. partitioning the inference engine;
2. sharing the same knowledge presentation structure; and
3. using meta rules to group backward chaining rules as rule sets according to the purpose they serves.

Development of individual knowledge bases. The above discussion describes TAA development as a whole, since the six knowledge bases were not developed evenly. The following description focuses on the development of individual knowledge bases.

In the first knowledge base ("1. Get Information"), only procedure-decision making knowledge has been coded. The grade scheme of testees' educational level is not completed. There is no instructional material in this knowledge base yet and assistance in how to use TAA is weak. Additional variables and their values, in the category of test design preconditions, should be defined and incorporated into this knowledge base in future TAA development (e.g., to define possible values of the research purpose).

In the second knowledge base ("2. Formulate Behavioral Objectives"), instructional materials have not been coded completely into this knowledge base, for instance prerequisite knowledge has not been defined for the task of formulating behavioral objectives. With regard to the user interface, typing in a verb from a given verb list for each level of performance should be changed into clicking a verb from a verb list menu.

More inference knowledge should be incorporated into the third knowledge base ("3. Choose Item Format") so that the selection of an item format will be more sensitive to various test conditions. The structure of the prerequisite knowledge for the task of choosing item format is completed, but many slots need to be filled.

In the fourth knowledge base ("4. Construct Test Specifications"), task, advisory and instructional functions are the most fully developed among the six knowledge bases.

In the fifth knowledge base ("5. Decide Length of a Test"), there are no instructional materials constructed yet. Also, few justifications on TAA's actions at this design phase are coded. Some areas of the user interface needs to be imrpoved, for instance, two matrixes representing the planned test specification and the actual score distribution are displayed on the screen. Among them, the actual score distribution matrix should be given a matrix heading, like the one the planned test specification matrix. In the

future development, when an objective is added, the score calculation should be connected with the inference done by the second knowledge base ("formulate behavioral objectives"); at present these two knowledge bases are separated.

In the sixth knowledge base ("6. Write Test Items"), procedure-decision making knowledge is completed. A great proportion of the assistance concerns advice on how to word a particular type of item and more theoretical knowledge for instructional purpose should be added.

CHAPTER 6

Evaluation

Expert System Evaluation

Although there may be different definitions of evaluation in education and computer science, it is defined here as the whole testing process in software design and development.

There is common agreement among researchers and professionals in software design as to which aspects of conventional software and expert system should be evaluated. The two general questions asked in the evaluation process are (Beohm, 1979):

1. Are we building the right product?
2. Are we building the product right?

The first question concerns validation. Validation of the system determines whether or not the designed system will solve the desired problems for worthwhile reasons. During the validation phase the requirements of a system and the criteria for success or failure are verified (Parsaye & Chignell, 1987, Sommerville, 1989) and validated. The second concern in software evaluation is verification. Verification (Sommerville, 1989) or Testing (Walters & Nielsen,1988) is to see whether the system is doing things properly. This refers to whether or not the system is functioning efficiently and effectively all the time. While a computer program may perform well during testing it is not necessarily a validated one for it may be producing the right result for the wrong reasons. On the other hand, if a program cannot perform well during its operation, there is no way that the program can be validated.

Walters and Nielsen (1988) consider debugging as the third type of question in software evaluation. Debugging involves checking the coding in order to find out the reasons underlying the problems encountered in the verification test. I consider

debugging to be follow up corrections of errors found in the evaluation process, instead of being within evaluation.

Smart (1987) outlines expert system evaluation in more detail. He stated that four basic aspects should be checked during the evaluation of an expert system:

1. It does the right things which include the essential and desirable features.
2. It works all the time thus it is error free.
3. It is user-friendly.
4. It is well documented. It should have updated documentation of a user's manual and development documentation.

These four questions are easily fit into the framework of the validation and verification, because "user friendly" is one of desirable features in any man-machine system which is a verification question, and "good documentation" is a means to efficient system development.

Although evaluation in expert system development and in conventional software development share the same basic focii, several differences should be noticed. For instance, while conventional software evaluation may mainly verify the performance of a system against the pre-specified system requirement, expert system evaluation has to validate and refine its system specification. The difference results from the fact that, at the very beginning of system development, we often have only a vague or skeletal idea of how an expert system will solve the targeted problem. As a matter of fact, the understanding of domain knowledge progresses along with the development of an expert system. In other words, unlike conventional software development in which the design and implementation is according to the system specification, the expert system starts from very general objectives about what problems the system is going to solve, and system specification is derived from the prototype (Sommerville, 1989). Along with a better understanding of a targeted domain knowledge, the focus of an expert system evaluation will be shifted from validation to verification.

A great deal of time and effort is required to specify how an expert will solve a problem, so expert systems usually have long and unique apprenticeship in their life cycles. The systems are tested extensively by the end-users and domain experts using both prepared scenarios and real-life problems. Inevitably, a domain experts' involvement with an expert system design process is much more intensive than that required for conventional software development. As a result, another aspect of expert system evaluation is introduced: evolution of the internal knowledge representation.

Internal knowledge representation affects the problem solving capability (validation) and external performance (verification) of a system. The internal knowledge representation of an expert system for teaching purposes should be validated in terms of psychological validity which refers the extent to which the structure, content and expression of knowledge representation are similar to that of the experts. On the other hand, appraising the internal adequacy of the knowledge representation, (without reference to the real-world tasks), is a question of verification. The internal adequacy means to what extent the structure and the expressions of knowledge representation support efficient recording, retrieving and analyzing data. Without validating the internal knowledge representation, the compilation of the knowledge for an expert system's ongoing development may be off track.Therefore, in its prototype stage, the emphasis in the evaluation of an expert system should be placed on the validation of the internal knowledge representation, and the verification of the knowledge representation has its meaning only when the knowledge representation is valid.

In certain aspects, the testing of the prototype of an expert system is very similar to the formative evaluation of instructional materials in education. For instance, the purposes of both are to improve the quality of the programs at their formative stages. To identify directions and areas for programs' revision, the comments and suggestions are collected from the target population and subject matter experts (domain experts) as references. In both testing situations, testing starts with a very small population, (e.g., on

a one-to-one basis) and a participant is asked to go through the instructional materials or system operations to detect problems in the products. When the tested population is small, qualitative data analysis techniques are used for both types of products. There are some differences between the data collecting and analysis techniques for testing two products: While an evaluation of instructional materials may use quantitative statistical analysis if pretests and posttests are employed, testing expert system capabilities almost always uses qualitative case testing to determine whether the system can reach the proper conclusions (Walters & Nielsen, 1988). Seldom is a big group of subjects involved for expert system testing, so the comments given by the participants are not suitable to be measured by scores.

Evaluation Questions

Evaluation questions mainly concern pragmatic validation issues, and verification questions are used to make the validation possible. Evaluation questions are classified into three categories:

1. Problem solving (Validation)
2. External Performance (Verification)
3. Internal Representation (Validation and Verification)

To test the TAA's problem solving capabilities, questions concerning the purpose of the system were asked, such as "Does the system solve non-trivial problems", "By using the system can the quality of a design test be improved and ensured?" and "Can the system improve the users' understanding and competence in test design?".

To test the TAA's the reliability of external performance, questions concerning meeting task requirements were asked, such as "Can the system ask questions in a consistent manner", "Can the system output results in a reasonable amount of time?" and "Are the screen displays clear?" .

To validate the TAA's internal knowledge representation, questions concerning adequacy of the knowledge representation structure were asked, such as "To what extent

the knowledge representation structure adequate to retrieve, store and analyze the data efficiently?"

To answer the above questions, three types of questionnaires (See Appendix 3) were designed, respectively, for subject matter experts, computer experts and end-users. Some questions were asked in all three of questionnaires when the evaluations concerns for the participants were overlapped among the subject matter experts, the computer experts and the end-users.

The matrix in Figure 8 shows the system evaluation design and its relation to the questions in the questionnaires.

Evaluation Design

The system was tested separately as six subsystems corresponding to the six subtasks in the design process. This was necessary because integration of the six knowledge bases of TAA into a single system was impossible, which is due to the development stage of the TAA and memory limitations with the Intelligent Compiler shell.

The evaluation was conducted by the system designer. The evaluation was done on a one-to-one basis with the participation of two subject matter experts, two computer experts and four end-users as subjects. All of them went through the TAA's operation.

Before the operation started, the system designer gave the guidelines verbally, indicating the focus and methods of evaluation. All of the problems arose during the system operation were recorded by both the evaluation participants and the evaluator. A discussion and a questionnaire were completed for each evaluation session. Because some of the questions in the questionnaires were very general and sometimes speculative, each evaluation participant had a verbal discussion with the system designer on a one-to-one basis during and at the end of the evaluation session. In the discussion, the participants gave specific explanations of their answers to the open-ended questions, and

Aspects	Questions	Experts of		Subjects	Methods		
Problem-Solving		domain	computer	end-user	operation	review code	discussion
effectiveness	solve non-trivial problems	*	*	*	*		*
	any important issues are missing	*	*	*	*		*
	any important features are missing	*	*	*	*		*
External Performance							
efficiency	reduce design time greatly	*	*	*	*		*
flexibility	easy access to alternatives	*		*	*		*
ask user questions	consistency	*	*	*	*		*
justification	logical, show underlying reasoning	*		*	*		*
guidance and advice	make sense, easy to understand	*	*	*	*		*
user-friendly	adaquate assistance, easy to use	*	*	*	*		*
assistances meet user's needs	meaningful, natural, relevant			*	*		*
screen layout	clear, easy to use	*	*	*	*		*
Knowledge Representation							
completeness	objects, variables, values, relation	*				*	*
consistency	logical and coherent structure	*	*			*	*
psychological validity	natural to the experts	*				*	*
efficiency	easy to retrive, store, analyze data		*			*	*
clear	easy to read	*	*			*	*

Figure 8. Matrix of Evaluation Design

they also gave suggestions for further system revision. These comments were recorded too.

The results of the evaluation were analyzed qualitatively and used to validate the system design and reveal the directions for further modification.

Evaluation With Subject Matter Experts

Two domain experts participated in the evaluation. They were professors in the Education Department, Concordia University. When the evaluation was done with a domain expert, s/he was given all of the possible values of constants/objects, and from them s/he chose a combination of these values as input forming a test case. While monitoring the system operation, the subject matter expert and the system designer discussed the effects, problems and possible improvements of the system. For the questions related to the knowledge representation, the TAA's conceptual models and the coding in computer language were given to the domain experts for their comments.

Evaluation With Computer Experts

Two compu experts, of Concordia University, participated the evaluation. One of them was a professor in Computer Science and the other was a professor in Educational Technology. The evaluation with a computer expert was quite similar to that with subject matter experts, except that the testing issues were concerned more with the internal knowledge representation category. Being given the names of the objects and their possible values, the computer experts went through the system operation on a one-to-one basis with the system designer. Also, they were given the TAA's conceptual models and their coding in the computer language for their comments.

Evaluation With End-Users

The end-users were graduate students and professors in Educational Technology, Concordia University. They were either then or future instructional designers and instructors. The evaluation with them was mainly concerned with the questions on the problem solving aspect and some in the external performance aspect. No questions were

asked about the TAA's internal knowledge representation. Information on the appropriateness of the features and capabilities of the system were collected, such as "to ease the difficulties in test design", "to increase the quality of it" and "to teach deep knowledge about the topic". After discussing the evaluation guidelines with the subject, the system designer only observed user's performance. During the TAA's operation, an end-user was not given a set of possible values of constants. Instead, s/he entered whatever s/he thought was appropriate. Intervention from the evaluator was kept to a minimum until the subject had difficulty proceeding or s/he initiated a comment. At the end of the TAA's operation, the evaluator and an end-user discussed the confusions and problems of the system and how it could be improved while s/he completed the questionnaire .

CHAPTER 7

Results And Discussion

Results

The results of the evaluation were generated qualitatively based on observations during TAA operations as well as on the verbal and written comments given by the participants. Results recorded in this chapter are the comments given by the participants during TAA's operation and their answers to the open-ended questions in the questionnaires. Please refer to Appendix 4 for the participants' ratings on the Likert scale questions. A few questions in the questionnaires were not answered because the participants felt it was too early to give definite judgement while TAA was still a prototype.

During TAA's operation, there were no any interruptions due to bugs of TAA's coding, but the operations were halted several times because there was not enough memory. After the evaluation, only spelling and grammar mistakes found in the output messages were corrected.

Regarding the strength of TAA, the end-users and subject matter experts rated TAA as potentially a very useful tool to design and develop a test of high quality as well as to improve user's understanding of test design process. The end-users stated that they liked the structure of the assistance both in performing the tasks and in teaching theoretical knowledge. The end-users also stated that TAA would greatly reduce the time required to complete administrative tasks in test design, such as calculating score distributions, and deciding which and how many test items would be included in a test.

The domain experts and computer expert perceived TAA as having potential for accommodating multi-level user-machine interactivities. By using TAA, a high quality test could be ensured, however, the overall efficiency of developing a test might be

decreased, because TAA would force the user to go through a thinking and planning process that most classroom instructors did not go through in their test design process. As a result, by using TAA, an instructor might spend more time in developing a test than without using TAA to regulate the design process. In terms of TAA knowledge representation, the domain expert considered that its structure was basically psychologically valid, but the content should be enriched and fine tuned.

Weaknesses were found in two areas: one was the quality of the advice to the user and the other was the quality of the user interface. Regarding the weaknesses in the advice that TAA gave, the expert considered that the messages were relevant but not directive enough. Also, the content of the advice was rather flat, which did not accommodate different shades of a user's understanding and a test's circumstances. The communication concerning learning was rigid and not interactive enough.

The weakness in the user interface was evident in that the evaluation participants had difficulty communicating with TAA and they could not proceed through TAA operation without extra help from TAA's designer. The difficulties resulted from insufficient and sometimes misleading on-line assistance regarding how to use TAA. For instance, in the "choosing from menu" tasks, TAA didn't explain to the user the implications and the consequences of a particular choice. Without being aware of the consequences of his or her action, an evaluation participant sometimes was bewildered at what s/he was doing and how s/he should proceed. Another problem that the evaluation participants encountered during TAA operation was the difficulty in seeing the pattern of flow of messages. All of the participants, in various degrees and at different points, were overwhelmed by a string of messages because the relationship of the messages and transition between two tasks were not communicated clearly to them.

Some problems identified in the evaluation were due to TAA's developmental stage. For instance, the assistance function was developed unevenly among the six knowledge bases, which made the evaluation participants very uncomfortable when the reality did

not match what they were expecting during separated TAA operations. Occasionally, the evaluation participants were trapped in a procedure because the system operation control function was not built, (e.g., the functions of going back to the previous step and terminating an operation). Also, because the function of retrieving files was not developed, a participant could not always review what products s/he had created; therefore s/he could not view where s/he stood in a test design process. Generally speaking, TAA, at this developmental stage, is not capable of helping an end-user to grasp the whole picture of the designed test.

To sum up, the evaluation participants considered TAA to have great potential to be further developed into an effective and a practical tool, to assist instructors in achievement test design and in developing their own test design skills. The domain experts considered the conceptual model of the domain, generally speaking, to be psychologically valid; and the computer experts observed that the knowledge representation supported efficient system operations. The weaknesses identified in the evaluation were in the user interface and the quality of the advice TAA output. In the following section, aspects and steps of further development are recommended.

Recommendations

Based upon the results of the evaluation, TAA should be modified and further developed in five major areas with the following sequence:

1. user interface at performing individual tasks level;
2. content of knowledge base;
3. quality of advice to the user;
4. user interface at system operation level;
5. integration of six separated knowledge bases as a final product.

The modification should first focus on the user and machine interface at the task level (vs. at system operation level). The following tasks should be completed not only in this

stage, but also being carried out when the new knowledge is compiled in the future system development:

1.1. Give a definition to every technical term and key word that can be interpreted by a user with various meanings (no definitions are given at present);

1.2. Give an explanation at every point when a user is supposed to make his or her choices from a menu. The message should tell the user the main implications and the consequences of a particular choice (no such message is given at present) ;

1.3. To make transitions between tasks smooth, at the end of each task insert messages clearly stating what has been done and what will be done next (at present, such messages exist unevenly among the six knowledge bases) ;

1.4. Label each output message with its message category title, such as "advice", "instruction", or "procedure", etc., so that a user would know what category the message belongs to and will not be overwhelmed by a flow the messages (all of the messages have been labelled internally and some of message have been labelled externally in such a way);

1.5. Monitor user's "type-in" (vs. "choose from a menu") input and give him or her feedback when the input is in an illegitimate form and give the user an opportunity to re-enter a legitimate one. For instance, to tell TAA the available testing time, a user may overlook the instruction of input test time by minute, therefore, s/he may type in "1" for 1 hour instead of "60" for 1 hour. In this situation, TAA should be capable of asking the user to re-enter the test time in minutes or convert the hour into minutes for the user (only a few inputs are monitored at present).

At present, the inference knowledge is not sufficient, so in the second stage of the modification, more inference knowledge should be added and fine tuned. The main concerns are as follows:

2.1. Heuristic knowledge compiled through field practice should be extracted from domain experts. For instance, how to balance the reliability of a test and the reliability of an item;

2.2 More variables in the achievement test design should be defined in order to increase the sensitivity of the conclusions that TAA draws. For instance, right now there is only one objective list that the user can choose from to formulate general objectives. It is suggested that parallel objective lists be developed according to the subject matter of a test. In this way, the objective list represented to the user can be more relevant to the test content and the system inference will be more to the point. Please refer to Appendix 6 to see the objects and their values that TAA uses at present;

After the knowledge bases are more or less completed for the design tasks, a user model should be designed and built to monitor his or her understanding of a topic. With the user model, conditions of a particular message to the user will be refined; thus, the advice given to the user can be more relevant and directive. The following changes are advisable:

3.1 The user model should be kept simple, and be structured around common misunderstandings;

3.2 the user model should deal with misunderstandings locally instead of trying to detect all misconceptions in the achievement test design process as a whole.

At the fourth stage of the modification, interface regarding system operation control (vs. performing individual tasks) should be incorporated as part of the interface. At the moment, there is no such function. TAA control interface will override the execution of the individual test design tasks. The interface might consist of certain key-strokes and pull-down menus. The capabilities and functions of the system operation control are defined as follows and may be constructed in the following sequence:

4.1. Allow a user to use TAA partially. For instance, if a user wants TAA to assist him or her only in formulating performance objectives, TAA should be able to start the

operation at the task of formulating performance objectives instead at the very beginning To serve this purpose, a menu, which has the six major tasks of the achievement test design listed on it, may be built so that the user can make his or her choice. After a choice is made, TAA should ask the user for all the information needed to complete the task(s). The development of this function requires the system designer to thoroughly analyze the logic of the paths that a user may choose. The analysis should look at the information dependency among the six major tasks. For instance, if a user wants TAA to choose only the item formats, s/he has to input the performance objectives in the forms that TAA requires. In this circumstance, if the user does not have a list of performance objectives written yet, TAA should suggest that s/he use TAA to formulate performance objectives first. From a completed objective list, TAA can select an item format for each objective.

4.2. Allow a user to make cross-reference of the concepts and principles of test design. A pull-down menu of on-line help may serve this purpose. In the menu, the topics of explanations and definitions of theoretical knowledge should be displayed. The menu can be indexed by the theoretical knowledge structure and constructed according to the levels of abstraction. The already designed theoretical knowledge structure may be used for this task.

