AC 2010-1620: A WEB-BASED BAYESIAN VAN HIELE PROBLEM SOLVER FOR COMPUTER PROGRAMMING

J. Wey Chen, Southern Taiwan University

Dr. J. Wey Chen is a Visiting Professor in the Department of Information System at Southern Taiwan University. He formerly served a two-year appointment (2007-2009) as the Department Chair of the Department of Information Management at Southern Taiwan University and was the Computer Science Department Chair at Western State College of Colorado. His scholarly interests range widely, from computer science curriculum design to e-learning and software engineering practices.

A Web-based Bayesian van Hiele Problem Solver for Computer Programming

Abstract

Computer programming teaching is often based upon the traditional lecture format. However, this methodology may not be the best way to help many students actively understand underlying concepts. This paper formulates an alternative pedagogical approach that encompasses the van Hiele Model, cognitive model, and Bayesian network to design a **w**eb-based intelligent van Hiele Problem Solver (IVHPS). The system takes full advantage of Bayesian networks (BNs), which are a formal framework for uncertainty management to provide intelligent navigation support, and to make individualized diagnosis of student solutions in learning computer programming. In addition, we describe the architecture of the system and the roles of seven modules contained in the system. They are all integrated into the environment to increase student satisfaction and achievement by stimulating student motivation and encouraging the perception of problem solving and programming concepts.

Introduction

"Programming" is a complicated business. For many of the students who take programming courses, programming is a scary new subject. In his classic article on teaching programming Dijkstra[1] argues that learning to program is a slow and gradual process of transforming the "novel into the familiar".

Most computing educators also find that programming is not a single skill. It is not a simple set of discrete skills; the skills form a hierarchy[2], and a programmer will be using many of them at any point in time. A student faced with learning a hierarchy of skills will generally learn the lower level skills first, and will then progress upwards[3]. In the case of coding, even though it is a small part of the skill of programming, it implies that students will learn the basics of syntax first and then gradually move on to semantics, integrated structure, and finally style.

As complicated as programming is, computer programming teaching is often based upon the traditional lecture format. However, this methodology may not be the best way to help many students actively understand underlying concepts. This paper formulates an alternative

pedagogical approach that encompasses the van Hiele model of geometric thought, the cognitive model, and Bayesian network to design a web-based intelligent van Hiele Problem Solver(IVHPS). The system takes full advantage of Bayesian networks (BNs) , which are a formal framework for uncertainty management to provide intelligent navigation support, and to make individualized diagnosis of student solutions in learning computer programming. In addition, we describe the architecture of the system and the roles of seven modules contained in the system. They are all integrated into the environment to increase student satisfaction and achievement by stimulating student motivation and encouraging the perception of problem solving and programming concept.

The Modified van Hiele Model for Computer Science Teaching

Adopted by Soviet educators for use in their geometry curriculum, the van Hiele model has stimulated considerable research, interest has risen in the United States[4] as more and more researchers have attempted to adapt the van Hiele model [to] learning in other mathematical areas such as economics and chemistry[5].

The van Hiele theory is partially based on the notion that student growth in geometry takes place in terms of identifiable levels of understanding and that the instruction in geometry is most successful when directed toward the student's level. Hence, the hierarchical structure of the van Hiele levels has been verified by Fuys, Geddes, Lovett, and Tischler[6] and Usiskin[7].

Computer programming has long been viewed as a vehicle for teaching students about their problem-solving process. The literature review revealed that the van Hiele model of geometric thought may be properly modified to apply to the learning and teaching of computer programming because both tasks have many features in common. Soloway and his colleagues further confirmed that computer programming and mathematical problem solving skills were mutually transferable[8,9].

Chen[10] conducted a study on the Taiwanese technological university students by applying the modified van Hiele's five-phases of geometric thought for learning and teaching computer programming and found that the use of van Hiele's modified five-phases of learning model for teaching computer programming may produce a higher level of computer programming thinking and a significantly higher achievement in learning the Java programming language.

The five-levels of geometric thought for learning and teaching computer programming were dubbed: "visual", "descriptive", "theoretical", "form logic modification and analogy", and "abstraction and modeling". In addition, the five sequential instructional steps, which they assert will take students through a reasoning level, will be integrated into the model to help students progress from one level to the next higher level.

The Cognitive Theory

It is widely known that programming, even at a simple level, is a difficult activity to learn. Why is this so? Are novice difficulties really inherent in programming or are they related to the nature of the programming tools currently given to novices? Bonar and Soloway[11] presented evidence that current programming languages do not accurately reflect the cognitive strategies used by novice programmers. Instead, Bonar and Soloway[11] have found that novice programmers possess knowledge and experience with step-by-step specifications in natural language. This knowledge and experience gives them powerful intuitions for using a programming language. Programming languages, however, are not designed to appeal to these intuitions.

