1993

An approach to the analysis and deisgn of an intelligent tutoring system using an object-oriented methodology

Jeanne C. Hiesel

Rochester Institute of Technology
School of Computer Science and Technology

An Approach to The Analysis and Design of An Intelligent Tutoring System Using An Object-Oriented Methodology

by
Jeanne C. Hiesel

A thesis, submitted to
The Faculty of the School of Computer Science and Technology
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science.

Approved by:

_____
Dr. Walter A. Wolf

_____
Dr. Kevin Donaghy

_____
Dr. J. E. Heliotis

Acknowledgments

My thanks and gratitude go to Dr. Wolf for his patience and guidance, to Dr. Donaghy and Dr. Heliotis for accepting my request to be part of my thesis committee, to Dr. Anderson for his understanding and to my friends for their support during the creation of this paper. Some of whom are: Mike Swank, Alan Dray, Greg Schwartz, Phil White, Roberta Baidy, David Bogdan and Amy Dawes.

Abstract

A true Intelligent Tutoring System is difficult to produce in today's technological environment. This thesis reviews various theoretical methods and strategies that could be employed in performing the analysis and design of an Intelligent Tutoring System. An overview of the basic concepts of Object-Oriented Analysis and Design are provided in this thesis. The notation system provided by these concepts are utilized. The Object-Oriented Analysis and Design methods that are employed create a basis for an implementation of an Intelligent Tutoring System.

Key Words and Phrases

Intelligent Tutoring Systems (ITS), Object-Oriented Analysis (OOA), Object-Oriented Design (OOD), Interface Module, Student Module, Instructor Module, Expert Module.

Computing Review Subject Codes

ACM Computing Review Codes

D.2 Software Engineering
D.2.1 Requirements/Specifications
D.2.10 Design

I.2 Artificial Intelligence
I.2.1 Applications and Expert Systems

K.3 Computers and Education
K.3.1 Computer Uses in Education

K.6 Management of Computer and Information Systems
K.6.1 Project and People Management

Inspec (IEEE/IEE) Computer and Control Abstracts Codes

1230 Artificial Intelligence
6110 Systems Analysis and Programming
6170 Expert Systems
6180G Graphical User Interfaces
7110 Education

Table of Contents

1. Introduction and Background

1.1 Problem Statement

The objective of this thesis is to discuss the evolution of Intelligent Tutoring Systems (ITS) and to perform the analysis and design of an ITS. The central aim of the analysis and design is to create an ideal or generic ITS. The inherent difficulty with this problem is the wide variety of methods that are used in education. An object-oriented methodology will be used to perform the analysis and design.

An interest was developed in tutoring applications while an undergraduate student at RIT in the early 1980's. At that time I knew nothing about computer science, artificial intelligence or intelligent tutoring systems. When I began my graduate studies in 1986 I had chosen to do a project. The intent was to create a portable medical record for anyone. This has now been done and is commercially available. During my search for another topic I became exposed to AI and in turn to ITSs. My original idea was to create a generic ITS. Research showed that it was highly unlikely that I would be able to accomplish that within the scope of a Masters thesis. I was told and now agree that it would be an appropriate endeavor for a Doctoral thesis. This made it necessary to limit my idea to a specific knowledge domain. Leukemia was selected for the knowledge domain for several reasons: my undergraduate degree is in medical technology; I have access to hematology experts because of my medical background and it is a very limited knowledge domain within the larger domain of hematology. In addition, I would have liked to have had a tutor available for use when studying hematology slides in my undergraduate days and I think that this tutor could eventually benefit both the School of Computer Science and Information Technology and my alma mater the College of Science. After further research the decision was made to perform the analysis and design without a specific knowledge domain. This will allow the creation of documents that other students can use to implement an ITS in any knowledge domain that they choose.

ITSs are a form of computer software that attempts to tutor an individual in an intelligent manner. They are an application of artificial intelligence (AI) in the field of education and involve steps beyond the area of traditional computer-assisted instruction. A more formal definition says that an ITS is a computer program capable of competent problem solving within a knowledge domain where the program can infer the student's approximation of competence and is able to reduce the difference between its own level of competence and the students level through the application of various tutoring strategies. The essential purpose of an ITS is closely tied to the area of cognitive science. An ITS must try to infer how the student organizes and processes knowledge. A cognitive model is a theoretical representation of that inference. (17, pp. 1,263)

There are four major components or modules that comprise an ITS: the Expert or domain knowledge module which contains the knowledge that the ITS attempts to teach; the Student/Learner module which is a diagnostic module that makes inferences about the user's state of knowledge; the Instructor/Tutor module or tutorial planning module that identifies deficiencies in the student's knowledge and selects strategies to present that knowledge through instructional interactions with the student; and the Interface/Communication module which forms the instructional environment that controls the interactions between the system and the student for the purpose of channeling tutorial communications. (14, pp. vi; 17, pp. 2)

The Expert module's knowledge can be represented by any of the traditional AI methods, such as frames, semantic nets and rule-based systems. The module should include not only surface knowledge but also "the ability to construct implicit representational understanding from explicit observations and other information". (14, pp. vi, vii)

Expert knowledge can be represented in several forms within an ITS. There is the 'black box' form, where the knowledge remains opaque to the student. A system

that employs this form can answer questions about the knowledge but is unable to explain to the student how the answer was derived. Most early ITS's utilized this form. Another form is called the 'glass box', where the knowledge is represented in a manner that more closely matches that of human capabilities. This kind of system creates better possibilities for explanations of how answers to questions were derived. (14, pp. vi, vii; 17, pp. 2-5)

The Student module represents the changing knowledge and growing skill of the student. It needs to be capable of diagnostic processing so that it can deduce what the student knows from their interaction with the system as the student deals with the tasks that are being posed by the system. There are several approaches that can be used to provide this capability, including the overlay model, deviation modeling, the mental model and graduated models.

The overlay model represents the student's knowledge as a subset of the experts knowledge. Deviation modeling is an approach that measures the students knowledge as a deviation from the experts knowledge. The mental model approach views the expert knowledge as the world and the student's knowledge as deviations from that world. The graduated model is an approach that uses a qualitative process which grows in correspondence with the students capabilities. (14, pp. vii; 17, pp. 5-7)

The Instructor/Tutor module, also known as the tutorial planning component, is closely linked to the Student module because it uses the knowledge about the student in conjunction with its own goal strategy to determine which instructional activities will be utilized. Instructional activities include hints, advice, support, explanation, new material, practice tasks and tests. There are several tutorial strategies, which will be described in following sections. An important distinction among them is in the learning method. Learning methods are generally either didactic or discovery-oriented.

Didactic learning tells the student what will be learned. This kind of system initiates and controls the activity of the system. The activity focus is directed toward the

instructional goals of the system. The advantage of didactic learning is its goal oriented approach to learning. The disadvantage is the lack of flexibility within the curriculum being used.

Discovery learning lets new knowledge be presented in terms that the student, rather than the system, specifies. Student employed terms are in the form of those concepts and capabilities that the student already knows. As the student degree of freedom within a curriculum is increased the difficulty of modeling the students knowledge also increases. (14, pp. viii; 17, pp. 7-11)

The Interface/Communication module controls the interaction between the student and the system. This module is usually graphical in nature. Graphical interfaces provide a more user-friendly interactive environment by substituting pictures and pointing for text and typing. Graphical interfaces also provide greater concreteness to the information that is presented to the student. Concreteness is meant to convey that the information is more dense when presented in a pictorial form. A picture is worth a thousand words. (14, pp. viii; 17, pp. 11-13)

1.1.1 Project Scope

A top level analysis of an ITS describes the interaction between the modules. The student or user interacts with the Interface/Communcation module. The interaction can take the form of verbal, mouse or keyboard input from the user in response to a comment, question or description displayed by the Interface module. The Interface module displays the instructions that are determined within the Instructor module. The Instructor module makes its decisions based upon the information resident in the Student module and the teaching algorithms within itself. The knowledge that the Instructor module seeks to impart is resident in the knowledge base. As the system is used, the Student module grows in order for the Instructor module to make intelligent decisions about what the user has learned.

Figure 1 is an early representation of the object model for representation of an Intelligent Tutoring System. There will be other examples of early analysis work to show starting and ending points for the analysis process. This diagram is an illustration of how the data and control flow of an ITS functions. It is not an OOA diagram but is the first formulation of analyzing the problem of an ITS.

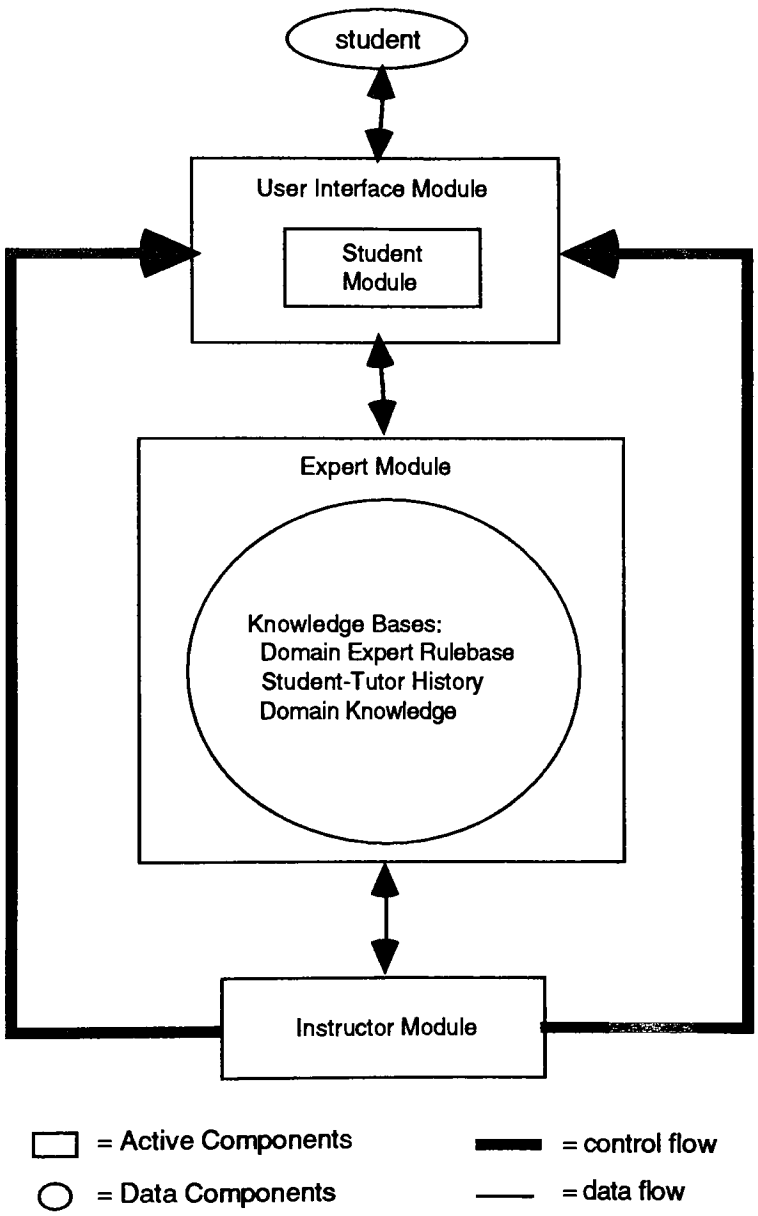


Figure 1: Rough Draft of Object Model

These modules represent the top level objects or the classes of the system. It is assumed that the Interface module would be graphical in nature. The other modules are invisible or transparent to the student. The knowledge base is some form of data and/or rule-base that the instructor would be accessing for instructional material. The Student module is both a database and some algorithmic logic that interacts with the instructor. The Instructor module is comprised of algorithms that incorporate a variety of teaching methodologies.

1.2 Previous Work

The driving force behind my research into ITS's was to acquire information on the theoretical background and the development path of Intelligent Tutoring Systems. Purely theoretical and/or informational papers were difficult to locate. Most articles pertained to specific applications of different ITS theories and methods.

The problem statement provided a fairly short explanation of the four main components that make up an ITS. Research and development of ITS's focuses on the improvement of those modules to make an ITS more like interaction with a human than a machine. The most effective teaching method known is considered to be one-on-one tutoring with a human. The objective of an ITS is to duplicate as closely as possible that type of effectiveness.

Currently the least researched module in an ITS is the Expert module. This is because expert systems are employed in other areas of artificial intelligence research and applications development. Most of the current research focuses on the Student module, the Instructor module and the interaction between them. This interaction seems to be the most difficult to implement with current technology. An ITS will display more of the strengths and features of a good teacher as more is learned about how to create effective instruction. The Interface/Communication module has recently

received a great deal of attention. The effective use of interfaces as related to their appearance and structure are the main points of discussion.

Computer assisted instruction has been available for about twenty years and significant advances have been made in "the intersection of human cognitive processes and intelligent computer-assisted instruction (ICAI), partly as a result of the convergence of investigations of those areas in the emerging paradigm of cognitive science". (7, pp. 169)

1.2.1 Expert Module

Within the Expert module there has been much progress made in knowledge representation methods "...in problem-solving domains comprising essentially formal systems, such as geometry, algebra, electricity, physics, theorem proving and computer languages...". (7, pp. 171) This does not hold true for the progress made with such non-formal domains as those involving conceptual knowledge, such as biological and medical sciences and those involving both conceptual and perceptual knowledge, such as computer-based systems supporting decision making. (7, pp. 172) The Expert module is considered to be the backbone of an ITS. The instructional system cannot exist without the domain knowledge.

In many ITSs the Expert module is incomplete because they provide only part of the information needed in a knowledge domain. An example of this is an ITS called Steamer which is used to train engineers about steam propulsion plants. The system "...knows a great deal about the mathematical properties of steam but rather little about how to operate a steam plant...". (17, pp. 21) This is because a large amount of effort must be expended to uncover and codify all areas of the domain knowledge. Most complex domains contain a very large amount of knowledge and this creates a very labor-intensive environment when developing an Expert module.

Encoding the knowledge into the Expert module encompasses some basic

options. One option is to try to find a way of reasoning that does not require coding of the knowledge. An example of this is an early ITS called SOPHIE (developed by Brown, Burton and deKleer in 1982) (23, pp. 227-279), which uses the SPICE simulator for electronic circuits. SOPHIE performs its calculations using mathematical equation-solving that techniques to produce what humans do through symbolic processes. (17, pp. 22) There is no knowledge of electronic currents within the system but it reasons about them through the use of a mathematical model. This is an example of the 'black box' type of model that could be used for the Expert module.

Another option is to implement a standard expert system in which the knowledge is extracted from an expert and that data is coded and applied. An example of this form is GUIDON by Clancey (1982) (17, pp. 31), which is based on MYCIN. The drivers behind the basic instruction of the system are t-rules. T-rules are an extension to Burton and Brown's issue-oriented recognizers (23, pp. 79-98) and are defined both on the difference between the expert's behavior and the student's behavior and on the expert's reasoning process. A disadvantage of GUIDON is the mismatch between the control structure of MYCIN and the control structure which a human uses. This makes it difficult to explain what will be done next. That disadvantage along with the difficulty of justifying the highly compiled rules of reason that MYCIN uses, and the complexity of trying to teach those rules to novices, led to the design of NEOMYCIN, which used a different control structure on the domain knowledge. The lesson learned from GUIDON is that it is necessary to pay attention not only to the knowledge itself but also to how it is used. The Expert module should use its knowledge with the same restrictions as a human in order to tutor effectively. (17, pp. 29-32)

Another option is to make the Expert module into a simulation. The simulation should model the way that an expert uses the knowledge at some level of abstraction. A simulation is the most difficult option to implement because technology has not yet

been able to find a way to accurately model human thought processes. It also seems the best option to produce a truly effective ITS. Simulation is also known as a cognitive model.

Cognitive systems use different types of knowledge: procedural, declarative and causal. Procedural knowledge is knowledge of how a task is performed. Declarative knowledge is facts and more general knowledge that is not specialized for a particular use. Causal knowledge uses qualitative models to reason about the behavior of a device.

An example of modeling procedural knowledge in an ITS is LISP Tutor (developed by Reiser, Anderson, and Farrell in 1985) (17, pp. 38). It uses a rule-based approach to tutoring. A rule-based approach makes possible the use of a tutoring method known as model tracing. Model tracing tries to place the student's "...surface behavior in solving a problem in correspondence with a sequence of productions that are firing in the internal..." student model. (17, pp. 38) The correspondence is then used to create an interpretation about the students behavior. Procedural knowledge representations are use-specific.

Declarative knowledge modeling is generally used to establish an understanding of the knowledge domain. The student should be able to reason with the basic principles and facts of a knowledge domain. No significant level of expertise is obtained with any one application within the domain. Understanding natural language is the weak point of any effort in declarative tutoring. It is necessary to use mixed initiative dialogues when tutoring with this method. This requires the Instructor module to ask a series of questions to which the student responds. There are very few ITSs that use the declarative knowledge model because of the natural language problem encountered in developing them. (17, pp. 40-45)

Dijkstra defines declarative knowledge as being categorized into conceptual and causal knowledge. Conceptual knowledge relates to facts and to class and

relational concepts. (5, pp. 20) Causal knowledge reflects changes in objects or situations that are described in conditional and biconditional statements. (5, pp. 23)

Causal or qualitative process models are useful in their ability to employ trouble-shooting or to use the causal structure of a device to reason about where potential trouble spots in learning may be. Qualitative models can also generate clear and effective simulations of a particular system. A simulation can illustrate the qualitative transformations assumed in the qualitative simulation. "How to include qualitative simulations in a tutoring paradigm has yet to be worked out." (17, pp. 47)

1.2.2 User/Student/Learner Module

The Student module creates an inference of the student's current understanding of the subject matter and uses this individualized model to adapt the instruction to the students needs. The inference process of the student model is called diagnosis. The Student module is where an ITSs diagnostic system uncovers a hidden cognitive state (the student's knowledge of the subject matter) from observable behavior. The power of student models has been limited by the characterization of a students problem-solving knowledge being based solely on observations of overt actions. It is not at odds with the purpose of an ITS to ask the student for assistance in order to attempt to determine the intentions and reasoning of the student. "Inviting students to rationalize their actions and to identify their hypotheses and goals can enormously enrich a program's knowledge of the student." (18, pp. 448,449) Current research indicates that complex learner models may not be available in the near future "because of the lack of knowledge on deep cognitive processes". (6, pp. 64) ITSs need to have knowledge about the possible environment of the student to be of use during the learning process.

The Student module contains a history of what the student has done in previous sessions with the tutor. It records information in terms of what goals were

accomplished and what plans and/or functions were used in obtaining the goal. Goals may have plans that were attempted but did not work, or plans that were attempted which do not exist in the Expert module. An assumption is made that either the student knows or does not know about information within the domain. "By definition, if something exists in the User Model, even if it is wrong, then it is known. If it does not exist, then it is not known." (22, pp. 76) In actual practice this assumption is found to be inadequate because "knowledge is not binary" and "tutoring can occur across a broader continuum." (22, pp. 76) The evaluation and recording of the student knowledge in a less quantified manner would be of greater benefit than the use of simple categorical levels.

Learner control research has provided some insights into different approaches to the problem as to whether an ITS is more effective when control is in the hands of the system or the user. "The use of learner and/or program control can also be described as the locus of instructional control, with the control of instruction being either external (program control) or internal (learner control)." (15, pp. 99) Learner control most often refers to choices made by a student when working on a single lesson. It also refers to the student being able to make choices at the curriculum level, to study a lesson for as long as desired and to be able to select and sequence a variety of processing strategies.

Milheim (15) discusses three theories of learner control: learner control of pacing, leamer control of sequence and leamer control of content.

"Learner control of pacing typically allows students to control the speed of presentation of the instructional materials to a degree that most suits their learning needs." (15, pp. 102) Learner control of pacing is based on motivation theory and can be provided in most situations. The advantage of this method is that the student may derive greater satisfaction from the learning process if allowed to spend more time on those topics that relate to their personal needs and goals .

