1 INTELLIGENT TUTORING SYSTEM

1.1 Introduction
The great variation in the quality of education exists due to various factors like social background of students, parents’ attitude and awareness, different standards of teaching and teachers training programs throughout in India [43, 53, 120]. Further the presence of great social diversity in India, it is difficult to change the social background of students, parents’ attitude and their socio-economical conditions [63, 120, 126]. Also increasing demand to acquire multi-disciplinary, multilingual knowledge and skills, both to teacher/tutor as well as student/trainer have led to increase pressure on academician to ensure transfer of adequate quality knowledge and skills [4, 18, 80]. This has increased the use of computer based education during the last two decades.

In the early 1960s, Programmed Instruction (PI) was aroused a new alternative to educational system [44, 96, 107, 122]. This kind of pedagogy can be related to any structured, goal-oriented instruction. According to Bounderson (1970), PI required the program designer to specify input and output in terms of entering skills and terminal behaviours of the learner.

In general, PI refers to any instructional methodology that utilizes a systematic approach to problem decomposition and teaching [21]. In some cases, PI was embedded in a computer program, known as Computer Assisted Instruction (CAI or Computer-Based Training, CBT). Some similarities between PI and CAI are that both have well-defined curricula and branching routines (intrinsic branching for PI, conditional branching for CAI).
A major distinction between the two is that CAI is administered on a computer. CAI also evolved from Skinnerian stimulus response psychology: “... the student’s response serves primarily as a means of determining whether the communication process has been effective and at the same time allows appropriate corrective action to be taken” (Crowder, 1959). In other words, at every point in the curriculum, the computer program evaluates whether the student’s answer is right or wrong and then provides right direction to student. Built-in remediation loops tutor/students who are attempting to answer a question incorrectly. If learners answer correctly, they are moved ahead in the curriculum. Figure 1.1 illustrates a typical flow of events in CAI.
As can be seen in Figure 1.1, there are several places where this simple model may be expanded to create more flexibility and, hence, render it adaptive to individual learners. For instance, various mastery criteria can be imposed, where subjects have to answer a certain proportion of items correctly before moving on. Failure to reach a criterion would force the student back into remediation mode (see “If incorrect” branch) where a different problem is presented, rather than the problem that caused the error.

To distinguish between simple versus more adaptive CAI (i.e., Intelligent Computer-Assisted Instruction (ICAI)) Wenger (1987) pointed out that actually there is no explicit demarcation between the two. Instead, there’s a scale, from linear CAI to more complex branching CAI, to elementary ICAI, to autonomous (or stand-alone) ICAI. This scale is often misconstrued as representing a worse-to-better progression.

All above approach is restrictive in that it does not consider the diversity of students’ knowledge states and their particular needs. [86]. Moreover, such systems are not more adaptive and are unable to provide individualized attention that a human instructor can provide [42, 65, 139, 116]. It recognizes the fact that knowledge is interrelated in many complex ways, and there may be multiple good paths through the curriculum. AI programming techniques empower the computer to manifest intelligence by going beyond what’s explicitly programmed, understanding student inputs, and generating rational responses based on reasoning from the inputs and the system’s own database.

However, for more complex knowledge domains, such as history, or the scientific debate over the extinction of dinosaurs, the complexity of
alternatives is beyond enumeration. And it is the complexity of this branching that really provides a qualitative break between older forms of PI and CAI and newer Intelligent Tutoring System (ITS). Not only the branching in ITS complex, it is also algorithmic and not enumerated, predefined, or hand crafted. With this qualitative increase in complexity comes a flexibility of interaction and potential for communication that, better than anything else before, begins to qualify for the word “intelligent”.

Hartley and Sleeman (1973) argued that ITS must possess: (a) knowledge of the domain (expert model), (b) knowledge of the learner (student model), and (c) knowledge of teaching strategies (tutor). It is interesting to note that this simple list has not changed in more than 20 years (see Lajoie & Deny, 1993; Poison & Richardson, 1988; Psotka, Massey & Mutter, 1988; Regian & Shute, 1992; Sleeman & Brown, 1982).