On a semantic and pragmatic level, there are incompatibilities between the way natural and programming languages are used. Many novice programming bugs can be directly traced to an inappropriate use of natural language specification style or strategy.

In order to provide a novice with powerful intuitions for using a programming language, the researcher represented and arranged programming knowledge according to its level of difficulty in four cognitive levels: Lexical and Syntactic, Semantic, Schematic, and Conceptual[12].

- The Lexical and Syntactic levels are self-explanatory. Syntax refers to mistakes in spelling, punctuation, and the order of words in a program. Syntax errors are frequently identified by the compiler, but the error messages may not give the students the information needed to fix the code.
- The Semantic level (as adapted to the programming domain) deals with the semantics of individual statements.
- The Schematic level, through the use of programming plans, allows multiple statements to be grouped into semantically meaningful knowledge units.
- The Conceptual level deals with definable functions within the problem domain of the application being programmed.

A Combined Model

The van Hiele model asserts that the learner moves sequentially through five levels of understanding. The Cognitive Theory finds a more natural way to give novice powerful intuitions for using a programming language by further representing and dividing programming knowledge according to its level of difficulty in four cognitive categories. Figure 1 shows a combined model used by the study to represent the knowledge structure of every learning node(concept).

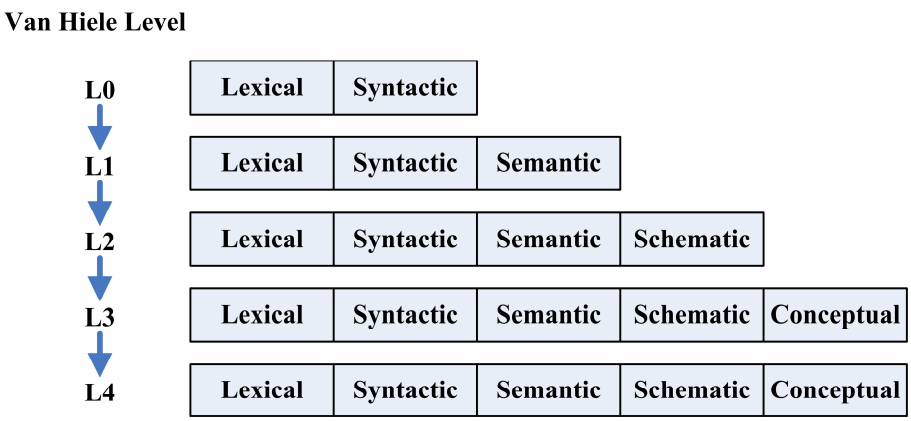

Figure 1. Knowledge structure for each learning node

The study found that instruction developed in this structured sequence prompted the acquisition of the next higher level of computer programming learning. Moreover, students were found to commit less of the common programming mistakes if these five levels of thought were properly modeled.

The Subjects and Bayesian Training Data

This study employs the data generated by sixty freshman students (2 classes) majoring in Management Information System (MIS) in the Information Management Department at Southern Taiwan University of Technology (STUT) and Kun Shan University (KSU). The subjects come from two sophomore-level Java programming classes with approximately 30 students each. The subjects are selected from these two technological universities because they are representative of most technological universities in Taiwan.

The diagnostic test plays a critical role in implementing the system. The Java curriculum content is structured into levels and categories based on the combined van Hiele model of geometric thought and the Cognitive theory. The computerized diagnostic test is structured similarly. Each

test for a particular module is structured into topics and questions. Three questions at most will be used to represent knowledge of a cognitive category within a van Hiele level of understanding.

The Bayesian training data consists of a set of diagnostic test items and the actual answers compiled from sixty freshman Information Management majors. This database was selected because it is a classic dataset used as evidence for the existence of programming bugs. Since these test items are designed to map programming bugs with learning topics, it is highly likely that our anticipated bugs will occur in this test set reasonably frequently. Such a design will help to conclude robust statistics.

The Study Module

To enable communication between the system and learner at the content level, the domain model of the system has to be adequate with respect to inferences and relations of domain entities with the mental domain of a human expert[13]. In this sense, the domain knowledge of IVHPS is represented in a conceptual network that depicts the interrelations between several learning nodes (concepts) of the Java programming language. Concept knowledge in each learning node is further divided into five van Hiele levels of understanding, and each van Hiele level of understanding is then represented by two to three cognitive categories depending on its level of difficulty. Representing the Java domain knowledge in a structured way ensures that the system "knows" the dependencies between concepts, and uses this knowledge to provide customized instruction and feedback to errors.