"Learner control of sequence typically allows students to choose the order of the content presented to them, although all information within the lesson must eventually be studied." (15, pp. 103) There are two levels of control associated with sequence control, either macro or micro. At the macro level sequence control is only in terms of choosing complete chapters and at the micro level control refers to being able to sequence specific sections within a chapter. In those situations where the materials have a specific prerequisite order, sequence control should not be provided.

"Learner control of content allows a student to choose only those materials that he or she wishes to study within a particular lesson." (15, pp. 103) The student selects only material within a lesson that the student wants to study or to avoid those sections that the student knows or has no interest in. There are two situations when this type of control should not be allowed: when all topics are required and when there is a hierarchical order that should be followed.

1.2.3 Instructor/Tutor Module

The Instructor/Tutorial planning module should have certain specific characteristics. Control should be maintained over the representation of the instructional knowledge to be presented. This is for the purpose of selection and sequencing of the subject matter. The Instructor module should also be capable of responding to the student's questions, about the content and the goals of instruction. Strategies should be provided by the instructor to determine when a student needs help and to deliver that help appropriately.

Two learning or instructional modes are learning-by-example and learning-by-doing. Learning-by-example is also known as didactic learning and learning-by-doing is known as discovery-oriented learning. With learning-by-example, "the system displays the best search tree it can generate, given the level of expertise chosen by the student". (6, pp. 65) The tree is a set of examples. When using this mode the

student is able to query the system about its action and receive an explanation at either a factual or strategic level. The primary advantage of didactic learning is its goal-oriented approach. In the learning-by-doing mode, the student develops their own search tree by selecting at every step an expression and a transformation rule to apply. When learning-by-doing is used, the student asks for help and receives an explanation of the actions that the system would do in the situation. The primary advantage of discovery-oriented learning is that new knowledge is constructed in the terms of the concepts and capabilities the student already possesses. (6, pp. 65; 14, pp. viii)

At any point during instruction a tutoring system must select from many possible instructional actions. The problem of control is to select the most effective action given the current lesson objectives, student model, tutorial strategy, subject matter and resources available. An important decision in designing an ITS is the choice of architecture for this control mechanism. Examples of control mechanisms are discourse management networks and blackboard architectures. (16, pp. 107)

A tutoring system should have a plan and be able to modify that plan. There should be a plan that will properly manage its time and generate a coherent instruction. "Proper time management prevents spending too much time on relatively unimportant topics or packing too many topics or exercises into one lesson." (16, pp. 108) Coherence connects topics in a logical manner that is sequenced and presented in a manner suitable to the students level of knowledge, interest and motivation, the time available and the tutors objectives.

Three levels of instructional planning are curriculum planning (planning an extended sequence of lessons for a subject), lesson planning (determining the subject matter to present in a single lesson and the order of presentation) and discourse planning (planning communicative actions between the tutor and student within a lesson). (16, pp. 110) In practice, these instructional levels are not clearly separated.

Problems with implementation of planning are characterized by the environment being worked with. Some of those problems are incomplete knowledge (the tutor does not know the student's knowledge complexity at any point), uncertainty (what the tutor does know, it does not know with complete certainty), dynamic knowledge (the student's knowledge changes during and between sessions), multi-agent (the tutor and student cooperate to facilitate the student's learning) and the results of actions are uncertain (the tutor cannot predict with certainty the results of its actions). (16, pp. 123,124)

Early in the development of ITSs, researchers worked without established standards with which to build instructional strategies into workable systems. As research progressed some paradigms emerged such as: mixed-initiative dialogues; coaching; diagnostic tutors; microworlds and articulate expert systems.

Mixed-initiative dialogues were one of the earliest of these paradigms. It provides an environment in which the tutor and student engage in a two-way conversation "...utilizing the Socratic method of guided discovery, whereby the computer provides the student with questions that guide him or her through the processes of debugging their own misconceptions...". (19, pp. 31) A tutor that is able to plan is more able to deal with a mixed-initiative dialogue than one that is not able to plan.

Coaching uses the method of observing the student's performance and offering advice when and if the student asks for it. The purpose of the advice is to increase the student's understanding of the topic. This method allows more freedom for the student than the mixed initiative dialogues where the system responds to each initiative of the student.

Diagnostic tutors attempt to identify any misconceptions that a student may have and counter them with appropriate feedback. The appropriate feedback 'debugs' a student's work. The misconceptions are labeled as 'bugs'. A bug catalog is

maintained that holds these misconceptions. The catalog is used in later learning sessions to identify previous or existing misconceptions. Diagnostic tutors utilize the bug library technique.

Microworlds use a computational tool that allows a student to explore a given domain free of teaching constraints, thus learning through their own creativity. This allows the student to learn at a maximum of freedom of choice. The principle of microworlds stems from the assumption that students learn by their own choice.

Articulate expert systems use knowledge representation schemes that are able to explain as well as tutor their own reasoning. It is the explanation of the systems instructional reasoning that makes it articulate.

The preceding methods are not exclusive of one another. Most ITSs combine elements from several of them. In addition, the transition between topics should be smooth and not contradictory with either previous or future instruction. (19, pp. 31,32)

1.2.4 Interface/Communication Module

Before going on to the Interface/Communication module it is necessary to provide some information on the topic of interactivity. In order to instruct a student there must be some type of interaction between the student and the tutor.

Factors that affect interactivity are: immediacy of response, non-sequential access of information, adaptability (information upon which adaptations are made), feedback, options, bi-directional communication and grain-size (the length of time required of a given sequence before allowing further input). (3, pp. 12,13) Each one of these factors has an impact on the effectiveness of an ITS.

The Interface module attempts to make the human-computer interface transparent or 'user-friendly'. The interaction that occurs within the Interface module is considered to be a communications problem.

There are some basic styles of design. First-person interfaces allow the student

to become a direct participant in the knowledge domain. An example of this type is an Apple Macintosh icon. Another design type allows the student to control the knowledge domain by instructing the system to carry out any requested actions by whatever means that have been implemented for use by the developer. Some options that may be used are: icons; keyboard input; mouse input or menu selection.

In addition to design styles there are several attributes of screen design which affect the legibility and meaningfulness of a screen such as: capital versus lower case letters, interlinear space, the screen page as a unit, strategies for recall and location of information.

In the debate of capital versus lower case, the argument is made that lower case letters provide better visual cues than upper case letters. This is due to lower case letters having ascenders and descenders which create a specific shape. Since capitals do not have them, words become more of a solid block. This reduces the readability of a word unless presented by itself. (1, pp. 54)

The amount of interlinear space used is designed to promote readability. The ascenders and descenders of lower case letters are more prominent when there is extra space between lines. Too much space between lines can interfere with the concept of continuation. The amount of interlinear space needs to be consistent in throughout the screen in order to avoid continuation problems. The interlinear space can also be used to create some kind of hierarchical order within the screen by grouping information in relation to its content. The information could also be divided according to ideas, separating major points from minor ones. (1, pp. 54)

The attribute of the screen page as a unit, refers to where on a screen page, each line should have a brief, self-contained message. The length of a line affects readability and should contain a brief, self-contained message. The student needs to be provided with adequate prompts and retrieval cues. This is a crucial aspect since the use of stored information relies on these strategies. (1, pp. 55)

Strategies for recall refers to ragged versus right justified margins, chunks of information and the location of information in relation to graphics. The physical location of information should be consistent throughout the system. (1, pp. 55,56) A major point to the interface is that the student can "...benefit from a layout which, in addition to organizing content, provides visual cues from locations on the screen...". (2, pp. 91) The greatest benefit from efficient screen design is most evident during the student's process of retrieval.

1.2.5 Instructional Design Tools

Another area under research for Intelligent Tutoring Systems is the use of an instructional design tool that helps to develop an instructional product. Instructional design is considered to be a bottleneck in the area of developing instructional materials. Due to the lack of a systematic approach for instructional design, instructional products have tended to be expensive and of a lower quality than desired.

An example of this is the 'ID Expert 2.0'. One of the main purposes for creating the tool was to improve instructional design theory. "The process involves the transformation of broad general instructional design and learning principles into specific knowledge frames and production rules sufficient to enable the computer to reason with those rules and recommend specifications for the development of a particular instructional product." (12, pp. 60) Unfortunately, this tool is only a prototype and is not available for general use.

Another example of an instructional design tool is the Research Reference Interface (RRI). RRI is also considered to be a prototype that "...defines a structure for generating explanations and justifications based on research literature that is universal to any knowledge-based instructional design system...". (11, pp. 16) This tool was created to generalize with rule-based systems and is not able to adequately

support diagnostic reasoning.

1.2.6 ITS Evaluation Standards

Evaluation of the effectiveness of an ITS should be a primary concern of anyone creating an ITS. If it is not effective then it is not doing what it was developed for.

Some evaluation criteria has been developed. One is to compare the performance of a classroom control, a human tutorial and an ITS. Another is to assess the impact of an extensive ITS application on performance. The system can be evaluated against O'Shea's principles of ITS design, developed in 1984, of which there are thirteen. The principles are: robustness, helpfulness, simplicity, perspicuity, power, navigability, consistency, transparency, flexibility, redundancy, sensitivity, omniscience and docility. (25, pp. 129,130) Lastly, large groups of users should be used when performing any evaluation. (10, pp. 58,60,61)

2. Analysis

The analysis of a problem includes understanding the problem domain and the system's responsibilities within the problem domain. Analysis is the process of understanding and modeling the problem domain. This process studies a problem domain, leading to a specification of externally observable behavior. The specification is a complete, consistent and feasible statement of what is needed by the system. Both functional and quantified operational characteristics are covered. (31, p 8) The design of the problem adds the details that are necessary for a particular implementation. The implementation fulfills those requirements that were delineated during the analysis phase. Using object-oriented methodology the object-oriented analysis (OOA) "...layers model the problem domain and the system's responsibilities..." and the object-oriented design (OOD) "...expansion of the OOA layers model a particular implementation...". (31, p 178) The expansion occurs within the components developed during analysis.

Analysis usually begins with a requirements document. This document is not required for this thesis due to the lack of a customer. It can be argued that a form of the requirements document could have been developed for this paper. In lieu of this document the basic requirements needed to develop an ITS will be utilized.

The draft object model for the ITS (Fig. 1 page 5) will be refined and expanded during the formal analysis process.

Figure 2 is the first attempt at analyzing the internal processing of the Instructor module. This diagram is not part of OOA. It is the next step that was undergone in analyzing the Instructor module of an ITS before beginning the formal analysis process. It is included in this paper to show progress in the analysis process.

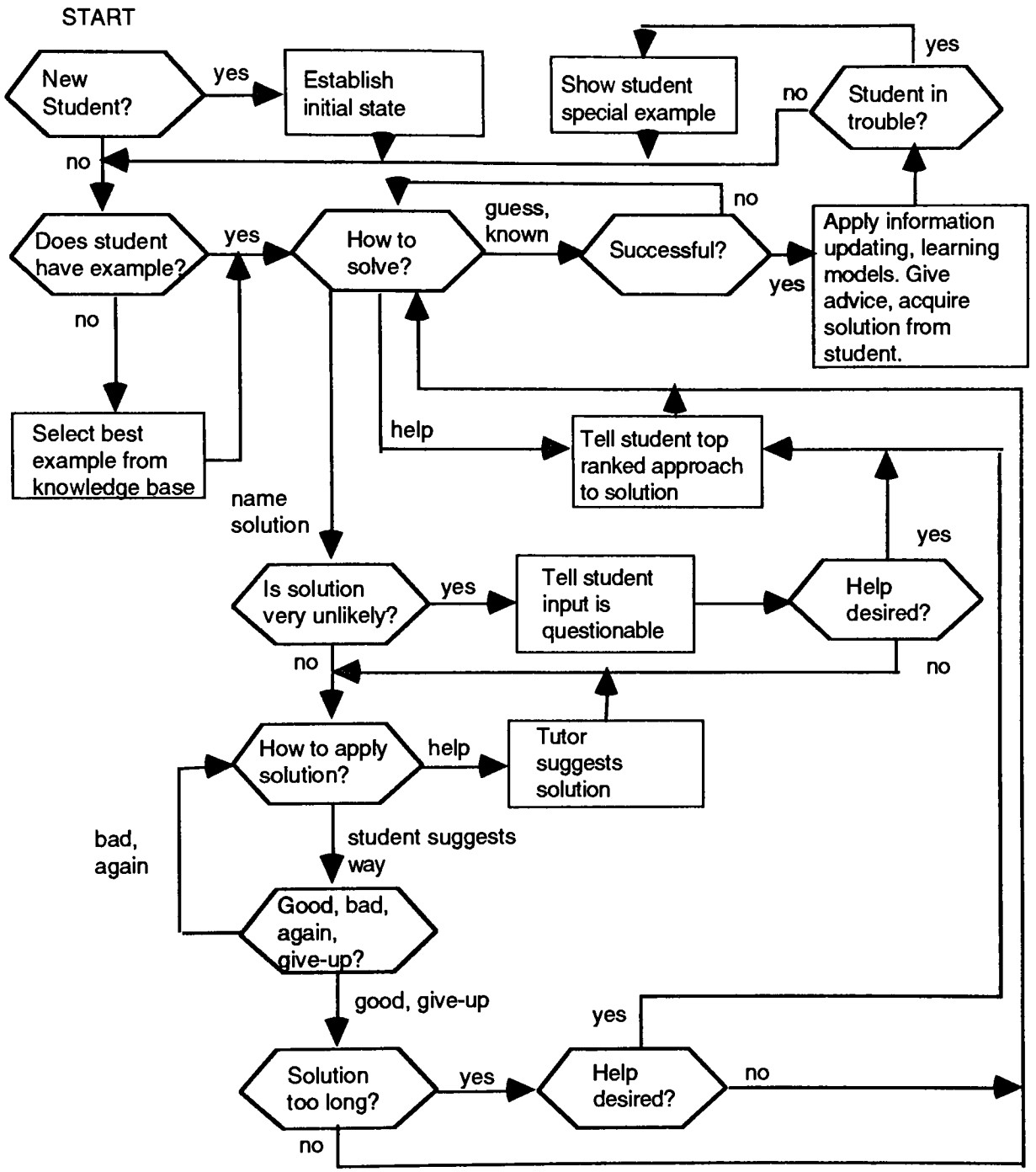


Figure 2: Elementary Flow Diagram of the Instructor

There are several approaches that can be used in analysis, such as functional decomposition, data flow or structured analysis, information modeling and object-oriented. The best approach is a combination that will produce the most complete

analysis of the problem.

Functional decomposition focuses on the processing that is required for the system. The specifications that are created by functional decomposition are indirectly mapped to the subject matter.

Data flow is also known as structured analysis and the focus is on mapping the problem into data flows and bubbles (events that occur within the data flow). The flow of data is mapped into the specifications. The structured analysis approach is weak in the data structure area.

Information modeling focuses on the creation of the entity-relationship diagram. Problem domain contents are captured by modeling it in data. 'Object' is equivalent to 'entity' in this approach. This approach is considered to be a partial method. It does not include the systems behavior or the interface to depict process interdependency. (31, p20-28)

Object-oriented analysis uses concepts from information modeling, object-oriented programming languages and knowledge-based systems. Figure 3 is a pictorial representation of the merging of these disciplines. Information modeling provide constructs analogous to Attributes, Instance Connections, Generalization-Specialization and Whole-Part. Object-Oriented Programming Languages and Knowledge-Based Systems provide the encapsulation of Attributes and exclusive Services, communication with messages, Generalization-Specialization and Inheritance. (31, p 31) "OOA directly maps problem domain and system responsibility directly into a model." (31, pp. 32)

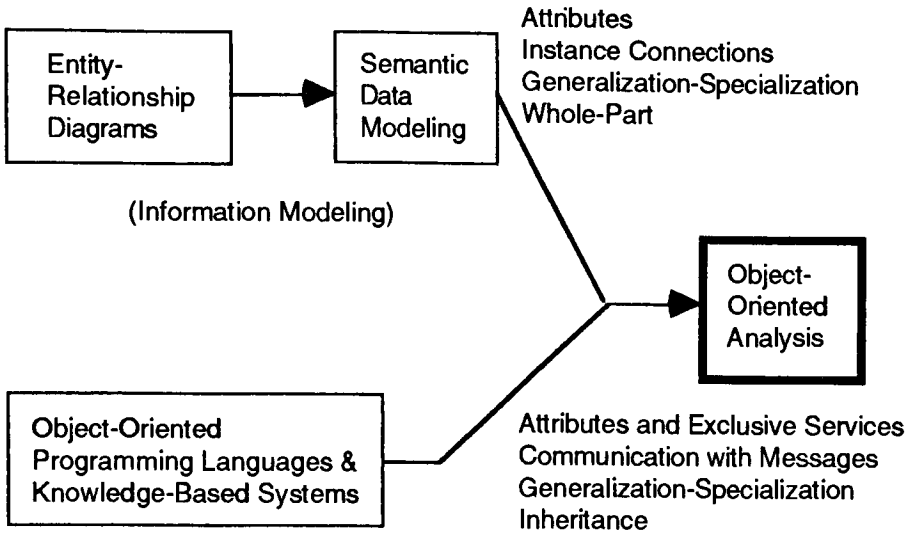


Figure 3: OOA Concepts
(31, pp. 31 Figure 1.4)

An Intelligent Tutoring System (ITS) fits well into the OOD model that Coad and Yourdon are proponents of. (31) The Multi-Layer, Multi-Component Model consists of five layers with four major components. The five layers are: subject, class-&-object, structure, attribute and service. These layers correspond to activities involved in OOA: finding class-&-objects, identifying structures, identifying subjects, defining attributes and defining services. The layers are overlapped and while it is helpful to move from one layer to the next, it is not necessary to do so. The layers or activities are not sequential steps and can be performed in any order. The four components of OOA are: human-interaction (equivalent to the Interface module of an ITS), problem domain (equivalent to the Expert module of an ITS), task management (equivalent to the Instructor module) and data management (also equivalent to the Expert module of an ITS). (31, p179) Objects are viewed as an abstraction of the real world. This provides a focus on the significant aspects of the problem domain and the systems responsibilities.

Figures 4 through 10 are provided to give background on the notation, documentation and functionality that OOA should produce. They are for reference use to understand the diagrams in the analysis and design sections of this paper.

Figure 4 lists the major principles espoused by Coad and Yourdon (31) for managing complex analysis.

1 Abstraction
    a Procedural
    b Data
2 Encapsulation
3 Inheritance
4 Association
5 Communication with messages
6 Pervading methods of organization
    a Objects and attributes
    b Whole and parts
    c Classes and members, and distinguishing between them
7 Scale
8 Categories of behavior
    a Immediate causation
    b Change over time
    c Similarity of functions

Figure 4. Principles for Managing Complexity
(31, pp. 33 Figure 1.6)
(The numbers are used in Figures 5 and 6.)

Figure 5 compares OOA with other approaches that have been mentioned, while Figure 6 shows which OOA constructs are used with what principles. Definitions of the principles and constructs can be found in the glossary.