The definition given by Katie Hafner (2000) is that, an ITS is educational software containing an Artificial Intelligence (AI) component. The software tracks student’s work, tailoring feedback and hints along the way. By collecting information on a particular student’s performance, the software can make inferences about strengths and weaknesses, and can suggest additional work.

Today, Web Based interactive distance learning technologies and its more general synonymous term e-Learning are two of today’s buzz-words in the academic and business world [71, 75]. These buzz-words allow students to better control the process of learning because they can decide when, where and how fast to learn and are more cost effective than traditional learning strategies [2, 76, 91]. Web-based learning environments are becoming very popular where an array of media resources are employed to facilitate
learning and provide channels of communication between the learners and tutors [2, 10, 65, 76].

Typical web-based learning environments, such as Virtual-U, Blackboard, WebAssign, MUVE, QuizPACK and Web-CT, include course content delivery tools, synchronous and asynchronous conferencing systems, polling and quiz modules, virtual workspaces for sharing resources, white boards, grade reporting systems, logbooks, assignment submission components, etc. [17, 27, 36, 70, 156, 159].

All such systems can appear to imitate intelligence by being able to adapt to student misconceptions. However, result of the system designer anticipating all possible errors that the student may make. These are built into the system at design time and encoded within the branching structure. If the designer of the system has not anticipated an incorrect interpretation of the instructional material then the system will not be able to provide feedback to help the examiner resolve the misunderstanding.

Such systems are incapable of dynamically generating a response to a particular situation as a human tutor would be able to do. Further, for a computer based educational system to provide such attention, it must implement more advanced algorithms to parse the knowledge domain and related learner feedback.

During last few years there have been various specialized tutoring systems developed in a particular knowledge domain (e.g. PROUST, MENO-II and ELM-PE developed to teach programming) [71, 148, 162]. Until now the majority of such ITSs are specialized in a specific knowledge domain. For example, Kumar’s model-based tutor asks students to predict C++ programs’ output and identify semantic and run-time errors [100]. It provides explanations of program execution line by line to help students
understand code behaviour. Adaptive navigation based on student modelling is used in a web-based system called ELM-ART II to provide individualized annotated hyperlinks and curriculum sequencing [163].

Expert critiquing systems provide useful feedback to users’ work in many domains. Such feedback includes alerts to problematic situations, relevant information to the task at hand, directions for improvement, and prompts for reflection [143]. One of these systems is “Java Critiquer” that uses an incremental authoring approach to amortize the high development cost [125]. Other systems like “CTutor”, “Prolog Tutor” and “Java Intelligent Tutoring System” adopts advanced code analysis to see what is the intention of student and give feedback based on this analysis [64, 118, 155].

There are also tutoring system based on natural language like CIRCSIM-Tutor that is a language-based intelligent tutoring system for first-year medical students to learn about the reflex control of blood pressure [33]. Students solve small problems and are tutored by dialog with the computer. Despite of the great variety of ITS in the literature researcher chose to focus attention on tutoring system that define a generic architecture.

It is possible, in principle, to inject knowledge base of every kind in this type of system so that the process of learning can start in whatever domain of knowledge. This choice will enable us to develop a system for knowledge acquisition that tries to be independent from the specific domain in which an ITS can be developed.

In the next sections, after a brief introduction to the ITS, researcher has discussed the two major systems that are state-of-the-art for these generalist ITS architectures: Constraint Based Tutor and Cognitive Tutor.
1.2 Intelligent Tutoring Systems: An Overview

The developments of ITSs have to be seen in the context of AI and cognitivist educational theory. Until very recently, workers in the AI community have performed the majority of work on ITS with little interaction with educational researchers. Despite of a broad variety of the developed systems, an explicit and exhaustive definition of an ITS still does not exist. However, it is possible to list the most often mentioned characteristics of systems of such kind [12, 16, 26, 27, 38, 52, 129].

D. R. Benyon and D. M. Murray (1993) defined ITS as it is an adaptive system as it alters aspects of its structure, functionality or interface for the particular user and his/her changing needs over time [13]. It emulates a human teacher, tries to provide benefits of individual (one-to-one) tutoring, and is based on the theory of learning and cognition.