For our purposes, we identified a set of concepts that are taught in our Java programming language course at the Southern Taiwan University. Each concept is represented by a node in the graph. We add a directed line from one concept (node) to another, if knowledge of the former is a prerequisite for understanding the latter. Thus, the DAG can be constructed manually with the aid of the course textbook. For example, consider one instance of the if statement in Java such as:

```
if((a <= b) && (b <= c))
      return true;
else
      return false;
```

Even though it is as small as it can be, one can see that the if statement has quite a lot to it. This is because Java is a real industry-strength language, and even the smallest portion of a program needs some heavyweight ingredients. To understand the if statement, one must first develop some basic concept of programming, the Java programming environment, the concepts of data types, variable assignment, Relational operators, and logical operators. These relationships can be modeled as depicted in Figure 2. Naturally, Figure 2 depicts a small portion of the entire DAG implemented in the study.

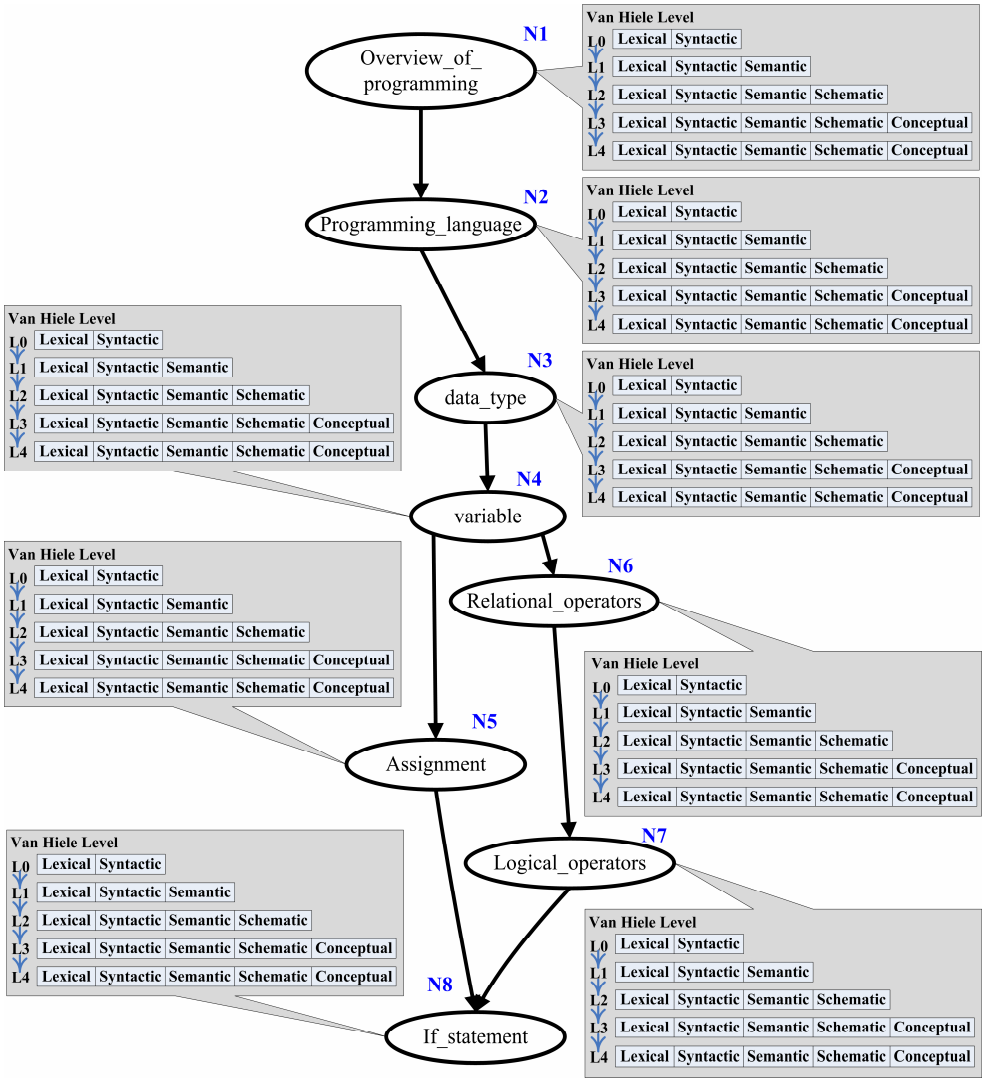

Figure 2 Sub-DAG for the if statement

The next task in the construction of the BN is to specify a conditional probability distribution (CPD) for each node given its parents. For variable NRLNi (Next Related Learning Node, a child node) with parent set CPItemi (Conditional Probability of each Item), a CPD p(NRLNi|CPItemi)

has the property that for each configuration (instantiation) of the variables in CPItemi, the sum of the probabilities of NRLNi is 1.0. In Figure 2, the parent set of the if statement is {N1-Overview_of_programming, N2-Programming_language, N3-Data_type, N4-Variable, N5-Assignment, N6-Relational operators, N7-Logical operators}. The corresponding CPD