4.3. Allow a user to view where s/he is at a particular point. There are two viewpoints regarding this type of information: one is from the procedure-decision point of view and the other is from the theoretical structure of test design, either of which should be available. The already designed structures of the procedure-decision knowledge and the theoretical knowledge can be used for this purpose.

4.4. Allow a user to retrieve what s/he has created. To develop this capability requires identifying the products a user has possibly made at each stage in a test design process, such as a list of performance objectives.

4.5 Allow the user to control the system operation. The functions of the system operation control includes:

a. terminating the operation completely at any point of the operation;

b. exiting a particular task and going back to the main menu to choose another task;

c. terminating the instruction session and going on with the design tasks;

d. saving/retrieving products that a user has created;

e. printing products that a user has created.

By using either key-strokes or a pull-down menu, TAA control will override the operation of a particular task.To build these functions, one may have to use C language, because the Intelligent Compiler shell does not directly support building such a system control;

4.6 Allow a user to go back to the previous stage and revise the data s/he has put in. However, the point at which the user is allowed to do so cannot be a free will choice, but it must be well thought out. This is because test design is an iterative refinement process and any changes at one point will cause a chain reaction in its following decisions. More specifically, if a user freely revises the decisions that s/he has made, it might result in heavy recursion and confusion in TAA operation. Therefore, it is suggested to select a few of logical points where a user can revise what s/he has put into TAA.

In the last stage, the six separated knowledge bases should be integrated into one entity. At the end of this stage, a stand alone application should completed. Please refer to Appendix 6 to see the objects and their values for each knowledge base.

On-going formative evaluation is necessary in all phases in order to ensure that the modification and development is on the right track. The focus of an evaluation should be shifted gradually from validation to verification, along with TAA requirements becoming more clear.

The requirements for the modifications and future development apply to all six knowledge bases. Since the six knowledge bases have been developed unevenly, some of the knowledge bases need more effort in their modifications than others do.

The above recommendations concern TAA as a stand alone application. Beyond that, for practical purposes, TAA may be used as a subsystem incorporated into a system which assists a user in instructional design. It can also be used with a parallel system which analyzes the validity and reliability of an achievement test based upon the testees' scores.

Future study may be directed developing a user model as an "intelligent" coaching system. It should reflect the differences between computer based on-the-job assistance and computer aided learning.

To sum up the recommendation section, according to the system design, the TAA's user-machine interface at performing individual tasks level should be modified and the knowledge content incorporated into TAA should be enriched. To improve the quality of advice and the user-machine interface, a user model and a system operation control should be designed and implemented. The heuristics of the knowledge domain should be fine tuned to increase the quality and sensitivity of TAA's inference. More studies should be carried out to explore strategies and tactics of designing a user model as an "intelligent" coaching expert system.

Discussion and Conclusions

The purpose of this study was to explore practical design strategies for expert systems development. The study focused on expert systems employed to deal with open-ended design tasks. Ideally, such an expert system would communicate information concerning the heuristic and theoretical knowledge of the domain, and the weight of each type of knowledge assigned according to the purpose of TAA.

For the purpose of the study, a prototype of an expert "coaching" system assisting the user in achievement test design has been constructed. TAA primarily deals with the targeted problem, while also teaching the end-users the heuristic and theoretical knowledge involved. As a result of the study, the domain knowledge (achievement test design) has been formalized and constructed into a computable model. This model can be

further used as a subsystem of an instructional design application; or it can be used as a parallel system to applications of the research design and evaluation.

Three expert system design strategies were employed in this study and may be very useful for future expert systems development. The first design strategy was to control the abstraction level of the output solutions in an application of design tasks. When constructing such expert systems, we face the problem that explicit solutions are impossible to be produced because specific goal states cannot be predetermined. In this study, TAA was not expected to output explicit solutions; rather TAA was designed to output solutions at an appropriate abstract level, keeping the conclusion abstract (but as explicit as possible) and advisory (but as decisive as possible). The abstract and advisory guidance was further backed up by examples so that the messages of such guidance could be meaningful to the user in order to make proper decisions. This approach may be used to reduce the required searching space in an application of design tasks, and to improve the end user understanding of the domain knowledge in any expert system applications.

The second design strategy used in this study was to keep knowledge representation structure consistent with the purpose of TAA. As a practical tool, TAA was supposed to primarily solve the targeted design problem; and second, to communicate heuristic and theoretical knowledge to the users. To serve the user in that manner, TAA was designed from the perspectives of cognitive strategy (control strategy) and procedural knowledge in the domain. The domain knowledge was modeled from three angles. The first was from a procedural knowledge perspective which reveals the procedural relationship among the subtasks in the achievement test design process. The second was from heuristic knowledge consisting of rules of thumb and standard practices in the field. This type of knowledge reveals how decisions and choices are made in a professional test design process.The third was the control strategies or theoretical knowledge of test design. To keep the primary purpose of the application and the primary knowledge representation structure consistent, TAA used an algorithmic specification

representing the procedural knowledge as its primary knowledge representation structure. The procedural knowledge in the domain was further coded in a form of meta rules to ensure that TAA efficiently selects the most pertinent sets of rules in its inference. The heuristic knowledge was then incorporated into the slots in a particular step expressed by the procedural knowledge. Also, the procedural knowledge was used as an indicator to partition the inference engine for efficient system development. By bringing different types of knowledge into a coherent structure to fulfil different purposes, the coding of an application is easy to comprehend by the experts and modified by TAA designer. The above mentioned approach can be used in the design of any expert system.

The third design strategy used in the study was to design a flexible as well as effective coaching scheme to meet different users' needs. The coaching function was designed and developed according to the The Elaboration Theory of Instruction (Reigeluth & Stein,1983). Although it was an excellent tool to design instructions at the macro level, the scheme of the Elaboration Theory of Instruction was not well suited to such a complicated problem-solving application. To remedy this, the assistance regarding communication of the procedure and decision-making knowledge was further defined into three categories:

a). purpose - The messages of the purpose were to answer the "why" questions that a user might have in mind. It was not a goal of a set of backward chaining rules, but it was the control strategy of an action, such as "the purpose of constructing test specifications is to ensure content validity in a test design";

b). procedure - A procedure consists a set of steps to complete a task successfully. A procedure answered a user's "how to" questions before s/he begun to perform a task;

c). advice - Advice was the output solutions which could be advisory or decisive. The messages of the advice category were different from that of the procedure category: although the messages of both categories concern "how to" questions, the messages of

the advice category only addressed inference knowledge (vs. procedural knowledge) in a problem-solving process.

Backing up the design making in a test design, instructions were used as a supporting component.

Partitioning types of knowledge not only enables a user to choose the type(s) and quantity of coaching function s/he needs, but it also facilitates efficient system maintenance and updating. This strategy can be used as an alternative to designing a comprehensive user (learner's) model to monitor the user's understanding of the domain knowledge, but further studies on this approach are necessary.

Given the purpose of TAA, it is concluded that an expert system approach is appropriate to deal with this type of open-ended design problem. Because the application purpose(s) and the end users' needs are different from system to system, it is inappropriate to have a universal formula in the design an expert system application. However, an expert system should have a minimum proportion of each problem-solving function and coaching function. Beyond that minimum, the weight and the structures of these functions should be designed according to the purpose of a system. Again, the main concept here is balance.

References

Alty, J. J., & Coombs. M. J. (1984). Expert systems Concepts and examples. England: NCC Publications.

Bloom, B. S. (1956). Taxonomy of Educational Objectives: Handbook 1. Cognitive Domain. New York: David McKay Co., Inc.

Brown, H., Tong, Tong, C., and Foyester, G. (1983) PALLADIO: an exploratory environment for circuit design. IEEE Computer, December, 41-45

Clancey, W. J. (1982). Tutoring rules for guiding a case method dialogue.In D. Sleeman & J. S. Brown (Ed.), Intelligent tutoring systems. (pp. 130-136). New York: Academic Press.

Clancey, W. J. (1987). Methodology for building an intelligent tutoring system.In G. Kerarsley (Ed.) Artificial intelligence and instruction: Applications and methods. Massachusetts: Addison-Wesley Publishing Company.

Carlsen, S., & Stokke, G. (1987). Conceptual modelling + prototyping = functional specification. In G. Guida. (Ed.), Proceedings of The Third International Expert Systems Conference (pp49-53). Oxford: Learned Information.

Dincbas, M. (1980). A knowledge-based expert system for automatic analysis and synthesis in CAD. Information Processing 80, IFIPS Proceeding, (pp. 705-710).

Fosrsyth, R. (1987) The architecture of expert system. In R. Forsyth (Ed.), Expert systems: Principles and case studies (pp9-12). London, New York: Capman and Hall.

Freksa, C. (1986). Knowledge representation for interactive aircraft design. In T. Bernold (Ed.), Expert systems and knowledge engineering (pp221-229). North Holland: Elservier Science Publishers P. V.

Gammack, J. G., (1987). Modelling expert knowledge using cognitively compatible structures. In G. Guida. (Ed.), Proceedings of The Third International Expert Systems Conference (pp191-200). Oxford: Learned Information.

Gilbert, T. (1978). Human competence . McGraw-Hill, Inc.

Graham, N. (1979). Artificial Intelligence: Making machine "think". PA: Tab Books Inc.

Harmon, P. (1987). Intelligent job aids: How AI will change training in the next five years. In G. P Kearsley (Ed.), Artificial intelligence and instruction: Applications and methods (cha. 8) Addison-Wesley Publishing Company.

Hu, D. (1987). Programmer's reference guide to expert systems. Indiana: Howard W. Sam & Company.

Hughes, S. (1987). Domains, tasks and questions: An approach to knowledge analysis In G. Guida. (Ed.), Proceedings of The Third International Expert Systems Conference (pp37-45).Oxford: Learned Information.

Keravnou, E. T., & Johnson, L. (1986). Competent expert system: A case study in fault diagnosis. London: Kogna Page Ltd.

Grounlund, N.E. (1981). Measurement and evaluation in education and psychology (2nd ed.). New York: Holt, Rinehart and Winston.

Hartely, R. T. (1985) Representation of procedural knowledge for expert systems. In Proceedings of 2nd Conference on AIA: The engineering of knowledge-based systems, (pp.526-531). IEEE, Miami Beach,

Mitchell, T.M., Steinberg, L.I., An intelligent aid for circuit redesign. Proceedings AAAI, (pp. 274-278)

Oliver, A. (1987). How to make rapid prototyping effective when developing expert systems. In G. Guida. (Ed.), Proceedings of The Third International Expert Systems Conference (pp45-48). Oxford: Learned Information.

Parsaye, K., & Chignell, M. (1988). Expert systems for experts. John Wiley & Sons, Inc.

Rayment, P. J., & Thomas, S. (1987). Advanced embedded expert system techniques. In G. Guida. (Ed.), Proceedings of The Third International Expert Systems Conference (pp77-82). Oxford: learned Information.

Reigeluth, C.M., & Stein, F. S. (1983) The elaboration theory of instruction. In C. M. Reigeluth (Ed.), Instructional-design theories and models: An overview of their current status (pp335-381). London, New Jersey: Lawrence Erlbaum Associates, Inc.

Self, J. A. (1988). Bypassing the intractable problem of student modelling. In Proceedings of Intelligent Tutoring Systems (pp18-24). Montreal: University du Montreal.

Shaw, M., & Gaines, B. (1987). Advances in interactive knowledge engineering. In N. Shadbolt. (Ed.), Research and development in expert systems (pp111-118) England: British Information Society Ltd.

Steels, L. (1986). Research and development in expert systems III. In M. A.Bramer (Ed.), Second generation expert systems.

Summerville, I. (1989). Software engineering. Addison-Wesley Publishing Company.

Van Horn, M. (1986). Understanding expert systems Toronto: Bantam Books.

Walters, J. R., & Neilsen, N. R. (1988). Crafting knowledge based systems: Expert systems made easy/realistic New York: John Wiley & Sons.

Wolfram, D. D. (1987). Expert systems for the technical professional. New York: John Willey & Sons, Inc.

Woolf, B. P. (1988). 20 years in the trenches: what have we learned? Intelligent tutoring systems June 1-3, 1988, (pp33-35), Association for Computing Machinery

Appendix 1

Task Map of Test Design Process

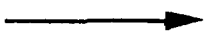
Legend of Figures Cited In Appendix 1

represents transformation sentre where an input data flow is transfored to an output.

represents a data store.

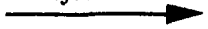
reprents user interactions with the system. The interaction may provide input or receive output.

represents the start or exit point of the operation.



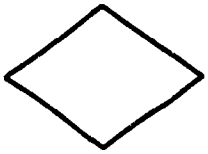
shows the direction of data flow.

general objectives



Words on an arrow describe the name of data.

represents a screen display.



represents an evaluation action.

5.2.1

represents the number of a task. The digit indicate the level of a task. "5 " reprents the fifth task among the six major tasks of test design. The '2" represent the second subtask of "5" and the "1" represents the first subtask of "2".

'X'

represents a variable.