According to P. Brusilovsky and C. Peylo (2003) it is an intelligent system because it uses principles and methods of AI such as knowledge representation, inference mechanisms and machine learning in its structure and operation [23].

Jianping Zhang (2005) defined ITS as it is a computer-based instructional system with models of instructional content that specify what to teach, and teaching strategies that specify how to teach.

The Association for the Advancement of Artificial Intelligence (AAAI) (2010) defined ITS as, An intelligent tutoring system is educational software containing an artificial intelligence component. The software tracks students’ work, tailoring feedback and hints along the way. By collecting information on a particular student’s performance, the software can make
inferences about strengths and weaknesses, and can suggest additional work (AAAI, AI Topics/Intelligent Tutoring Systems).

Furthermore, ITSs are characterized by the fact that they store three basic kinds of knowledge: domain knowledge, knowledge about learners, and pedagogical knowledge [27, 49]. An ITS, as any other software intensively communicating with users, needs a part of the architecture responsible for the interaction between the system and the learner. It is a communication module or interface which controls screen layouts, interaction tools, etc.

However, each system can contain additional components; the presence of which depends on the factors such as features of problem domain, locking down of separate functions of the basic constituent parts in the isolated components of the structure, technology used for system implementation, and additional functional capabilities of the system. The general architecture of an ITS is shown in Figure 1.2.
Figure 1.2: General Architecture of Intelligent Tutoring System

The interaction scheme between these five modules is shown in Figure 1.2. Knowledge representation and tutoring methodologies are areas suitable for the application of intelligence. Typically this kind of systems can be seen as a number of independent components which are discussed here.

1.2.1 Student Module

The student diagnosis module has the function of capturing the level of a student’s understanding of the domain knowledge. It keeps track of information that is specific to each individual student, such as his mastery or competence of the material being taught, and his misconceptions. In effect, it stores the computer tutor’s beliefs about the student. All these information are used by the pedagogical module to tailor its teaching to the individual needs of the student.
This module concerns to identifying a student’s current state of understanding of the subject domain which is commonly known as student modelling, that is how to store the current student knowledge. This information is very often related to the domain knowledge as its subset. Student modules can be classified according to the functions they can perform. Following are the six main functions of student modules [120]:

- Corrective: Feedback intended at repairing a misunderstanding of the student. In this case, the module must identify a difference between the student’s understanding and the correct knowledge, and provide this information to other parts of the system.

- Elaborative: Extending the knowledge of the student. In this case, the module should identify areas where the student can be introduced to new material or a refinement of her current understanding.

- Strategic: Changing the approach to teaching at a higher level than local tactics. This requires the student module to provide more general information about the student, such as her success rate with the current teaching strategy as opposed to a previous teaching strategy.

- Diagnostic: Analysis of the state of the student. In some sense, all aspects of student modelling are diagnostic. What is meant here is the explicit use of the student module to refine information about the student in order to make a decision. If, for example, the tutor wishes to introduce a new topic, but the student module is unable to indicate whether the current level of understanding of the student is adequate, the module can be requested to generate diagnostic examples which can be presented to the student.
• Predictive: Using the module to anticipate the effect of an action upon the student. This requires the student module to act as a ‘simulator’ to simulate the behaviour of the student, given a particular action.

• Evaluative: Providing an assessment of the level of achievement of the student. This requires the system to make some aggregation across the information that it has.

Student modules can also be classified by their modes of interpretation: process or state models. Process models are capable of simulating the process by which the learner solves a problem and can therefore perform the predictive function of student modelling. They are also called executable or runnable models.

A student module is executable if its present state can be utilised by a certain interpreter to simulate the behaviour of the modelled student when he is solving a problem. Cognitive tutors [26], which will be presented in next section, use this kind of student modelling. On the other hand, state models do not have the capability of simulating and contain only state information. Examples are Constraint-based Tutors [105].