P(if statement| N1-Overview_of_programming, N2-Programming_language, N3-Data_type, N4-Variable, N5-Assignment, N6-Relational operators, N7-Logical operators)

Regarding relationships among concepts and questions, for each test item it would be necessary to specify the probability of correctly answering the question given all possible combinations of mastering/not mastering the related concepts. The training data will be used by the Bayesian inference engine to provide intelligent, personalized tutoring and support to the student.

A Bayesian network (BN), which consists of directed acyclic graph (DAG) and a corresponding set of conditional probability distributions (CPDs) was used in this study to perform the following three functions: (1) to construct and validate the course content map represented in DAG format, (2) to model the students' prerequisite information and to guide the student in navigating through the Java programming concepts based on the structure depicted in Figure 2, and (3) to keep track of student knowledge regarding each concept.

General architecture of intelligent van Hiele Problem Solver

The Web-based IVHPS that was evaluated is a system that aims to teaching the Java programming language. The IVHPS incorporates techniques from the Intelligent Tutoring System and Hypermedia links to tailor instruction and feedback to each individual student. Individualization in IVHPS is based on two models: the domain model (representing knowledge about the domain of the Java programming language) and the student model (representing knowledge about the individual student). We will describe how these two models are used in order to provide personalized diagnosis and instruction.

The literature reveals that novices need additional support to solve a problem. The idea is simple but important to computer science pedagogy as learning language syntax, code flow, data representation, and appropriate design are daunting tasks for novices.

The Web-based IVHPS to promote effective collaborative learning relies on six technologies, as

shown in Figure 3. The function and design principle of each component is described as follows:

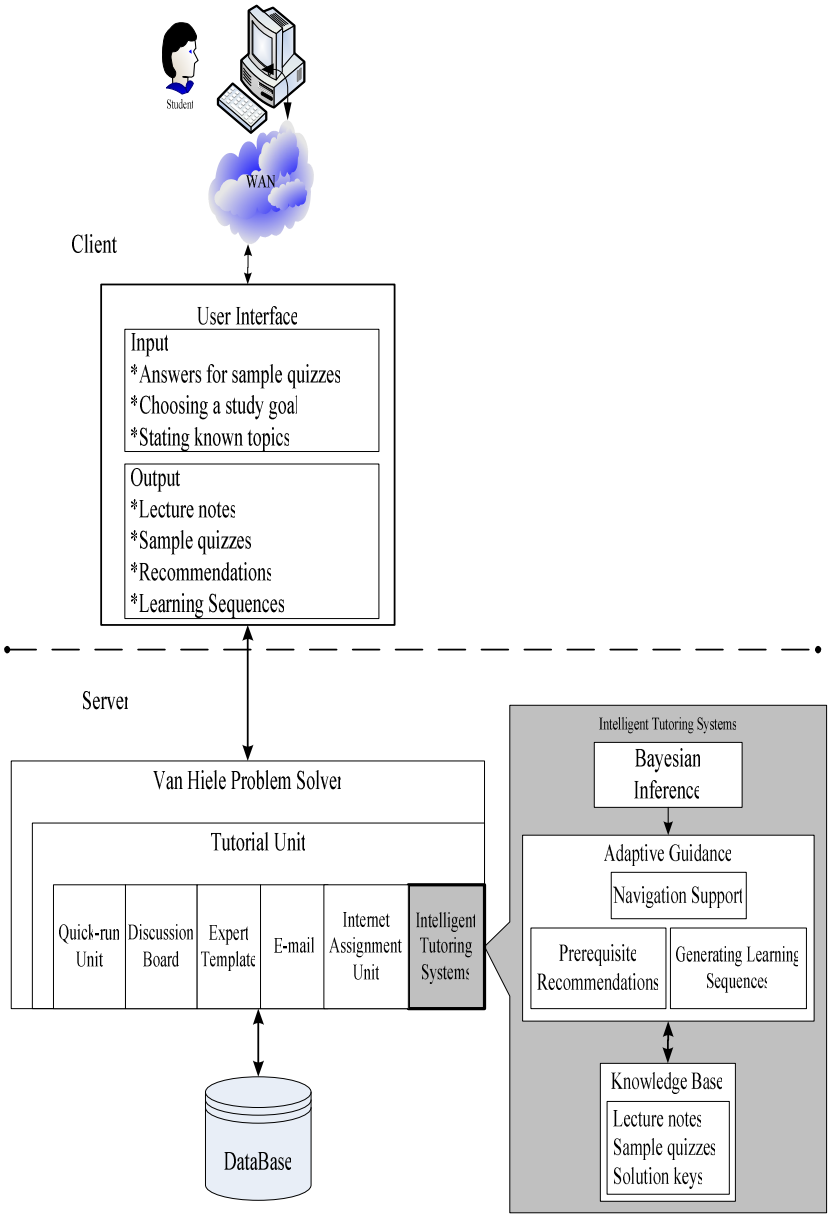

Figure 3. Architecture of VHPS

The Intelligent Tutoring System

The system provides remote access for students to take the diagnostic test and based on the overall picture of the test result, the system will provide the learner with intelligent navigation support and learning recommendations, and integrates the features of electronic hypermedia course

material with intelligent tutoring tactics. The system can propose learning goals and guide users by generating reading sequences for them.

Based on the information contained in the student model the system provides intelligent, personalized tutoring and support to the student. In particular, based on information concerning the knowledge level of the student in each concept of the domain knowledge, the system provides individualized support when s/he navigates through the course material.

The system uses the direct guidance technique to inform the learner whether s/he is ready to visit the corresponding topic or if the studying of a page is unnecessary due to the fact that the student has already mastered the concept that is associated with this test frame. With the direct guidance technique, the system suggests and leads the student to the particular learning level the system considers as the most appropriate for the student to visit.

In order to select the next learning topic to present to the student, the system consults the individual student model based on the diagnostic test data. In particular, it uses the information that represents the knowledge concept of the student in each domain concept. First, it looks for domain concepts related to parts of the topic that the student has already mastered. Then, among them, it looks for topics that the student has difficulty with when answering test questions. The tutor then selects a category for which the student has the greatest probability of making a mistake while using it in a test. In this way, we ensure that students are not always asked to solve either too easy or too difficult learning topics. Figure 4 is a screen shot of IVHPS displaying the diagnostic report when the student completed all the questions in the diagnostic test.

E-mail . E-mail is used for communication between the instructor and individual students or the entire class. Specific communication types supported by e-mail include: (1) time-sensitive announcements to the class (i.e. scheduling and assignment changes), and (2) faster student questions and instructor response interactions.

Discussion board. Discussion boards provide an efficient means of communication and creating student support outside of the class. This also allows the shy students an opportunity to fully participate since they may feel more comfortable in the electronic environment. The instructor can then post a “thread,” or topic, for discussion and have the students respond to the topic within a specified period of time. Similar to being in a chatroom, the instructor will monitor the discussion for content and appropriateness.

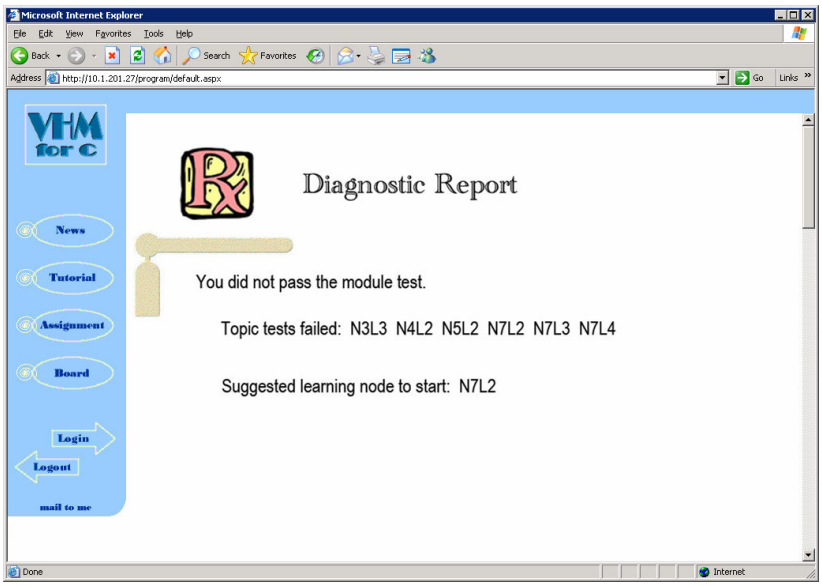

Figure 4. A screen shot of IVHPS displaying the diagnostic report

Internet assignment units: Detailed information presented in this unit can reduce the classroom time allocated to technical and project support which, in turn, allows greater depth and breadth of topics to be covered in class.

Tutorial unit: The tutorial unit is the realization of van Hiele's five-levels of thought to learning computer programming. The five sequential instructional steps will be integrated into the model in order to facilitate the students' progress from one level to the next higher level. Hypertext and icon techniques will be used to switch from tutorial mode to display or run mode so that students may obtain immediate visual feedback to enhance their learning. Figure 5 is a screen shot of IVHPS displaying the lecture notes for the concept "Data Types".