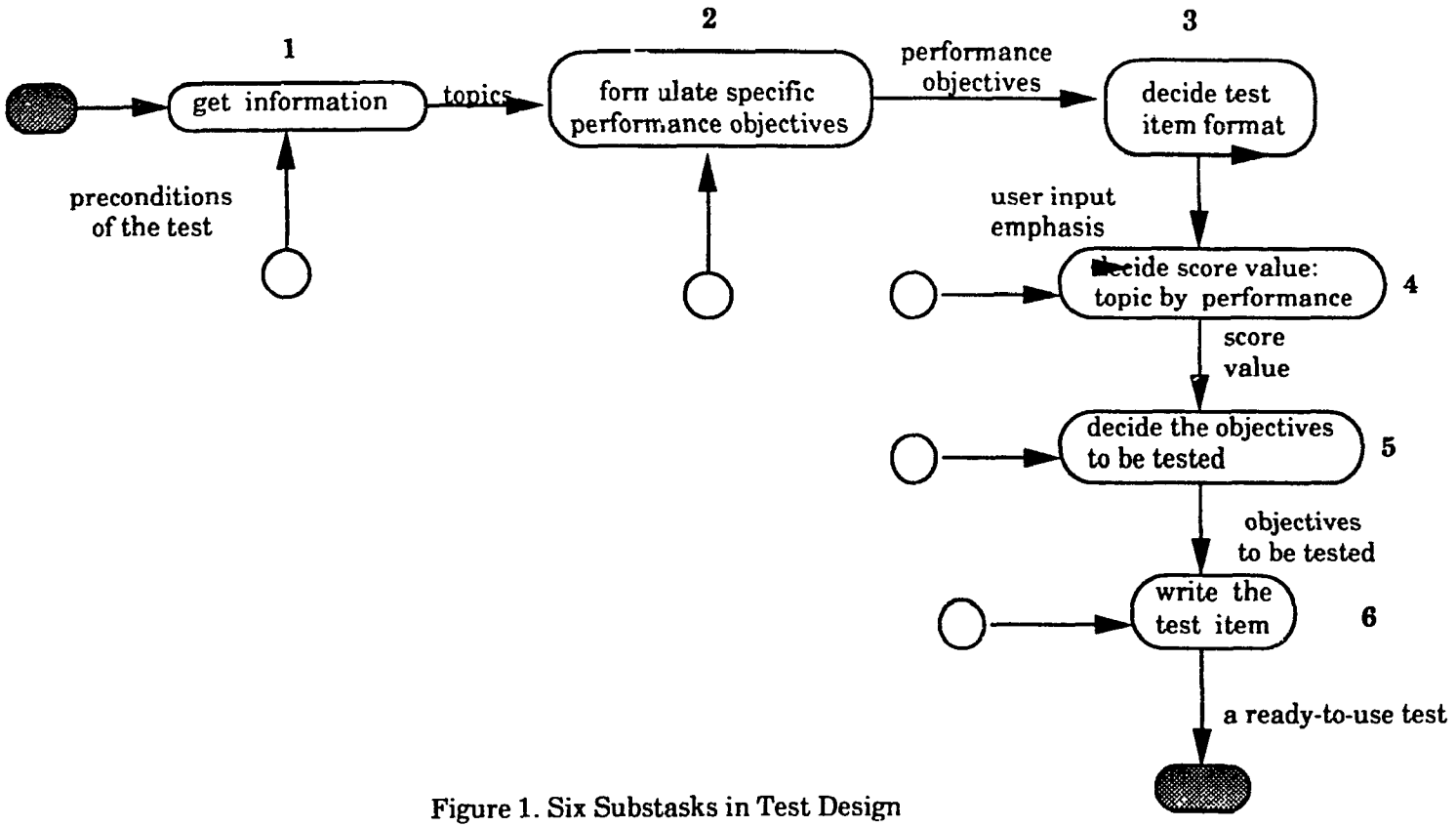


Figure 1. Six Substasks in Test Design

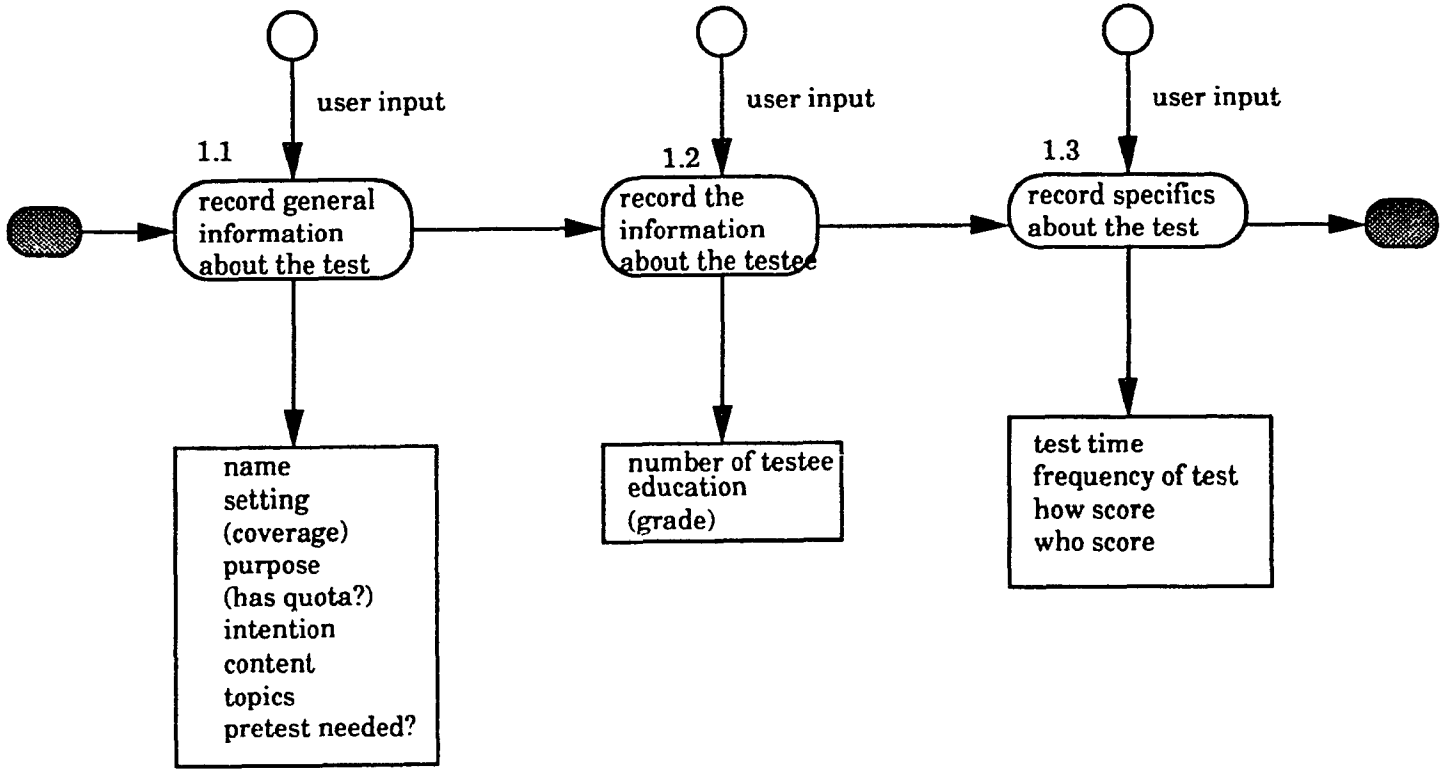


Figure 1.1 Get Information

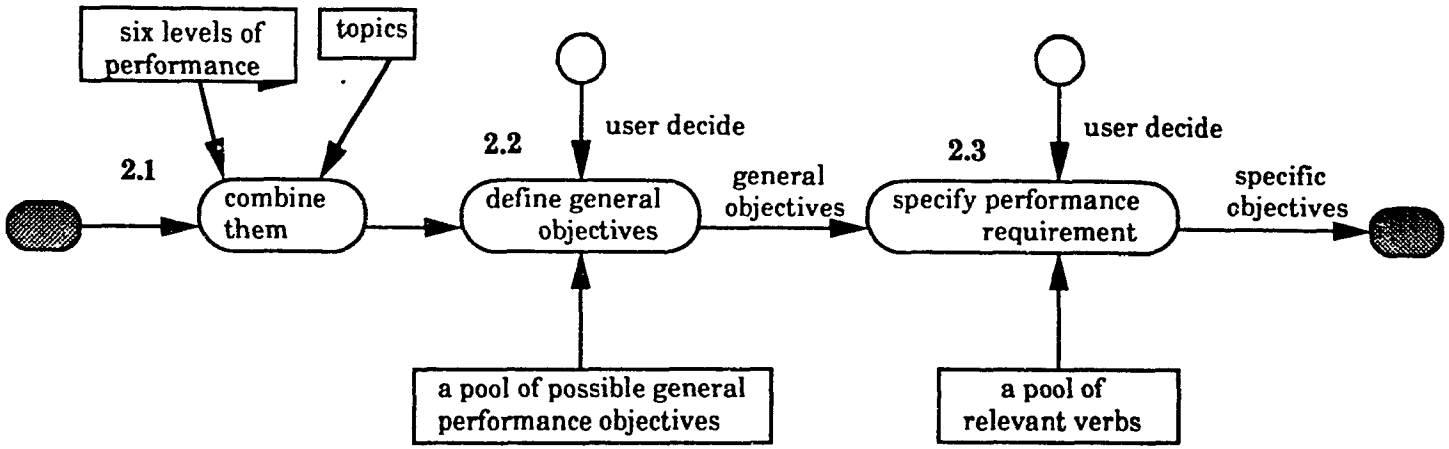


Figure 2.1 Formulate Performance Objectives

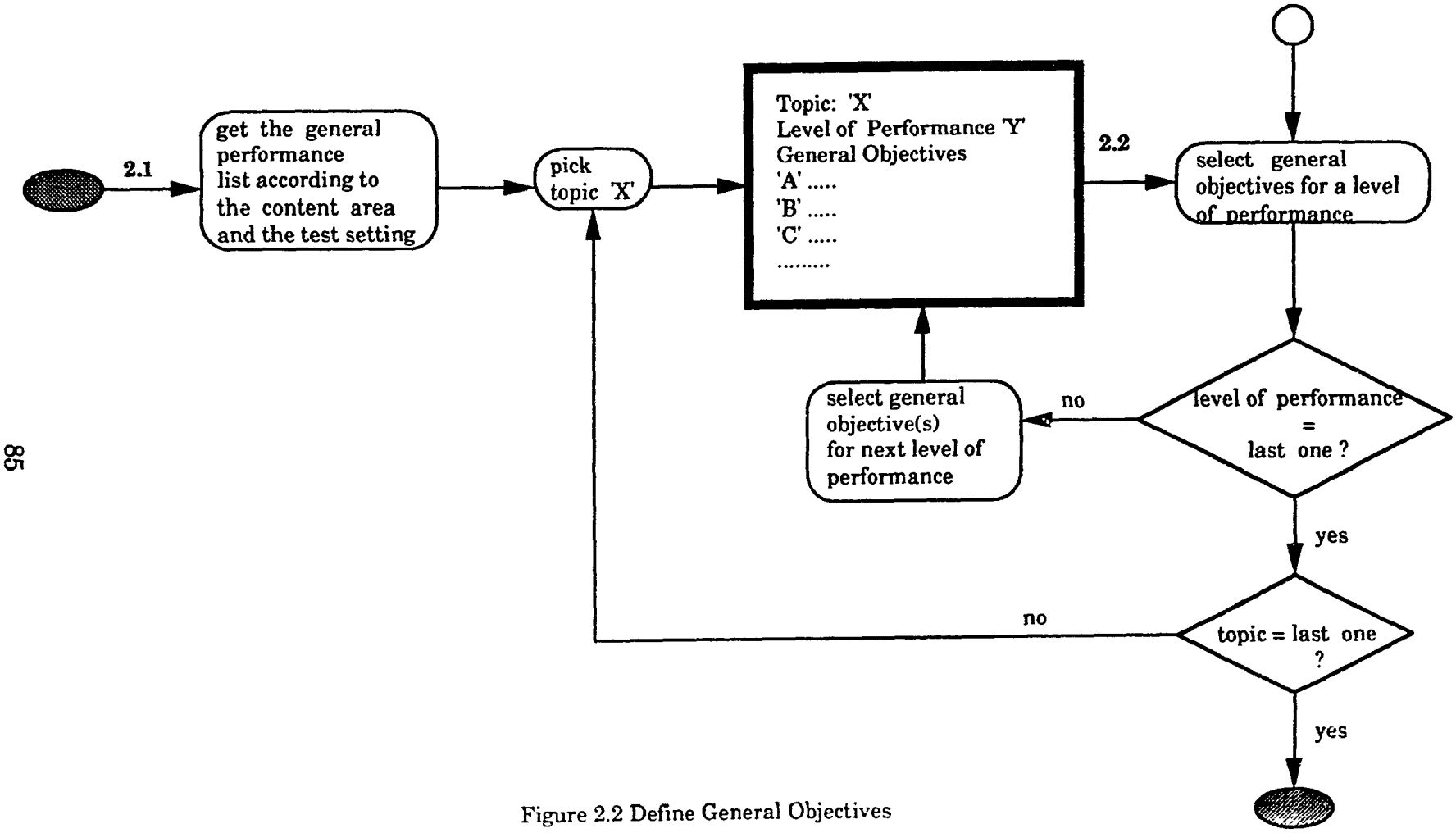


Figure 2.2 Define General Objectives

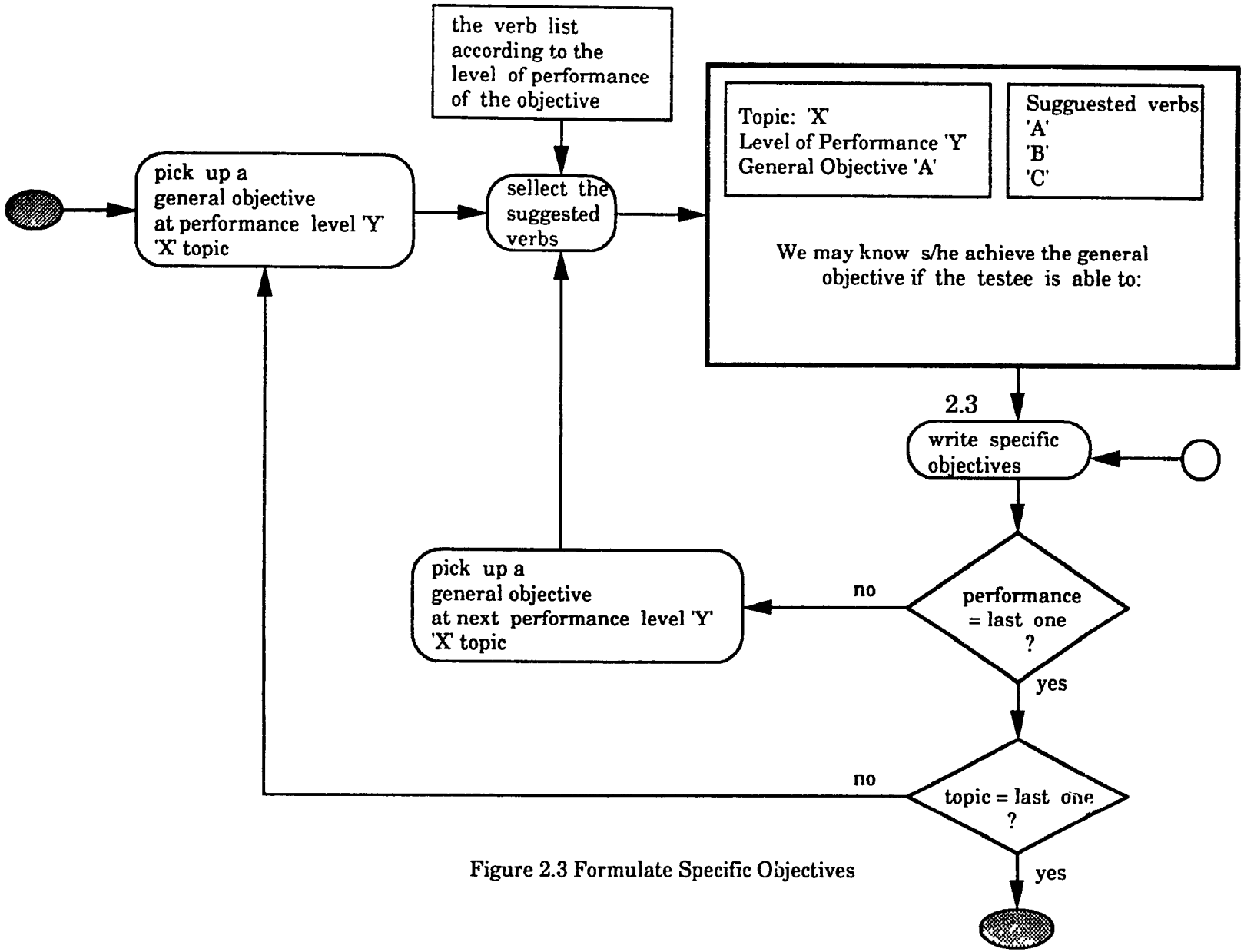


Figure 2.3 Formulate Specific Objectives

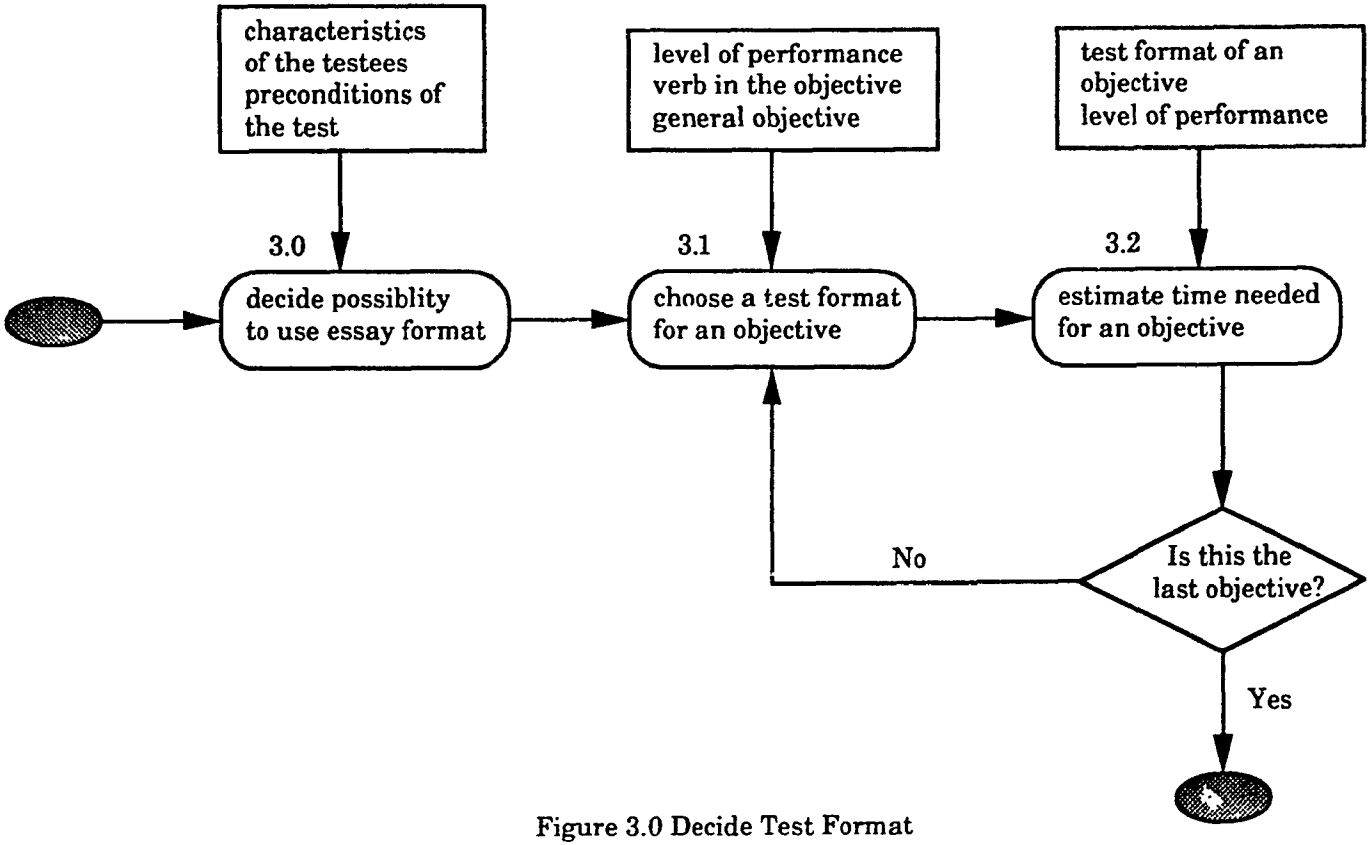


Figure 3.0 Decide Test Format

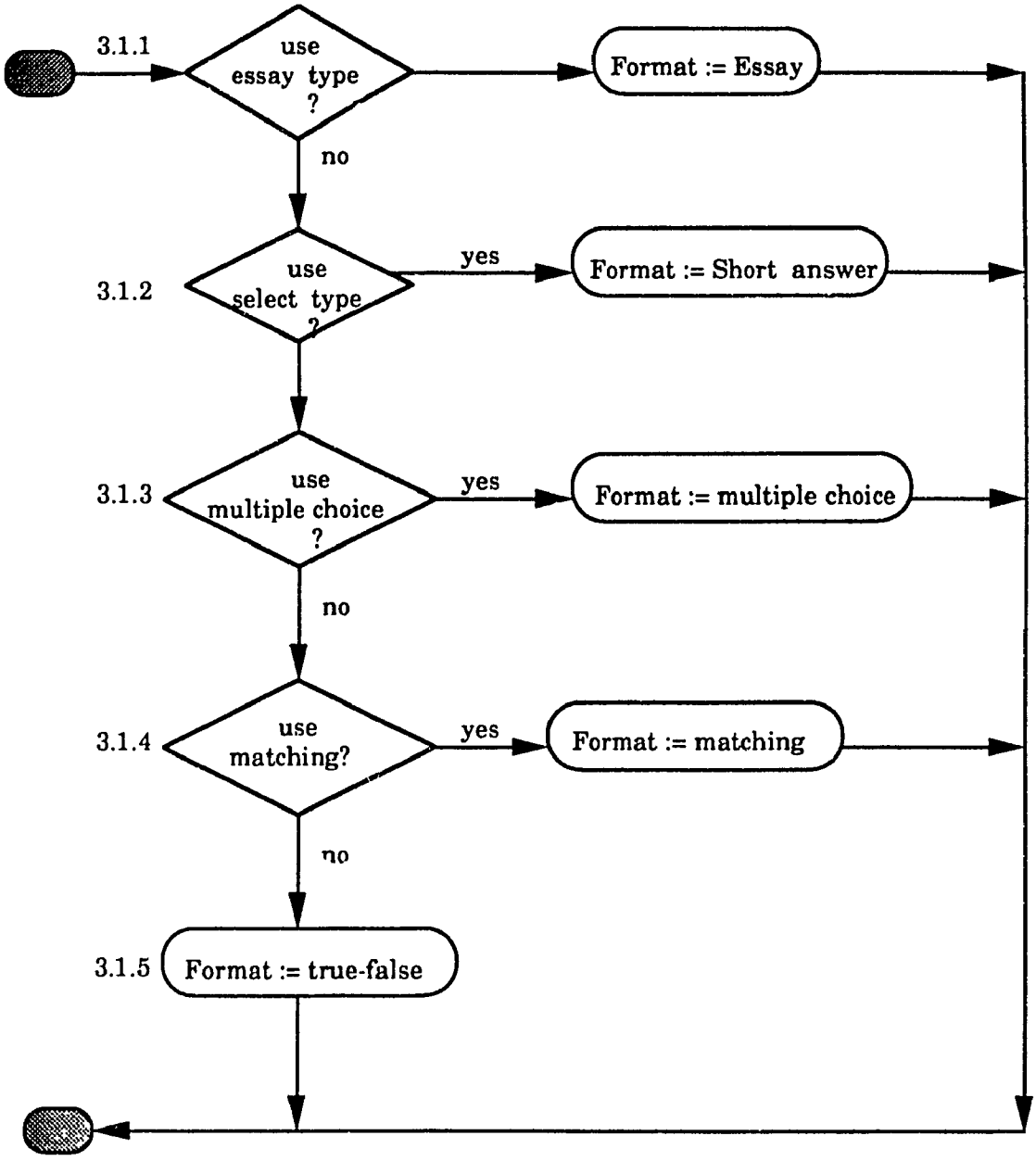


Figure 3.1 Choose Test Format

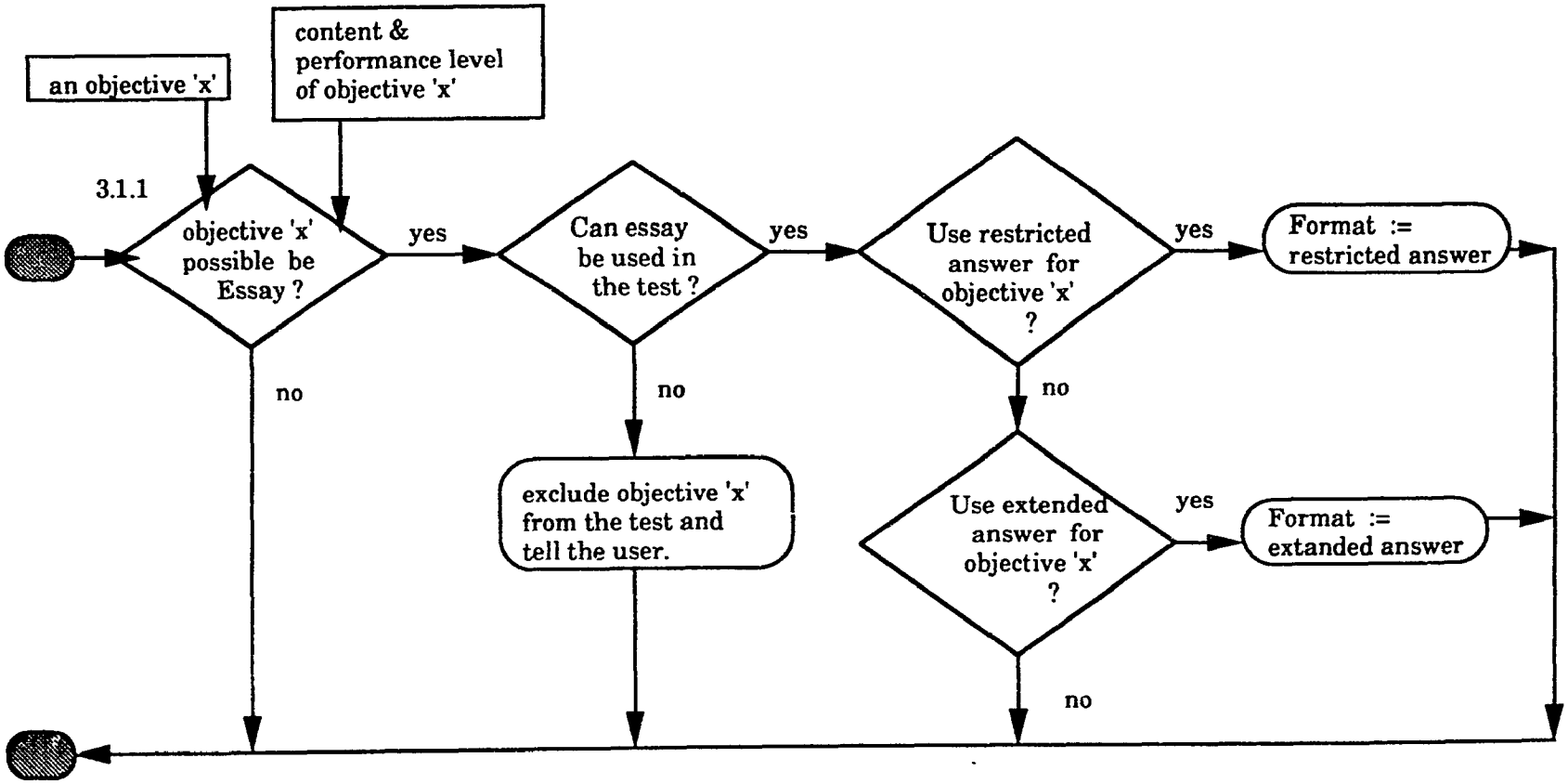