Methods	Principles of Managing Complexity												
	1a	1b	2	3	4	5	6a	6b	6c	7	8a	8b	8c
Functional Decomposition	X												
Data Flow	X									X	X		
Information Modeling					X		X	X	X				
Object-Oriented	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 5: Comparison of Analysis Methods

(31, pp. 33 Figure 1.7)

OOA Construct	Principles for Managing Complexity												
	1a	1b	2	3	4	5	6a	6b	6c	7	8a	8b	8c
Class-&-Object		X	X				X						
Gen-Spec Structure				X					X				
Whole-Part Structure								X					
Attribute		X	X	X			X		X				
Service	X	X	X	X					X		X	X	X
Instance Connection					X								
Message Connection			X			X					X		
Subject								X		X			

Figure 6: OOA's application of principle for managing complexity

(31, pp. 34 Figure 1.8)

OOA was chosen for the analysis method for several reasons: it has improved the analyst and problem domain expert interaction; the internal consistency of analysis results are increased: commonality of attributes and services is explicitly represented; specifications can be built that are resilient to change; it provides a consistent

underlying representation for analysis and design; it incorporates parts of the other approaches and seems to be a more generalized and flexible approach. (31, pp. 3,4)

The OOA documentation set consists of the following:

- The five layer OOA model

  Subject, Class-&-Object, Structure, Attribute, Service

- The Class-&-Object specifications
- Supplemental documentation, as needed

  Table of critical threads of execution

  Additional system constraints

  Services/States table (31, pp. 164)

The notation and approach of OOA builds upon three methods of organization. "In apprehending the real world, people constantly employ three methods of organization, which pervade all of their thinking:

(1) the differentiation of experience into particular objects and their attributes - e.g., when they distinguish between a tree and its size or spatial relations to other objects,

(2) the distinction between whole objects and their component parts - e.g., when they contrast a tree with its component branches, and

(3) the formation of and the distinction between different classes of objects - e.g., when they form the class of all trees and the class of all stones and distinguish between them." (33, pp. 1)

Figures 7, 8 and 9 represent the notation used with the documentation set.

specification
attribute
attribute
attribute

externalInput
externalOutput

objectStateDiagram

additionalConstraints

notes

service <name & Service Chart>
service <name & Service Chart>
service <name & Service Chart>

and, as needed,

traceabilityCodes
applicableStateCodes
timeRequirements
memoryRequirements

Figure 7: Class-&-Object Specification Template

(31, pp. 197 Figure A.2)

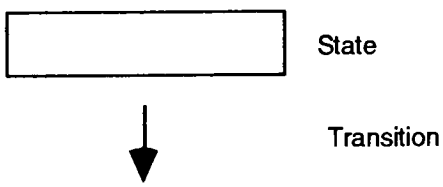


Figure 8: Object State Diagram Notation
(used within the template)

(31, pp. 197 Figure A.3)

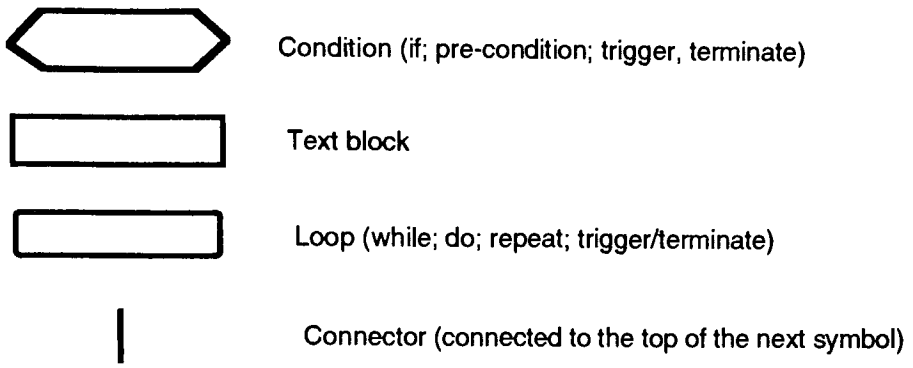


Figure 9: Service Chart notation
(used within the template, for each Service)

(31, pp. 197 Figure A.4)

The notation used for Object State diagrams and Service Chart notations is similar to that found in a standard flow chart. A service is equivalent to a method or function of a program.

Figure 10 illustrates the individual symbol notation that will be used in OOA diagrams within this paper. The symbols represent OOA constructs.

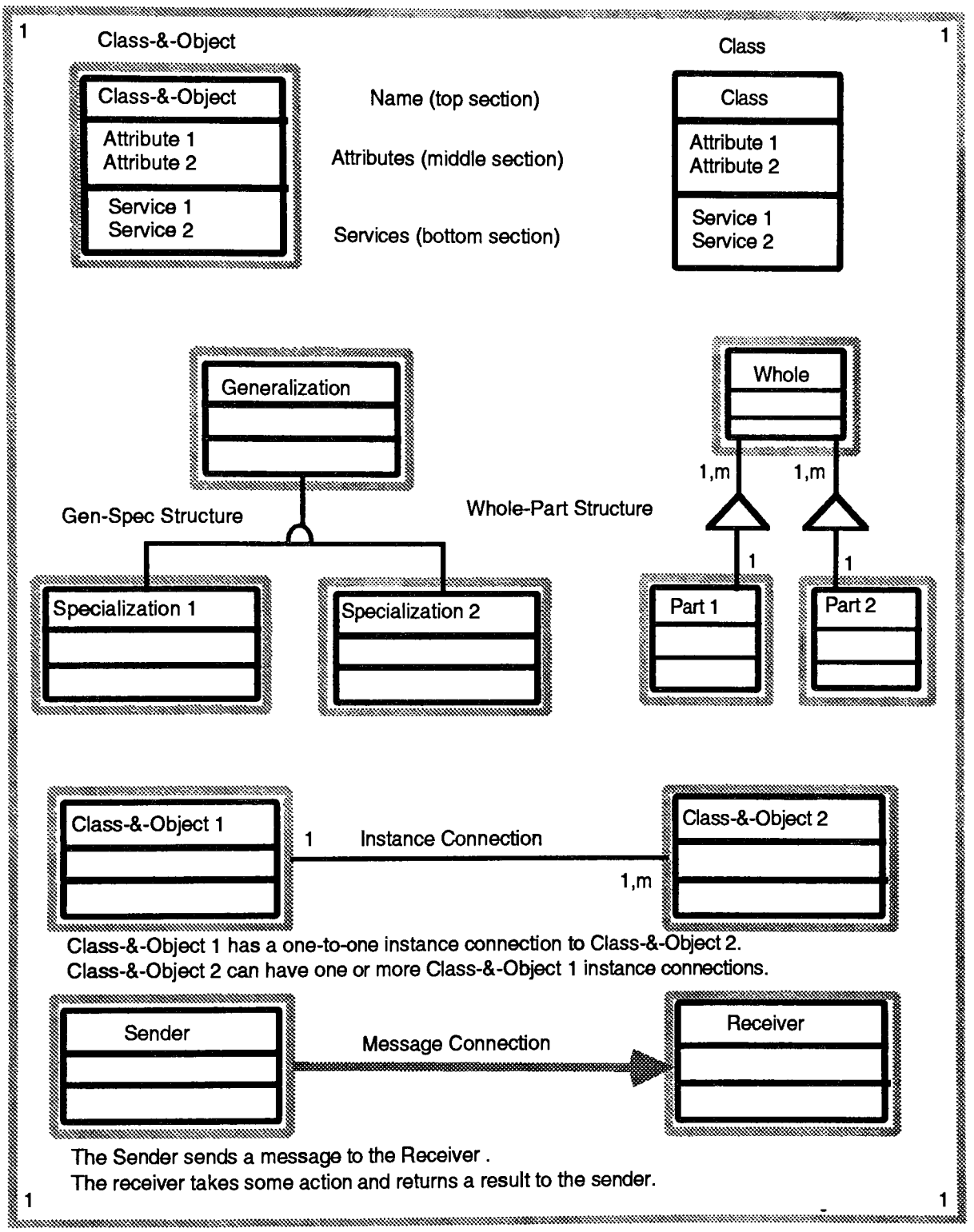


Figure 10: OOA Notations

(31, pp. 196 Figure A.1)

2.1 Thought Flow

The following section is to illustrate and place into print some of the thought flow that occurred during the analysis process.

The initial operation of an ITS begins with the user starting the program and selecting a main level topic to receive instruction on. The instructor receives this selection and checks the student history data to determine what level of instruction needs to be presented. The instructor then checks the expert rule-base to determine what instruction is to be selected based on the determined level and then retrieves that instruction from the domain knowledge base. The selected instruction is presented to the student at the interface. Instruction selection and presentation continue in this manner until the student selects another topic or ends the learning session.

Figure 11 represents the first of the analysis diagrams for the OOA of the ITS being represented in this paper. This figure represents the Subject and Class-&-Object layers for an ITS. It is the first activity in this analysis process. This activity defines the top level subject and the main classes and objects in the system. The large outer square represents the subject layer and the inner rounded squares represent the classes and objects. The lines within the inner squares represent class segments. The first line is the class name. The second line is for the attributes of the class. These will be entered in the definition of attributes activity. The third line is for the services of the class. These will be entered during the definition of services activity.

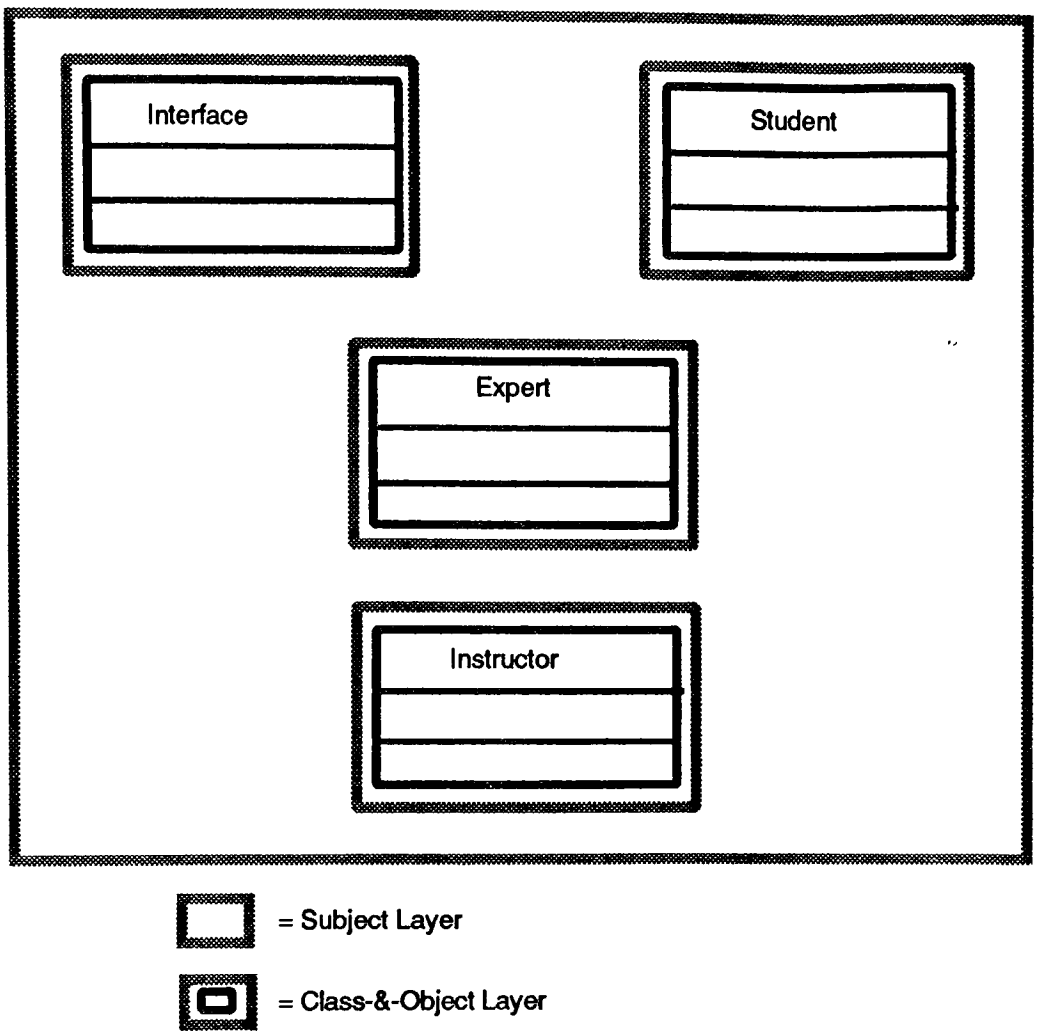


Figure 11: Initial Subject and Class-&-Object Layers for the ITS

The analysis will attempt to present a feasible method for implementation of the system by other students. Conclusions made during the analysis process may not necessarily be the best for Object-Oriented Analysis (OOA) nor better than conclusions determined by other approaches. The OOA process led to the conclusions contained in this paper. These conclusions are based on information gathered about Intelligent Tutoring Systems (ITS) and OOA. The most common form of ITS was followed in performing the analysis. The analysis of the ITS in this paper stays within the scope of

already existing systems.

While analyzing the problem of an ITS one of the first realizations was that the Interface module is a communications medium and performs no processing of its own. Its primary purpose is to display those instructions selected by the Instructor module that will facilitate the learning of the student. In the analysis of the Interface module that primary purpose was kept in mind. When implementing the precepts of OOA it became apparent that the Interface module was an object within the ITS as a whole which does not contain methods. It is a virtual or abstract class. The Interface module obtains the information it displays from the domain knowledge base within the Expert/school module. The displays are contained as instructional material within the Expert module. The Interface module has no intelligence built into it and is the visual aspect of the Instructor module. The Interface module is a communications medium between the Instructor module and the student using the system. The displays that are observed by the student are selected by the Instructor module based upon the lesson plan. The lesson plan is determined by the students previous sessions, if any, or by a standard startup session that would be established by those implementing the system. The Interface module would use as many displays that have been created or are available within the domain knowledge base. The Interface module becomes the largest but least complex piece of the ITS.

The Instructor module is the most complex part of an ITS. Most of the processing occurs within the Instructor module. The Instructor module decides what lessons to display based on the teaching rules within itself. On initial analysis of the ITS the domain rulebase was placed within the Expert module. After analysis it was determined that the domain rulebase belongs within the Instructor module. This decision was made by following the concepts of OOA and OOD. The view that was taken to learn this was to see the ITS as a public school system. This placed the Instructor module as the teacher, the Expert module as the school, the Student module

as the student and the Interface module as the blackboard. To get information about a student the teacher accesses the school records. Therefore the Instructor module accesses the Expert/school modules student-history database. To get instructional material the teacher acquires textbooks or other instructional material from the school. Therefore the Instructor module accesses the domain knowledge from the Expert/school module. Those rules or methods that the teacher uses to determine a lesson plan are an integral part of the teacher and not of the school. As related to a human, those rules are in the mind of the teacher. Therefore the domain rulebase is part of the Instructor module and not the Expert/school module. The Instructor module makes the decision about what instruction or display will be presented to the student/user. The intelligence of the system is contained within the Instructor module. The Instructor module employs the rules or methods of instruction used and can be thought of as the teacher. The instructional material that the Instructor module uses comes from the Expert module. The analysis of the Instructor module was both the hardest and the longest. This was due to its complexity and the attempt to cover all aspects of the instructors processes.

The Expert module can be referred to as the school which contains the student records and textbooks or other instructional material. After initial analysis the Expert module had three sub-modules. The domain knowledge base, domain rulebase and the student-history database. After further analysis the domain rule base was moved to the Instructor module. This follows the precepts of OOA where objects should accommodate real world views, such as the school analogy. The teacher contains the knowledge of rules to apply within their minds. Within the ITS program the database that contains the knowledge of how to teach should be part of the Instructor module. This reduces the amount of I/O for the Expert module. The sub-modules left in the Expert module after the last analysis step and the first design step are the domain knowledge base and the student-history database. The student-history database is

accessed by the Instructor and Student modules. The domain knowledge database is accessed only by the Instructor module.

The Student module is a data structure that contains a temporary picture of the student that is representative of the current session that the student is in. A final record is written when the student changes the topic or ends the session. The Instructor module sends the data to the Student module that it needs to hold and also sends the signal to the Student module to write the record into the student-history database. The success ratio is the calculated value that the Instructor module uses to determine the lesson plan for the current topic or sub-topic.

Figure 12 illustrates the messaging and instance connections between the objects in the ITS. This is the second OOA diagram for an ITS. It shows the interaction of the modules. This diagram was created before the school analogy was realized.

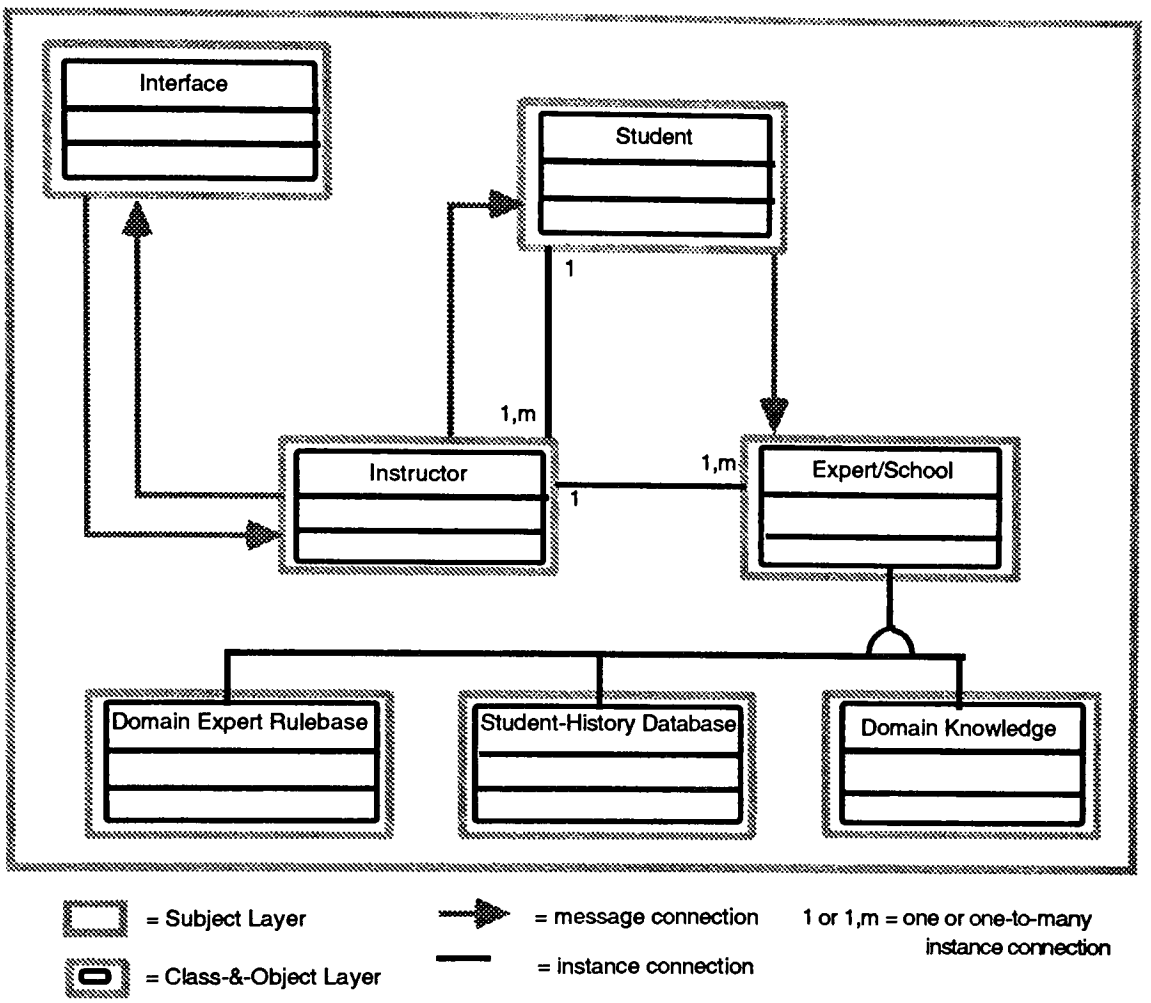


Figure 12: Subject, Structure and Class-&-Object Layer for the ITS

In researching of Intelligent Tutoring Systems, one of the reasons for the topic of this thesis was being unable to locate an application that applied a formal methodology to the problem of ITS. OOA and OOD was selected because of personal interest and popularity within professional circles. The OO paradigm seems to fit the picture of an ITS and the application of the OO paradigm to an ITS seemed worthy of investigation.