Quick-run unit: Constructive scaffolding is used in this component. For example, we may give code that solves a given problem to the students. The code, in this example, is the scaffolding mechanism that students will build upon. We then have them insert comments to describe the semantics of the code. Alternatively, we may give comments (the scaffolding) to the students that describe an algorithm and have students write a code that corresponds to the comments. The ultimate goal is for students to generate all the codes and think about an efficient solution. Once the students have finished their code, they can then immediately test and run their code in the integrated programming environment and receive the immediate feedback. Gomes and Mendes[14] consider the use of programming patterns is one of the most effective practices to teach computer programming language especially when the student shows incapacity to solve a

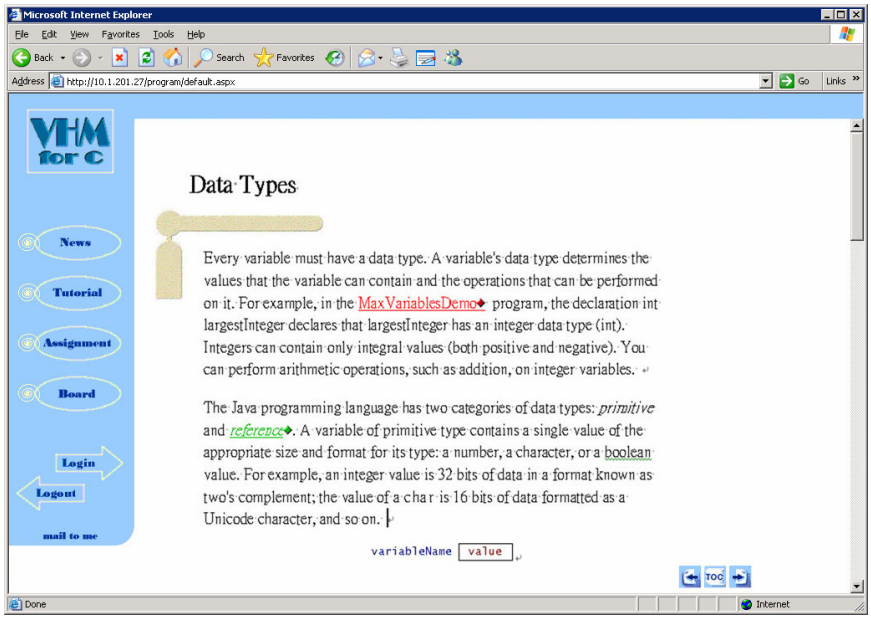

Figure 5. A screen shot of IVHPS displaying the lecture notes for the concept “Data Types”.

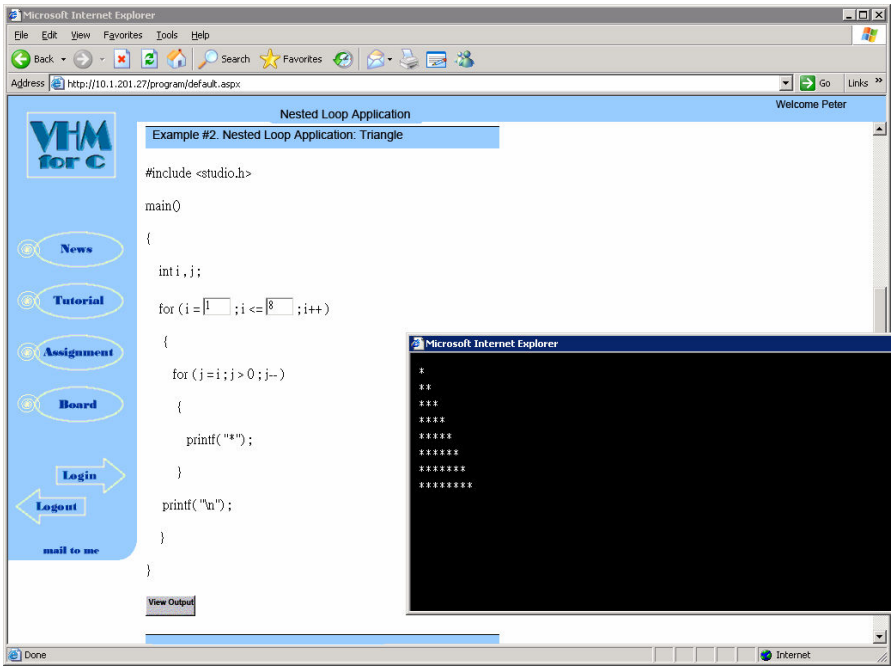

Figure 6 A screen shot of IVHPS displaying a typical quick-run sample output

particular problem or a part of it. Figure 6 is a screen shot of IVHPS displaying a typical quick-run sample output.