1.2.2 Domain Knowledge and Expert Module

A crucial aspect in the development of an ITS is how related knowledge is represented and how reasoning for problem solving is accomplished. The domain module contains a representation of the information to be taught. It provides input into the expert module, and ultimately is used to produce detailed feedback, guide problem selection/generation, and as a basis for the student module.
The domain module may take many forms, depending on the knowledge representation used, the domain it represents, and the granularity of the information being presented. For example domain knowledge can be stored at the page and topic level, and provides basic information about the content of the page, which aids in problem selection and course sequencing. In Cognitive tutors [79, 105], the domain module consists of low-level production rules that completely describe the expected student behaviour down to atomic thought components.

Simulation-based systems, use the domain module to describe how each component of the simulation should behave (i.e. what actions are possible with this object, and what the consequences of each action should be), and how components are interrelated [130]. Constraint-based systems describe the possible valid states that an answer may occupy.

The expert module uses the domain knowledge to advise other parts of the system. It may indicate the relative difficulty of curriculum sections or problems, such that the pedagogical module can select the next task. In Cognitive tutors it identifies whether or not the student’s current solution is on track and, if not, assess what has gone wrong. It may also be able to run the domain module to solve the problem from a given state. In constraint-based systems it evaluates the student solution against the constraints to determine what subjects have been misunderstood. More about Cognitive tutor has been discussed in next topic.

1.2.3 Pedagogical Module

The pedagogical module uses information from the student module to determine what aspects of the domain knowledge should be presented to the learner. This information, for example, may be new material, a review of previous topics, or feedback on the current topic. One pedagogical concern for an ITS is the selection of a meta-strategy for teaching the
domain. Once the meta-strategy is selected, low level issues, such as the exact example to use, can be decided. The tutor must decide the content of the material to be presented to the student. This involves decisions about the topic, the problem, and the feedback.

To select a topic to present, the tutor must examine the student module to determine the topics on which the student needs to focus. Many possibilities exist for the most appropriate topic on which a student should work. For example, if the meta-strategy indicates that review is in order, the tutor will select a topic the student has already learned. On the other hand, if new information is to be presented, the tutor will choose a topic that the student does not yet know.

Once the topic has been selected, a problem must be generated for the student to solve. The grain size of the problem is determined by the domain knowledge representation. For example the student can be asked to deduce some facts from a simulation of the world concerning the domain, or the student can be given a simple problem, such as adding two fractions. Whatever the granularity of the problem generated, it is important that the difficulty be appropriate for the student’s level of ability, which can be determined from the student module.

Most of the tutors work smoothly as long as students get everything right. Problems arise when the student has difficulties and needs help from the tutor. In these situations, the tutor must determine the kind of feedback to provide. The issue of how much help to provide the student is also a very complex issue as too little feedback can lead to frustration and floundering while too much feedback can interfere with learning.

Once the system decides how much feedback to give, it must determine the content of the advice. The feedback should contain enough information
so that the student can proceed to the next step in solving the problem. Furthermore, the advice given to the learner should be appropriate for her ability level. For example the more proficient the student is at a particular skill, the more subtle the hint is. On the other hand, a student with low proficiency in a skill would be presented with a more obvious hint.

High level strategy selection in ITSs has proven a formidable problem. Most ITSs do not explicitly identify the strategies they are using for teaching and implicitly implement a hard coded strategy. A better method is to use the student module to select an appropriate strategy from those maintained by the system. Ideally a student’s module could track the instructional strategies that are most effective for teaching him.

1.2.4 Communication Module

The last module is the interface or Communication Module which has the role of a communication bridge between the ITS and the student/learner. The interface becomes a very important part of an ITS. Natural communication between the system and its users should be generated to learn, educate and evaluate implicitly and naturally to produce more convenient interaction between human and computer [140, 171]. This believes that user interface has a great impact on student motivation.

Communication Module controls the communication between the learners, teachers, and the system as well as follow-up the behaviours among them. It is capable to send and receive messages using certain standard. It is also responsible for receiving the queries from agents, passing them and forwarding the content to other module.

The user interface design is based on a presentation format implemented in many popular Integrated Development Environments. Upon connecting to the teacher/tutor website, the student’s browser displays the working
environment for the teacher/tutor. An appropriate skill-level problem is selected by the expert agent module or the problem that last attempted is presented to the student.