Each set of OOA diagrams for a module is comprised of: the Class-&-Object, Attribute and Services layers diagram, the specification of each object, the service chart diagrams and/or the object state diagrams. Some of the modules do not have

object state diagrams as they were not applicable. The service charts could have been done with bullets instead of diagrams. Diagrams were chosen because pictorial representation has some advantages over textual representation. 'A picture is worth a thousand words.'

2.2 Interface Analysis

2.2.1 Factors and Problems to Consider

The graphical interface is a communication medium between the student and the rest of the system. It is a two-way communication with the student providing input from the screen and the system providing output to the screen. This is the least complex piece of the system but possibly the largest due to the number of display types that could be used.

When analyzing the problem of the Interface module there are several factors that need to be considered: whether there is effective communication, the look and feel of the interface, the ease of use of the interface, and the field placement of items within the interface. These can arguably be considered to be design decisions but because of the nature of a graphical user interface these shall be part of the analysis process. Other problems to be dealt with include how well the interface addresses the semantics of its task and the problem domain and how well it addresses the students' knowledge and abilities. While current interfaces may have problems with the menus and messages, those problems are centered on the content, not the mechanics of the interaction. It is useless to compute the correct instruction to be presented if the instruction is not communicated to the student properly.

Future systems may approach the level of human communication if the interaction was extended from simple textual and graphical techniques to include natural language, voice, animation, three-dimensional graphics and video. At present, an interface designer uses design knowledge, experience and task analysis to determine what information to present to the student and organizes that information into a formatted display. The display illustrates the important aspects of the information being presented. This does limit the student to a particular set of displays and does not always take into account the needs of a particular student or the students' particular set of tasks. Unfortunately, if the number of displays is increased,

the number of potential choices of displays could be large enough to render implementation unfeasible. (29, pp. 5-9) This choice is implementation dependent and beyond the scope of this paper.

There are several methods that can be used for user interface management. A multi-media interface supports more than one medium through which the student and the system communicate. A collaborative interface exploits knowledge about tasks in ways that help the student to accomplish those tasks effectively. Collaborative interfaces use context to aid in interpreting ambiguous inputs and to phrase output in a manner sensitive to the students' situation, and provide advice on efficient ways to accomplish the students' goals. (29, pp. 294) Interfaces that are part of an ITS are a collaboration between human and computer. These graphical knowledge-based models create an interface that allows the computer to present an accurate description of the current situation and the result of its simulations or heuristic analysis and also allows the student to express details of the situation and decisions on how to proceed. This approach allows the intelligence of the system to be embodied in the knowledge-base rather than in the interface.

A graphical knowledge-based model is the type that will be used within the analysis because of its predominance within already existing Intelligent Tutoring Systems and the embodiment of the intelligence remains within the knowledge-base rather than the interface. The number of displays to be used is implementation dependent. The method by which the displays are utilized by the instructor is a design issue.

Figure 13 is the Class-&-Object, Attribute and Service layer diagram for the Interface module. The attributes of the Interface module are the InputBuffer, OutputBuffer and DisplayType. The services are InputFromStudent and DisplayOfInstruction. The specification gives a brief description of the attributes and services, the external input/output sources and the service chart notations for each

service.

When the system is started by a user, the instructor sends a message to the interface to display the name request form. This is a use of DisplayOfInstruction and the OutputBuffer. The user enters their name. This is a use of the InputFromStudent and the InputBuffer. The user is identified by the instructor as a new or not-new student. A new student is shown the standard list of topics. A not-new student is asked if they wish to continue with the topic from the last session. The appropriate list is displayed depending on the students' selection. Once a topic has been selected, the instructor decides what instruction will be displayed. The next steps are under the Instructor module analysis section.

2.2.2 OOA Diagrams

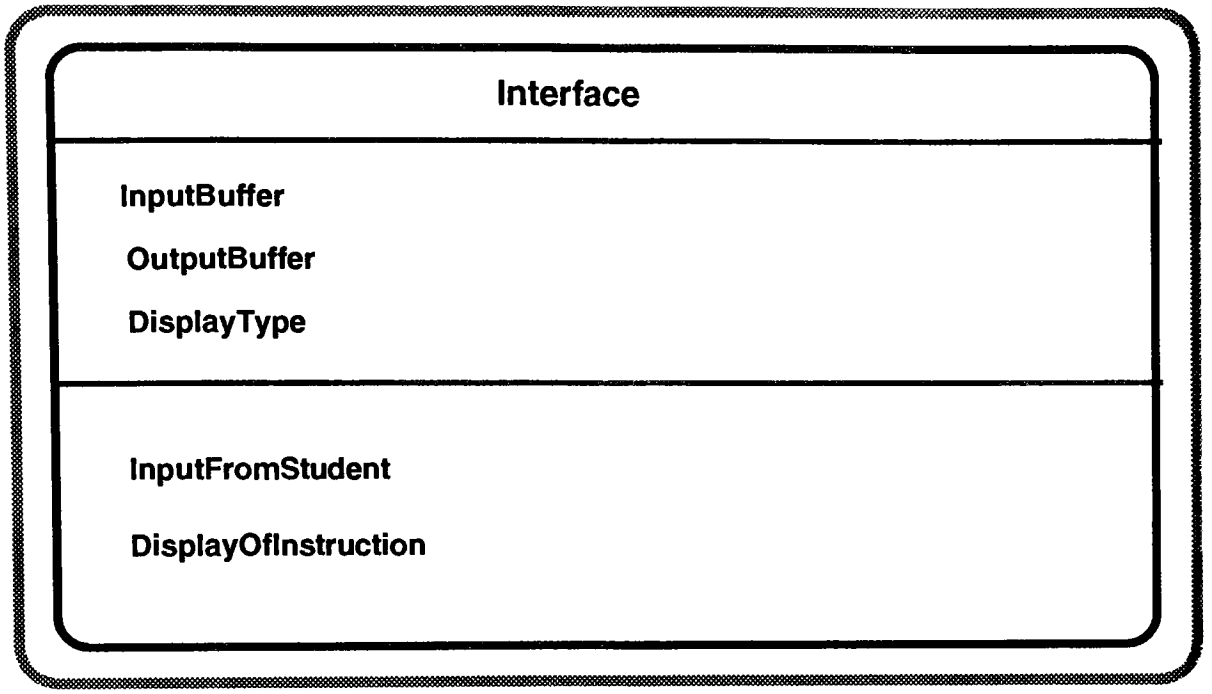


Figure 13: Interface Module - Class-&-Object, Attribute and Service layers

specification Interface

attribute InputBuffer: holds the student input from the keyboard

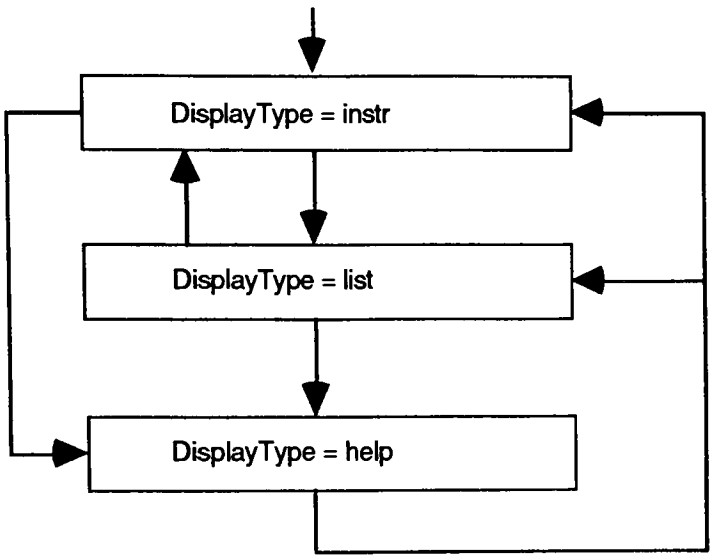
attribute OutputBuffer: holds the instruction to be displayed

attribute DisplayType: the display type of the interface (text field, selection list, help screen, picture) Implementation dependent.

externalInput KeyboardInput: the input from the keyboard

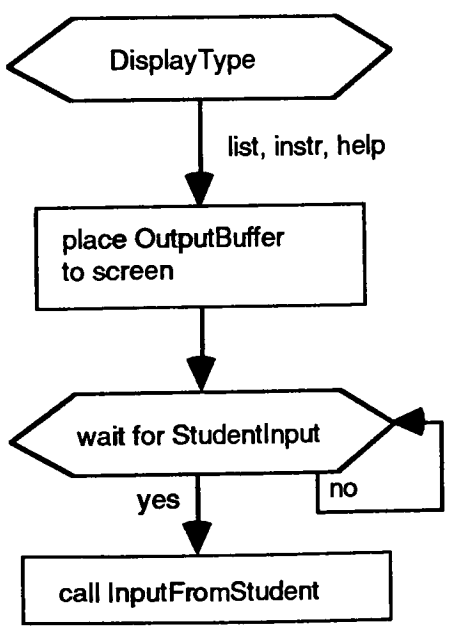
externalOutput ScreenOutput: the data from the output buffer

objectStateDiagram

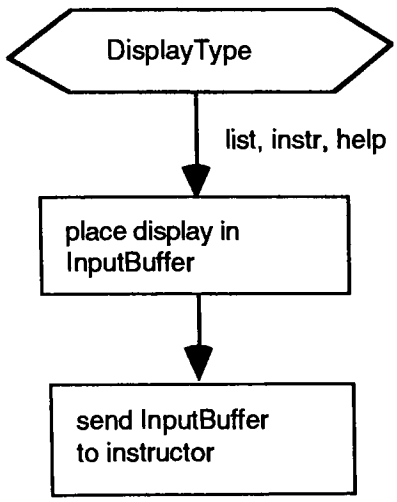


instr = text, graphic, name and/or
other to be determined by implementation

service DisplayOfInstruction



service InputFromStudent



2.3 Instructor Module Analysis

2.3.1 Factors and Problems to Consider

The major factor to consider for the Instructor module are theories of teaching. One theory is the simple communication of subject matter. This is a traditional view of teaching and involves conversing with the student. Another is the remediation of incomplete or incorrect mental representation of the student and is the current view of teaching. It is considered the current view because most ITSs that have been implemented are based upon this method. The facilitation of knowledge construction is a future view of teaching. Using any teaching method involves making a decision about how and what to teach. These decisions are the basic knowledge requirements for each of these teaching theories. (28, pp. 26)

Simple communication includes deciding on the subject matter to teach and which way to present that topic. Deciding what to teach is a recurring process that occurs at many levels. Topics can vary in scope and are generally nested. For example, the topic of medicine contains large sub-topics such as neurology and hematology, which have sub-topics themselves. Presentation decisions include the sequencing of the sub-topics, selection of exercises, forming explanations and determining the graphical details for visual illustrations. (28, pp. 27)

This theory generates two issues pertaining to educational research; subject matter analysis and the effectiveness of different modes of presentation. Subject matter analysis involves the use of a computer program that can solve problems within the knowledge domain of the system. In attempting to create these programs it has been discovered that for most knowledge domains no standard codification of that knowledge exists. "The stringent criteria of clarity and completeness that computer implementation imposes on subject matter analysis have led to the conclusion that teachers have always worked with inadequate codification of the knowledge they were supposed to teach. It is obvious that a clear and precise codification of the subject

matter is an important requirement for effective teaching, whether the teaching is delivered by a human or a machine." (28, pp. 28) Without clear and precise codification the system is unable to teach effectively due to a lack of complete or accurate knowledge.

Remediation includes the decisions of subject matter and topic representation. In addition, remediation deals with correcting deficiencies in the mental representation of the student at the current time. (28, pp. 35)

This theory aims to correct the students mental representation of the subject matter. The teachers job is to remediate the discrepancy between the students view and the complete and "correct" view. The basic knowledge requirements of subject matter and topic representation are included in this theory. An essential difference between the theories of communication and remediation is in representation of the topic. In remediation, the expert makes the decision based on what needs to be corrected. Teaching is considered an a corrective action rather than communication of a knowledge item. Corrective action takes many forms such as stating or restating the knowledge item, asking a question or providing a counter example. Communication teaches a concept while remediation teaches how to correct an error.

A knowledge requirement introduced by remediation is the codification of the relevant subject matter and the theory of mental representation. These requirements are also the basic parts of an expert model. This requirement causes some problems. Often, a standard codification of the subject matter is not available, or is incomplete enough that it is unusable. Also, the encoding of the subject matter into a particular format can have more than one solution. How the expert encodes the subject matter is another problem and the solution requires collection of that information directly from the expert. "The purpose of the expert model is to facilitate the description of errors. Once a model of the correct representation is available, the learner's mental representation of the subject matter can be described as a deviation from the correct

one." (28, pp. 36) Other knowledge requirements that are generated by this theory are: "(a) knowledge of the format(s) of mental representations and the processes that operate upon them, (b) knowledge of the mental representation of somebody who knows the subject matter well, (c) knowledge of the different perversions of the subject matter that are likely to occur in human learners, and (d) knowledge of how to identify which of these perversions a particular learner is suffering from at a particular moment." (28, pp. 37) 'Perversions' is used to describe a deviation from the state that the system is attempting to produce from the student. Satisfaction of all these knowledge requirements is difficult and there are few knowledge domains that this can be done in.

The basic process of remediation is that instruction is tailored to the knowledge state of the student. This presents some problems of its own. The instruction being taught can be misunderstood in more than one way. An instruction that corrects one knowledge item may not be the same instruction that corrects another. The Instructor needs to decide which instruction will correct the misunderstanding from the view of the correct representation. "The description of a deviation between student's mental representation of the subject matter and the correct representation does not entail any conclusion about which tutorial message might cause the learner to correct that deviation." (28, pp. 42) The expert model and the student model are actually based on performance but the instructions being delivered should be based on a theory of learning. Many current ITSs are only partially based on learning theory. "The advantages of theory-based instructional design cannot be cashed in, as it were, unless *all* components of the tutoring system are designed in accordance with theory." (28, pp. 43)

Facilitation includes selection of topic, the method of teaching the topic and identification of the students' deficiency. The essential difference for this method from remediation is that it deals with selecting a method to correct the deficiency in an

intelligent manner rather than simply correcting the deficiency by explanation. This theory wants to identify which productive learning sequences are possible in the students current knowledge state, under which circumstances will those sequences occur and how can those circumstances be brought about. (28, pp. 44)

This theory attempts to help the student improve their current view of the subject matter. Improvements have many forms: the connection of unconnected knowledge; the resolution of conflicts after identification of the conflicting information; the clarification of vague information; the identification of the difference between similar pieces of information; etc. Ohlsson (28) calls these processes *learning events*. A sequence of these is called a *learning sequence*. A *productive learning sequence* leads to a better or improved view. Teaching with this theory involves arranging situations in which a productive learning sequence would occur.

Facilitation brings the knowledge requirement that the instructor must know how learning happens. Most modern learning theory is focused in procedural knowledge but the core content of most knowledge domains consists of conceptual knowledge. At this time there are no precise learning theories that can explain the construction of conceptual knowledge. Instruction in concepts cannot be based on theory because of the lack of an appropriate theory. (28, pp. 44-46)

It has been shown that different teaching theories have different requirements. The traditional theory implies that the instructor knows the subject matter. The current and future theories build upon this knowledge of the subject matter by adding a more complex aspect to teaching. The current theory uses the subject matter to remediate the students errors. The future theory facilitates knowledge construction with the use of the subject matter. The design of the ITS to be done in this paper will use the current theory of learning due to it being the most usable solution.

The Instructor module decides what instruction to present by utilizing the expert and domain knowledge bases. Once an instruction has been presented to the student

that information is recorded in the student-history database.

Figure 14 is the Class-&-Object, Attribute and Service layer diagram for the Instructor module. The attributes of the Instructor module are the StudentData (a structure), DisplayType, InstructionData (a structure) and StudentExists. The services are Initialize, SendDisplayTypeToInterface, ReceiveStudentName, SearchForStudent, ReceiveTopicSelected, InstructionPlanning, ReceiveAnswerToInstruction, ReceiveNewTopicRequest and EndSession. The specification gives a brief description of the attributes and services and the service chart notations for each service.

Once a topic or sub-topic has been chosen by the user, the Instructor module sends the name and topic data to the Student module and creates a lesson plan by accessing the domain rulebase. When an instruction has been selected for display the Instructor module tells the interface to display that instruction. The user inputs their answer and that information is sent to the Instructor module. The answer is evaluated for correctness. The success ratio is calculated. The lesson plan is checked. The answer and correctness are sent to the Student module. The next instruction is sent to the interface. This cycle continues until the user elects to end the session or to change the topic. If the user chooses a new topic the Student module is told to write the record to the student-history database and the cycle goes to the point of displaying a list of topics. If the user chooses to end the session then the Student module is told to write the record to student-history and the program exits.

2.3.2 OOA Diagrams

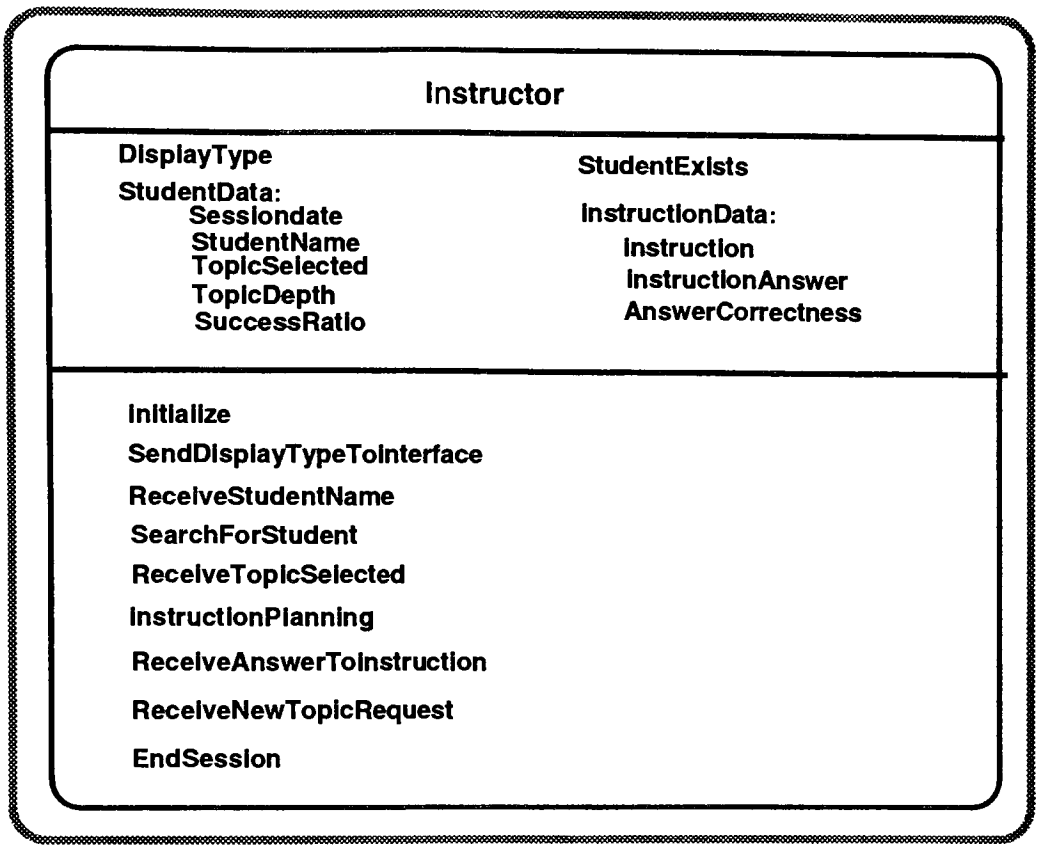


Figure 14: Instructor Module - Class-&-Object, Attribute and Service layers

specification Instructor

attribute Instruction: the instruction to be presented

attribute TopicSelected: the topic selected by the student

attribute StudentName: the name of the student

attribute SessionDate: the date of the current session

attribute SuccessRatio: the success ratio of the current session,

correct/total

attribute TopicDepth: the depth reached in the topic selected

attribute InstructionData:

the current instruction, the answer to instruction,

the correctness of instruction

attribute StudentData: the session date, the student name,
the topic selected, the topic depth reached, the success ratio

attribute DisplayType: the type of instruction being displayed
(list, instr, help)

attribute StudentExists: whether the student has entered the system

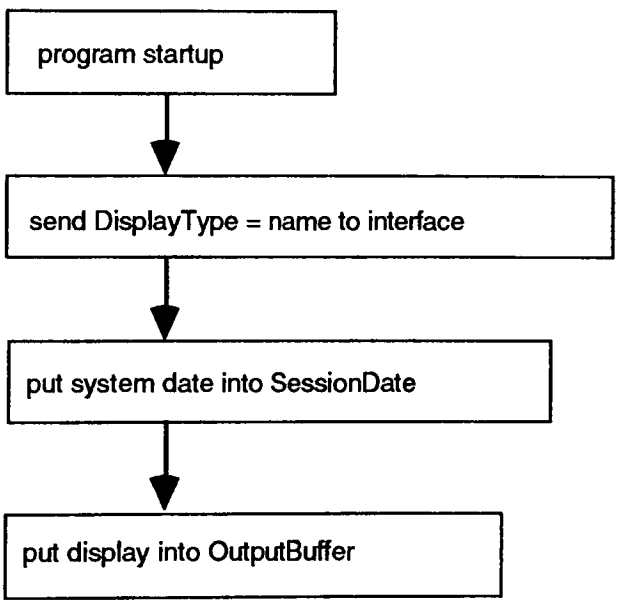
attribute InstructionAnswer: the student answer to the current instruction

externalInput InterfaceInput: the input from the interface

externalOutput OutputBuffer: the display data

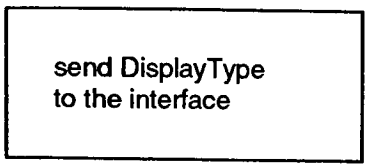
DetailRecord: the InstructionData

service Initialize: start up the system by asking for the students' name



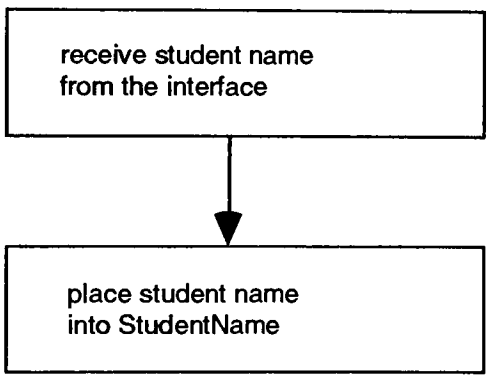
service SendDisplayTypeToInterface:

the display type is sent to the interface



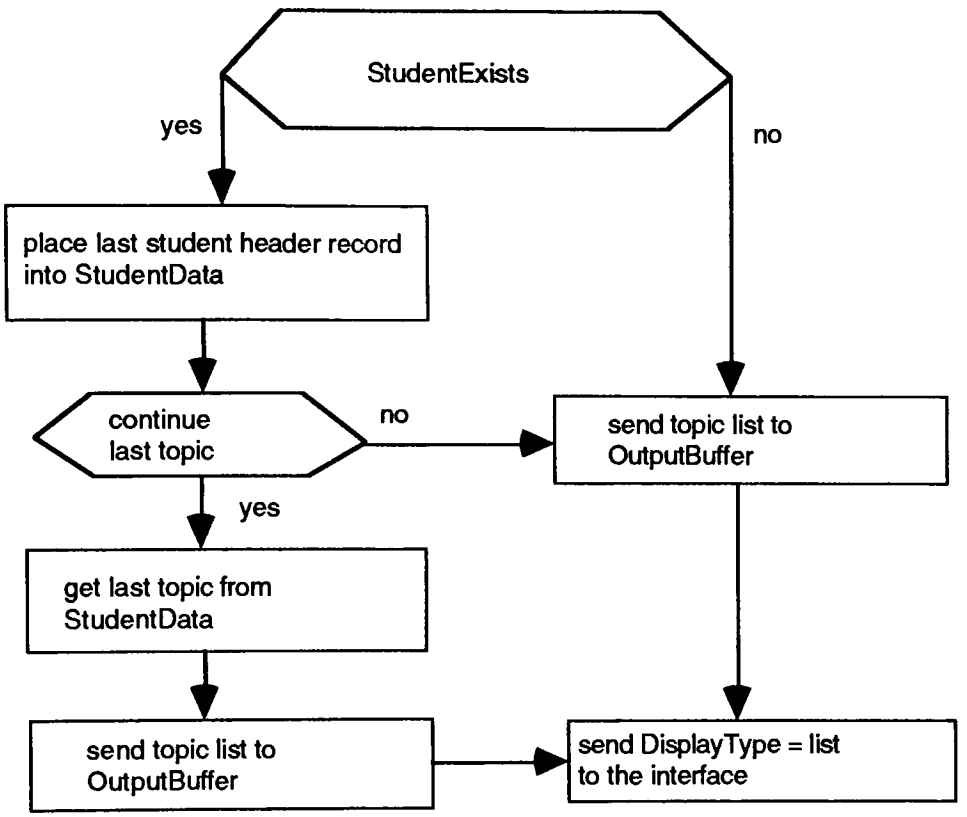
service ReceiveStudentName:

the students' name is received from the interface



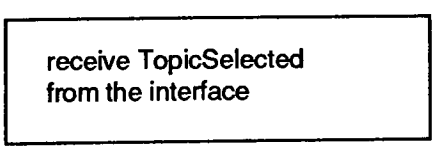
service SearchForStudent:

the student-history database is searched for the student name

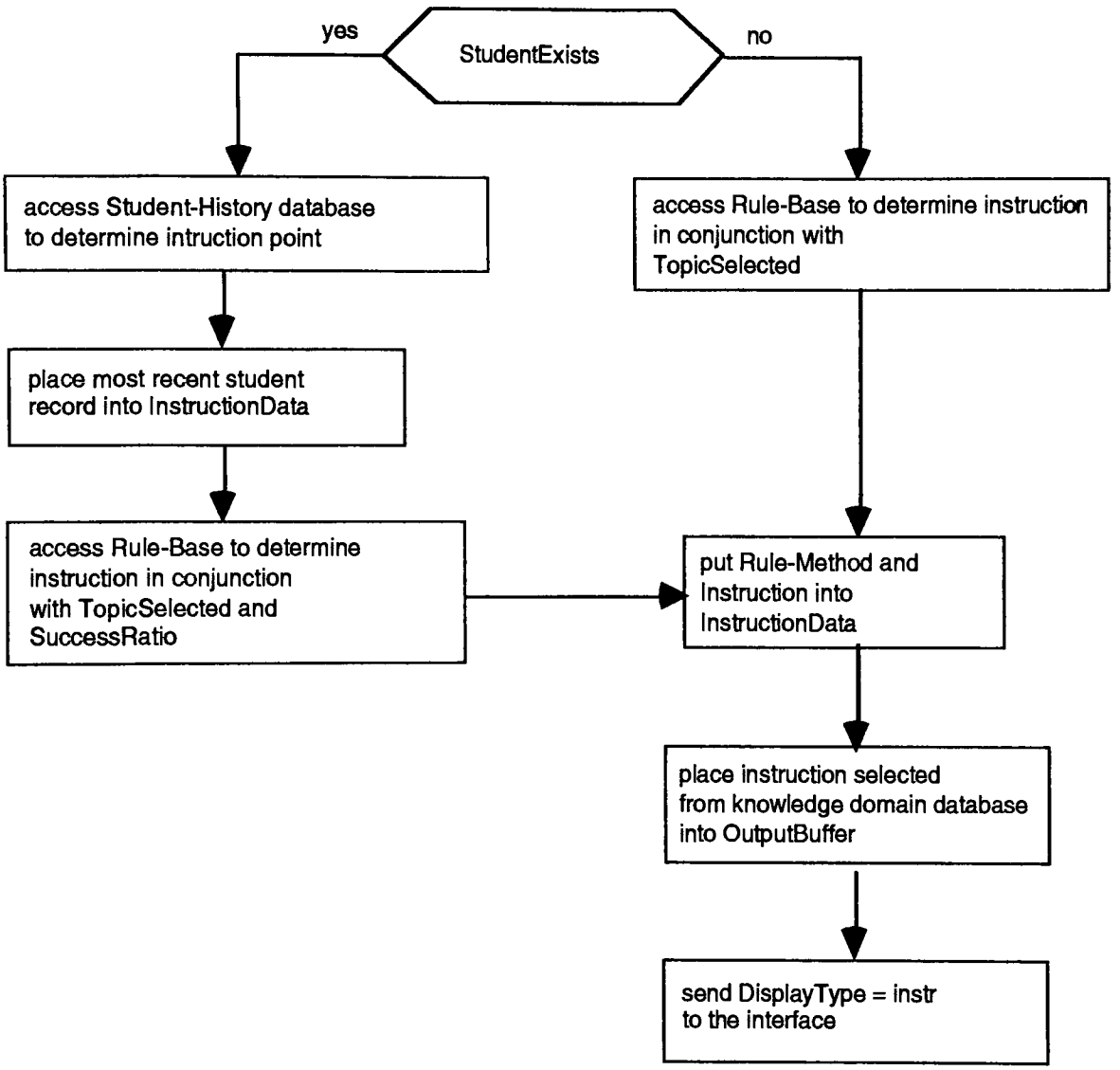


service ReceiveTopicSelected:

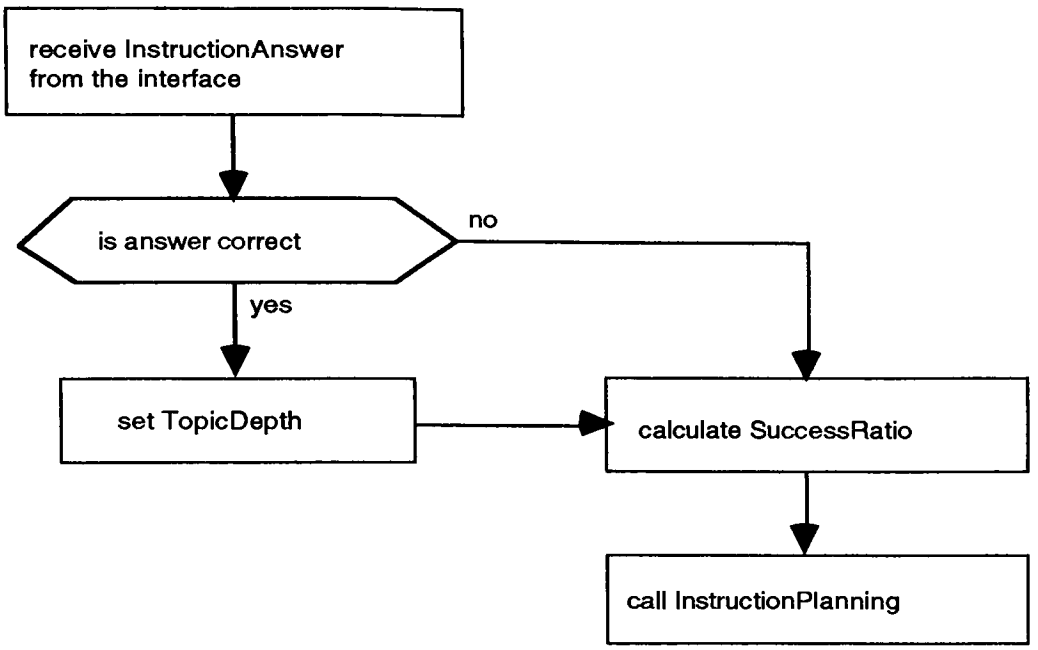
the topic selected is received from the interface



service InstructionPlanning: the instruction plan is created or modified based upon the topic selected, the topic depth and the student history

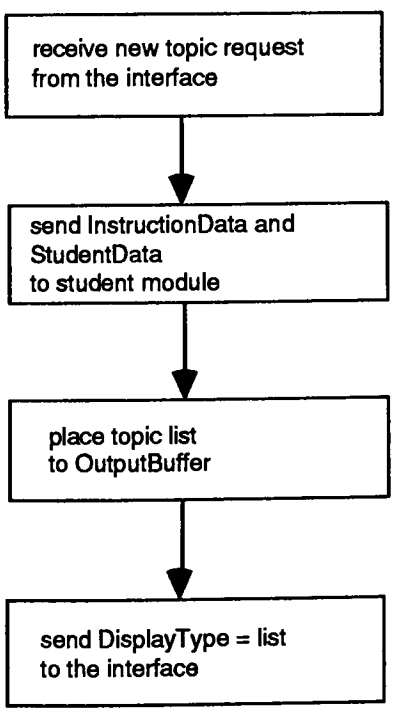


service ReceiveAnswerToInstruction: the student response to the current instruction is received from the interface

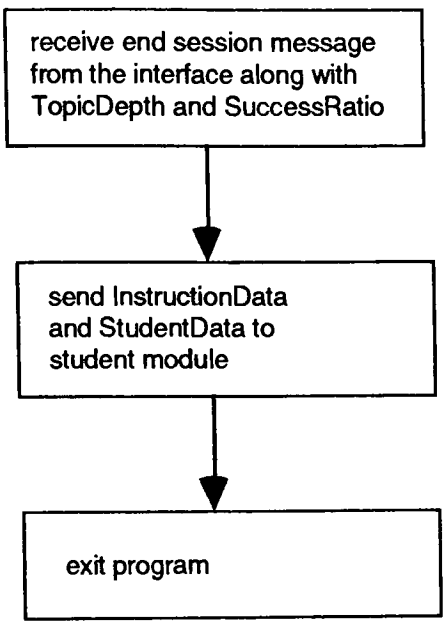


service ReceiveNewTopicRequest:

a new topic is requested by the student



service EndSession: the end session message is received



2.4 Student Module Analysis

2.4.1 Factors and Problems to Consider

The Student module is the component of an ITS that represents and records the student's current state of knowledge. Inferring the student model is a diagnostic process that determines the student's knowledge of the subject matter. The Student module and the Instructor module are interdependent. The Student module is a data structure and diagnosis is the process that manipulates it.

The Student module inputs records to the history knowledge base that keeps track of the students activities on the system. The student-history database contains information on methods or rules used, topics covered, the depth of a topic and the success/performance data for what has been covered. The input is garnered through interaction with the student. The information available to the Student module depends on the overall ITS application. Information could be, for example, answers to questions posed by the system, commands to an editor or the student's educational history.

Common uses for the Student module are advancement, offering unsolicited advice, problem generation and adapting explanations. Advancement is a structured curriculum. The student is moved to the next topic when the student has mastered the topic. Advancement is useful with linearly structured and componentially structured curricula. Unsolicited advice is offered only the system decides that the student requires it. When the student is performing well, the system does not respond. The ITS must know the state of the student's knowledge in order to offer instruction at the right time. The Student module is read in order to accomplish instruction at the correct time. Problem generation by the system is a dynamic process. The student is presented with problems to solve rather than sequencing through a pre-defined list of problems or letting the student invent problems to solve. A good problem is usually slightly beyond the student's current capabilities. The Student module is read in order

to determine the student's capabilities. Adapting explanations requires that explanations use concepts that the student already understands. The system can issue good explanations after determining what the student already knows by referring to the Student module. (17, pp. 55-57)

Figure 15 is the Class-&-Object, Attribute and Service layer diagram for the Student module. The attributes of the Student module are the TopicSelected, StudentName, SessionDate and InstructionAnswerStructure (a structure). The services are Initialize, ReceiveTopic, ReceiveInstructionAnswerData and WriteStudentHistoryRecord. The specification gives a brief description of the attributes, the external input/output sources and the service chart notations for each service.

Information is received from the Instructor module. The session information is held until the call is received from the Instructor module to write the record to the student-history database. This occurs when the user selects instruction on a new topic or decides to end the current session.

2.4.2 OOA Diagrams

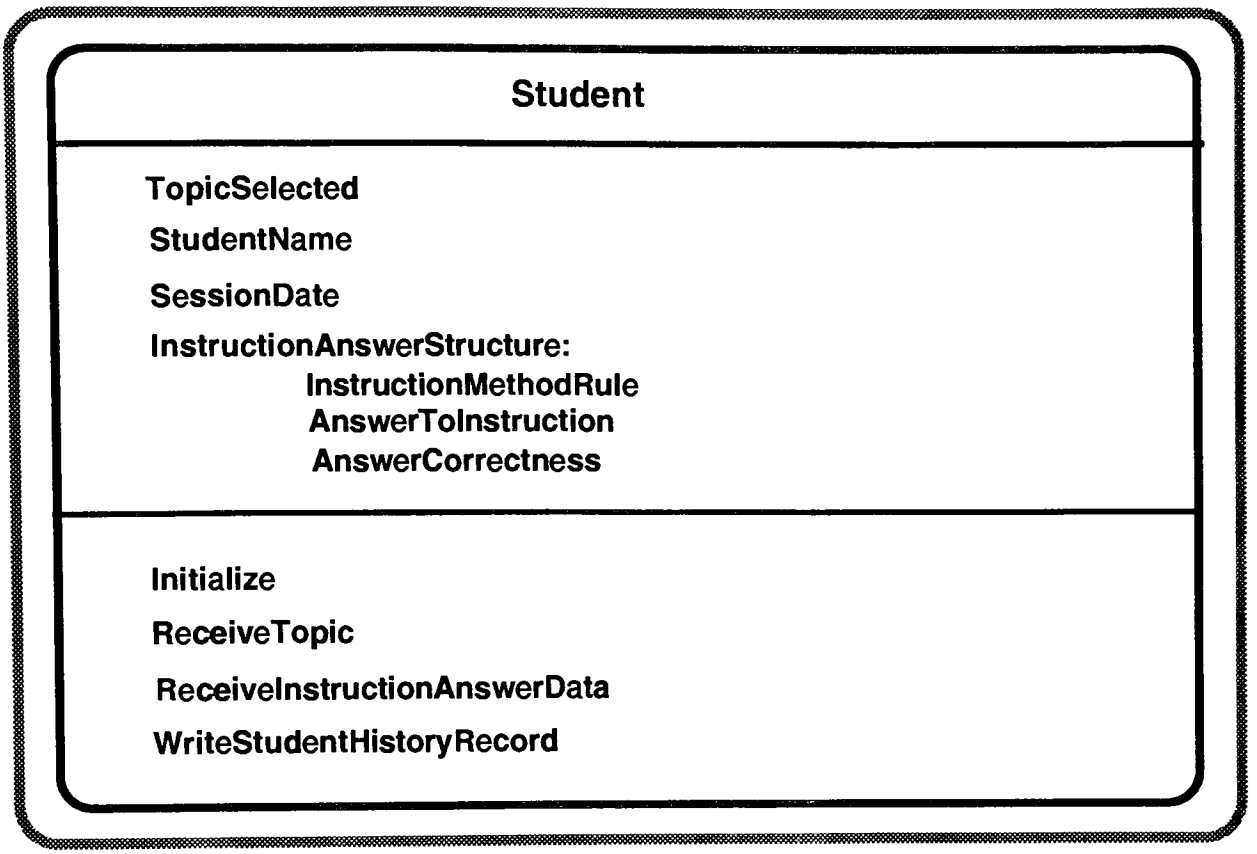
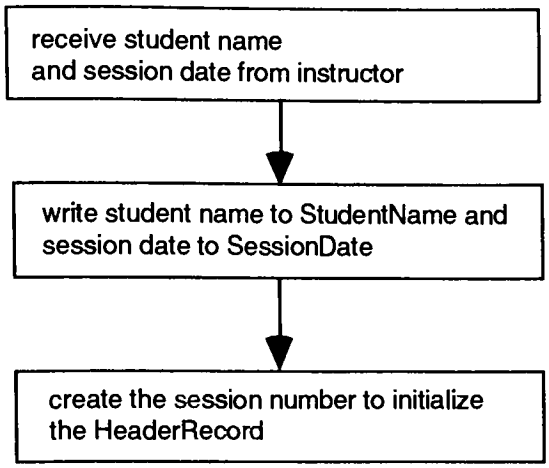


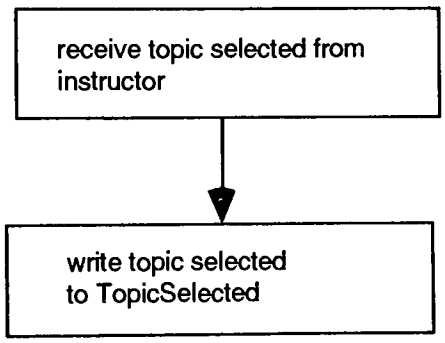
Figure 15: Student Module - Class-&-Object, Attribute and Service layers

specification Student
attribute InstructionAnswerStructure: the InstructionMethodRule, AnswerToInstruction and AnswerCorrectness
attribute TopicSelected: the topic selected
attribute StudentName: the students' name
attribute SessionDate: the date of the current session
externalInput InstructorData: the final success ratio of the student for the current session, the topic depth reached in the current session
externalOutput
HeaderRecord: the session number, the session date, the student name, the topic selected, the topic depth reached, the final success ratio
DetailRecord: the session number, the InstructionAnswerStructure

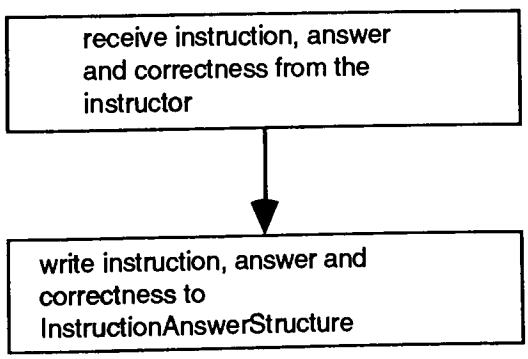
service Initialize



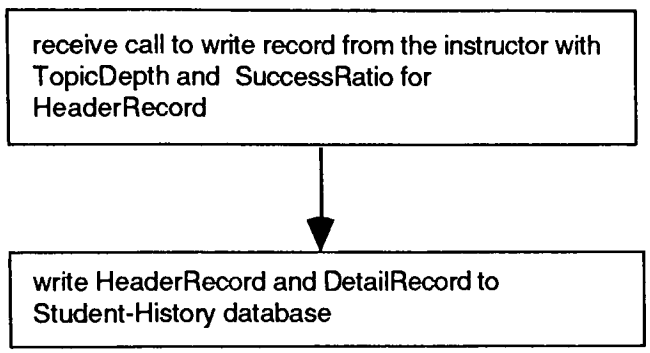
service ReceiveTopic



service ReceiveInstructionAnswerData



service WriteStudentHistoryRecord



2.5 Expert Module/Knowledge Base Analysis

2.5.1 Factors and Problems to Consider

The Expert module or knowledge base module interacts with the Instructor module to decide which instruction to present and the domain knowledge that corresponds to that instruction.

