Case-Based Instructional Planning for Learning in a Context

Kai Warendorf, Haiping Xu and Antoon Verhoeven
School of Applied Science
Nanyang Technological University
Singapore 639798
Email: {warendorf, haiping, antoon}@ntu.edu.sg

Abstract

This paper describes an architecture of Intelligent Multimedia Tutoring System, called IMTS, which provides an environment that facilitates students to learn the course of Data Structure in a context. Three main teaching strategies on the basis of cognitive apprenticeship are discussed, which are program generation strategy, program completion strategy and program understanding strategy. To generate contexts for learning, we also proposed a structure of case-based instructional planner, in which IMTS can make instructional plans for a particular student by reusing his past teaching experience.

1. Introduction

Deeply embedded in research for ICAI (Intelligent Computer Aided Instruction) is a profoundly misleading belief, that an intelligent tutor can replace the human teacher. Although such an intelligent tutor is ideal, it is not realistic, at least for the present, for it depends on the ability of natural language processing. Although many systems claim to use some forms of "Socratic dialogue" - a question and answer sequence directed towards uncovering the underlying misconception of the student, it is not always the case that the dialogue is truly Socratic [4]. However, practically, an intelligent tutoring system could be a great assistant to teachers. As a feasible scheme, we attempt to develop an Intelligent Multimedia Tutoring System (IMTS) for laboratory support in software engineering education [7].

ITSs (Intelligent Tutoring Systems) clearly abandoned the objective of only providing courses for students to concentrate on building systems that provide supportive environments for more limited topics [5]. The limited topics in our endeavor are complex data structures and algorithms. To build such an IMTS, we follow a natural division of tasks for ITS into four distinct components, each corresponding to a distinct section: Knowledge Base, Tutoring Module, Student Model and Communication Module. Similar divisions have been proposed by [8].

2. A Generic Architecture for IMTS

In most traditional ICAI systems, all of the instructional components (i.e. domain knowledge, student status and teaching module etc.) are stored and implemented in a single structure. Although some systems have separate modules to store the instructional component independently, their operational procedures are still determined by specific pieces of information and by predefined algorithmic processes; IMTS, on the other hand, is generative in that students can take the initiative while learning, and can choose learning material by interest or be coached by an intelligent tutor who can respond to their cognitive status. This requires that IMTS can furnish students with a learning environment that has the ability of organizing domain knowledge, diagnosing student's cognitive status and instructing student by a suitable teaching strategy.

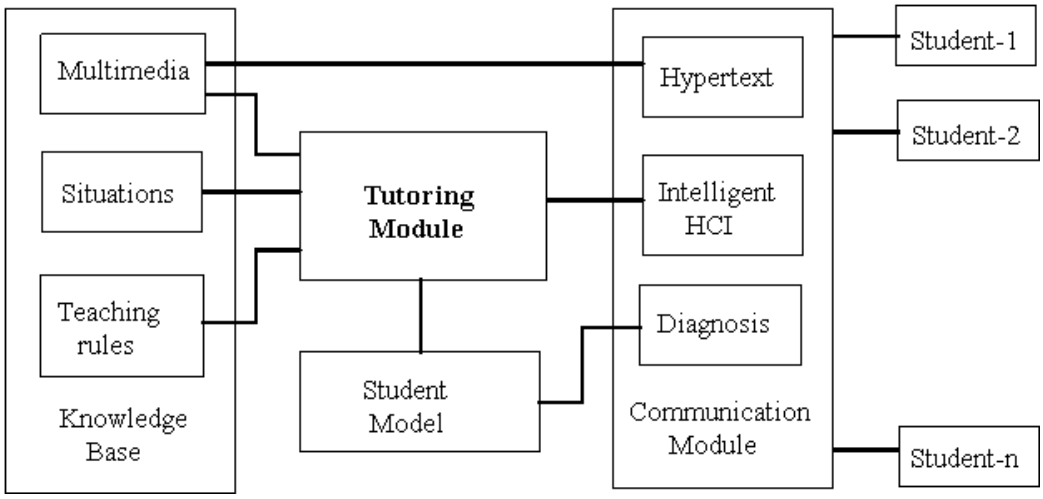

Fig. 1 The Generic Architecture of IMTS

As shown in Figure 1, the Intelligent Multimedia Tutoring System consists of four major components:

1. The **Knowledge Base** contains three types of knowledge that are domain knowledge, situations and teaching rules.

   - *Domain Knowledge* means concepts, algorithms and their different forms of expression. We organize them in a form of multimedia and plan to move them to the Internet in a future version that can be navigated by a WWW browser.
   - *Situations* are namely tasks which are given to the student during a laboratory session. The task could be developing a small program or operating algorithm in a graphical interface.

- *Teaching rules* are also called expertise knowledge, which are production rules transformed from teaching skills/strategies.

2. The **Tutoring Module** is the heart of IMTS. It is actually an instructional planner for didactic decisions. This module fulfills tasks of collecting student information from the student model, selecting situations for the student, presenting learning materials and giving instructions at proper time according to the teaching rules.
3. The **Student Model** includes aspects of the student's behavior and knowledge status that have repercussions for his performance and learning. As with most existing ICAI systems, we build the student model as variants of an "expert proficiency" in the domain.
4. The **Communication Module** can be divided into three parts; these are hypertext, intelligent HCI (Human-Computer Interface) and performance diagnosis.

    - Hypertext gives students an opportunity to learn by themselves and retrieve information from the multimedia section in the knowledge base.
    - Intelligent HCI provides a limited natural language interface and other communication methods such as menu-driven interface.
    - Performance Diagnosis is used to observe student's performance and transform his/her behavior and knowledge status into parameters which are stored in the student model.

An architecture as Fig.1 not only meets the structural requirements of an ICAI system, but also has the following benefits for teaching and learning:

- To represent domain knowledge in the form of multimedia makes the concepts and algorithms more vivid and understandable. Moreover, learning material in IMTS can be retrieved by a hypertext browser.
- Given a situation, students gain a deeper understanding of what they are learning because they can learn concepts and algorithms in context, and learn how to apply the new knowledge.

3. Learning in a Context

All researchers are aware that ITSs are not built and used on a large scale. It seems to us that the great days of Intelligent Tutoring Systems are over. One ICAI researcher, Elshout, even said: "... and I do not, to end, see much of a role for intelligent tutoring systems. I think the task is too difficult, diagnosis is too difficult, conversation is too difficult." [6]. Although there are still many researchers who continue to build ITSs, a major trend has emerged, which is moving away from "individual" cognition towards cognition embedded in a much wider context. One representative in this trend is A. Colins, who has left the classic ITS behind and now focuses on situated learning environments, simulation programs and microworlds, to provide learners with a context.

3.1 Cognitive Apprenticeship and Instructional Technology

Cognitive apprenticeship employs the modeling, coaching and fading paradigm of traditional apprenticeship, but with emphasis on cognition, rather than physical skills [2]. The differences between formal schooling and apprenticeship methods are many, but the most important one is that, in school, skills and knowledge have become abstracted from their uses in the world; while in apprenticeship learning, target skills are not only continually in use by skilled practitioners, but are instrumental to the accomplishment of meaningful tasks [3]. There are three main characteristics of cognitive apprenticeship; these are situated learning, modeling & explaining and coaching.

3.1.1 Situated Learning

Situated learning is the notion of learning knowledge and skills in contexts that reflect the way that the knowledge will be useful in real life. The contexts range from everyday life to the most theoretical endeavors. It is the computer that makes it possible for us to create environments, mimic situations in the real world that we cannot otherwise realize in a classroom.

3.1.2 Modeling & Explaining

Modeling is showing how a process unfolds, while explaining involves giving reasons why it happens that way. Using computers, we can represent process in ways that are difficult for human teachers to adopt because of the limitations of the lecture format (e.g. using animation technique to illustrate the process of searching a binary tree).