1.3 Cognitive Tutor

Cognitive tutors are based on the ACT-R theory of mind. Because the tutoring effort is so strongly tied to the ACT theories of skill acquisition (initially the ACT* theory; J. R. Anderson, 1983; and now the ACT-R theory; J. R. Anderson, 1993), it is worth reviewing the principal tenets of that theory. The central principle of this theory is that the processes of thought can be modelled using declarative and procedural knowledge.

Declarative knowledge corresponds to things are aware to know and can usually describe to others. Examples of declarative knowledge include “Dr. Rajendra Prasad was the first president of the Republic of India” and “The halting problem is not computable”. Procedural knowledge is knowledge which we show in our behaviour but which we are not conscious of. For example, usually no one can describe the rules by which we speak a language but we can do it. In ACT-R, declarative knowledge is represented in structures called chunks whereas procedural knowledge is represented in productions. Thus chunks and productions are the basic building blocks of an ACT-R model.

Tutoring is achieved using a method known as model tracking. As the student works at the problem, the system traces his progress along valid paths in the model. If he makes a recognisable off-path action, he is given an error message indicating what he has done wrong, or perhaps an indication of what he should do. If the action is identified as being off-path but cannot be recognised, he may only be told that it is incorrect, but not why.
Knowledge tracing is used to monitor the knowledge that students have acquired from problems. A Bayesian procedure is used to estimate the probability that a production rule has been learned after each attempt. This assessment information is used to individualize problem selection and optimally route students through the curriculum.

1.3.1 PAT: Algebra Tutor

The PUMP (Pittsburgh Urban Mathematics Project) Algebra Tutor (PAT) was originally developed by the Pittsburgh Advanced Cognitive Tutor Center at Carnegie Mellon University with support from NSF, Darpa, and foundations in Pittsburgh. PAT is a cognitive tutor that modelled algebra problem solving and student’s path towards a solution. PAT is based on ACT-R, based on cognitive architecture that accommodates different theories [78]. ACT-R models problem solving, learning and memory and integrated theories of cognition, visual attention, and motor movement.

The main purpose of the PAT Tutor is to help students to develop algebraic skills which they can use in the context of real-life problem situations. Students work through PAT problem situations by reading its textual description and a number of questions about it. They investigate the situation by representing it in text (Verbal Representation), tables (Discrete Representation), graphs (Graphical Representation), and symbols (Algebraic Representation) and using these representations to answer the questions. The major focus of the tutor is helping students to understand and use multiple representations of information.

Students are presented in PAT Tutor screen, divided in four windows (First: (Upper Left): Problem Statement, Second: (Lower Left): Worksheet, Third: (Upper Right): Grapher, Fourth: (Lower Right): Equation Solver), each containing a tool for solving the problem (in addition to the window showing the current text of problem). While students work on a problem,
tutor traces their activities and if something goes wrong (student makes error), the system shows the relative feedback in various form, depending on the type of error.

An experiment was conducted on 470 students using this system and the experimental classes outperformed students in comparison classes by 15% on standardized tests and 100% on tests targeting the PUMP objectives. This was helped individualizing instruction and targeting each student’s strengths and weaknesses, intelligent tutors can maximize both the student’s and the teacher’s use of time in the classroom. Students are shown immediately whether their actions will be successful in constructing a solution. They can more easily focus on correction of errors and development of skills that they find difficult.

1.3.2 LISP Tutor

The LISP tutor based on Cognitive tutor [2], was a WWW tutorial that helps the student in the process of learning the LISP language. In ACT* (Anderson 1983), learning is accomplished by forming new procedures through the combination of existing production rules. The student is given a description of a small program to encode in LISP, which he then writes with the system’s help.

In the LISP Tutor the expert model was created as a series of correct production rules for creating LISP programs and a learner model was built as a subset of these correct production rules along with common incorrect ones. The tutor acts as a problem solving guide but never states the productions to be learned.

Students interact with a language editor that shows the structure of program to be coded. As the user builds his solution, the system inserts
tags that describe the general form of the program, for example (user input in bold):
(defun fact (n)
(cond ((equalp) <ACTION>)
<RECURSIVE-CASE>))

In the above case, the student was asked to code the program that compute the factorial of a given number. At this point the system interrupts the student and asks for the use of ZEROP function, that tests a number against zero, instead of the general EQUALP. Then the tutor interacts with the student in similar manner until the final solution is reached.