An Expert module must have an abundance of knowledge. A great deal of effort is expended in discovering and codifying the domain knowledge. The amount of knowledge required in most domains ensures that the development of the Expert module is labor-intensive. An estimate has been made that over fifty per cent of the work in an ITS application is spent in encoding the domain knowledge.

Some options for encoding the knowledge domain are a black box model, an expert system or a simulation. The black box model is a method of reasoning about the domain that does not require the actual codification of the knowledge. A system might use mathematical equation solving processes which produces output through numerical processes what the student would achieve through symbolic processes. The system reasons by simulating the knowledge with its mathematical model. An expert system is created by extracting information from a human expert and devising a way of codification and application of that knowledge. The method in which the knowledge is applied does not have to correspond to the method that the expert applied it. Simulations are used to illustrate a process in the same terms that a student should use in reasoning about the process. The way a student uses the knowledge is simulated at some level of abstraction by the system.

Intelligent tutoring systems are usually built with an expert system. Most Intelligent Tutoring Systems are designed for domains that have an already existing expert system available. ITSs teach these topics because they satisfy a need for robustness, establish prerequisite knowledge and teach part of a skill. For example, it is considered valuable to be able to spell even though spelling correctors exist. It is

necessary to possess basic math skills in order to learn calculus, and the instructor is at least capable of teaching part of a skill. (17, pp. 21-26)

Figure 16 is the Class-&-Object, Attribute and Service layer diagram for the Expert module. The attributes of the Expert module are the StudentHistoryData, InstructionData and RuleData. The services are the ProcessRequestForInstruction, ProcessRequestForStudentData and ProcessRuleRequest. The specification gives a brief description of the attributes and the service chart notations for each service.

The domain rulebase is accessed by the Instructor module when the lesson plan and instruction being selected for display is determined. The domain knowledge base is accessed by the Instructor module to retrieve the instruction selected by the lesson plan. The domain knowledge base or domain rulebase is written or added to outside of the scope of this paper. The maintenance of the data in the system is considered a separate issue from the ITS as a whole. Data is retrieved from the student-history database when the Instructor module is looking for a users previous sessions. Data is written to the student-history database by the Student module when the Instructor module passes that message.

2.5.2 OOA Diagrams

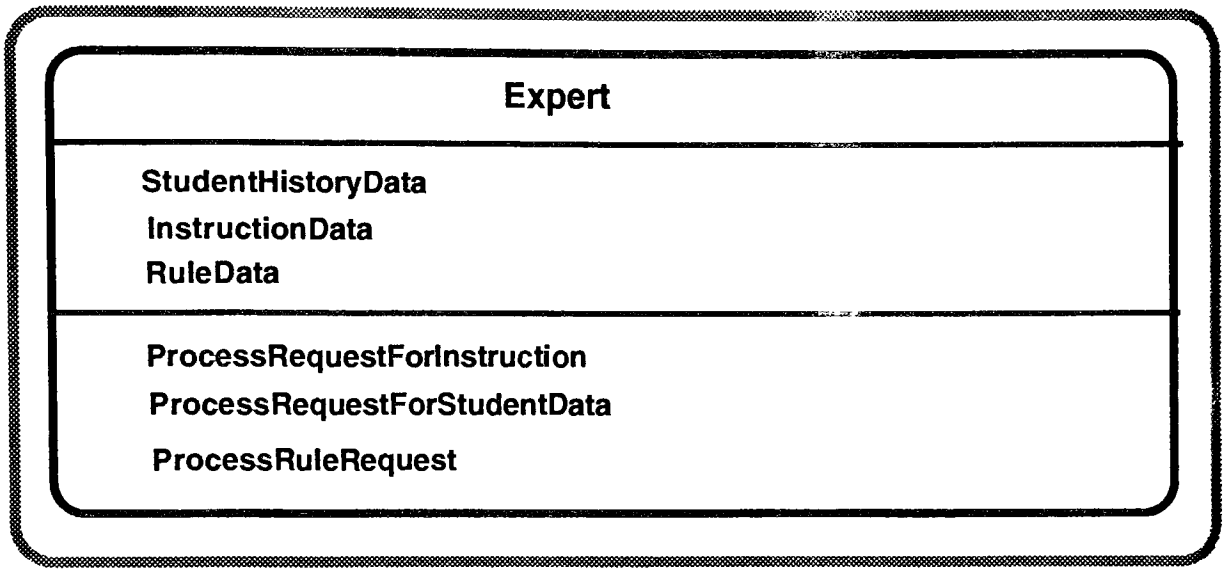


Figure 16: Expert Module - Class-&-Object, Attribute and Service layers

specification Expert

attribute StudentHistoryData:

the most recent student record that matches the topic selected

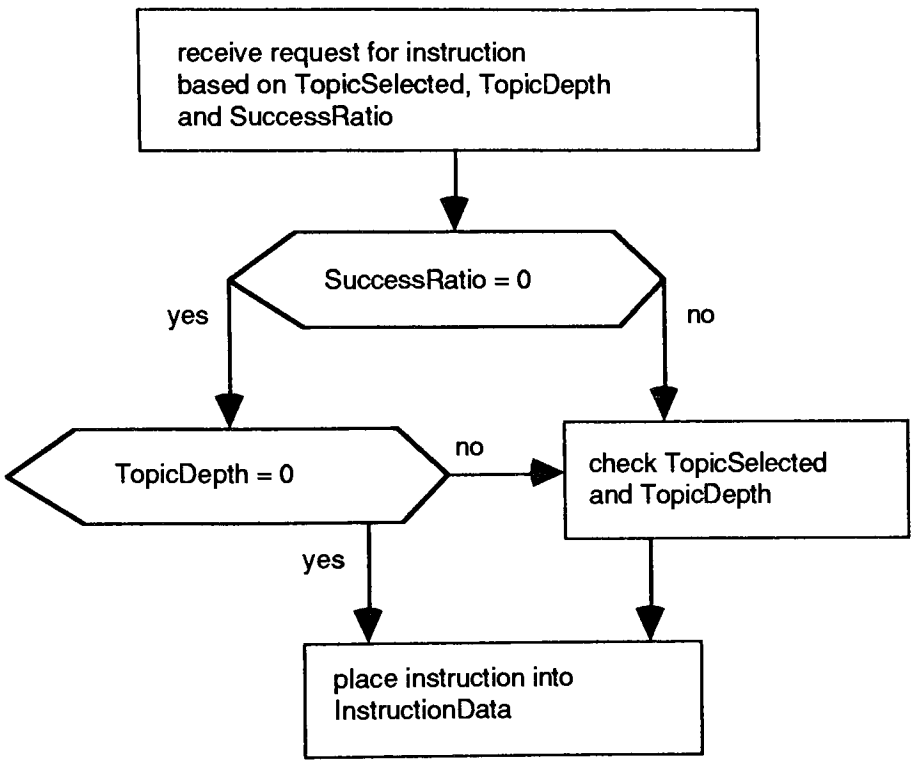
attribute InstructionData:

the instruction data from the domain knowledge database

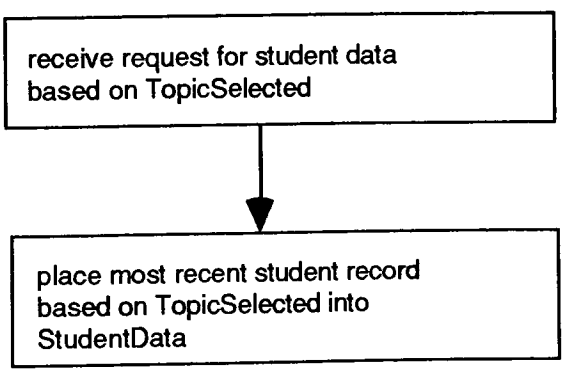
attribute RuleData

the rule or method used to determine an instruction

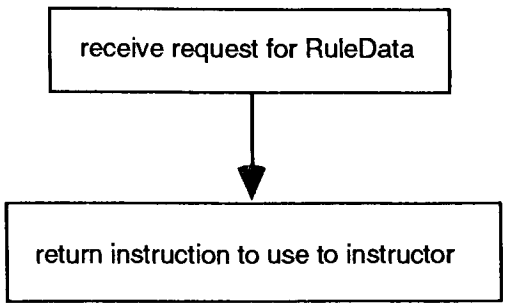
service ProcessRequestForInstruction: the request for the instruction is received from the instructor



service ProcessRequestForStudentData: the request for student data is received from the instructor



service ProcessRuleRequest: the request for application of a rule or method to determine what instruction to use is received from the instructor



3. Design

3.1 Factors and Problems to Consider

Object-Oriented Analysis (OOA) identifies and defines the classes and objects that directly reflect the problem domain and the system's responsibilities within that domain. Object-Oriented Design (OOD) expands the OOA layers that model a particular implementation. It identifies and defines additional classes and objects, reflecting an implementation of the requirements. There are no major differences between the OOA and OOD notations. The difference between analysis and design becomes a question of which activities are being performed. This means that the software professional would be unable to point to the shape of a data flow bubble or a structure chart box and say that they are specifically doing analysis or design. (33, pp. 21)

An Intelligent Tutoring System (ITS) fits well into the OOD model that Coad and Yourdon propose. (31) The Multi-Layer, Multi-Component Model consists of five layers with four major components. The model has the same five layers as OOA; subject, class-&-object, structure, attribute and service. Figure 17 is an illustration of the components as vertical slices of the overall model.

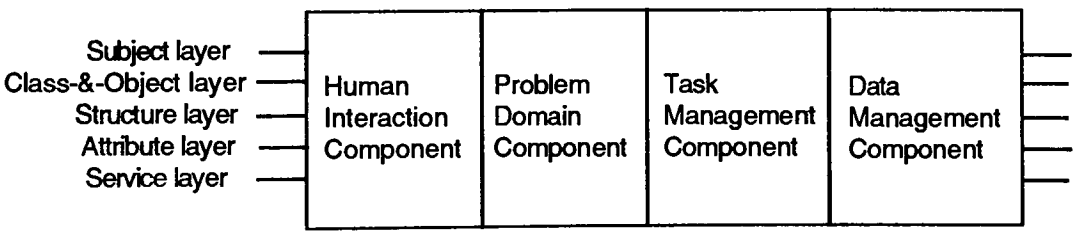


Figure 17: Four Components
(33, Figure 2.4 pp. 26)

There are also four activities. The four activities are: designing the problem domain component, designing the human interaction component, designing the task management component and designing the data management component.

OOA results are placed in the problem domain component (PDC). There is the potential need to manage combinations and division of certain OOA Classes-&-Objects, Structures, Attributes and Services. The splits are made using specific engineering criteria and/or tools needed to capture each decision. Criteria include reusing design and code classes, grouping problem-domain-specific classes together, establishing a protocol by adding a generalization class, accommodating a level of inheritance, improving performance, supporting storage management and adding lower-level detail.

The human interaction component (HIC) includes the displays and inputs that are needed for effective human-computer interaction. Classes will vary depending upon the graphical user interface being used. (e.g. Motif, Smalltalk Presentation Manager, etc.)

The task management component (TMC) includes program or task definition, program/task communication and program/task coordination. Hardware allocation considerations, external system and device protocols are also part of task management.

The data management component (DMC) includes access and management of persistent data. The data management approach is isolated in this component, whether it is flat file, relational, object-oriented or some other one. (33, pp. 21-26)

The key objectives to OOD are improved productivity, increased quality and elevated maintainability. Improved productivity can be accomplished by focusing effort on the up-front activity of software design. Increased quality can occur through analysis and design. Time spent on analysis and design helps to reduce coding errors and/or problems which will produce quality early in the development cycle. As requirements for a system are always changing, a design that is resilient to change provides for elevated maintainability. (33, pp. 14-17)

Following the recommendations made by Yourdon and Coad the process has

proceeded smoothly from analysis to design. As the analysis process progressed it was found that design decisions were being made. As stated previously, most of the OOA documents are placed in the problem domain component (PDC). The Interface module is conceptually part of the human interaction component (HIC). The Interface module is that part of an ITS that performs the human interaction role. The Instructor module is conceptually part of the task management component (TMC). The Instructor module performs the task of an ITS, which is to teach. The Expert module is conceptually part of the data management component (DMC). The Expert module contains the data that an ITS utilizes.

Figure 18 is actually a complete, unexpanded illustration of the OOD components that are created by an ITS. Each of these components has an expanded version in their appropriate component section. This figure is provided as part of the overall OOD process.

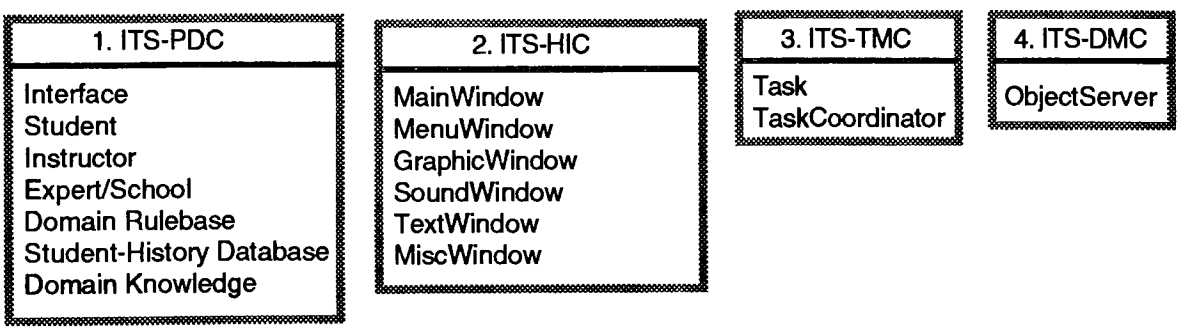


Figure 18: Intelligent Tutoring System, OOD Components

3.2 Problem Domain Component

The problem domain component (PDC) uses the strategy of applying OOA and then adding to and improving the results during OOD. (32, pp. 37) The diagrams and documents created in the analysis section are placed here according to Yourdon and Coad. Revised, final and/or improved diagrams are included in this section.

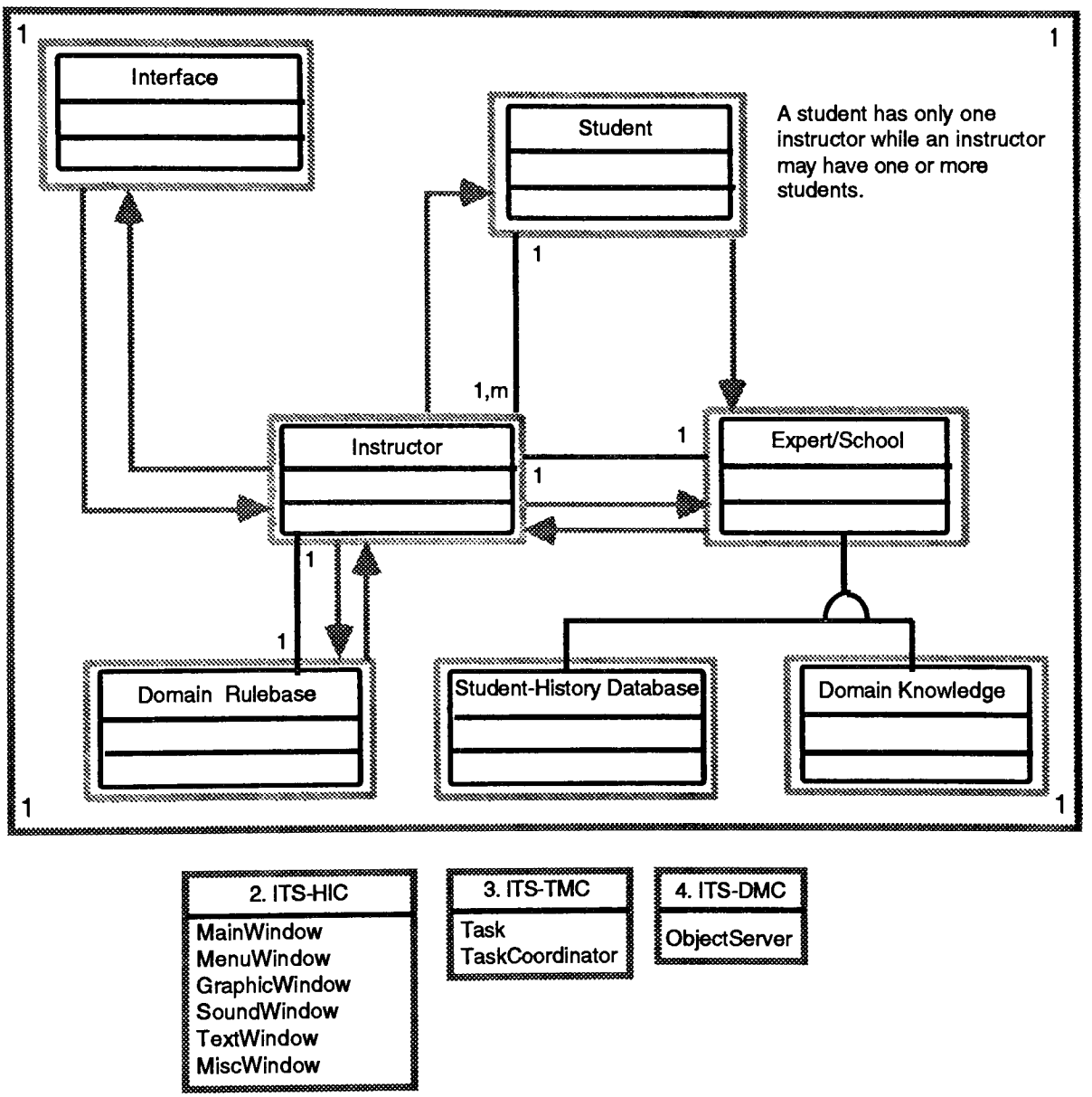


Figure 19: Intelligent Tutoring System, PDC expanded

Figure 19 is the expansion of the PDC component. There are similarities to the Class-&-Object, Attribute and Service layers diagram of Figure 12 on page 34. The

Attribute and Service layers are not included in this diagram due to space considerations. They are usually included in this diagram. The change from Figure 12 is the Domain Rulebase being placed within the Instructor module and removed from the Expert module. After initial analysis the Expert module had three sub-modules; the domain knowledge base, the domain rulebase and the student-history database. After further analysis the domain rule base was moved to the Instructor module. This follows the precepts of OOA where objects should accommodate real world views, such as the school analogy.