3.1.3 Coaching

Computer systems can patiently observe students as they try to carry out tasks, and provide hints or assistance when needed. This kind of personal attention is simply not feasible in most school classrooms. Moreover, a computer coach can give students assistance when they are in real difficulties, for it can observe them over a long period of time in order to discover what problems the student is really having and can remember perfectly what the student did before.

3.2 Context for Learning Data Structures

The most frequently encountered context for data structures is programming. Most programmers have had the experience of only seeing the implications of some concepts after they begin to program. For instance, by applying the concept of "pointer" in different settings, they finally realize that a pointer is actually an object stored in the memory that represents an address of another object. From this point of view, programming is highly recommended as the best way to learn the course of data structures.

Since programming is a time-consuming task, we will adopt three eclectic methods, namely understanding how the data structure is built and how operations are performed on it and generating a small program to implement the algorithm. The realization of these methods is discussed in detail as following.

3.2.1 Building the Data Structure

The general user interface for the IMTS is shown it Figure 2 and comprises three windows. The top window is used for the dialog with the student. The left window (interactive mode) lets the student draw data structures using an abstract representation. All the input from the student will come from interactively manipulating the data structures. Generally the student does not need to type in any instructions or answers, as the actions on the data structures will be analyzed by the IMTS. The tutor will only intervene when the student chooses to build a data structure that has constraints, e.g. a binary search tree (BST). While the student is drawing the data structure, simultaneously in the right window (passive display) the appropriate C data structure is displayed.

3.2.2 Operating on Data Structures

After the student is acquainted with the data structure s/he might choose to try an operation (algorithm) on it. This might include deleting an element from a linked list or searching for a node in a BST. The student has to tell the IMTS what kind of algorithm the student wants to perform. The student has to fulfill all constraints needed to complete the operation successfully, e.g. include help pointers when required. The tutor intervenes if a mistake occurs, and guides the student towards the correct answer.

3.2.3 Program Generation Strategy

It is assumed that students have experience in C. The student chooses to build a small program from the algorithms s/he has just mastered. The display will change slightly. The interactive (left) window will now display the C data structure which can be manipulated. The right window will be used to display the student's C code s/he has come up with to solve the problem. In order to give step by step guidance on how to write a solution for a particular algorithm, the student will not write his/her own C code, but choose the appropriate lines from a vast collection displayed in the lower part of the right window.