Figure 3.2 Choose Essay Versus Objective Type

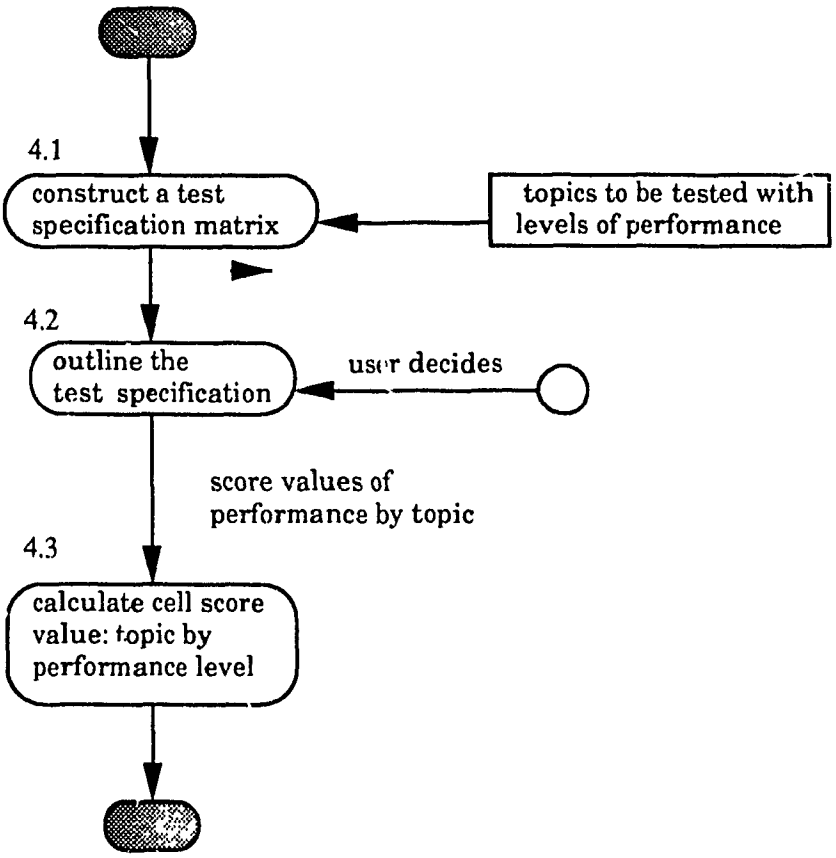


Figure 4.1 Construct Test Specifications

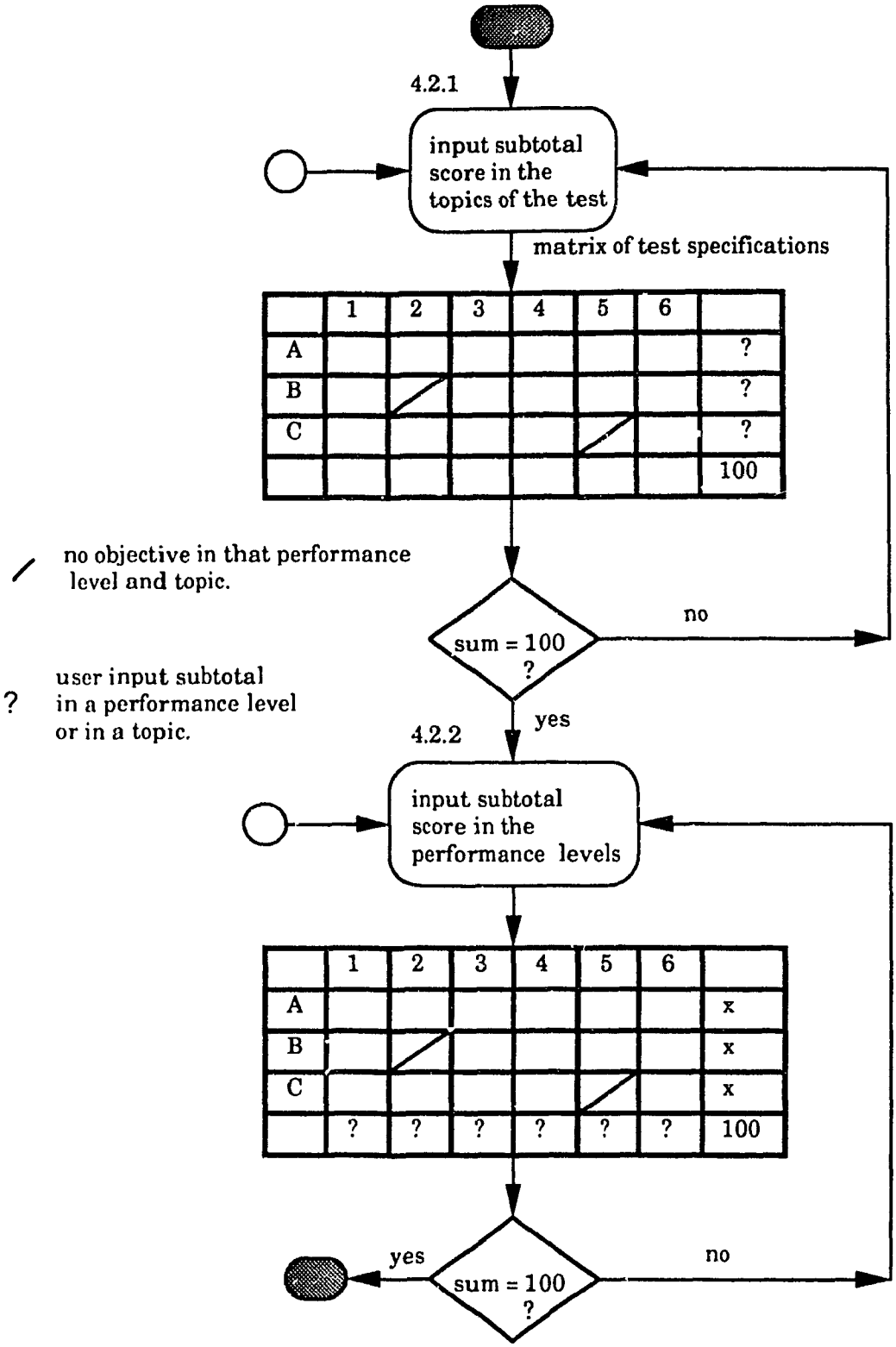


Figure 4.2 Outline The Test Speficiation

4.3.1 adjust by topic

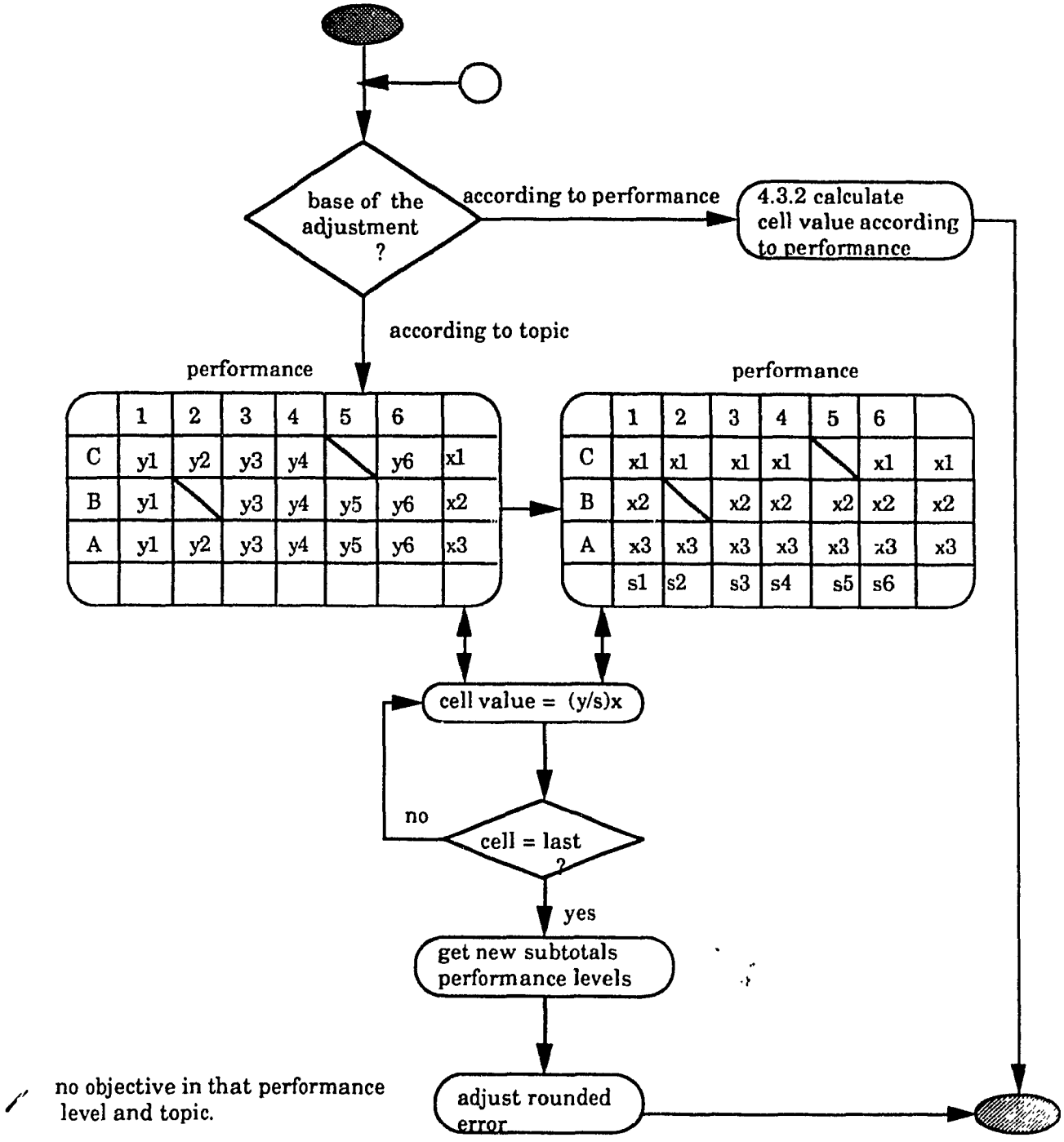
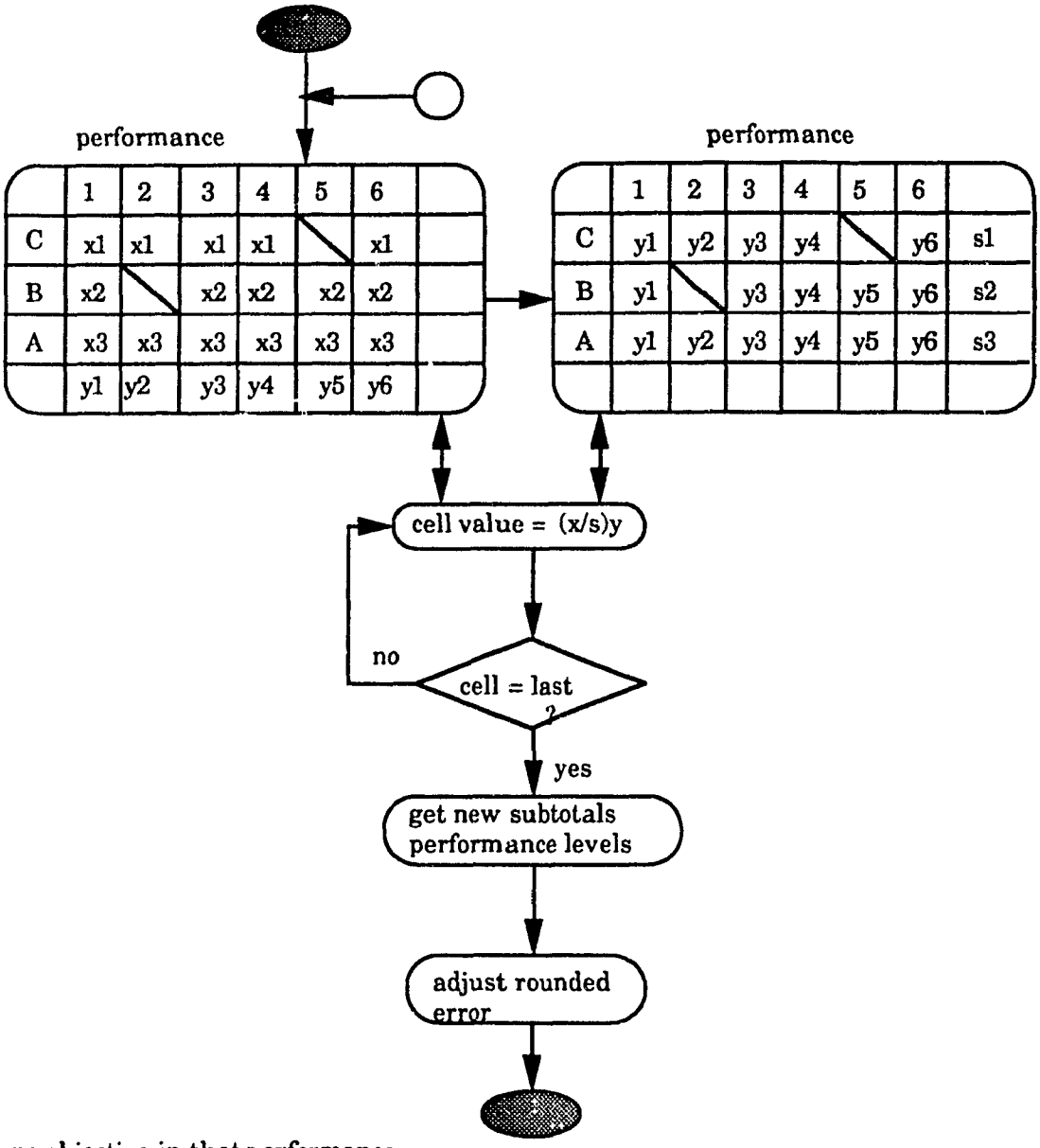


Figure 4.3 Calculate Cell Value According to Topics

4.3.2 adjust by performance



	1	2	3	4	5	6	
C	x1	x1	x1	x1		x1	
B	x2		x2	x2	x2	x2	
A	x3	x3	x3	x3	x3	x3	
	y1	y2	y3	y4	y5	y6	

	1	2	3	4	5	6	
C	y1	y2	y3	y4		y6	s1
B	y1		y3	y4	y5	y6	s2
A	y1	y2	y3	y4	y5	y6	s3

/ no objective in that performance level and topic.