The change to Figure 12 that is reflected in Figure 19 caused some changes to the Expert module Class-&-Object, Attribute and Service layers diagram. The RuleData attribute and ProcessRuleRequest service from the Expert module have been incorporated into the Instructor module. Figure 16 has been replaced by Figure 20. The Instructors InstructionPlanning service now directly accesses the Domain Rulebase.

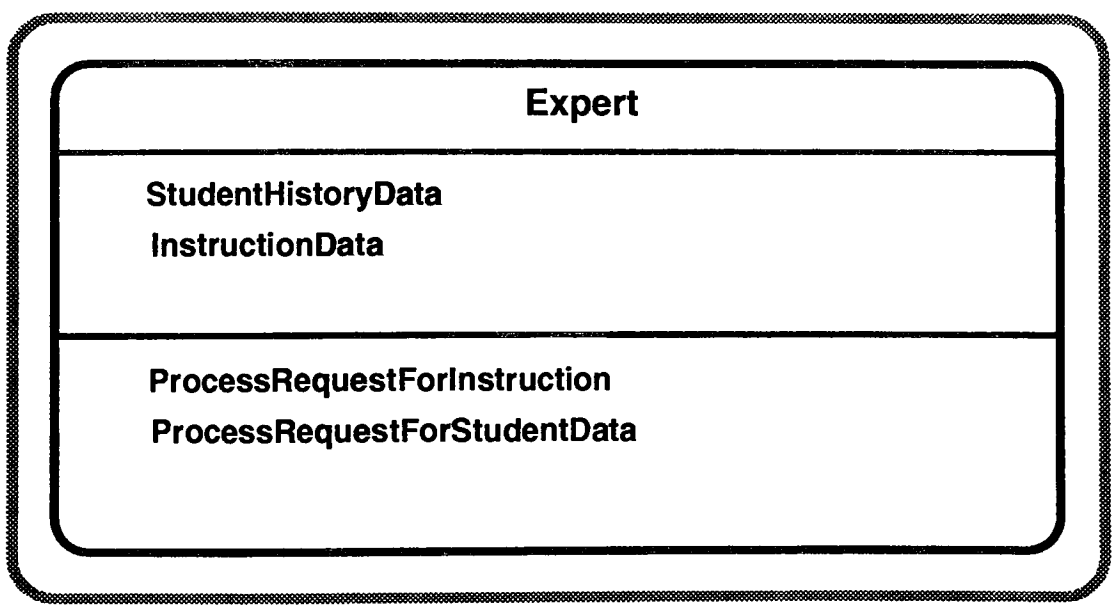


Figure 20: Expert Module - Class-&-Object, Attribute and Service layers

3.3 Human Interaction Component

The strategy for design of the human interaction component (HIC) uses the following steps: classify the users, describe the users and their task scenarios, design the command hierarchy, design the detailed interaction, continue to prototype (not applicable to this paper) and design the HIC classes. Classification refers to the identification of who will be using the system. The description of those users incorporates who they are and how they will be using the system. The command hierarchy may be presented, for example, as a series of menu screens, a menu bar or a series of icons that take actions when something is dropped onto them. The detailed interaction is based upon a number of criteria. These criteria include using consistent terms, minimizing the number of steps to complete a task, providing a sense of closure to the users actions, providing an undo capability and creating a look and feel that the user enjoys. (32, pp. 57-67) The design of the HIC classes will vary somewhat depending upon the graphical user interface that is used. A basic design will be provided for the purposes of this paper.

Figure 21 represents the expansion of the HIC component. These classes of windows are not completely defined. The person who implements this system can create as many window classes as needed.

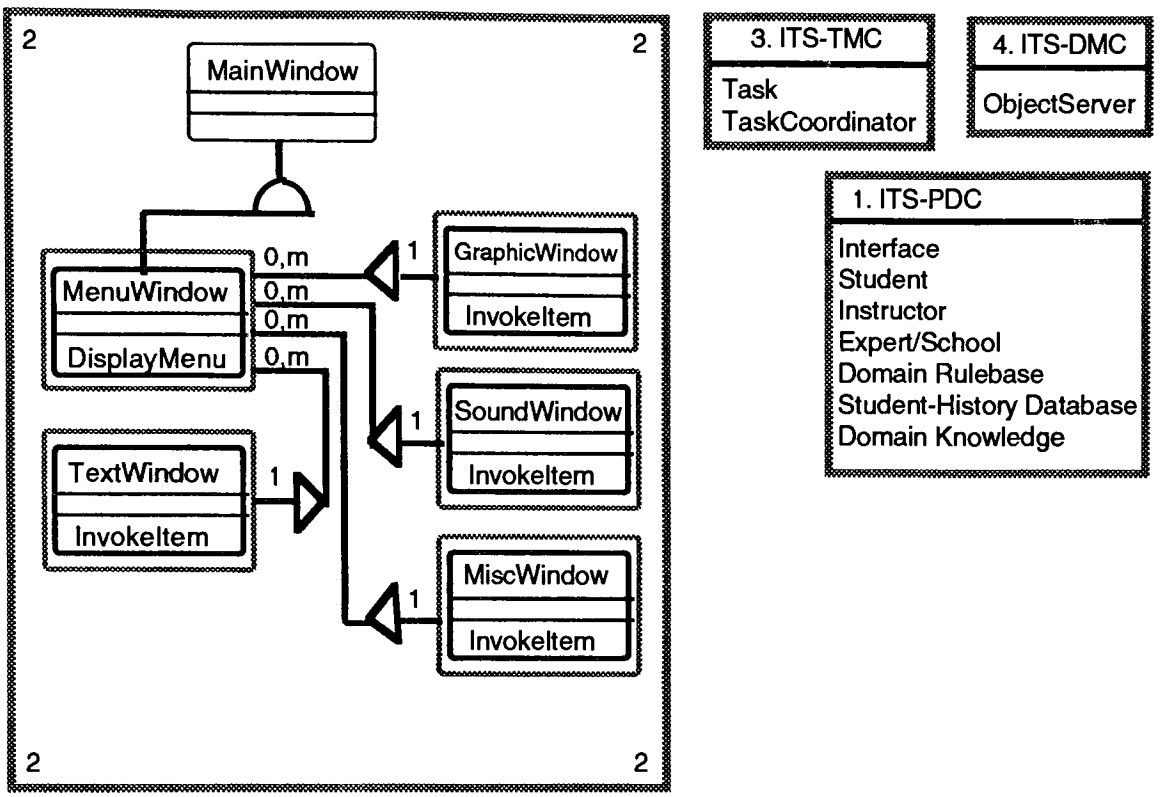


Figure 21: Intelligent Tutoring System, HIC expanded

The HIC class hierarchy is listed below:

MainWindow

MenuWindow

GraphicWindow

SoundWindow

TextWindow

MiscWindow

The user initially sees the MainWindow and is asked for their name. The MenuWindow allows the user to list the current topic list, to end the session and to select a new topic.

3.4 Task Management Component

The task management component (TMC) uses the following strategy: identify event-driven tasks, identify clock-driven tasks, identify priority tasks and critical tasks, identify a coordinator, challenge each task and define each task. (32, pp. 73) An ITS has only one task, to display an appropriate instruction to the student. It is an event driven task.

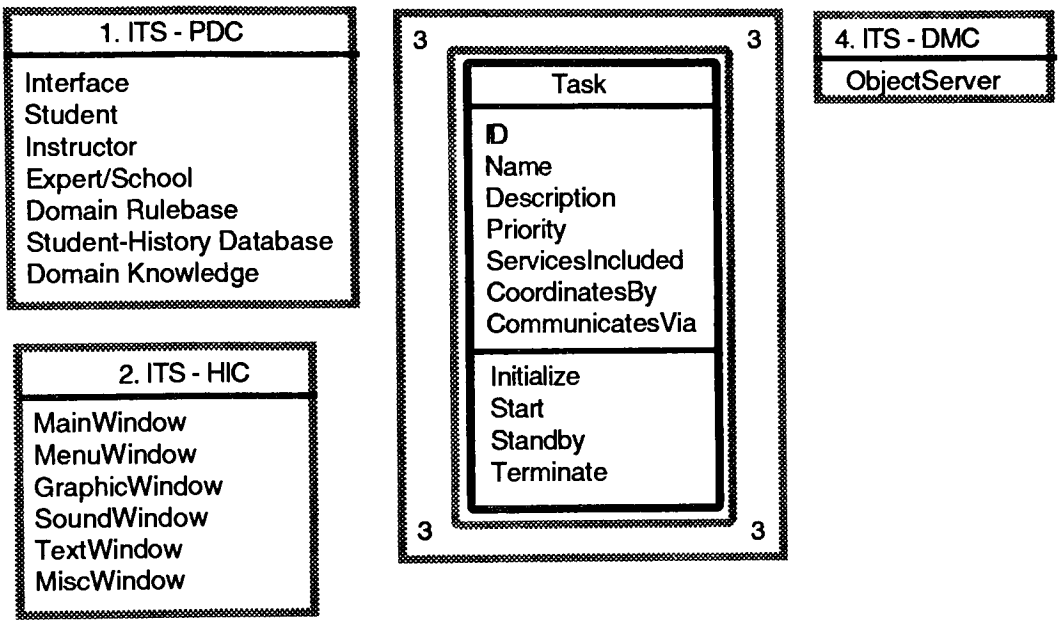


Figure 22: Intelligent Tutoring System, TMC expanded

Task 1

Name: DisplayInstruction

Description: This task is responsible for displaying the instruction selected by the instructor.

Services Included:

Interface.InputFromStudent

Interface.DisplayOfInstruction

Priority: low

Coordinates by: event driven, by human interaction

Communicates via: Gets values from the input buffer.

3.5 Data Management Component

The data management component (DMC) provides the infrastructure for the storage and retrieval of objects from a data management system. The purpose of this component is to isolate the impact of the data management scheme. The data management scheme could be flat file, relational, object-oriented or some other one. (32, pp. 80)

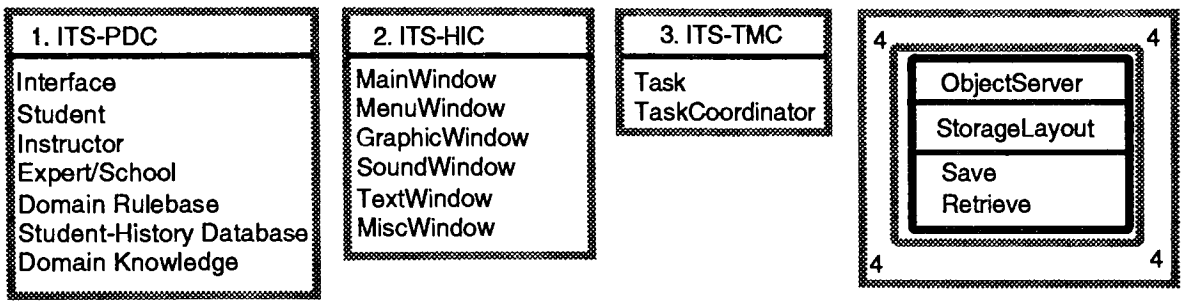


Figure 23: Intelligent Tutoring System, DMC expanded

Figure 23 is the expansion of the DMC component. The storage layout recommendation follows:

The data management for this system is object-oriented.

A separate table will be used to store information for each Class-&-Object.

Tables:

Student-History table

Domain Knowledge table

The data management for this system is object-oriented using the extended relational approach. Extended relational allows the storage of non-textual media such as, video, sound and graphical images. Third normal form tables would be used for textual information. Third normal form tables are table that have atomic values for attributes, one or more of the attributes is identified as the primary key and every nonkey attribute depends only on the primary key attribute(s). To establish third normal form tables, each class and its attributes are listed and the list is translated into third normal form. In a practical application of these steps, the performance and storage needs would be determined at this point. To remain within the theoretical

limits of this paper these implementation details are not set. Tables that would be needed for this system correspond to the knowledge domain, student-history data and the domain rulebase.

4. Conclusions

4.1 Problems Encountered

When this paper was started I had no knowledge of ITS or OOA/OOD. I did have some background theory in OOP which helped in understanding the knowledge that was acquired in learning OOA/OOD. Acquiring the knowledge necessary to complete this paper was the easiest segment to accomplish. There is more depth to the knowledge of ITS than there is for OOA and OOD.

This paper is theory oriented. There is no prototyping or programming involved. The lack of prototyping is a definite problem and limits the design aspect of this paper. The design of a system has a tendency to be intertwined with some form of implementation. This means that my design specifications are vague. Many of the specifications are necessarily implementation dependent.

The lack of prototyping is a shortcoming of the system. If prototyping had been included in this paper the design section would have been more complete. This shortcoming can be remedied by other students prototyping using the analysis section from this paper.

There are many implementation dependent aspects in doing design work on a system using OOD, such as what kind of displays would be used within the system, the completeness of the table specifications of the data management component and the vagueness of the classes within the human interaction component. Their could be as many classes as a designer can create.

4.2 Lessons Learned

An ITS fits very will into an OO paradigm because the system is made up of modules. The modules have classes, methods and interdependencies.

Performing analysis and design from a theory oriented perspective means that certain details are not able to be done. The analysis and design created in this paper

is not perfect. It would be preferable to see another student use these preliminary documents and expand on them to either implement or prototype the system.

Analysis and design should not be performed alone. A team work effort should be employed to perform analysis and design. What one person may miss another may not. This reduces the possibility of missing or incorrect aspects of the system. More than one persons' viewpoint is used to create the analysis and design documents.

4.3 Epilogue

The analysis and design documents within this paper are not as complete or correct as they could be. It would not be feasible to use these documents in a professional capacity. There is no prototyping and there are implementation aspects that would have to be dealt with for professional use.

The documents are appropriate for students to use. The level of detail is not high and leaves flexibility for the students to expand on what already exists. Another cycle or two of the analysis and design process are needed to create an analysis and design document that can be implemented. Prototypes could be created and the design details could be fully defined such as, the human interaction classes and the table definitions of the data management component. Any design decisions made in this paper are purely recommendations and are changeable by whomever takes this paper to the next step. The availability of an individual who has created an ITS for a reference source would go a long way to making these documents more accurate.

I learned a lot during the creation of this thesis. Not just about ITS or OOA/OOD but in how to coordinate analysis and design with the development of a system. This is my first attempt at formal analysis and design of a system. My usual capacity as a professional is to receive completed analysis and design documents and implement those decisions. The front end process of system development is not something that I have had experience with.

The most important lesson that I learned in writing this paper was to realize that analysis and design should not be done alone. A team of individuals should work together to create a system. At least one individual who would be using the system, at least one person who will be developing the system and at least one person who knows how to analyze problems (a systems analyst) should comprise the team. These individuals are needed to ensure that the system is fully developed in both usability and functionality. The most important aspect of any developed system is that the intended audience for the software will use it.

5. Glossary

Abstraction. The principle of ignoring those aspects of a subject that are not relevant to the current purpose in order to concentrate more fully on those that are.

Advancement. The use of an algorithm to determine whether to advance the student to the next curriculum topic.

Analysis. The practice of studying a problem domain, leading to a specification of externally observable behavior; a complete, consistent and feasible statement of what is needed; a coverage of both functional and quantified operational characteristics (e.g., reliability, availability, performance).

Artificial Intelligence (AI). The study of techniques and principles for applying computers to issues normally reserved for human intelligence.

Association. The union or connection of ideas.

Attribute. An attribute is some data or state information for which each Object in a Class has its own value.

Authoring system. A domain-independent component of an ITS that allows the developer to enter specific domain knowledge into the tutor's knowledge base.

Backward chaining. A pattern matching technique that tries to prove the condition part of rules whose actions match the conditions of proven rules. (See *forward chaining.*)

Bandwidth. The amount of the student's activity available to the diagnostic model. The three categories of bandwidth in ITS's, from low to high are: final states, intermediate states and mental states.

Black box expert system. A procedure that generates correct behavior over a range of tasks in the domain, but whose mechanism is inaccessible to the ITS. (See *glass box expert system.*)

Bug catalog or bug library. A set of well-analyzed and carefully collected patterns of typical errors. A user-expert difference model that generates bugs from fragments of valid rules.

Bug library technique. A user-expert difference model that represents misconceptions. It augments an expert model with a list of bugs.

Bugs. User misconceptions in declarative or procedural knowledge.

Case-Based reasoning. Problem solving based on a collection of individual experiences rather than general rules.

Causal stories. Causal stories, in troubleshooting contexts, are elaborate knowledge structures and narratives drawn from those structures, that relate observable evidence and symptoms to causes of faults through various models and knowledge about the device in question.

Class. A description of one or more objects with a uniform set of Attributes and Services, including a description of how to create new Objects in the Class.

Class-&-Object. A term meaning "*a Class and the objects in that Class.*"

Coach. A form of user modeling in which the ITS intervenes only when it is fairly sure that the user is doing something wrong. The intervention is with graduated hints and examples.

Coarse-grained user or student model. A user/student model that does not describe cognitive processes at a detailed level.

Cognitive fidelity. The measure of correlation between the cognitive model and actual human problem solving strategy.

Cognitive model. A representation of human cognitive processes in a particular domain.

Communication or interface module. The component of an ITS that specifies or supports the activities that the user does and the methods available to accomplish those activities. Also known as the environment of an ITS.

Computer Based Instruction (CBI). The use of computers for instruction and training. Generally this refers to instruction in which no expert system or production rules are used to order the sequence of information presented. It often results in linear sequences, or chains, of presented material. (See *microworlds.*)

Concept hierarchy. A graph of more and less general topics or ideas.

Curriculum module. The component of an ITS that selects and orders the material to be presented to the user/student.

Curriculum selection techniques. Techniques that deal with selecting problems to exercise those areas in the curriculum where the student is weak.

Daemons. Rules that actively wait for their conditions to become true and fire in dynamic systems.

Data Abstraction. The principle of defining a data type in terms of the operations that apply to objects of the type, with the constraint that the values of such objects can be modified and observed only by the use of the operations.

Decision tree technique. A diagnostic technique used in the user model that creates a tree of paths. Each diagnosis corresponds to a path from the root to some leaf.

Declarative knowledge. Knowledge represented as basic principles and facts of a domain. It is usually contrasted with knowing how to use facts or procedural knowledge.

Deep-level tutoring. Tutoring that can provide explanation of the internal reasoning of its expert module.

Design. The practice of taking a specification of externally observable behavior and adding details needed for actual computer system implementation, including human interaction, task management and data management details.