Expert template. This unit is provided to model an expert's idea of solving a specific problem. The unit will be designed in template format which follows the steps a human expert takes in solving problems of this type. This allows students to compare his/her solution with the expert's solution in hopes of providing a learning experience that is transformative. Besides the students' effort, the success of this process depends on two critical components: the design of the problematic scenarios and the entire learning activities and the guidance provided by human experts. The practical difficulty of this component is, even though the expert examination knowledge can be reproduced in the same context, students are neither able to transform it to similar scenarios, nor apply it to an actual situation which calls for action. Figure 7 is a screen shot of IVHPS displaying a typical practice sample from the expert template.

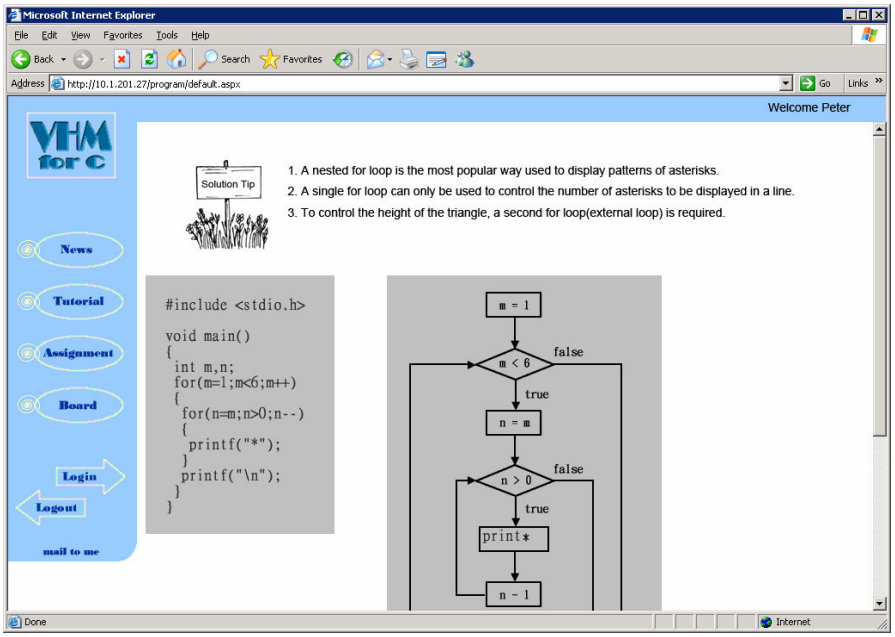

Figure 7. A screen shot of IVHPS displaying a typical practice sample from the expert template.

A Preliminary Experiment

We have conducted a preliminary experiment with the IVHPS in the Fall Semester, 2008 in the Information System (MIS) Department at the Southern Taiwan University. A total of 52 MIS students from the above randomly selected Java class became the subjects of this study. The

students were asked to complete a 57-question Diagnostic Test and received a diagnostic report generated by the system to indicate Java topics for which they failed to pass and the topic which the system suggest for them to start with.

Forty eight out of 52 (92.31%) participants reported that they are satisfied with the feedback from the system's feedback to their mistakes. The results from the preliminary experiment indicated that the use of the Bayesian inference engine and the structured diagnostic test can be effectively used to provide intelligent navigation support, and to make individualized diagnosis of student solutions in learning computer programming.

A follow-up interview was further conducted to collect information regarding how individuals reacted to the difficulty in answering diagnostic test questions. Table 1 shows the result of the students' responses when asked the question" What kind of suggestion and help do you need to solve programming questions of this kind?"

Table 1 Students' Responses to the Desired Need for Programming and Problem Solving

Students' Responses	Tally
I want to see the entire curriculum picture of this test on-line	9
I want to communicate with my friends who can restate/explain the problem in my vocabulary	7
I thought a fill-in-the-blank type of question would be helpful	5
I just want to keep trying to solve it	2
I like to take this test on-line with C++ compiler	4
I need to refer to my text	5
I want to discuss questions with my classmates	8
I need somebody to layout the detailed problem plan for me	3
I want to see more real examples of the kind	5

The information collected from the follow-up interview provided useful feedback and guidelines for the researcher to construct a second generation IVHPS by considering the traditional didactical questions of why, what, how and for whom on the construction of the diagnostic test. It is also essential to keep the students' responses in mind in knowing that collaborative learning and the utilization of some types of interaction are vital for students when solving a programming

problem.