Figure 4.4 Calculate Cell Value According to Performance

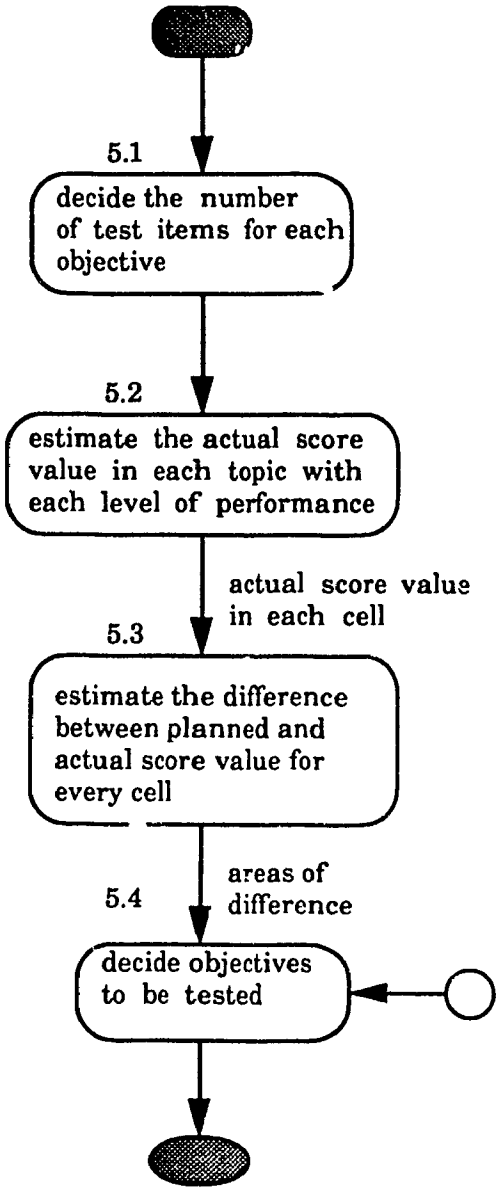


Figure 5.1 Decide Length of The Test

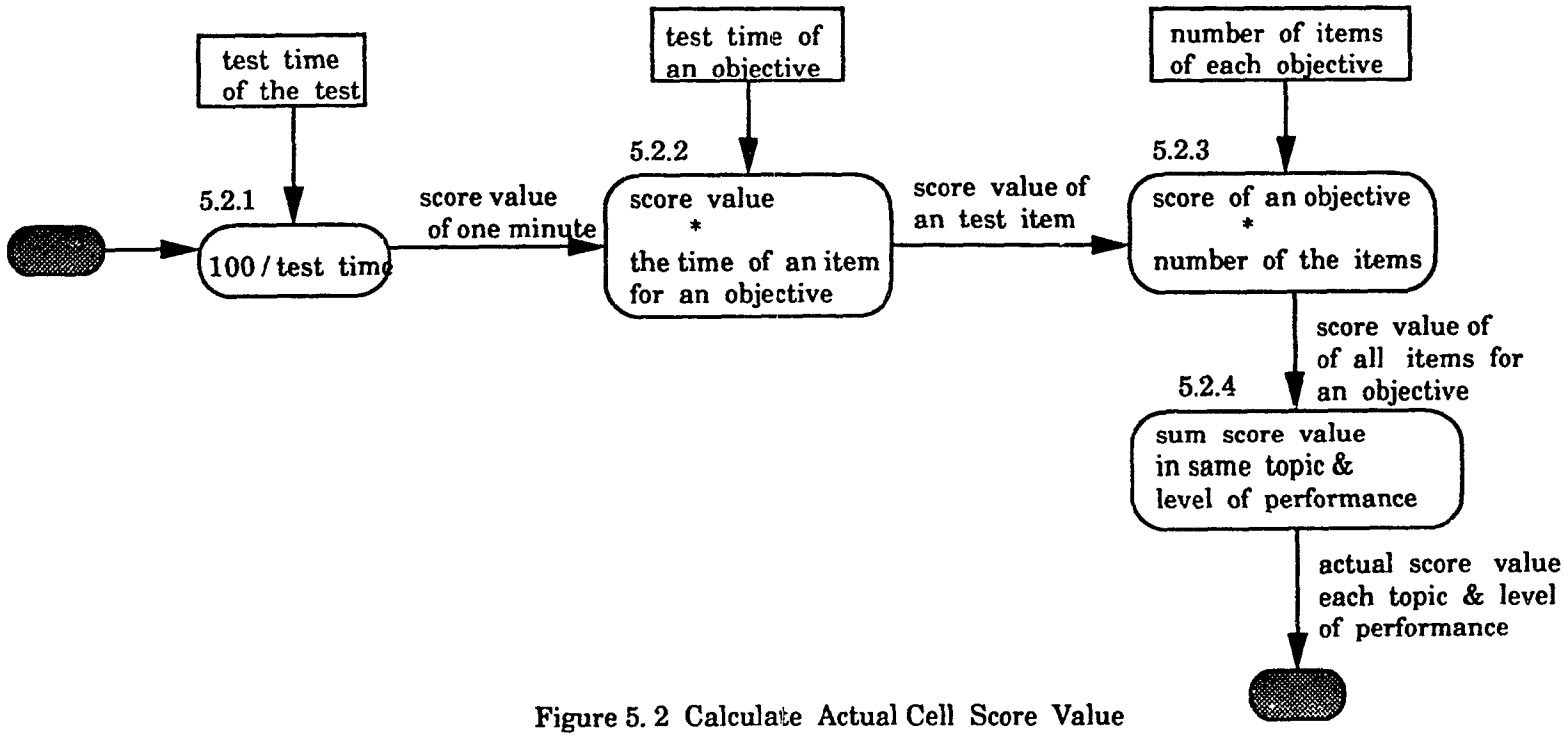


Figure 5. 2 Calculate Actual Cell Score Value

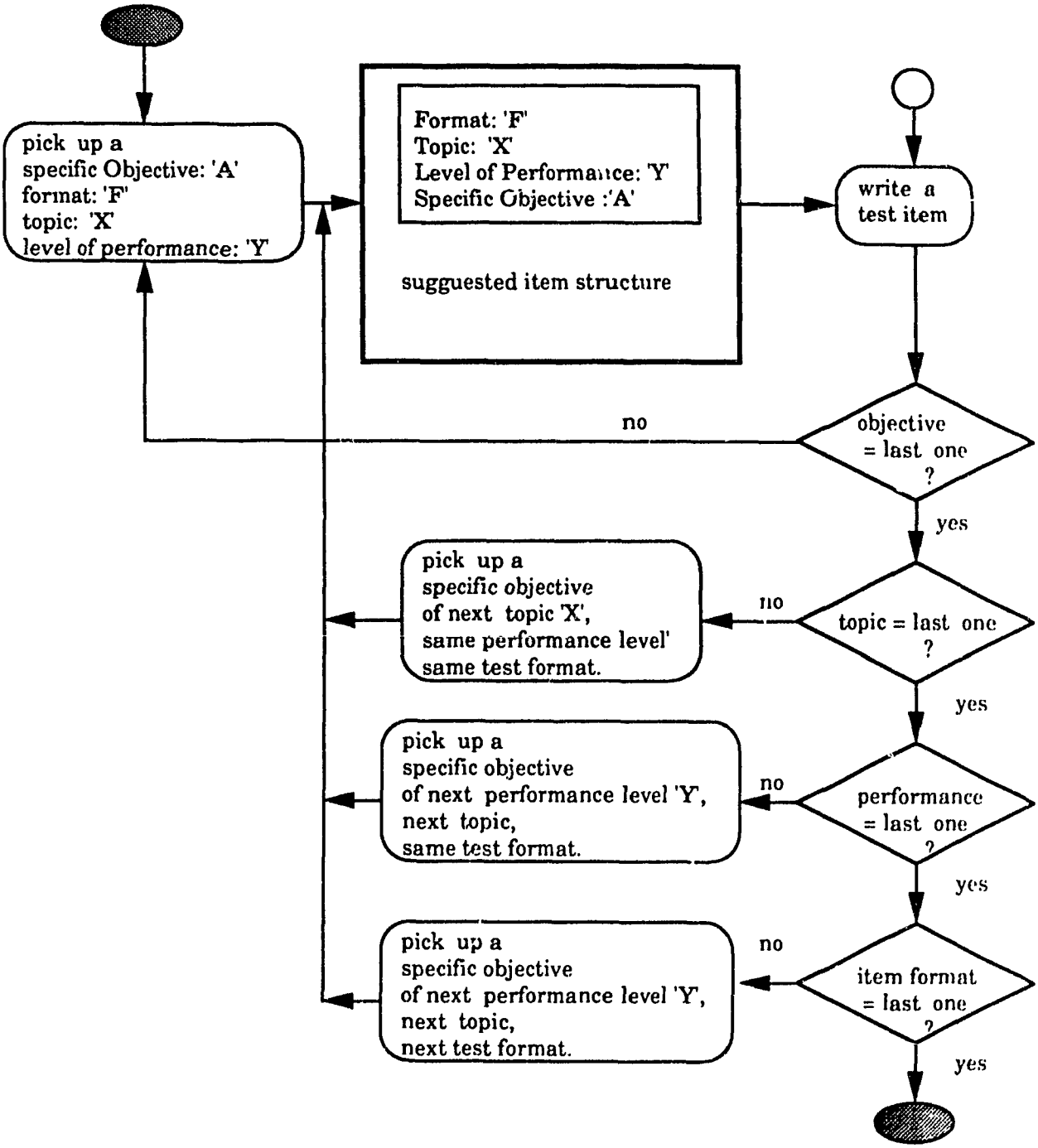


Figure 6.1 Write Test Items

Appendix 2

Samples of Knowledge Representation

```
3. ^_Choose Test Format^_
if
0.0 ^_Introduction^_ Select Test Format, 30005 and
3.0 ^_Coaching^_ and
3.0 ^_condition Essay^_ and
^_tell^_ 30042, Task and ^_tell^_ 30043, Advice and ^_tell^_
30045, Unfamiliar and
'A' := (Objective Total of Test) and
'All' := 'A' + 1 and
'Obj' := 1 and
^_repeat^_ and
3.1 ^_select-format^_ 'Obj' and
'Format' := (Format of 'Obj') and
^_tell^_ 30044, Unfamiliar and
3.2 ^_estimate time needed^_ 'Obj', 'Format' and
'T' := (Test Time of 'Obj') and
^_afterwords^_ 'T', 'Format', 'Obj' and
'Obj' := 'Obj' + 1 and
^_ask to^_ see the test format for next objective and
^_cls^_  and
'Obj' = 'All'  ;

3.0 ^_condition Essay^_
if
^_tell^_ 30040, Task and ^_tell^_ 30041, Task and
^_condition^_ 1 and
^_condition^_ 2 and
^_condition^_ 3 and
(Essay of Test) = Yes
or
^_Decide Essay^_  ;

^_Decide Essay^_
if
^_system^_ 30036   and
^_tell^_ 30033, Advice and ^_move^_ and
^_read-yes^_ "Would like to exclude the essay format from
the test?" and
(Essay of Test) := No
or
(Essay of Test) := Yes ;

^_condition^_ 1
if
(How Score of Test) = Both or
(How Score of Test) = Mannual or
(Resource of Test) := No and (Essay of Test) := No and
^_tell^_ 30008, Advice ;
```

Figure 1. Backward Chaining Rules Used to Choose Item Format

```
^_condition^_ 2
if
'G' := (Grade of Testee) and 'G' < 4  and
^_read-no^_ "Have the testees learnt how to write an essay?"
and
^_tell^_ 30009, Advice and
(Education of Test) := No and
(Essay of Test) := No or ^_true^_ ;

^_condition^_ 3
if
'N' := (Number of Testee) and  'W' := (Who Score of Test)
and
'A' := 'N' / 'W' and  'A' < 30
or
(Essay of Test) := No and
(Resource of Test) := No and
^_Advice3^_ ;

^_Advice3^_
if
(Advice of User) = Yes and
^_cls^_ and
'N' := (Number of Testee) and  'W' := (Who Score of Test)
and
^_write^_ "Number of Testee: ", 'N' and
^_write^_ "How many people to socre: ", 'W' and
^_system^_ 30010 ;


3.0 ^_Coaching^_
if
(Assistance of User) = Yes and
^_tell^_ 30001, Instruction and
^_tell^_ 30002, Purpose and
^_tell^_ 30021, Procedure and
^_tell^_ 30003, Advice and
^_Example^_ 30022, Example, Advice and
^_tell^_ 30015, Unfamiliar and ^_difference^_ or ^_true^_ ;

^_difference^_
if
(Instruction of User) = Yes and
^_system^_ 30023 and
^_read-yes^_"Would you like to know the features of
objective test & essay test?"
and ^_system^_ 30016 and ^_system^_ 30017 and
```

Figure 1. Backward Chaining Rules Used to Choose Item Format

```
^_more concepts^_ Comparing Format, A and
^_a concept^_ 3compare1, Performance Measured, Comparing
Format and
^_a concept^_ 3compare2, Preparation of Questions, Comparing
Format and
^_a concept^_ 3compare3, Sampling of Course Content,
Comparing Format and
^_a concept^_ 3compare4, Control of Response, Comparing
Format and
^_a concept^_ 3compare5, Scoring, Comparing Format and
^_a concept^_ 3compare6, Influence of Learning, Comparing
Format and
^_a concept^_ 3compare7, Reliability, Comparing Format
or ^_true^_ ;

3.1 ^_select-format^_ 'Obj'
if
It ^_is likely^_ Essay, 'Obj' and
^_Choose Essay^_ 'Obj' and ^_tell user^_ 'Obj'
or
It ^_is^_ Short Answer, 'Obj'   and
(Format of 'Obj') := Short Answer         and
(Include of 'Obj') := Yes
or
It ^_will be^_ Multiple Choice, 'Obj' and
(Format of 'Obj') := Multiple Choice   and
(Include of 'Obj') := Yes
or
It ^_will be^_ Matching, 'Obj'         and
(Include of 'Obj') := Yes
or
It ^_will be^_ True-False, 'Obj' and
(Format of 'Obj') := True-False       and
(Include of 'Obj') := Yes
or
^_for revise^_ 'Obj' ;

^_for revise^_ 'Obj'
if
'G' := (General of 'Obj') and
'L' := (Level of 'Obj') and
'S' := (Specific of 'Obj') and
'V' := (Verb of 'Obj')  and
^_go^_ 28, 15 and
^_write^_ "Write down the information of this objective."
and
^_go^_ 1, 1 and
^_write^_ "Performace level: ", 'L' and
```

Figure 1. Backward Chaining Rules Used to Choose Item Format

```
^_write^_ "Objective: ", 'G' and
^_write^_ "Specific objective: ", 'V' , " ", 'S'  and
(Format of 'Obj') := unable to decide and ^_go on^_ and
^_read^_ 'A' ;

% ************************************
It ^_is likely^_ Essay, 'Obj'
if
not (Level of 'Obj') = Knowledge  or
not (Level of 'Obj') = Comprehension ;

^_Choose Essay^_ 'Obj'
if
It ^_is^_ Restricted Response, 'Obj', 'Format' and
(Format of 'Obj') := Restricted Response
or
It ^_is^_ Extended Response, 'Obj', 'Format' and
(Format of 'Obj') := Extended Response ;

^_tell user^_ 'Obj'
if
(Essay of Test) = Yes and (Include of 'Obj') := Yes or
(Include of 'Obj') := No and
^_show test format^_ 'Obj'  and
^_system^_ 30037 ;

It ^_will be^_ Multiple Choice, 'Obj'
if
(Verb of 'Obj') = select or
(Verb of 'Obj') = discover or
(Verb of 'Obj') = identify or
(General of 'Obj') = justify methods/procedures or
(General of 'Obj') = interpret cause-effect relationships;

It ^_will be^_ Matching, 'Obj'
if
^_consider^_ Matching, 'Obj' and
^_make sure^_ Matching, 'Obj' ;

^_make sure^_ Matching, 'Obj'
if
^_cls^_ and
'G' := (General of 'Obj') and
'S' := (Specific of 'Obj') and
'V' := (Verb of 'Obj') and
^_write^_ "", 'Obj',  ". ", 'G' and
^_write^_ "Specific Objective: ", 'V', " ", 'S' and
^_display^_ 30031, Instruction, 40, 1 and
```

Figure 1. Backward Chaining Rules Used to Choose Item Format

```
^_system^_  30030 and
^_read-yes^_
"Sufficient number of homogenous premises & responses can
obtained" and
(Format of 'Obj') := Matching or
^_tell^_ 30032, Task and (Format of 'Obj') := Multiple
Choice ;

^_consider^_ Matching, 'Obj'
if
(Verb of 'Obj') = match or
(Verb of 'Obj') = relate or
(General of 'Obj') = identify relationship between two
things ;

It ^_is^_ Restricted Response, 'Obj', 'Format'
if
(Verb of 'Obj') = explain or
(Verb of 'Obj') = categorize or
(Verb of 'Obj') = describe or
(Verb of 'Obj') = present or
(Verb of 'Obj') = formulate or
(Verb of 'Obj') = state ;

It ^_is^_ Extended Response, 'Obj', 'Format'
if
(Verb of 'Obj') = compose or
(Verb of 'Obj') = write or
(Verb of 'Obj') = plan or
(Verb of 'Obj') = design or
(Verb of 'Obj') = produce or
(Verb of 'Obj') = organize or
(Verb of 'Obj') = express or
(Verb of 'Obj') = integrate or
(Verb of 'Obj') = create or
(Verb of 'Obj') = evaluate ;

It ^_is^_ Short Answer, 'Obj'
if
(Verb of 'Obj') = compute or
(Verb of 'Obj') = calculate or
(General of 'Obj') = interpret diagrams/graphs or
(General of 'Obj') = translate verbal material to
mathematical formulas or
(General of 'Obj') = solve mathematical problems ;

%*****************************************
^_afterwords^_ 'T', 'Format', 'Obj'
```

Figure 1. Backward Chaining Rules Used to Choose Item Format

```
if
^_show test format^_ 'Obj' and
'G' := (General of 'Obj') and
'L' := (Level of 'Obj') and
'V' := (Verb of 'Obj') and
not 'Format' = unable to decide and
^_3tell^_ 30006, Task, 'L', 'G', 'V' and
^_may-want explanation^_ 'Format' or ^_true^_ ;
%^_may change format^_ 'Obj' or ^_true^_ ;

^_show test format^_ 'Obj'
if
^_cls^_ and
'Format' := (Format of 'Obj')  and
'T' := (Test Time of 'Obj') and
'G' := (General of 'Obj') and
'V' := (Verb of 'Obj') and
'S' := (Specific of 'Obj') and
'I' := (Include of 'Obj') and
^_write^_ "Objective ", 'Obj', ". ", 'G'  and
^_write^_ "Spefic Objective:", 'V', " ", 'S' and
^_write^_ "Format: ", 'Format' and
^_write^_ "Time Needed: ", 'T' , " min. " and
^_write^_ "Included in The Test: ", 'I' ;

^_may change format^_ 'Obj'
if
^_show test format^_ 'Obj' and
^_read-no^_ "Do you want to change the test format for the
objective?" or
^_change it^_ 'Obj' and
^_show test format^_ 'Obj' ;

^_may-want explanation^_ 'Format'
if
^_see this^_ 'Format' and
^_see other^_ 'Format' ;

^_see this^_ 'Format'
if
(Instruction of User) = Yes and
('Format' of Format) = Yes and
^_ask again^_ 'Format' ;

^_see other^_ 'Format'
if
(Review of Format)  = Yes and
^_may-want-all^_ or ^_true^_ ;
```

Figure 1. Backward Chaining Rules Used to Choose Item Format

```
^_ask again^_ 'Format'
if
^_read-yes^_ "Would you like the explanation on this test
format?" and
('Format' of Format) := Yes and
^_give explanation on format^_ 'Format'
 or
('Format' of Format) := No ;

^_may-want-all^_
if
^_cls^_ and
^_read-yes^_ "Whould you like the explanation on the other
test formats?" and
^_show-the-user-all^_ or
(Review of Format) := No ;

^_show-the-user-all^_
if
^_clean slot^_ Overview Format or
0.0 ^_Choose Assistance^_ Explanation on Formats, F,
Overview Format and
^_for-every^_ Overview Format ^_has-a^_ 'Slot'
^_do^_ ^_search for Yes^_ 'Slot' ;

^_search for Yes^_ 'Slot'
if
'X' := ('Slot' of Overview Format) and
'X' = Yes and
^_give explanation on format^_ 'Slot' or ^_true^_ ;

^_give explanation on format^_ 'Format'
if
^_cls^_ and ^_go^_ 20, 1 and
^_write^_ "Choose Aspects of Instruction On ", 'Format' and
^_clean slot^_ 'Format' or
1. ^_choose aspects^_ 'Format' and
2. ^_give aspects^_ 'Format'    ;

1. ^_choose aspects^_ 'Format'
if
^_get the set^_ Explanation on Format, C and
^_list-member^_ 'M', C and
('M' of 'Format') := Yes and ^_fail^_  or  ^_no-backtrack^_
;

2. ^_give aspects^_ 'Format'
```