The LISP tutor performs very well. In an initial mini-course at CMU, students using it solved a series of exercises in 30% less time than those in a standard LISP environment, and performed one standard deviation better on their final test. However, later evaluations failed to conclusively prove that the style of teaching used improved performance.

The main reason for improved performance in post tests previously was probably because the LISP tutor enabled students to cover more exercises in the same amount of time, which subsequently gave them an advantage. However, this is, in itself, considered a worthwhile outcome, since enabling students to achieve their learning in less time gives them more time to learn additional material, or to do other things.

1.4 Constraint-Based Tutor
These kind of ITS is based on Ohlsson's theory of learning from performance errors. Ohlsson S. (1994) addresses Constraint-based Modelling (CBM) as it is a student modelling approach that somewhat eases the knowledge acquisition bottleneck by using a more abstract
representation of the domain compared to other commonly used approaches. CBM focuses on faulty knowledge, realizing that it is not sufficient to describe what the student knows correctly.

The basic assumption is that diagnostic information is not hidden in the sequence of student’s actions, but in the problem state the student arrived at. This assumption is supported by the fact that there can be no correct solution of a problem that traverses a problem state, which violates fundamental ideas, or concepts of the domain.

1.4.1 **SQL-Tutor**

SQL-Tutor is a knowledge-based teaching system which supports students learning SQL. The intention was to provide an easy-to-use system that will adapt to the needs and learning abilities of individual students. The tailoring of instruction is done in two ways: by adapting the level of complexity of problems and by generating informative feedback messages.

It consists of definitions of several databases, and a set of problems and the ideal solutions to them. The student are presented a question asking to write an SQL statement, a description of the database schema and the structure of the query consisting of all clause SELECT, FROM, WHERE, ORDER BY, GROUP BY, HAVING. The graphic interface can be seen in Figure 1.3. When the student submits the solution, it is passed to the system that corrects errors, updates the student model and selects the next action (i.e. feedback or another question). In the early version the system had several limitations:
The domain admits more than one solution to a given problem but the system was unable to recognize a different solution. This translates in misleading feedback that can confuse the student. This problem has been changing the representation of constraints reducing it to simple pattern matching and developing a problem solving procedure able to produce a piece of solution from a constraint [52]. The partial solution replaces a piece of the student solution that violates the constraint and the procedure is iterated until no more errors exist.

System had a limited problem set. Authoring a set of questions that cover all the defined constraint (over 500) is a very difficult task. Automatic problem generation in constraint-based tutors have been developed a method, based on problem solving capability described in the previous point that generates a set of problem starting from the constraint set defined in the system [23]. The researcher has to manually transform
the generated solutions into natural language for presentation to the student.

Authoring a constraint based tutor is a task reserved to ITS engineers or expert with a necessary knowledge to represent constraints. In a system called WETAS was developed to help the process of system authoring and WETAS was augmented with the capabilities of extracting constraints from ontology [152].

1.4.2 NORMalization Intelligent Tutor (NORMIT)

NORMalization Intelligent Tutor (NORMIT) is first in the series of constraint-based tutors developed at Intelligent Computer Tutoring Group (ICTG) that teaches data normalization, which is a procedural task. NORMIT is a problem-solving environment, in which students can improve their skills. This system helpful for students would already be familiar with the data normalization theory from lecture, but the system does provide some support for acquiring domain knowledge.

NORMIT requires the student to determine candidate keys, the closure of a set of attributes, prime attributes, simplify functional dependencies, determine normal forms, and, if necessary, decompose the table. The sequence is fixed: the student will only see a Web page corresponding to the current task.

NORMIT is an adaptive system based on a centralized architecture, as many other existing Web-enabled ITSs (e.g. ELM-ART [120], AST [136] and SQLT-Web [109]). The tailoring of instruction is done in two ways: by adapting the level of complexity of problems and by generating informative feedback messages. Student models are kept on the server, and all tutoring functions are also executed on the server.
The amount of information that needs to be transferred from the browser to the server is not large, and we believe that such architecture is appropriate. NORMIT is developed in AllegroServe, an extensible Web server that allows the components of the system to be developed in Lisp. A special component of the system called the session manager ensures that a student’s actions are associated with her/his student model, thus enabling the system to be used by multiple students simultaneously.