Conclusions

High failure rates in many introductory programming courses may be a reflection of the limited research that has been invested and developed in the computer programming discipline. The high failure rates have propelled a substantial proportion of college students to assume that there is a hidden mind somewhere in computer programs which prevents students from completing the task[15]. We hope the proposed Modified van Hiele Model for Computer Science Teaching can help unveil the mystery of the "hidden mind" and provide a logical link for students to inductively learn problem-solving and programming skills. The success of this model is attributed to the extensive review of the available literature and to the exploratory interviews with students who participated in the first phase of study. The exploratory interviews with students and the observations generated a set of constructive data for us to devise a functional environment to support collaboration for learning programming in our second phase of study.

This paper discusses a new architecture of designing a van Hiele-based intelligent tutoring system for computer programming using Bayesian technology. Centering on the explicit knowledge structure and the way to use Bayesian training data for diagnostic and recommendation purposes, we described the theory and implemented the prototype of the suggested system. The current study is designed to be able to: (1) demonstrate a measurement scheme to detect misconceptions employed by the students, and (2) provide a convenient descriptive tool for diagnosing students' programming abilities by representing a set of bugs in the networks. More specifically, the system is able to utilize Bayesian network techniques in modeling the student knowledge based on the structure depicted in Figure 2.

Future work will involve incorporating the more sophisticated concepts of Java into the system. We also hope to extend the suggested system by incorporating other programming languages such as C++ and MS Visual Basic.

Acknowledgement

This work is funded by the National Science Council in Taiwan, under the "Science Education" Program, Project No. NSC 97-2511-S-218-005-MY2.

Bibliography

1. Edsger W. Dijkstra. (1989). On the Cruelty of Really Teaching Computing Science. Comm. ACM, Vol.32, pp 1398-1404.
2. Sloane, K. & Linn, M. (1988). Instructional Conditions in Pascal Programming Classes. In R. E. Mayer (ed), "Teaching and Learning Computer Programming", Lawrence Erlbaum Associates, pp 207-235,8.
3. Bereiter, C. and E. Ng. (1991). Three Levels of Goal Orientation in Learning. Journal of the Learning Sciences, Vol. 1, pp 243-271.
4. National Council of Teachers of Mathematics.(1989). Curriculum and evaluation standards for school mathematics. Reston, VA: Author.
5. Crowley, M. L. (1987). The van Hiele model of the development of geometric thought. In M. Lindguist & A. Shutle (eds.), Learning and teaching geometry, K-12, (1987 Yearbook of the National Council of Teachers of Mathematics) (pp. 1-16). Reston, VA: NCTM.
6. Fuys, D., Geddes, D., Lovett, C., and Tischler, R. (1988). The van Hiele model of thinking in geometry among adolescents. Journal for Research in Mathematics Education(Monograph No. 3). Reston, VA: National Council of Teachers of Mathematics.
7. Usiskin, Z. (1982). Van Hiele levels of achievement in secondary school geometry(Final report of the Cognitive Developemtn and Achievement in Secondary School Geometry Project). Chicago: University of Chicago.(ERIC Document Reproduction Service No. ED220288)
8. Ehrlich, K., Soloway, E., & Abbott, V. (1982). Transfer effects from programming to algebra word problems: A preliminary study (Rep. No. 257) New Haven: Yale University Department of Computer Science.
9. Soloway, E., Lockhead, J., & Clement, J. (1982). Does Computer programming enhance problem solving ability? Some positive evidence on algebra word problems. In R. Seidel, R. Anderson, & B. Hunter (EDs.) Computer literacy, New York: Academic Press, 1982.
10. Chen, J. & Lin, C. (2006). A van Hiele Web-based Learning System with Knowledge Management for Teaching Programming, Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies, pp. 114-116, (ICALT2006).
11. Bonar, J. & Soloway, E. (2001). Uncovering Principles of Novice Programming Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, 10 – 13.
12. Liffick, B. & Aiken R. (1996) A novice programmer's support environment. Proceedings of the 1st conference on Integrating technology into computer science education, Volume 28 , 24 Issue SI , 1-3
13. Peylo, C., Teiken, W., Rollinger, C., and H. Gust. (2000). An otology as domain model in a web-based educational system for prolog, in Proceedings of the 13th International Florida Artificial Intelligence Research Society Conference, eds. J. Etheredge and B. Manaris, AAAIPress, Menlo Park, CA, pp. 55-59.

14. Gomes, A. and Mendes, A. (2007). "An environment to improve programming education", ACM.International Conference on Computer Systems and Technologies – CompSysTech07, IV. 19-1 – 19.6.
15. Pea, R.D. (1986). Language-independent conceptual “bug” in novice programming. Journal of Educational Computing Research, (2) 1, pp. 25-36.