Figure 1. Backward Chaining Rules Used to Choose Item Format

```
if
^_cls^_ and ^_go^_ 20, 1 and
^_write^_ "Explanation On ", 'Format'    and
'Format' = Multiple Choice and ^_Multiple Choice^_ or
'Format' = Matching        and ^_Matching^_        or
'Format' = True-False      and ^_True-False^_      or
'Format' = Short Answer    and ^_Short Answer^_    or
'Format' = Restricted Response and ^_Restricted Response^_
or
'Format' = Extended Response and  ^_Extended Response^_ or
^_true^_ ;

^_Multiple Choice^_
if
^_a concept^_ 3multi1,   Characteristics, Multiple Choice
and
^_a concept^_ 3multi2,   Characteristics, Multiple Choice
and
^_a concept^_ 3multi3,   Usage,     Multiple Choice and
^_a concept^_ 3multi4,   Usage,     Multiple Choice and
^_a concept^_ 3multi5,   Advantages, Multiple Choice
and
^_a concept^_ 3multi6,   Limitations,  Multiple Choice and
^_a concept^_ 3format,   Constructing, Multiple Choice ;

^_Matching^_
if
^_a concept^_ 3format, Limitations and
^_a concept^_ 3format, Advantages and
^_a concept^_ 3format, Usage   and
^_a concept^_ 3format, Characteristics and
^_a concept^_ 3format, Constructing ;

^_True-False     ^_
if
^_a concept^_ 3format, Limitations and
^_a concept^_ 3format, Advantages and
^_a concept^_ 3format, Usage   and
^_a concept^_ 3format, Characteristics and
^_a concept^_ 3format, Constructing ;

^_Short Answer   ^_
if
^_a concept^_ 3short1,   Characteristics, Short Answer and
^_a concept^_ 3short2,  Usage, Short Answer   and
^_a concept^_ 3short3,  Usage, Short Answer   and
^_a concept^_ 3short4,  Usage, Short Answer   and
^_a concept^_ 3short5,  Advantages, Short Answer and
```

Figure 1. Backward Chaining Rules Used to Choose Item Format

```
^_a concept^_ 3short6,  Advantages, Short Answer and
^_a concept^_ 3short7, Limitations, Short Answer and
^_a concept^_ 3short8, Limitations, Short Answer and
^_a concept^_ 3short9, Limitations, Short Answer and
^_a concept^_ 3short10, Limitations, Short Answer ;

^_Restricted Response^_
if
^_a concept^_ 3format,   Limitations and
^_a concept^_ 3format,   Advantages and
^_a concept^_ 3format,   Usage   and
^_a concept^_ 3format,   Characteristics and
^_a concept^_ 3format,   Constructing ;

^_Extended Response^_
if
^_a concept^_ 3format, Limitations and
^_a concept^_ 3format, Advantages and
^_a concept^_ 3format, Usage   and
^_a concept^_ 3format, Characteristics and
^_a concept^_ 3format, Constructing ;

^Z^Z
```

Figure 1. Backward Chaining Rules Used to Choose Item Format

```
4. ^_Test Specification^_
if
4.0 ^_Types of Test^_  and
0.0 ^_Introduction^_ Make A Plan For The Test  and
4.0 ^_Review^_ and
4.1 ^_draw^_ the matrix and
4.2 ^_draw^_ test outline and
4.3 ^_get^_ the planned score and ^_tell^_ 40004, Task and
^_cls^_ and
4.0 ^_Summarize^_ and ^_cls^_ and
4.0 ^_Synthesize^_ ;

4.0 ^_Types of Test^_
if
'X' := (Purpose of Test) and
4.0 ^_Which Type^_ 'X' ;

4.0 ^_Which Type^_ 'X'
if
^_It is^_ Placement, 'X' and (Type of Test) := Placement or
^_It is^_ Formative, 'X' and (Type of Test) := Formative or
^_It is^_ Summative, 'X' and (Type of Test) := Summatove or
^_It is^_ Diagnostic, 'X' and (Type of Test) := Diagnostic
;

^_It is^_ Placement, 'X'
if
'X' = Training Readiness or 'X' = Learning Readiness or
'X' = Job Assignment or
'X' = Certifying and
'Y' := (Has Quota of Test?) and  'Y'= No;

^_It is^_ Formative, 'X'
if
'X' = Training Evaluation and 'Y' := (Coverage of Test) and
'Y' = Instruction Unit or
'X' = Evaluation and 'Y' := (Coverage of Test) and 'Y' =
Instruction Unit;

^_It is^_ Summative, 'X'
if
'X' = Certifying and 'Y' := (Has Quota of Test?)  and 'Y' =
Yes or
'X' = Grading and 'Y' := (Coverage of Test) and 'Y' = Term
Course or
'X' = Evaluation and 'Y' := (Coverage of Test) and 'Y' =
Term Course ;
```

Figure 2. Meta Rules Used to Index Production Rules

```
% Coaching ********************************

* Variables are marked by '    '.

^_display^_ 'X', 'Y', 'Col', 'Line'
if
('Y' of User) = Yes and
^_msg^_ 'X', 'R' and
^_move-cursor-to^_ 'Col', 'Line' and
^_write-text^_ 'R' and ^_stop^_ or ^_true^_ ;

^_stay^_ 'X', 'Col', 'Line'
if
^_msg^_ 'X', 'R' and
^_move-cursor-to^_ 'Col', 'Line' and
^_write-text^_ 'R'  or ^_ture^_ ;

^_stop^_
if
^_go^_ 19, 21 and
^_write^_ "Press Enter Key to Continue" and
^_read^_ 'New' and
^_clear space^_  21, 22;


^_Press^_
if
^_go^_ 19, 21 and
^_write^_ "Press Enter Key to Continue"  ;

^_a concept^_ 'num', 'item', 'concept'
if
('item' of 'concept') = Yes and
^_msg^_ 'num', 'ref' and  ^_pop-text^_ 'ref' and ^_go on^_
or ^_true^_ ;


^_Example^_ '1', 'Y', 'X'
if
('Y' of User) = Yes and
('X' of User) = Yes and ^_go on^_ and
^_msg^_ '1', 'Ref' and
^_pop-text^_ 'Ref', '1'  or ^_true^_ ;
```

Figure 3. Production Rules with Variables

```
^_1tell^_ 'X', 'Y', '1'
if
('Y' of User) = Yes and ^_go on^_ and
^_msg^_ 'X', 'Ref' and
^_pop-text^_ 'Ref', '1'  or ^_true^_ ;

^_5tell^_ 'X', 'Y', '1', '2', '3', '4', '5'
if
('Y' of User) = Yes and ^_go on^_ and
^_msg^_ 'X', 'Ref' and
^_pop-text^_ 'Ref', '1', '2', '3', '4', '5' or ^_true^_ ;


^_3tell^_ 'X', 'Y', '1', '2', '3'
if
('Y' of User) = Yes and ^_go on^_ and
^_msg^_ 'X', 'Ref' and
^_pop-text^_ 'Ref', '1', '2', '3' or ^_true^_ ;

^_tell^_ 'X', 'Y'
if
('Y' of User) = Yes and ^_go on^_ and
^_msg^_ 'X', 'Ref' and
^_pop-text^_ 'Ref' or ^_true^_ ;


^_more^_ 'A'
if
^_read-yes^_ "Would you like further explanation on this
topic?" and
^_msg^_ 'A', 'Ref' and
^_pop-text^_ 'Ref' and ^_go on^_ or ^_true^_ ;

^_change it^_ 'Obj'
if
^_go^_ 45, 1 and ^_msg^_ Format, 'R' and ^_write-text^_ 'R'
and
^_pop-text-input^_ 'F',1, {Please type in the number of the
format you choose.}
and ^_translate^_ 'F'  and
(Format of 'Obj') := 'F' and
3.2 ^_estimate time needed^_ 'Obj', 'F' ;




^_translate^_ 'F'
```

Figure 3. Production Rules with Variables

```
if
'F' = 1 and  'F' := Multiple Choice  or
'F' = 2 and  'F' := Matching or
'F' = 3 and  'F' := True-False or
'F' = 4 and  'F' := Short Answer or
'F' = 5 and  'F' := Restricted Response or
'F' = 6 and  'F' := Extended Response  ;


^_more concepts^_  'Topic', 'Set List'
if
^_read-yes^_ "Would you like further explanation on it?" and

^_get the set^_ 'Topic', 'Set List' and
^_list-member^_ 'M', 'Set List' and
('M' of 'Topic') := Yes and ^_fail^_ or ^_no-backtrack^_ ;

^_system^_ 'A'
if
^_go on^_ and
^_msg^_ 'A', 'Ref' and ^_pop-text^_ 'Ref' ;

%  User Choose ************************
0.0 ^_Introduction^_ 'X', 'Y'
if
^_ask to^_ 'X' and
^_tell^_ 'Y', Unfamiliar and ^_tell^_ 10008, Unfamiliar and

^_read-yes^_ "Would you like to have assistance for this
task?" and
(Assistance of User) := Yes and
^_display^_ 10008, Unfamiliar, 1, 1   and
0.0 ^_Choose Assistance^_ Assistance, Item, User and ^_cls^_
and
^_read-yes^_ "Would you like to have examples to illustrate
the explanation?"
and (Example of User) := Yes or ^_true^_ ;

^_clean slot^_ 'Frame'
if
^_for-every^_ 'Frame' ^_has-a^_ 'Slot'
^_do^_ ^_clean^_ 'Frame', 'Slot' ;

^_clean^_ 'Frame', 'Slot'
if
('Slot' of  'Frame') := No ;
```

Figure 3. Production Rules with Variables

```
0.0 ^_Choose Assistance^_ 'Screen', 'List', 'Frame'
if
^_get the set^_ 'Screen', 'List' and
^_list-member^_ 'M', 'List' and
('M' of 'Frame') := Yes and ^_fail^_ or ^_no-backtrack^_ ;

^_get the set^_  'Topic', 'Set List'
if
^_list-assign^_ "[]", Set List and
'X' ^_is a choice for^_ 'Topic' and
^_add to list^_ 'X', Set List and
^_fail^_
or
^_set^_ 'Topic', 'Set List' ;

^_add to list^_ 'M', 'L'
if
^_list-member^_ 'M', 'L' or ^_list-add-member^_ 'M', 'L' ;

^_set^_ 'Topic', 'Set List'
if
^_move^_ and
^_dialog-build^_ Get Set and
^_dialog-set-value^_ Get Set, Topic, 'Topic' and
^_dialog-execute^_ Get Set, 'Exit Button' and
^_dialog-get-set^_ Get Set, Set, 'Set List' and
^_dialog-dispose^_ Get Set and
^_no-backtrack^_  and
not  'Exit Button' = Esc and ^_no-backtrack^_ ;


* Variables are marked by '   '.
```

Figure 3. Production Rules with Variables

```
^_Frame^_: Test
^_Parent^_: Thing
^_Slot^_:  Name                 ^_Value^_:
^_Slot^_:  Setting              ^_Value^_:
^_Slot^_:  Purpose              ^_Value^_:
^_Slot^_:  Coverage             ^_Value^_:
^_Slot^_:  Has Quota?           ^_Value^_:
^_Slot^_:  Content              ^_Value^_:
^_Slot^_:  Time                 ^_Value^_:
^_Slot^_:  Min Score            ^_Value^_:
^_Slot^_:  Frequency            ^_Value^_:
^_Slot^_:  Need Pretest?        ^_Value^_:
^_Slot^_:  How Score            ^_Value^_:
^_Slot^_:  Who Score            ^_Value^_:
^_Slot^_:  Content Area         ^_Value^_:
^_Slot^_:  Type                 ^_Value^_:
^_Slot^_:  Nature               ^_Value^_:
^_Slot^_:  Focus                ^_Value^_:
^_Slot^_:  Difficulty Range     ^_Value^_:
^_Slot^_:  Difficulty Level     ^_Value^_:
^_Slot^_:  Score                ^_Value^_:
^_Slot^_:  Objective Total      ^_Value^_:
^_Slot^_:  Item Total           ^_Value^_:
^_Slot^_:  Explain              ^_Value^_:
^_Slot^_:  Items                ^_Value^_: 0


^_Frame^_: Testee
^_Parent^_: Thing
^_Slot^_:  Number       ^_Value^_:
^_Slot^_:  Education    ^_Value^_:
^_Slot^_:  Grade        ^_Value^_:
^_Slot^_:  Age          ^_Value^_:
```

Figure 4. Frames Used to Record Facts

```
^_Frame^_: Table
^_Parent^_:  Concept
^_Slot^_: Test Types  ^_Value^_:
^_Slot^_: Validity    ^_Value^_: Yes
^_Slot^_: Reliability ^_Value^_:
^_Slot^_: Usability   ^_Value^_:

^_Frame^_:  User
^_Parent^_: Thing
^_Slot^_:   Training      ^_Value^_: No
^_Slot^_:   Unfamiliar    ^_Value^_: No
^_Slot^_:   Assistance    ^_Value^_: No
^_Slot^_:   Instruction   ^_Value^_: No
^_Slot^_:   Purpose       ^_Value^_: No
^_Slot^_:   Procedure     ^_Value^_: No
^_Slot^_:   Task          ^_Value^_: No
^_Slot^_:   Advice        ^_Value^_: No
^_Slot^_:   Example       ^_Value^_: No

^_Frame^_: Review4
^_Parent^_: Table
^_Slot^_:   Content Validity   ^_Value^_:
^_Slot^_:   Test Types         ^_Value^_:


^_Frame^_: Types
^_Parent^_:  Table
^_Slot^_: Placement  ^_Value^_:
^_Slot^_: Formative  ^_Value^_:
^_Slot^_: Summative  ^_Value^_:
^_Slot^_: Diagnostic ^_Value^_:

^_Frame^_: Validity
^_Parent^_: Table
^_Slot^_: Content Validity  ^_Value^_:
^_Slot^_:  Criterion-Related Validity ^_Value^_:
^_Slot^_:  Construct Validity ^_Value^_:
^_Slot^_:  Relation With Other Concepts ^_Value^_:
^_Slot^_:  Factors Influencing Validity ^_Value^_:

^_Frame^_: Usablility
^_Parent^_: Table
^_Slot^_: Easy to Administrate   ^_Value^_:
^_Slot^_:  Time for Administration ^_Value^_:
^_Slot^_:  Easy to Score ^_Value^_:
^_Slot^_:  Easy to Interpretation ^_Value^_:
^_Slot^_:  Equivalent Forms ^_Value^_:
^_Slot^_:  Cost of Testing ^_Value^_:
```

Figure 5. Frames Used to Store Information Which has Inheritable Nature

```
^_Frame^_: Reliablity
^_Parent^_: Table
^_Slot^_: Stability  ^_Value^_:
^_Slot^_: Equivalence ^_Value^_:
^_Slot^_:  Internal Consistency ^_Value^_:
^_Slot^_:  Relation With Other Concepts ^_Value^_:
^_Slot^_:  Factors Influencing Reliability ^_Value^_:

^_Frame^_: Comparing Format
^_Parent^_: Concept
^_Slot^_:  Phase                    ^_Value^_: 3
^_Slot^_:  Performance Measured  ^_Value^_:
^_Slot^_:  Preparation of Questions ^_Value^_:
^_Slot^_:  Sampling of Course Content ^_Value^_:
^_Slot^_:  Control of Response ^_Value^_:
^_Slot^_:  Scoring ^_Value^_:
^_Slot^_:  Influence of Learning ^_Value^_:
^_Slot^_:  Reliability ^_Value^_:

^_Frame^_: Format
^_Parent^_: Concept
^_Slot^_:  Phase                    ^_Value^_: 3
^_Slot^_:  Short Answer    ^_Value^_:          Yes
^_Slot^_:  True-False      ^_Value^_:          Yes
^_Slot^_:  Multiple Choice ^_Value^_:          Yes
^_Slot^_:  Extended Response       ^_Value^_: Yes
^_Slot^_:  Restricted Response ^_Value^_:      Yes
^_Slot^_:  Matching         ^_Value^_:         Yes
^_Slot^_:  Review  ^_Value^_:  Yes

^_Frame^_: Overview Format
^_Parent^_: Concept
^_Slot^_:  Phase                    ^_Value^_: 3
^_Slot^_:  Short Answer    ^_Value^_:
^_Slot^_:  True-False      ^_Value^_:
^_Slot^_:  Multiple Choice ^_Value^_:
^_Slot^_:  Extended Response       ^_Value^_:
^_Slot^_:  Restricted Response ^_Value^_:
^_Slot^_:  Matching         ^_Value^_:

^_Frame^_: Short Answer
^_Parent^_: Format
^_Slot^_:  Phase              ^_Value^_: 3
^_Slot^_:  Advantages         ^_Value^_:
^_Slot^_:  Limitations        ^_Value^_:
^_Slot^_:  Characteristics    ^_Value^_:
^_Slot^_:  Usage              ^_Value^_:
^_Slot^_:  Constructing       ^_Value^_:
```

Figure 5. Frames Used to Store Information Which has Inheritable Nature

```
^_Frame^_: Multiple Choice
^_Parent^_: Format
^_Slot^_:  Phase                ^_Value^_: 3
^_Slot^_:  Advantages       ^_Value^_:
^_Slot^_:  Limitations      ^_Value^_:
^_Slot^_:  Characteristics  ^_Value^_:
^_Slot^_:  Usage            ^_Value^_:
^_Slot^_:  Constructing     ^_Value^_:

^_Frame^_: Short Answer
^_Parent^_: Format
^_Slot^_:  Phase                ^_Value^_: 3
^_Slot^_:  Advantages       ^_Value^_:
^_Slot^_:  Limitations      ^_Value^_:
^_Slot^_:  Characteristics  ^_Value^_:
^_Slot^_:  Usage            ^_Value^_:
^_Slot^_:  Constructing     ^_Value^_:

^_Frame^_: True-False
^_Parent^_: Format
^_Slot^_:  Phase                ^_Value^_: 3
^_Slot^_:  Advantages       ^_Value^_:
^_Slot^_:  Limitations      ^_Value^_:
^_Slot^_:  Characteristics  ^_Value^_:
^_Slot^_:  Usage            ^_Value^_:
^_Slot^_:  Constructing     ^_Value^_:


^_Frame^_: Matching
^_Parent^_: Format
^_Slot^_:  Phase                ^_Value^_: 3
^_Slot^_:  Advantages       ^_Value^_:
^_Slot^_:  Limitations      ^_Value^_:
^_Slot^_:  Characteristics  ^_Value^_:
^_Slot^_:  Usage            ^_Value^_:
^_Slot^_:  Constructing     ^_Value^_:
^_Frame^_: Concept
^_Parent^_: Thing
^_Slot^_: Number  ^_Value^_:


^_Frame^_: Extended Response
^_Parent^_: Format
^_Slot^_:  Phase                ^_Value^_: 3
^_Slot^_:  Advantages       ^_Value^_:
^_Slot^_:  Limitations      ^_Value^_:
^_Slot^_:  Characteristics  ^_Value^_:
```

Figure 5. Frames Used to Store Information Which has Inheritable Nature

```
^_Slot^_:  Usage              ^_Value^_:
^_Slot^_:  Constructing       ^_Value^_:
^_Frame^_: Concept
^_Frame^_: Restricted Response
^_Parent^_: Format
^_Slot^_:  Phase                    ^_Value^_: 3
^_Slot^_:  Advantages         ^_Value^_:
^_Slot^_:  Limitations        ^_Value^_:
^_Slot^_:  Characteristics    ^_Value^_:
^_Slot^_:  Usage              ^_Value^_:
^_Slot^_:  Constructing       ^_Value^_:
^_Frame^_: Concept
^_Parent^_: Thing
^_Slot^_: Number  ^_Value^_:
^Z^Z
```