1.5 The ITS Challenges
Learning, educating and evaluating is fundamental. One-to-One instruction by trained human tutors succeeds in helping learner to teach, but is expensive and sometimes unavailable. Efforts to duplicate the effects of one-on-one tutoring in large group settings have typically not matched the performance of human tutors.

However, it is difficult to make a fair online evaluation of how well the students understanding. There are several disturbances for realizing fair grading such as duplication of answers between the students or illegally pretending to be other persons to answer the exam.

An online question bank and examination system is a relatively new and rapidly expanding system. Although it is an effective solution for mass education evaluation, the fairness of the evaluation is still a big concern. Most of the present systems were designed to grade students based on how well they have done on their examination. These systems were designed with the subject of traditional paper based examination in mind. There is a need to use a range of different assessment methods, in order to prevent assessment being biased against students that have particular problems with one particular method.
Another drawback of present systems is that there is no flexibility and there are very limited options for the examination questions. Most of the systems were designed to deliver and mark multiple choice questions. These systems will not precise enough to represent the knowledge of individual users and to select problem to extend the user’s current level of understanding. On top of that, the definition of the level of difficulty for examination question often creates an argument.

Researcher does not have a clear mechanism to define the level of difficulty for each question. Hence, there is a need to come out with a system that can base on question difficulty assessment algorithm to determine the level of difficulty for each question.

Researcher has begun to create a development environment that addresses these difficulties. Our goal is to make tutor development both easier and faster for current developers and possible for researchers, teacher/tutor, student/trainer/learner who are not experts in cognitive psychology or AI.

To overcome above discussed problems, researcher has begun to create a development environment that addresses these difficulties. One of the most crucial and important options is to provide an individual or group of learner through standardize teaching learning environment using ITS. Researcher focus to create richer types of answers along with annotating the pedagogical material using metadata for facilitating its reusability to the teacher/tutor, student/trainer/learner if anything wrong or go right.

1.6 The Objective
This research is concerned with the design and construction of a heterogeneous environment, development in an open architecture and implementation of cost-effective and more secure ITS to teach procedural
knowledge as well as facilitating the acquisition of conceptual knowledge, in multiple subjects.

1.7 Statement of the Problem

By the mid-1980’s, much of enthusiasm in AI for creating “thinking” computers had declined as the field began to mature. Researchers turned to the more ordinary tasks of building expert systems that could function well in constrained domains, such as troubleshooting and diagnostic systems. At the same time, as ITS began to move out of the AI laboratories into classrooms and other instructional settings, they began to attract critical reactions. Some shortcomings of ITS became apparent as researchers realized that the problems associated with creating ITS were more intractable than they had originally anticipated.

Based on these concepts researcher has proposed to develop Intelligent Tutoring System as Human Knowledge Discovery Agent (HKDA) that makes good use of available resources. The title of the present study is:

“Standardization of Intelligent Tutoring System for Adaptive Information Retrieval and Knowledge Discovery by Monitoring Human Interaction”

Researcher attempted to develop various intelligent agents, who allows individual to interact with the system by asking some set of questions and start monitoring and diagnosing based on the user’s responses. They keep track many traits of a human being like knowledge, interest, pattern of answering, lack of awareness and so on. The system should shown enough intelligence to change the way questions should be asked to the individual based on the outcomes.
This thesis will explain what can be expected from ITS and although especially focused on the technological basis, will examine all kinds of requirements for ITS environments: pedagogical, functional and non functional requirements.

1.8 Conclusion
In Chapter 1, researcher has given an overview of what an Intelligent Tutoring System is, shown its main components that are student module, domain knowledge, pedagogical module, expert model and communication module. Researcher then reviewed the state-of-the-art ITSs which are Cognitive Tutors and Constraint-based Tutor and describe some instances of this type of tutor. Finally researcher discussed the ITS challenges and focus on how to overcome from this issues.