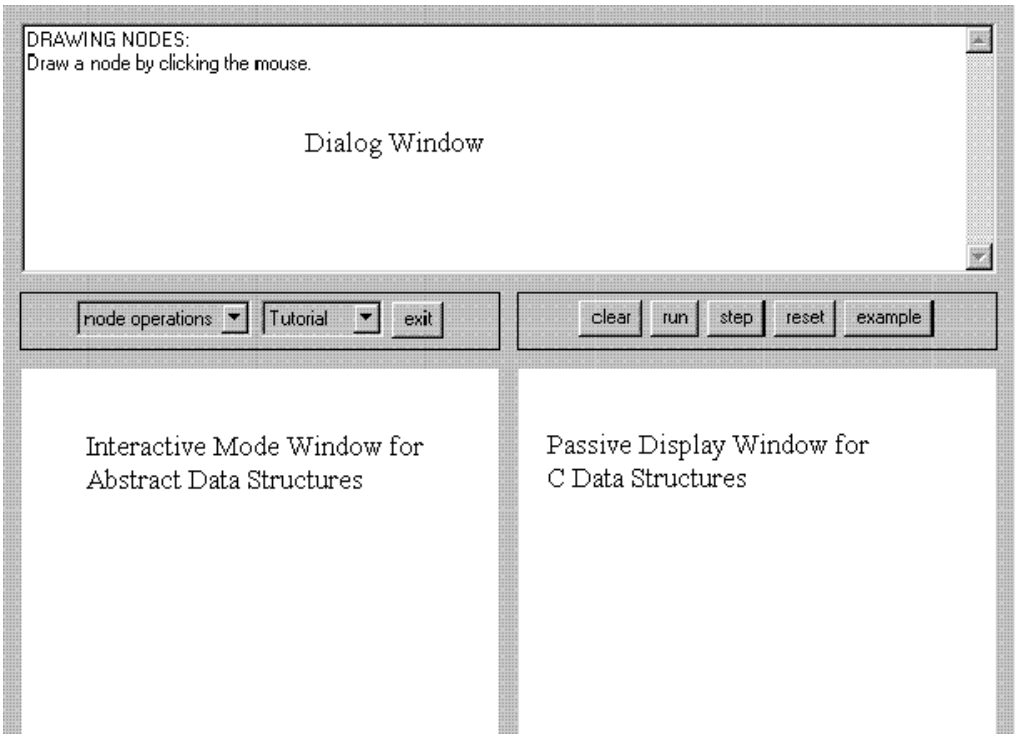

Figure 2: User Interface of the IMTS

During the exercise, every logical error will be reported by the tutor, and guidance will be given until the right solution is found. The data structure in the left window will be used to have the student repeat exercise and have him/her realize the correct order of the code, e.g. securing the tail of a linked list with a help pointer before deleting an element.

4. Case-based Instructional Planner

Every student has his/her own particular style of learning. For instance, some students are adept in reasoning and would like to solve problem by themselves, some students favor graphical interfaces, and others probably good at summarizing the teaching material. To cater to different learning styles, a tutor should have the ability to induce different cases (i.e. instructional plans); in other words, constructing an instructional plan for a particular student either by selecting a case from a case library or building a new one if his/her learning style can not be classified into any existing cases. One method to carry out this task is Case-Based Reasoning (CBR), which is a recent approach to problem solving and learning that has gotten much attention over the last few years.

4.1 Case-Based Reasoning (CBR)

Case-Based Reasoning is used to solve new problems by remembering a previous similar situation and by reusing information and knowledge of that situation [1]. A case-based reasoner generally has the following central tasks: identifying

the current problem situation, finding a past case similar to the new one, using that case to suggest a solution to the current problem, evaluating the proposed solution and updating the system by learning from this experience. The solution from a previous case may be directly applied to the present problem, or modified according to differences between the two cases.

4.2 Instructional Planning by CBR

A teacher making an instructional plan for a particular student is like a physician determining the disease and treatment for a patient. The success of a physician usually depends on his experience, namely the diagnosis and treatment of previous patients. In the same way, an experienced teacher will quickly recognize a student's character, and make suitable instructional plans for his/her learning. As a matter of fact, reasoning by reusing past cases is a powerful and frequently applied way to solve human problems.

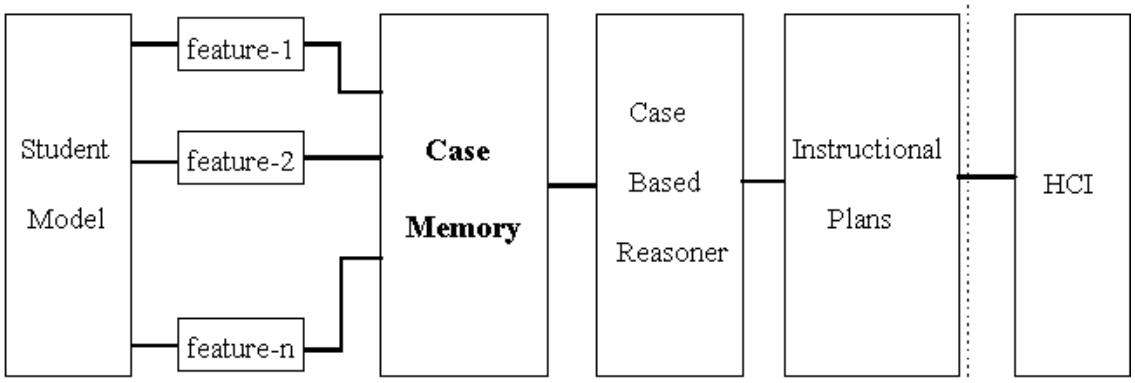

Figure 3: The Structure of the Instructional Planner

In Figure 3, we proposed a structure of the instructional planner. To remember past experiences, a case-based reasoner uses a case memory to store descriptions of previously solved problems and applied solutions. A case memory is defined as the following:

< case memory > ::= < case-1 > | < case-2 > | ... | < case-n >
< case > ::= < feature-1 > < feature-2 > ... < feature-m >
< feature > ::= < name > < value >

By definition, a case memory includes a number of cases, and the more cases the more experience the tutor has. A case consists of several features, which are described by a name and a value. For instance, we can define the features of a case as the following:

[feature-1] = [task] [the code for "insertion into a search tree"]
[feature-2] = [learning style] [the code for "graphical interface"]
[feature-3] = [teaching method] [the code for "program understanding"]
[feature-4] = [exercise] [the code for "related exercises"] . . .
[feature-n] = [effect] [value between 0~1]

In the last line, "effect" means the performance of a case that can be calculated by several factors, such as the time for finishing the task, the number of times for selecting and abandoning the case, and the score of related exercises done by the student. If the value of effect is too small (for example: less than 0.1), the case will be removed from the case memory to avoid case amount explosion.

In this example, the tutor first matches some features such as "learning style" and "task" for a particular student and then derives "program understanding" as an instructional plan from the case-based reasoner. In a practicable teaching process, the student will encounter the following two problems:

1. "Program understanding" is not enough or not suitable for the student to understand the process of "insertion into a binary tree" completely.
2. The student needs to take some remedial courses, such as "what is a binary tree?".

To solve these two problems, the case-based reasoner should find new cases for the student. For the first problem, the teaching method will be replaced by a new one; while in the second case, some sub-plans will be generated. From this point of view, the instructional plans are dynamic, and are modified frequently according to the student's reaction.

5. Conclusions

We have described the design of IMTS, an architecture that incorporates case-based reasoning to generate instructional plans for learning the course of Data Structure in a context. We discussed three main methods to build contexts that are program generation, program completion and program understanding, and coupled with hypermedia, we may provide supportive learning situations for students with an instructional paradigm of cognitive apprenticeship. Using case-based reasoning, IMTS can reuse its past teaching experience and make dynamic instructional plans for a particular student. Through an example in section 4.2, we conclude that plan-making by CBR is suitable to learning in context, and it makes the context a flexible one.

References

[1] Aamodt, A. and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", 1994, AICOM, Vol.7, No.1, pp.39-59.

[2] Collins, A., "Cognitive Apprenticeship and Instructional Technology", in Idol, L. Jones, B. F. (Eds.) Educational Values and Cognitive Instruction, Hillsdale, NJ: Lawrence Erlbaum and Associates, 1991.

[3] Collins, A., J. S. Brown and S. E. Newman, "Cognitive Apprenticeship: Teaching the Crafts of Reading, Writing and Mathematics", in Resnick, L. B.(Eds.) Knowing, learning and instruction: Essays in honor of Robert Glase, Hillsdale, NJ: Lawrence Erlbaum and Associates, 1989.

[4] Elsom-Cook, M., "Guided discovery tutoring and bounded user modeling", in John Self (Eds.) Artificial Intelligence and Human Learning, Chapman and Hall Computing,1988.

[5] Sleeman, D. and J. S. Brown, "Introduction: Intelligent tutoring systems". in D.Sleeman & J.S.Brown (Eds.) Intelligent Tutoring Systems, Academic Press, Inc.,1982.

[6] Sandberg, J. and Y. Barnard, Y., " Education and Technology: What do we know? And Where is AI?", AICOM,1993, Vol.6, No.1, pp.47-58.

[7] Warendorf, K., "Intelligent Multimedia Tutoring System for Laboratory Support in Software Engineering Education", Proc. of ED-MEDIA 96, 1996, pg.814.

[8] Wenger, E., Artificial Intelligence and Tutoring Systems, Morgan Kaufmann Publishers, Inc.,1987.