Figure 5. Frames Used to Store Information Which has Inheritable Nature

know common terms ^_is a choice for^_ At Knowledge Level
know specific facts ^_is a choice for^_ At Knowledge Level
know methods/procedures ^_is a choice for^_ At Knowledge Level
know principles ^_is a choice for^_ At Knowledge Level
know basic concepts ^_is a choice for^_ At Knowledge Level

understand facts/principles ^_is a choice for^_ At Comprehension Level
interpret charts/graphs ^_is a choice for^_ At Comprehension Level
interpret verbal material ^_is a choice for^_ At Comprehension Level
translate verbal material to mathematical formulas ^_is a^_ ^_choice for^_ At Comprehension Level
estimate consequences implied in data ^_is a choice for^_ At Comprehension Level
justifiy methods/procedures ^_is a choice for^_ At Comprehension Level

apply theories to pratical situations ^_is a choice for^_ At Application Level
apply principles to new situations ^_is a choice for^_ At Application Level
solve mathematical problems ^_is a choice for^_ At Application Level
construct charts/graphs^_ is a choice for^_ At Application Level
correctly use a procedure ^_is a choice for^_ At Application Level

recognize unstated assumptions ^_is a choice for^_ At Analysis Level
recognize logical fallacies in reasoning ^_is a choice for^_ At Analysis Level
distinguishe between facts/inferences ^_is a choice for^_ At Analysis Level
evaluate the relevancy of data ^_is a choice for^_ At Analysis Level
analyze organizational structure of a work ^_is a choice for^_ At Analysis Level

propose a plan for an experiment ^_is a choice for^_ At Synthesis Level
write a creative short story ^_is a choice for^_ At Synthesis Level
make a plan for solving a problem ^_is a choice for^_ At Synthesis Level
propose a plan for an experiment ^_is a choice for^_ At

Figure 6. Items of Lists

```
Synthesis Level
formulate a scheme for classifying... ^_is a choice for^_ At
Synthesis Level

judge the value of a work ^_is a choice for^_ At Evaluation
Level
judge the adequacy of a conclusion ^_is a choice for^_ At
Evaluation Level
judge the value of a work/idea^_is a choice for^_ At
Evaluation Level

Instruction ^_is a choice for^_ Assistance
Purpose ^_is a choice for^_ Assistance
Procedure ^_is a choice for^_ Assistance
Task ^_is a choice for^_ Assistance
Advice ^_is a choice for^_ Assistance

% 3. Choose Test Format
****************************************
Performance Measured   ^_is a choice for^_ Comparing Format
Preparation of Questions   ^_is a choice for^_ Comparing
Format
Sampling of Course Content   ^_is a choice for^_ Comparing
Format
Control of Response   ^_is a choice for^_ Comparing Format
Scoring   ^_is a choice for^_ Comparing Format
Influence of Learning   ^_is a choice for^_ Comparing Format

Reliability   ^_is a choice for^_ Comparing Format

Advantages ^_is a choice for^_ Explanation on Format
Limitations   ^_is a choice for^_ Explanation on Format
Characteristics   ^_is a choice for^_ Explanation on Format
Usage   ^_is a choice for^_ Explanation on Format


Multiple Choice ^_is a choice for^_ Explanation on Formats
Matching ^_is a choice for^_ Explanation on Formats
True-False ^_is a choice for^_ Explanation on Formats
Short Answer ^_is a choice for^_ Explanation on Formats
Restricted Response ^_is a choice for^_ Explanation on
Formats
Extended Response ^_is a choice for^_ Explanation on Formats


Multiple Choice ^_is a choice for^_ Review on Formats
Matching ^_is a choice for^_ Review on Formats
True-False ^_is a choice for^_ Review on Formats
Short Answer ^_is a choice for^_ Review on Formats
Restricted Response ^_is a choice for^_ Review on Formats
```

Figure 6. Items of Lists

```
Extended Response ^_is a choice for^_ Review on Formats

% 4. Table of Specification
************************************
Content Validity ^_is a choice for^_ Review Concept
Test Types ^_is a choice for^_ Review Concept

Content Validity ^_is a choice for^_ Validity
Criterion-Related Validity ^_is a choice for^_ Validity
Construct Validity ^_is a choice for^_ Validity
Relation With Other Concepts ^_is a choice for^_ Validity


Stability ^_is a choice for^_ Reliability
Equivalence ^_is a choice for^_ Reliability
Internal Consistency ^_is a choice for^_ Reliability
Relation With Other Concepts ^_is a choice for^_ Reliability



Easy to Administrate ^_is a choice for^_ Usability
Time for Administration^_ is a choice for^_ Usability
Easy to Score ^_is a choice for^_ Usability
Easy to Interpretation ^_is a choice for^_ Usability
Equivalent Forms ^_is a choice for^_ Usability
Cost of Testing ^_is a choice for^_ Usability

Placement Test  ^_is a choice for^_ Test Types
Formative Test  ^_is a choice for^_ Test Types
Summative Test  ^_is a choice for^_ Test Types
Diagnostic Test  ^_is a choice for^_ Test Types

  ^_is a choice for^_
  ^_is a choice for^_

Validity  ^_is a choice for^_ Topics
Reliability  ^_is a choice for^_  Topics
Usability  ^_is a choice for^_  Topics
Test Types ^_is a choice for^_ Topics
  ^_is a choice for^_
  ^_is a choice for^_
  ^_is a choice for^_
  ^_is a choice for^_
  ^_is a choice for^_
  ^_is a choice for^_

^Z^Z
```

Figure 6. Items of Lists

```
% Purpose  ************************************
^_msg^_  400, {
            Why Construct A Test Specification
A test, no matter how extensive, is almost always a sample
of the many possible test items that could be included. That
means that our limited samples must be selecte in such a way
that they provide as representative a smaple as possible in
each of the various areas for which the test is being
developed.
pur/400}

% Procedure  ************************************
^_msg^_ 40040, {   How to Build A Test Specification
A test specification is in the form of a table.
Building the table of test specification involves:
1. obtaining the list of instructional objectives
2. outlining course content
3. preparing a matrix specifying the nature of
   the desired test sample
pro/40040}

^_msg^_ 40001, {
The test specification is used a tool to obtain
a representative sample which include the objectives
with the greatest importance and reflect the purpose
of the test. It is the blue print of the test design.
in/40001}

% Intention
^_msg^_  4table1, {
4table1
    What is Table of Test Specification
The table of specification is set up as
a general evaluation plan. It is a blueprint
of a test which indicates the relative
emphasis given to each aspect of evaluation.
}

^_msg^_ 4table2, {
4table2
           Content Validity
The content of a course or curriculum may be
broadly defined to include both subject matter
content and performance objectives.
}
```

Figure 7. Canned Messages

```
^_msg^_ 4table3, {
4table3
          Content Validity
The content area is concerned with the topics
to be learned, and the performance objectives
are concened with types of performance learners
testees are expected to demonstrated
(e.g., knows, comprehends, applies).
}

^_msg^_ 4table4, {
4table4
           Content Validity
Both of these aspects are of concern in determining
content validity. We would like to any test that
we construct, or select, to provide resutls that that
are representative of both the content areas and the
objectives we wish to measure.
}

^_msg^_ 4table5 , {
table5
Table of Test Specification with Content Validity
Table of Specification is a device which is used
to obtain a representative sample of testees'
performance in each of the areas to be measured,
so we say the table of specification is a device
to ensure content validity in test design.
}

^_msg^_ 4table6, {
table6
 Table of Test Specification with Content Validity
The table indicates the sample of learning tasks to be
measured and the closer the test items correspond to
the specific sample, the greater the likelihood of
having satisfactory content validity.
}

% Procedure
^_msg^_ 40001   , {
40001
     How to Build A Test Specification
It is a two-way chart which relates the performance
objectives to the tested content area.
}
```

Figure 7. Canned Messages

```
^_msg^_ 40002, {
40002
   How to Build A Test Specification
It is a matter of analyzing the content and
tasks included in the evaluation instrument
and the domain of outcomes to be measured and
judging the degree of correspondence between
them.}

^_msg^_ 40017, {
400017
    What Do You Need Before Build A Test Specification
To construct a table of specification, we need
1. a list of instructional objectives
2. the course of content.
}

^_msg^_ 40003  , {
40003
      How to Build A Test Specification
The performance level of the general objectives
are listed across the top of the table, the major
areas of content down the left side.
}

^_msg^_ 40004  , {

     How to Build A Test Specification
The value of each cell indicate what proportion of
the test items should be devoted to each objective
and each area of content, which reflect the
emphysis and intention of the test designer.
}

% Instruction
**********************************************
^_msg^_ 40023, {
23  Syn
By developing a test plan using the test specification the
content validity is ensured. However, content validity is
only
one of several factors which effects the quality of a test
instrument.
}
```

Figure 7. Canned Messages

Appendix 3

Formative Evaluation Questionnaires

Evaluation Questions to Domain Experts

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 3 4 5 Agree
2. It reduces difficulties in the design task.	Not at all 1 2 3 4 5 Significantly
3. By using the system an instructor will develop a test of good quality.	Disagree 1 2 3 4 5 Agree
4. Are there any important issues missing?	
5. Are there any important features missing?	
6. What are the best features of the system?	
7. Where should it be modified ?	

External Performance

8. It gives the user easy access to alternatives.	Not at all 1 2 3 4 5 Significantly
9. It asks questions in a consistent manner.	Not at all 1 2 3 4 5 Very much
10. The advice is clear.	Not at all 1 2 3 4 5 Very
11. The advice is relevant and adequate.	Not at all 1 2 3 4 5 Very
12. The justification states underlying reasoning.	Not at all 1 2 3 4 5 Very much
13. The justification is adequate.	Not at all 1 2 3 4 5 Very
14. It is easy to use as software.	Not at all 1 2 3 4 5 Very
15. The screen display is clear.	Not at all 1 2 3 4 5 Very
16. It reduces time of the administrative tasks.	Unlikely 1 2 3 4 5 Very likely

Knowledge Representation Aspect

17. It depicts the expertise properly.	Not at all 1 2 3 4 5 Very much
18. The objects, variables & their values are complete	Disagree 1 2 3 4 5 Agree
19. The structure of representation is similar to your perception of the domain	Not at all 1 2 3 4 5 Very much

Evaluation Questions to Computer Experts

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 3 4 5 Agree
2. What are the best features of the system?	
3. Where should it be modified ?	
4. This content appears appropriate for using an expert system.	Not at all 1 2 3 4 5 Very

External Performance

5. It outputs results in a reasonable amount of time.	Disagree 1 2 3 4 5 Agree
6. It asks questions in a consistent manner.	Not at all 1 2 3 4 5 Very
7. The advice is clear.	Not at all 1 2 3 4 5 Very
8. It is easy to use as software.	Not at all 1 2 3 4 5 Very
9. The screen display is clear.	Not at all 1 2 3 4 5 Very

Knowledge Representation Aspect

10. The structure of representation is coherent.	Not at all 1 2 3 4 5 Very
11. The structure of representation is adequate to retrieve, store and analyze data easily.	Not at all 1 2 3 4 5 Very
12. The design of the inference engine is adequate for efficient inference.	Not at all 1 2 3 4 5 Very

Evaluation Questions to End Users

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 3 4 5 Agree
2. It reduces difficulties in the design task.	Not at all 1 2 3 4 5 Significantly
3. What are the best features of the system?	
4. Where should it be modified?	
5. I would like such a tool to assist me in my test design work.	Not at all 1 2 3 4 5 Very much

External Performance

6. It reduces design time of the administritive tasks.	Unlikely 1 2 3 4 5 Very likely
7. It gives me easy access to alternatives.	Not at all 1 2 3 4 5 Significantly
8. It asks questions in a consistent manner.	Not at all 1 2 3 4 5 Very
9. The advice is meaningful to me.	Not at all 1 2 3 4 5 Very
10. It improves my understanding of test design.	Not at all 1 2 3 4 5 Significantly
11. The assistance is tailored to my needs.	Not at all 1 2 3 4 5 Very much
12. I like the structure of the assistance.	Not at all 1 2 3 4 5 Very much
13. It is easy to use as software.	Not at all 1 2 3 4 5 Very
14. The screen display is clear.	Not at all 1 2 3 4 5 Very

Appendix 4

Evaluation Results

Evaluation Questions to Domain Expert 1

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 3 4 **5** Agree
2. It reduces difficulties in the design task .	Not at all 1 2 **3** 4 5 Significantly
3. By using the system an instructor will develop a test of good quality.	Disagree 1 2 3 4 **5** Agree
4. Are there any important issues missing?	decision orientation requirement on testing
5. Are there any important features missing?	
6. What are the best features of the system?	step by step guidance
7. Where should it be modified ?	

External Performance

8. It gives the user easy access to alternatives.	unclear yet
9. It asks questions in a consistent manner.	yes, not directive enough yet
10. The advice is clear.	sometimes
11. The advice is relevant and adequate.	is relevant, not adequate yet
12. The justification states underlying reasoning.	Not at all **1** 2 3 4 5 Very much
13. The justification is adequate.	Poorly **1** 2 3 4 5 Very well
14. It is easy to use as software.	cannot determine yet
15. The screen display is clear.	Not at all 1 2 3 4 **5** Very
16. It reduces time of the administrative tasks.	cannot determine yet

Knowledge Representation Aspect

17. It depicts the expertise properly.	Not at all 1 2 3 **4** 5 Very much
18. The objects, variables & their values are complete	earlier part - no; later part yes
19. The structure of representation is similar to your perception of the domain.	Not at all 1 2 **3 4** 5 Very much

Evaluation Questions to Domain Expert 2

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 3 4 **5** Agree
2. It reduces difficulties in the design task .	Not at all 1 2 **3** 4 5 Significantly
3. By using the system an instructor will develop a test of good quality.	Disagree 1 2 **3** 4 5 Agree
4. Are there any important issues missing?	a dynamic map to show where you are & doing what.
5. Are there any important features missing?	
6. What are the best features of the system?	
7. Where should it be modified ?	test data processing and reporting

External Performance

8. It gives the user easy access to alternatives.	Not at all 1 2 3 **4** 5 Significantly
9. It asks questions in a consistent manner.	Not at all 1 2 3 **4** 5 Very much
10. The advice is clear.	Not at all 1 2 3 **4** 5 Very much
11. The advice is relevant and adequate.	Not at all 1 2 **3 4** 5 Very much
12. The justification states underlying reasoning.	Not at all 1 **2** 3 4 5 Very well
13. The justification is adequate.	Poorly 1 **2** 3 4 5 Very well
14. It is easy to use as software.	Not at all 1 **2** 3 4 5 Very well
15. The screen display is clear.	Not at all 1 **2 3** 4 5 Very
16. It reduces time of the administrative tasks.	cannot determine yet

Knowledge Representation Aspect

17. It depicts the expertise properly.	Not at all 1 2 3 **4** 5 Very much
18. The objects, variables & their values are complete	Not at all 1 2 3 **4** 5 Very much
19 The structure of representation is similar to your perception of the domain.	Not at all 1 2 **3** 4 5 Very much

Evaluation Questions to Computer Expert 1

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 **3** 4 5 Agree
2. What are the best features of the system?	
3. Where should it be modified ?	user interface and explanations
4. This content appears appropriate for using an expert system.	Not at all 1 2 **3** 4 5 Very

External Performance

5. It outputs results in a reasonable amount of time.	Disagree 1 2 3 **4** 5 Agree
6. It asks questions in a consistent manner.	Not at all 1 2 3 **4** 5 Very
7. The advice is clear.	Not at all 1 2 **3** 4 5 Very
8. It is easy to use as software.	It is too early to judge.
9. The screen display is clear.	Not at all 1 2 **3** 4 5 Very

Knowledge Representation Aspect

10. The structure of representation is coherent.	It is too early to judge.
11. The structure of representation is adequate to retrieve, store and analyze data easily.	Not at all 1 2 3 **4** 5 Very
12. The design of the inference engine is adequate for efficient inference.	Not at all 1 2 3 **4** 5 Very

Evaluation Questions to Computer Expert 2

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 **3** 4 5 Agree
2. What are the best features of the system?	
3. Where should it be modified ?	
4. This content appears appropriate for using an expert system.	Not at all 1 2 **3** 4 5 Very

External Performance

5. It outputs results in a reasonable amount of time	Disagree **1** 2 3 4 5 Agree
6. It asks questions in a consistent manner.	Not at all 1 2 3 **4** 5 Very
7. The advice is clear.	Not at all 1 2 3 **4** 5 Very
8. It is easy to use as software.	Not at all 1 2 **3** 4 5 Very
9. The screen display is clear.	Not at all 1 2 **3** 4 5 Very

Knowledge Representation Aspect

10. The structure of representation is coherent.	Not at all 1 2 3 **4** 5 Very
11. The structure of representation is adequate to retrieve, store and analyze data easily.	Not at all 1 2 3 **4** 5 Very
12 The design of the inference engine is adequate for efficient inference.	did not know

Evaluation Questions to End User 1

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 3 4 **5** Agree
2. It reduces difficulties in the design task.	Not at all 1 2 3 4 **5** Significantly
3. What are the best features of the system?	
4. Where should it be modified?	
5. I would like such a tool to assist me in my test design work.	Not at all 1 2 3 4 **5** Very much

External Performance

6. It reduces design time of the administritive tasks.	Unlikely 1 2 3 **4** 5 Very likely
7. It gives me easy access to alternatives.	Not at all 1 2 3 4 **5** Significantly
8. It asks questions in a consistent manner.	Not at all 1 2 3 **4** 5 Very
9. The advice is meaningful to me.	Not at all 1 2 3 4 **5** Very
10. It improves my understanding of test design.	Not at all 1 2 3 4 **5** Significantly
11. The assistance is tailored to my needs.	Not at all 1 2 3 **4** 5 Very much
12. I like the structure of the assistance.	Not at all 1 2 3 4 **5** Very much
13. It is easy to use as software.	Not at all 1 2 **3** 4 5 Very
14. The screen display is clear.	Not at all 1 2 **3** 4 5 Very

Evaluation Questions to End User 2

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 3 **4** 5 Agree
2. It reduces difficulties in the design task.	Not at all 1 2 3 4 **5** Significantly
3. What are the best features of the system?	
4. Where should it be modified?	
5. I would like such a tool to assist me in my test design work.	Not at all 1 2 3 **4** 5 Very much

External Performance

6. It reduces design time of the administritive tasks.	Unlikely 1 2 3 4 **5** Very likely
7. It gives me easy access to alternatives.	Not at all 1 2 3 **4** 5 Significantly
8. It asks questions in a consistent manner.	Not at all 1 2 **3** 4 5 Very
9. The advice is meaningful to me.	Not at all 1 2 3 **4** 5 Very
10. It improves my understanding of test design.	Not at all 1 2 3 **4** 5 Significantly
11. The assistance is tailored to my needs.	Not at all 1 2 **3** 4 5 Very much
12. I like the structure of the assistance.	Not at all 1 2 3 **4** 5 Very much
13. It is easy to use as software.	Not at all 1 2 **3** 4 5 Very
14. The screen display is clear.	Not at all 1 2 **3** 4 5 Very

Evaluation Questions to End User 3

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 3 4 **5** Agree
2. It reduces difficulties in the design task.	Not at all 1 2 3 4 **5** Significantly
3. What are the best features of the system?	
4. Where should it be modified?	
5. I would like such a tool to assist me in my test design work.	Not at all 1 2 3 4 **5** Very much

External Performance

6. It reduces design time of the administritive tasks.	Unlikely 1 2 3 **4** 5 Very likely
7. It gives me easy access to alternatives.	Not at all 1 2 3 4 **5** Significantly
8. It asks questions in a consistent manner.	Not at all 1 2 **3** 4 5 Very
9. The advice is meaningful to me.	Not at all 1 2 3 4 **5** Very
10. It improves my understanding of test design.	Not at all 1 2 3 **4** 5 Significantly
11. The assistance is tailored to my needs.	Not at all 1 2 3 **4** 5 Very much
12. I like the structure of the assistance.	Not at all 1 2 3 **4** 5 Very much
13. It is easy to use as software.	Not at all 1 **2** 3 4 5 Very
14. The screen display is clear.	Not at all 1 **2** 3 4 5 Very

Evaluation Questions to End User 4

Problem Solving Aspect

1. The system deals with non-trivial problems.	Disagree 1 2 3 4 **5** Agree
2. It reduces difficulties in the design task.	Not at all 1 2 3 **4** 5 Significantly
3. What are the best features of the system?	
4. Where should it be modified?	
5. I would like such a tool to assist me in my test design work.	Not at all 1 2 **3** 4 5 Very much

External Performance

6. It reduces design time of the administritive tasks.	Unlikely 1 2 **3** 4 5 Very likely
7. It gives me easy access to alternatives.	Not at all 1 2 3 **4** 5 Significantly
8. It asks questions in a consistent manner.	Not at all 1 2 **3** 4 5 Very
9. The advice is meaningful to me.	Not at all 1 2 3 **4** 5 Very
10. It improves my understanding of test design.	Not at all 1 2 **3 4** 5 Significantly
11. The assistance is tailored to my needs.	Not at all 1 2 **3** 4 5 Very much
12. I like the structure of the assistance.	Not at all 1 **2 3** 4 5 Very much
13. It is easy to use as software.	Not at all **1** 2 3 4 5 Very
14. The screen display is clear.	Not at all 1 2 **3** 4 5 Very

Appendix 5

User's Manual

User's Manual

At present stage, TAA has not been wrapped up as a stand alone application, therefore, it is to be run in its programming mode. This manual explains how to run the six separated knowledge bases.