Direct manipulation. (See *first-person interface.*)

Divergence principle. A curriculum principle that states that there should be a broad representative sampling of exercises and examples in curricula for procedural tutors.

Dynamic systems. Complex mechanisms that require swift and effective interaction, so that instruction and tutoring must be terse and to the point, and more lengthy instruction delayed to a later debriefing.

Encapsulation. (Information Hiding) A principle, used when developing an overall program structure, that each component of a program should encapsulate or hide a single design decision... The interface to each module is defined in such a way as to reveal as little as possible about its inner workings.

Expert module. The module of an ITS that provides the domain expertise or the knowledge that the ITS is trying to teach.

Expert system. A computer program that uses a knowledge base and inference procedures to act as an expert in a specific domain. It is able to reach conclusions very similar to those reached by a human expert.

Explanation based simulations. Simulations or models whose design are predominantly driven by the need to provide explanations to students about device functions. Veridicality is subordinated to simplicity of explanations.

Expository tutor. A tutor that is concerned with declarative knowledge. Usually interactive dialogue is the instructional tool used in this type of tutor.

External evaluation. Evaluation of an ITS that focuses on the impact of the ITS on students' knowledge and problem solving.

External-internal task mapping problem. A problem in the human-computer interaction component of an ITS. It is a gap between what the user wants, the goal of the interaction and the actions the user must make to achieve the goal.

Fault diagnosis. A problem-solving technique used to uncover the source of system malfunction.

Fidelity. A measure of how closely the simulated environment in an ITS matches the real world. There are four kinds of fidelity: physical, display, mechanistic and conceptual.

First-person interface. An interface approach that provides visual simulations that can be altered to produce corresponding changes in the underlying symbolic representation. Actions and objects relevant to the task and domain map directly to actions and objects in the interface. An example of this type of interface is the icon.

Flat procedural knowledge. Procedural knowledge that is not organized by subgoals.

Forward chaining. A pattern matching technique that tries to prove the condition part of rules whose actions are then used to prove other rules. (See *backward chaining.*)

Glass-box expert system. An expert system that contains human-like representation of knowledge. This type of expert system is more amenable to tutoring than a black box expert system because it can explain its reasoning.

Graduated models. Qualitative models whose power and extension grow in some sort of correspondence with the capabilities of the user.

Grain-size of diagnosis. The level of detail used by the diagnostic technique for processing user/learner models. Closely related to *bandwidth.*

Hierarchical procedural knowledge. Procedural knowledge with subgoals.

Heuristics. Rules of thumb that are practical and often work, but are not based on a principled, theoretical understanding and therefore are not guaranteed to work.

Hypertext. A text-based system that goes beyond text to include graphics, video and sound (hypermedia) as well as links, cross references and hierarchical structures. It is interactive so that one word can be expanded on command into other media (hypermedia). The term was coined by Ted Nelson.

Individualization. A curriculum principle that states that exercises and examples should be chosen to fit the pattem of skills and weaknesses that characterize the student at the time the exercise or example is chosen.

Inheritance. A mechanism for expressing similarity among classes, simplifying definition of Classes similar to one(s) previously defined. It portrays generalization and specialization, making common Attributes and Services explicit within a class hierarchy or lattice.

Instruction. Actual presentation of the curriculum material to the user.

Instructional amplifier. A computer used to enlarge the scope and powers of teachers for instruction, that lets teachers personalize instruction more than they can now.

Instructional strategy. A general approach toward teaching or training, including objectives, plans and teaching style.

Instructional Systems Design (ISD). A systems engineering approach to the analysis, design, development, delivery and evaluation of instruction.

Instructor/tutor or tutorial planning module. The component of an ITS that infers and manipulates the user model. The selection of a diagnostic algorithm is dependent on the bandwidth of the system. Also known as a diagnostic module.

Intelligent Computer Assisted Instruction (ICAI). Synonym for Intelligent Tutoring System.

Intelligent Tutoring System (ITS). An advanced form of ICAI and CBI that tries to individualize instruction by creating a computer-based learning environment. It is a computer program that is capable of competent problem solving in a knowledge domain, can infer a learner's approximation of competence and is able to reduce the difference between its competence and the learner's through application of various tutoring strategies. Some sub-topics for ITS are knowledge representation, simulation, natural language, expert systems and induction.

ITS architectures. A systematic approach to structuring the many components that comprise an effective, working ITS. Usually these consist of a student model, an organized domain of knowledge, instructional principles and a tutorial interface.

Internal evaluation. Evaluation of an ITS that focuses on the relationship between the architecture of the ITS and its actual behavior.

Knowledge acquisition. The fundamental bottleneck in instructional design for informal systems: How does one acquire and organize the subject matter or knowledge base?

Knowledge base. Codified knowledge (usually represented on a computer) of a domain or subject matter.

Knowledge level analysis. An intemal evaluation method; it attempts to characterize the knowledge in the ITS and thus answers the question: *What does the ITS know?*

Knowledge representation. Computer-based techniques for storing and retrieving knowledge organized according to specific principles. Prominent techniques include frames, semantic networks and object-oriented techniques.

Manageability. A curriculum principle that states that every exercise should be workable and every example should be comprehensible to users who have completed previous parts of the curriculum. Manageability applies to procedural tutors.

Mental model. A popular theoretical construct for a knowledge representation form that supposes that people simulate their environments with models of the world that they are able to run in their minds. These runnable mental models can be used to predict the outcomes of thought experiments using novel conditions. Mental models can also be used to trace the causal connections of events and devices in the world.

Message. Any communication, written or oral, sent between persons.

Microworlds. Computer-based learning environments in which users are free to explore and discover the limits of their own understanding. The computer provides little direction or guidance, but it does narrow and constrain the topics for search to those that are valid within the current world. The environments can also raise sharply focused contrasts between alternative hypotheses about the world to facilitate insight and discovery.

Misconception. An item of knowledge that the user has and the expert does not have. A type of user-expert difference. A *bug.*

Missing conception. An item of knowledge that the expert has and the user does not have. A type of user-expert difference. (See *overlay model.*)

Mixed-initiative dialogue. A tutor and user engage in a two-way conversation utilizing the Socratic method of guided discovery.

Model-tracing. A diagnostic technique used to build a user model. It uses the user's surface behavior to infer the sequences of rules fired in a rule-based model of performance; that is, the user's actions traced a path through the rule base.

Object. An *abstraction* of something in a problem domain, reflecting the capabilities of a system to keep information about it, interact with it, or both; an *encapsulation* of Attribute values and their exclusive Services. (synonym: an Instance)

Overlay models. User modeling technique in which user performance is measured against the standard of an expert's model.

Path finding. A diagnostic technique used to find a path from one state to the next, which is a chain of rule applications. This is a way of representing the user's mental state sequence. The path is given to the model tracer.

Pedagogical. The adjective used to represent the study of ways of teaching or pedagogy.

Plan recognition. A diagnostic technique used in user models that represent hierarchical procedural knowledge. It is similar to path finding in that it is a front end to model tracing.

Predicate. A relation defined for a set of concepts.

Procedural knowledge. A form of knowledge representation distinct from declarative knowledge in which the knowledge is portrayed as active and functional. Production systems are sometimes viewed as a procedural form of knowledge to distinguish then from the organized declarative structures of semantic networks.

Procedural tutor. A type of tutor that teaches procedural knowledge and procedures. Usually exercises and examples are used by procedure tutors.

Process model. A model that reveals the mechanism behind behavior.

Production rule. A rule of the form, *condition-action pair*, used in modeling cognitive behavior. A set of production rules and an interpreter for processing them is termed a *production system.*

Propaedeutics. Knowledge that is needed for learning but not for proficient performance.

Problem Domain. A field of endeavor under consideration.

Qualitative models. A type of cognitive model where the computer-based simulation is composed of ordinal or even nominal metrics, such as 'good' and 'better', rather than higher-order mathematical models.

Rule-based model. An expert module of an ITS that is implemented with a rule-based (production) system. It is also known as a production model.

Second-person interface. A type of user interface where the user gives commands to a second party. Examples of this type of interface are command languages, menus and some natural language interfaces.

Semantic networks. A graph structure that links concepts with conventional links such as 'part-of', 'isa', 'instance', 'super', 'class', etc. Often seen as a declarative form of knowledge. (See *concept hierarchy.*)

Service. A Service is a specific behavior that an Object is responsible for exhibiting.

Situated learning. The context or situation of much expert activity directly supports learning the skills that the expert has. These skills are otherwise rarely invoked. The result is that learning by doing is cued and accelerated by the environment.

Step theory. A theory that states that the sequence of exercise and examples should reflect the structure of the procedure being taught and should thereby help the user induce the target procedure.

Structure. Structure is an expression of problem-domain complexity, pertinent to the system's responsibilities. the term "Structure" is used as an overall term, describing both Generalization-Specialization (Gen-Spec) Structure and Whole-Part Structure.

Subject. A Subject is a mechanism for guiding a reader (analyst, problem domain expert, manager, client) through a large, complex model. Subjects are also helpful for organizing work packages on larger projects, based upon initial OOA investigations.

Subject Matter Experts (SMEs). Experts that are knowledgeable in a domain and possess a fragmented, self-imposed organization of things that has considerable pragmatic value in dealing with everyday problems.

Surface-level tutoring. Tutoring that can be implemented with issue-oriented recognizers. Access to the internal reasoning of the expert module is not available.

System's Responsibilities. An arrangement of things accountable for, related together as a whole.

Target knowledge type. The type of knowledge that is represented in the expert and user model modules. Knowledge representation can be organized into three types: procedural (both flat and hierarchical), declarative and qualitative process model.

Tutorial domain analysis. An internal evaluation method for iteratively adding and subtracting requirements of the ITS design.

User Interface Management System (UIMS). A strategy that attempts to separate the interface component of an application program from the computational part.

User module. The component (a data structure) of an ITS that represents the user's current state of knowledge (mastery) of the domain. Various student modeling systems have been proposed: bug catalogs, overlay models, issue oriented models, coaching systems and psychometric systems.

User-expert differences. The difference between the expert's knowledge and the user's knowledge. There are two basic types of user-expert differences; missing conceptions and misconceptions. The three models used to represent user-expert differences are: overlay model, bug library technique and library of bug parts.

Web teaching. A curriculum approach where selection of materials is guided by two principles: relatedness, where priority is given to concepts that are closely related, and generality, where generalities are discussed before specifics. Web teaching applies to expository tutors.

Wizard-of-Oz system. Semi-automated tutors where a human tutor replaces some or all of the instructional functions of an automated tutor. Used in research and development of ITSs.

6. Bibliography

1. Aspillaga, M., "Implications of Screen Design Upon Learning", *Journal of Educational Technology Systems*, 1991, Vol. 20(1), 53-58.
Article on issues relating to how to make an interface effective. Somewhat useful.

2. Aspillaga, M., "Screen Design: Location of Information and Its Effects on Learning", *Journal of Computer-Based Instruction*, Summer 1991, Vol. 18, No. 3, 89-92.
Article on issues relating to how to make an interface effective. Somewhat useful.

3. Borsook, T. K. and Higginbotham-Wheat, N., "Interactivity: What Is It and What Can It Do for Computer-Based Instruction?", *Educational Technology*, October 1991, 11-17.
Article on interactivity with specific relation made to adaptive systems and simulations. Very informational but not useful until the actual thesis is underway.

4. Connop-Scollard, C., "The ID/D Chart: A Representation of Instructional Design and Development", *Educational Technology*, December 1991, 47-50.
An illustration of major concepts, theories, content areas and personalities in the field of instructional design and development. Very informational and helpful but not extremely useful at this point in my research.

5. Dijkstra, S., "Instructional Design Models and the Representation of Knowledge and Skills", *Educational Technology*, June 1991, 19-26.
Theoretical analysis of component parts relating to the design and knowledge representation for instructional purposes. Very good and easily understood. I wished it was longer.

6. Grandbastein, M., "Intelligent Tutoring Systems on Scientific Subjects: Are Prototypes Ready for Broad Experimentation?", *Computers & Education*, 1992, Vol. 18, No. 1-3, 63-70.
Covered current research topics for ITS's with some detail information on each prototype. Very useful.

7. Hamill, B. W., "Three Issues for Intelligent Tutoring", *Machine-Mediated Learning*, 1989, Vol. 3, 169-187.
Discourses on three specific issues within current ITS research: knowledge representation, instructional strategies and tutorial communication. Very useful.

8. Hendley, R. J. and Jurascheck, N., "CASCADE: Introducing AI Into CBT", *Computers & Education*, 1992, Vol. 18, No. 1-3, 71-76.
Informational article on application of artificial intelligence to computer-based training. Provided transitional data on going from CBT or CAI to an ITS. Useful.

9. Hofmeister, A. M., "Expert Systems and Decision Support in Special Education: Results and Promises", *Educational Technology*, October 1991, 23-28.
Article on implications of ITS use within decision support systems in special education. Provided information on the current usage status of ITS's in the real world. Useful.

10. Legree, P. J. and Gillis, P. D., "Product Effectiveness Criteria for Intelligent Tutoring Systems", *Journal of Computer-Based Instruction*, Spring 1991, Vol. 18, No. 2, 57-62.
An article on evaluation standards and procedures as applied (or not applied) to ITS's. Very useful.

11. Li, Z., O'Neill, Jr., H. F., and Baker, E. L., "Developing a Research Reference Interface for Knowledge-Based Instructional Design Tools", *Educational Technology*, August 1991, 7-16.
An article on the development of a prototype designed to be a knowledge-based tool that assists in the decision-making process of instructional design. Very interesting and useful.

12. Li, Z. and Merrill, M. D., "ID Expert 2.0: Design Theory and Process", *Educational Technology Research & Development*, Vol. 39, No. 2, 53-69.
Theory and design information on a prototype instructional design expert system whose purpose is to help develop instructional materials. Very interesting but only somewhat useful.

13. Looi, C., "Automatic Debugging of Prolog Programs in a Prolog Intelligent Tutoring System", *Instructional Science*, 1991, Vol. 20, 215-263.
Article on approach to debugging student written Prolog programs using an ITS written with Prolog. Very long article and unfortunately too specific to be very useful to me at this time.

14. Mandl, H., and Lesgold, A. (Eds.). (1988). *Learning Issues for Intelligent Tutoring Systems*. New York: Springer-Verlag.
Book on ITS learning issues. The chapters were written by cognitive scientists discoursing on how cognitive processes can be applied to ITS's. Extremely useful.

15. Milheim, W. D. and Martin, B. L., "Theoretical Bases for the Use of Learner Control: Three Different Perspectives", *Journal of Computer-Based Instruction*, Summer 1991, Vol. 18, No. 3, 99-105.
An article on user or learner control from three theoretical perspectives: motivation, attribution and information processing. Somewhat useful.

16. Murray, W. R., "Control for Intelligent Tutoring Systems: A Comparison of Blackboard Architectures and Discourse Management Networks", *Machine-Mediated Learning*, 1989, Vol. 3, 107-124.
An article comparing two types of control mechanisms for the tutor module of an ITS. This is an example of current research. Informational and useful.

17. Polson, M., and Richardson, J. (Eds.). (1988). *Foundations of Intelligent Tutoring Systems.* Hillsdale, NJ: Lawrence Erlbaum Associates Inc..
Book that describes itself as being a synthesis of information on the field of ITS's. It contains a collection of essays that impart information at the foundation level. Covers both theoretical and research issues. Extremely useful.

18. Psotka, J., Massey, D., and Mutter, S. (Eds.). (1988). *Intelligent Tutoring Systems: Lessons Learned.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc..
Book similar to Polson and Richardson but written from a more practical viewpoint. Deals more with maintenance, troubleshooting and programming issues than theoretical issues. Very useful.

19. Schaffer, J. W., "Harmony Coach: An Exploration of Microcomputer-Based Intelligent Tutoring Systems in Music", *Journal of Computer-Based instruction*, Winter 1991, Vol. 18, No. 1, 30-36.
Article about three tutorials designed to teach a specific music skill. More informational than useful.

20. Smeaton, A. F., "Using Hypertext for Computer Based Learning", *Computers & Education*, 1991, Vol. 17, No. 3, 173-179.
An article on the response to using a hypertext based system with computer based learning. Another transitional type of article going from CBT to ITS's. Somewhat useful.

21. Vassileva, J., "Pedagogical Decisions Within An ITS-Shell", *Computers & Education*, 1992, Vol. 18, No. 1-3, 39-43.
A discourse on a hierarchically organized method of domain knowledge representation for the tutorial component of an ITS. Very useful.

22. Wolz, U., McKeown, K. R., and Kaiser, G. E., "Automated Tutoring in Interactive Environments: A Task-Centered Approach", *Machine-Mediated Learning*, 1989, Vol. 3, 53-79.
Article on tutoring versus consulting. A consultant can dynamically adapt where a tutor cannot. Great article. Very useful.

23. Sleeman, D. and Brown, J. S. (Eds.) (1982). *Intelligent Tutoring Systems.* New York: Academic Press.
A foundation book for ITS. It provided a lot of background information and is referred to often by other researchers in the field.

24. Reiser, B. J., Anderson, J. R. and Farrell, R. B. (1985). "Dynamic Student Modeling in an Intelligent Tutor for LISP Programming". *Proceedings of the International Conference on Artificial Intelligence-85* (Vol. 1, pp. 8-14). Los Altos, CA: Morgan Kaufman.
An article providing specific information.

25. Nwana, H. S., "The Evaluation of an Intelligent Tutoring System", *Intelligent Tutoring Media*, 1990, Vol. 1, No. 3, 117-132.
Very specific article relating to evaluation criteria for ITSs'. Very informational and useful.

26. Ayel, M. and Laurent, J. (Eds.) (1991). *Validation, Verification and Test of Knowledge-Based Systems*. West Sussex, England: John Wiley & Sons, Ltd.
A source of evaluation methods that could be applied to ITSs.

27. Frasson, C. and Gauthier, G. (Eds.) (1990). *Intelligent Tutoring Systems: At The Crossroads of Artificial Intelligence and Education*. Norwood, New Jersey: Ablex Publishing Corporation.
A collection of papers from the International Conference of Intelligent Tutoring Systems of 1988 in Montreal, Canada. Useful.

28. Goodyear, Peter (Ed.) (1991). *Teaching Knowledge and Intelligent Tutoring*. Norwood, New Jersey: Ablex Publishing Corporation.
A collection of articles from professionals who are all involved in the study of teaching. Useful.

29. Sullivan, J. W. and Tyler, S. W. (Eds.) (1991). *Intelligent User Interfaces*. New York, New York: ACM Press.
A collection of articles on the challenges of creating interfaces that will complement the growing sophistication of software. Useful.

30. Van Der Veer, G. C. and Mulder, G. (Eds.) (1988). *Human-Computer Interaction: Psychonomic Aspects*. New York: Springer-Verlag.
A collection of articles concerning the different aspects of human-computer interaction and how those aspects can be fully utilized. Useful.

31. Coad, P. and Yourdon, E. (1991). *Object-Oriented Analysis*. Englewood Cliffs, New Jersey: Yourdon Press.
Essential reference book for object-oriented analysis. Extremely useful.

32. Bielawski, L. and Lewand, R. (1991). *Intelligent Systems Design: Integrating Expert Systems, Hypermedia, and Database Technologies*. New York, New York: John Wiley & Sons.

33. Coad, P. and Yourdon, E. (1991). *Object-Oriented Design*. Englewood Cliffs, New Jersey: Yourdon Press.
Essential reference book for object-oriented design. Extremely useful.

34. Koch, George (1990). *ORACLE: The Complete Reference*. Berkeley, California: Osbourne McGraw-Hill.
Used for relational database terminology.

35. Rogers, Ulka (1991). *ORACLE: A Database Developer's Guide*. Englewood Cliffs, New Jersey: Yourdon Press.
Used for relational database terminology.

7. Authors Note

The glossary is a compilation of my own definitions, those in reference 17, pages 531-534, those in reference 18, pages 261-266 and those in reference 31 pages 52, 53, 79, 106, 119, 143.