Figure 1 of this appendix is the algorithm for starting system operation.The messages in the user's mannual are detailed explainations of the tasks indicated in Figure 1, and the number of the following messages corresponding to the task number in the algorithm in Figure 1.

Figure 2 of this appendix is a list of files designated to each of the six knowledge bases. You may use the list to check whether or not all of the files of a particular knowledge base are loaded into that knowledge base before you can run the knowledge base.

In the following messages, **Bold** characters are those which should be typed in or pressed by you during an operation, and underlined ones are the Intelligent Compiler operation messages or the operation commands of Intelligent Compiler displayed in the pull dow menus.

If you want to know more about the operation of Intelligent Compliler, please consult its user's manaual in Computer Lab of Education Department , Concordia University.

1. Before run a knowledge base (application) make sure that the Intelligent Compiler version 2.1 has been properly installed on an IBM PC of = or > one megabyte memory and with a monochrome monitor. If you want to see the coding of the program, then you need a color monitor in order to tell the predicates from the other components of the program.

2.Please refer to the manual of Intelligent Compiler for the procedures on how to install the application.

3. You can enter a knowledge base by typing **ICE** (representing Intelligent Compiler Editor) at DOS level. Each of the knowledge bases is supposed to be named

uniquely and Intelligent Compiler will tell you the name of current knowledge base already loaded for operation.

A knowledge base cannot be run unless the knowledge base is loaded. and please make sure that the current knowledge base loaded is the one you want to run.

4. If the name under the current knowledge base is not the knowledge base you want, press **F10** key on the key board to go to the editing menu on the top of your screen. Then pull down Quit menu and use an appropriate arrow key to point to New knowledge base and press **return** (**<--'** or **ret**) to tell Intelligent Compiler that you want to load a knowledge base. After that, Intelligent Compiler will ask you to type in the name of the knowledge base you want to run under the title of Current Knowledge Base.

After you have tried to load a knowledge base, a message of Knowledge base does not exist indicates either a typing mistake(s) is in the knowledge base name that you just typed or no such a knowledge base has been established. If there is a typing mistake(s) in the knowledge base name, you may retype the correct one; otherwise, go to the step 4 to 'establish' a new knowledge base.

5. If a knowledge base is run for the first time on the machine you are using, you need to 'establish' that knowledge base there. To establish a knowledge you can type in any thing as the name of the new knowledge base and press **return** (**<--'** or **ret**) on your key board. Also, write down and remember that name for later operation.

6. To run a knowledge base you must have all of its files loaded, therefore, after you have set up a knowledge base, you need to check whether or not all the files of the knowledge base are loaded into the knowledge base.

Please refer to Figure 2 of this appendix to see the names of the files designated to each knowledge base and press **F2** on your key board to see which file(s) are already loaded into the knowledge base.

To load a file into the knowledge, please pull down the File menu and use an appropriate arrow key to point to Open File and press **return** (**<--'** or **ret**). Type in a

file name of the knowledge base files when Intelligent Compiler asks you the File Name. Repeat the above mentioned procedure until all of the files have been loaded into the knowledge base. You need load the files into the knowledge base only once instead of every time you run that knowledge base.

7. After you have loaded all of the files needed into a knowledge base, you should exit the Edit mode and go to the Run mode. To go to the Run mode, you press **F10** key to go to the editing menu. Then pull down the Compiler in the menu, use an appropriate arrow key to point to either Compile without saving or to Save and Compile and press **return** (<--' or **ret**). During the transaction, Intelligent Compiler will automatically load all the files into the compiling mode.

8. After the loading is completed, pull down the Compile menu on the top of the screen, use an appropriate arrow key to point to Run and press **return** (<--' or **ret**).

9. If you want to run another knowledge base, you should go back to the Edit mode to load the knowledge base into the operation. To go back to the Edit mode, use an appropriate arrow key to point to Editor and press **return** (<--' or **ret**).

10. If you want to run the knowledge base again, stay in the Run mode. Pull down the Run menu, use an appropriate arrow key to point to Rerun and press **return** (<--' or **ret**).

* To exit the operation in the Run mode, pull down the Quit menu, use an appropriate arrow key to point to Done, exit and press **return** (<--' or **ret**).

* To exit the operation in the Edit mode, you press **F10** key on the key board to go to the menu, then pull down the Quit menu, use an appropriate arrow key to point to Quit without saving and press **return** (<--' or **ret**).

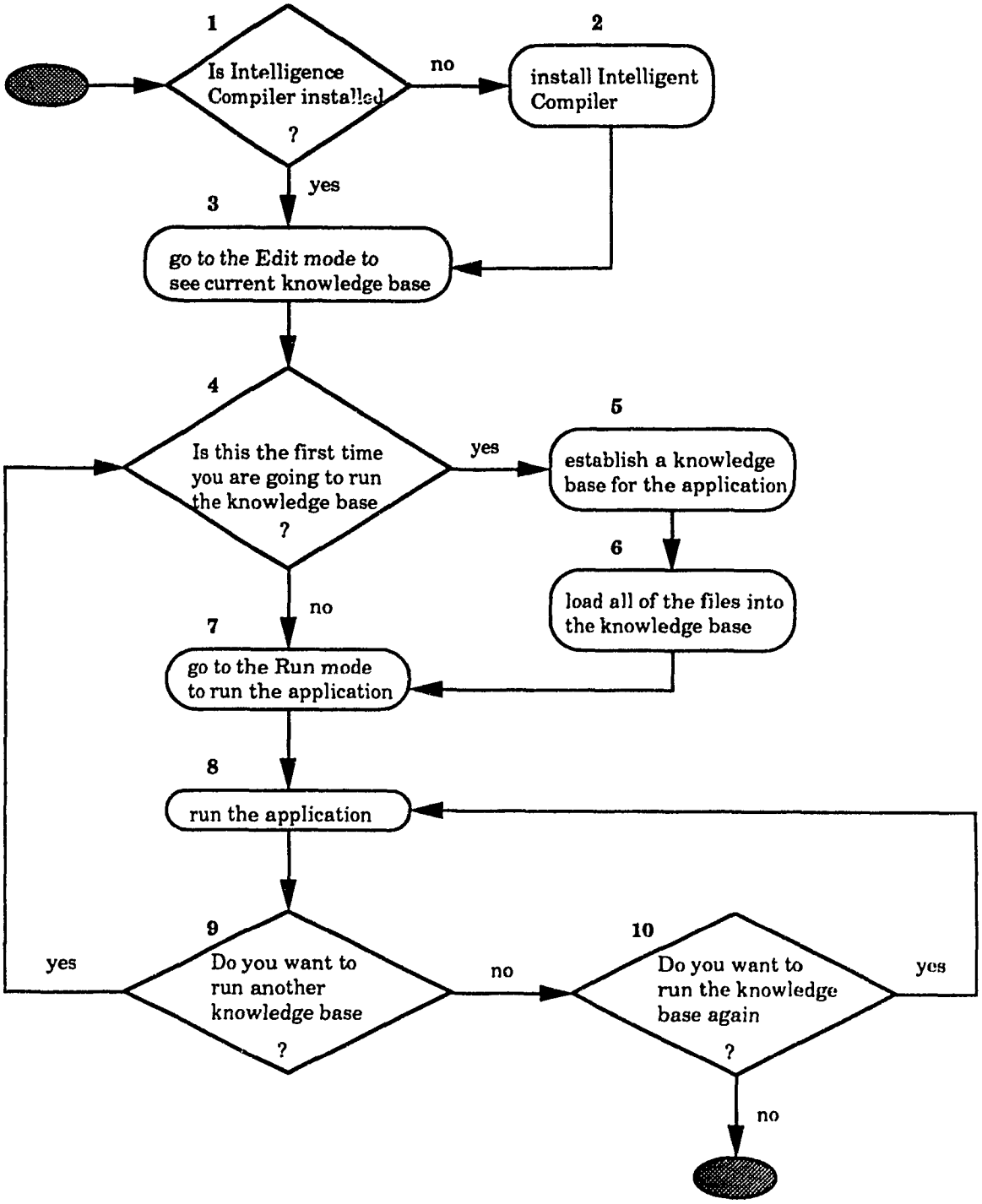


Figure 1. Algorithm for Starting System Operation

KB	TOP	BWD	BWD	FBS	FRM	LST
1	1	P	WHOLE	1	TEST1	
		P2		L	O	
					COACH	
KB	TOP	BWD	BWD	FBS	FRM	LST
2	2	OBJ	CONCEPT	2	TEST2	CHOICE
		SPECIFIC	WHOLE	L	O	SET
				SET	COACH	
				VERB		
				22		
				222		
KB	TOP	BWD	BWD	FBS	FRM	LST
3	333	FORMAT	WHOLE	3	O	SET
		TIME		L	O1	
				SET	TEST3	
					COACH	
KB	TOP	BWD	BWD	FBS	FRM	LST
4	4	LL0	WHOLE	L	COACH4	SET
				SET	TEST4	
				4	O	
				44	O3	
KB	TOP	BWD	BWD	FBS	FRM	LST
New4	4	LL	WHOLE	4	TEST4	SET
		LL1		L	O	
		LL2		44	O1	
		LL3		SET	COACH4	
		LL4				
		TOPIC				
KB	TOP	BWD	BWD	FBS	FRM	LST
5	5	A	OBJ	5	TEST5	SET
		A1	TIME	L	O	
		A2	WHOLE	SET	5	
					COACH5	
KB	TOP	BWD	BWD	FBS	FRM	LST
6	6	W	WHOLE	6	O	
		W1			O6	
		W2			COACH6	
		W3			TEST6	
		W4				
		W5				
		W6				
KB	=	knowledge base		FBS	=	fact base file
TOP	=	top file		FRM	=	frame file
BWD	=	backward chaining rule file		LST	=	list file

Figure 2. Files in Knowledge Bases

Appendix 6

Objects and Their Values

Objects and Their Values

TAA uses frames and lists to record factual knowledge for its inference and operation. Most of the objects are represented in a form of frames and other are represented in a form of list; therefor, in the following description, an object is equivalent to a list, and the values of an object in a frame are equivalent to the items in a particular list.

In the next six pages, there are six matrixes exhibiting the objects and their values used in TAA's inference. The objects and their values are displayed according to a input/out classification which does not reveal how the objects and their values interact with each other during the inference.

The six matrixes correspond to the six knowledge bases respectively. On the top of each matrixes is the description of the major task of a knowledge base. For instance, on the top of the first matrix, "1. Get Information" means the matrix describes objects and their values of the first knowledge base for getting information before developing a test. The number and title of each matrix are consistent with the figures in the appendix 1 (Task Map of Achievement Test Design Process)

Items following **User input** of the column A represent the information which will be put into TAA by the user.

Items following **Internal input** of the column A represent the information that the machine will retrieve from the fact base for its inference.

Items following **Internal output** of the column A represent the conclusions that TAA draw. The conclusions will be recorded into the fact base of the system, but they will not necessary be represented to a user upon the conclusions are formulated.

Items following **External output** of the column A represent the information or conclusions which will be represented to the user.

Bold Characters, such as "**purpose**" in the column B of "1. Get Information" sheet, is the name of an object. The items under an object are the values that the object takes.

An object may have a description in a bracket, for instance, in "**purpose (training)**", the values under it are restricted to a training environment and "training readiness, training evaluation, job assignment, and certify" are the four possible values assigned to the object "**purpose (training)**". An objective may take "yes" or "no" as its values. For instance, "**yes**" or "**no**" under the object of "**training of user**" indicates whether or not a user has had training in the achievement test design. For an object may take any value that a user puts in, such as giving a name to a test. The value of such an object will be indicated by a question mark (**?**). An object may take a value of "**not defined yet**" where the values of the object are not defined yet.

	A	B	C	D	E	F	G	H
1								
2				1. Get Information				
3								
4								
5								
6	user input	**assistance of user**	**training of user**	**unfamiliar (with TAA)**	**name of (test)**	**coverage**	**setting**	
7		yes	yes	yes	?	term course	education	
8		no	no	no		instruction unit	training	
9							research	
10								
11		**purpose (training)**	**purpose (education)**	**purpose (research)**	**intention**	**has quota?**	**need pretest?**	**frequency**
12		training readiness	learner readiness	not defined yet	knowledge	yes	yes	times (1,2,3,4)
13		training evaluation	evaluation		speed	no	no	
14		job assignment	grading					
15		certify	certify					
16			diagnose errors					
17								
18		**content area**	**content area**	**content area (research)**	**number of testee**	**education**	**test time**	**how score?**
19		science	administration	not defined yet	?	? grade	? (minitues)	machine
20		social science	supervision			(1,2,3,4 ...)		mannual
21		language	professional knowledge					both
22		arts						
23								
24	internal input	**intention**						
25		knowledge						
26		speed						
27								
28	internal output	**power test**						
29		yes						
30		no						
31								
32								
33								
34								
35				Objects & Their Values				

	A	B	C	D	E
1					
2			2. Formulate Objectives		
3					
4					
5					
6					
7	user input	**general objective**	**specific objective**	**verb (choose from a verb list)**	**performance level**
8		?	?	?	knowledge
9					comprehension
10					analysis
11					application
12					evaluation
13					synthesis
14					
15	internal output	number of objectives	row number of matrix		
16		?	?		
17					
18					
19					
20					
21					
22					
23					
24			Objects & Their Values		

	A	B	C	D	E	F	G	H	I	J
1										
2						3 Choose Test Format				
3										
4										
5	user input	**essay**	**write essay**			**has sufficient number of**				
6		**(to be included)**	**(testees can)**			**homogenous premises & responses**				
7		yes	yes			**(for a possible matching item)**				
8		no	no			yes				
9						no				
10										
11	internal input	**how score?**	**no. of testee**	**grade**	**performance level**	**general objective**	**specific**	**verb**	**verb**	**length**
12		machine	?	? 1, 2, 3, 4,	knowledge	justify methods/proceures	(objective)	select	compose	**(of an essay)**
13		mannual			comprehension	interpret cause-effect relationships	?	discover	write	? page (1,2,3,)
14					analysis	use the methods in a new situation		identify	plan	
15					application	identify relationship between two things		match	design	
16					evaluation	interpret diagrams/graphs		relate	produce	
17					synthesis	translate verbal material to mathematical formulas		explain	organize	
18						solve mathematical problems		categorize	express	
19								describe	integrate	
20								present	create	
21								formulate	evaluate	
22								state	compute	
23										
24	internal output	**resource of test**	**essay**	**education**	**item format**	**test time of each test item**				
25		**(to use essay)**	**(to be included**	**(for essay)**	extended response	min.(1,2,3,..)				
26		yes	yes	yes	restricted response					
27		no	no	no	multiple choice					
28					short answer					
29					matching					
30					true-false					
31										
32	output to user	*	*		*	*				
33										
34										
35										
36										
37										
38										
39										
40						Objects and Their Values				

	A	B	C	D	E	F
1						
2				4. Construct Test Specification		
3						
4						
5	user input	**subtotal**	**subtotal**			
6		(of topics)	(of levels of			
7			performance)			
8		?	?			
9						
10		100	100			
11						
12						
13	internal input	**purpose**	**purpose**	**coverage**	**has quota?**	**performance level**
14		training readiness	learner readiness	term course	yes	knowledge
15		training evaluation	evaluation	instruction unit	no	comprehension
16		job assignment	grading			analysis
17		certify	certify			application
18			diagnose errors			evaluation
19						synthesis
20						
21						
22	internal output	**type of test**	**cell values**	**coverage**	**difficult levle**	**focus**
23		placement	**(score values at each topic wit**	**(of a test)**	**(of a test)**	**(of a test)**
24		formative	**each level of performance)**	?	?	?
25		summative	?			
26		diagnostic				
27						
28	output to user	•	•	•	•	•
29						
30						
31						
32						
33						
34						
35						
36						
37						
38						
39						
40				Objects and Their Values		

			5 Decide Test Length							
user input	**change the specification**	**delete obj.**	**add obj.**							
	yes	yes	yes							
	no									
internal input	**general objectives**	**total**	**coverage**	**pretest?**	**time of test**	**testing time**	**score value**	**item no.**	**performance lev**	**obj. mark**
	translate verbal material to	(objectives)	term course	yes	min (30. .)	(an obj.)	per minitue	(each objective	knowledge	1,2,3,4...
	mathematical formulas;	?	instruction unit	no		min. (1, 2, 3,	(1, 2, 3, 4, ...)		comprehension	
									analysis	
	solve mathematical problems;								application	
	know vocabulary;								evaluation	
									synthesis	
internal output	**item (no.) of obj.**	**items of tes**	**Min. score value**	**item mark**	**obj. mark**	**tables**	**cell differenc**			
	1, 2, 3, 4, ...	1, 2, 3, 4, ...	1, 2, 3, 4, ...	1,2,3,4. .	1,2,3,4...	specification	1,2,3,4...			
						draft				
						difference				
output to user	**objectives (tested in the test**	*		*	*	*	*			
			Objects and Their values							

	A	B	C	D	E	F	G	H	I
1									
2				6. Write Test Items					
3									
4									
5									
6	user input	**test items**							
7		?							
8									
9	internal input	**objectives**	**objectives**	**item total of the test**					
10		page of essay	topic	1, 2, 3, 4...					
11		verb used	general						
12			performance level						
13			format						
14			item time						
15			number of items						
16			item mark						
17									
18									
19	output to user	**test items by format**							
20									
21									
22									
23									
24									
25									
26									
27									
28									
29									
30									
31									
32									
33									
34									
35									
36									
37									
38									
39									
40				Objects and Their Values					
41									