AC 2010-1196: CURRICULUM SEQUENCES CONSTRUCTION IN A WEB-BASED VAN HIELE TUTOR USING BAYESIAN NETWORK

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.

Curriculum Sequences Construction in a Web-based van Hiele Tutor Using Bayesian Network

Abstract

Educational content on the Internet is rapidly increasing. Educational institutions and businesses are placing more course material online to supplement classroom and business training situations. Prior researchers have reported that this new web-based training technology has not integrated sound pedagogical practices into the authoring process when developing new tutorials. This paper formulates an alternative pedagogical approach that encompasses the van Hiele Model, cognitive model, and Bayesian network to design the curriculum content and sequence, to provide intelligent navigation support, and to make individualized diagnosis of student solutions in learning computer programming possible.

Introduction

Programming is a vital area in computer science education and a fundamental part of the computer science curriculum[1] . Research shows that computer programming languages help students develop problem solving ability and analytical skills[2,3,4]. Ebrahimi[5] claims that the study of programming provides a golden opportunity for: 1) understanding human problem solving, 2) learning the important aspects of programming, and 3) contributing to the refinement of programming languages, training, tools, and design methods. In addition, programming experience as a part of IT education allows students to get a better understanding of software, which is an essential part of computers[6].

Although there are a variety of possible motivations for learning to program, the task can be very difficult for beginning students of all ages[7]. In addition to the challenges of learning to form structured solutions to problems and understanding how programs are executed, beginning programmers also have to learn a rigid syntax and commands that may have seemingly arbitrary or perhaps confusing names. Tackling all these challenges simultaneously can be overwhelming and often discouraging for beginning programmers.

The learning and teaching of computer programming and that of geometry have many common features. The van Hiele's five-phase learning model for teaching computer programming has the

capability to produce a higher level of computer programming thinking and a significantly higher achievement in learning computer programming[8]. The purpose of this study is to adapt the combined van Hiele model of geometric thought and the Cognitive theory to develop a Bayesian network-based van Hiele intelligent tutoring system for learning Java programming language.

The van Hiele Model of Geometric Thought

The van Hiele model of geometric thought has been regarded as an effective model for mathematical problem-solving. This model identifies five levels of thinking in geometry: (1) Visual, (2) Descriptive, (3) Theoretical, (4) Formal Logic, and (5) The Nature of Logical Laws. The van Hiele model also defined five phases of learning procedures in course instruction: "information," "guided orientation," "explication," "free orientation," and "integration"[9]. According to this model, the learner, assisted by appropriate instructional experiences, can be successfully promoted from a lower to a high level of geometric thinking. The van Hiele model has motivated considerable research and resultant changes in the geometry curriculum by Soviet educators, and in recent years, interest has increased in the United States. More and more researchers are trying to expand the van Hiele model to facilitate learning in other mathematical areas. For instance, in Holland, it is being applied to economics and chemistry.[10]

The Modified van Hiele Model for Computer Science Teaching

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[11,12]

Chen[8] conducted a study on the Taiwanese technological university students by applying the modified van Hiele's five-phase of learning for teaching computer programming and found that the use of van Hiele's modified five-phase of learning model for teaching computer programming may produce a higher level of computer programming thinking and a significantly higher achievement in learning the C++ programming language.

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. Based on the study of the van Hiele model and his experiences in teaching computer programming, Chen[13] proposed a modified model as a practical application to computer programming as shown in Figure 1.

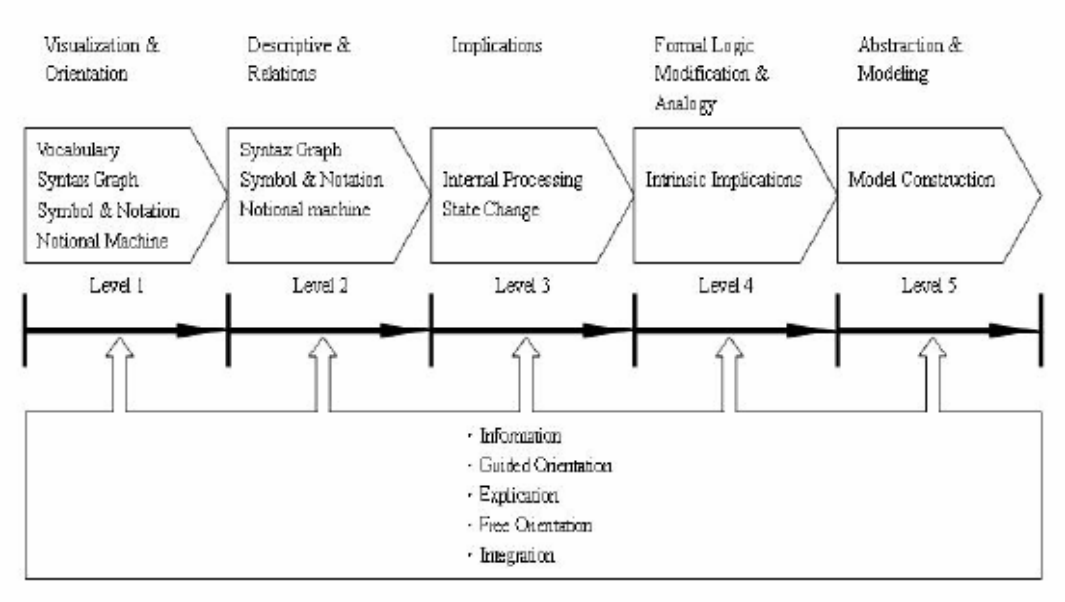

Figure 1. The Modified van Hiele Model for Computer Science Teaching

The modified van Hiele model of computer programming thinking still consisted of three major elements: (a) the nature of insight, (b) the levels of thought, and (c) the phases of learning. The five-levels of thought of learning 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 sequence is shown in outline form below.

1. Information: New topics are introduced through guided dialog.
   A. Teacher assesses student's vocabulary, interpretations, and prior knowledge.
   B. Students learn what direction further study will take.
2. Directed Orientation: Students explore the topic through activities and materials sequenced by teacher.
   A. Much of the material will be short tasks designed to elicit specific responses.
   B. Teacher provides clarification where needed.
3. Explication: Students refine their conceptualizations and vocabulary.
   A. Students express opinions about the structures observed.

B. Students observe relations within the structure.
C. Teacher moderates student discussion and helps them reach consensus.

4. Free Orientation: Students are challenged with more complex tasks of problem solving nature.
   A. Students discover new relationships and resolve problems on their own.
   B. Teacher acts as a guide for discovery.
5. Integration
   A. Students review and summarize their observations forming a synthesis of new concepts and relationships.
   B. Teacher provides direct explanations to assist students in refining and internalizing concepts and procedures.

The researcher found that instruction in this sequence promoted the acquisition of the next highest level in computer programming learning. Common programming mistakes committed by students also revealed their faulty explanation stances if these five levels of thought were not modeled properly. The five sequential instructional steps will be integrated into the Bayesian model to serve as a guiding framework to develop a rich and flexible web-based environment for Teaching and Learning Computer Programming languages.

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[14] presented evidence that current programming languages do not accurately reflect the cognitive strategies used by novice programmers. Instead, Bonar and Soloway[14] 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 [15].

- 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 2 shows a combined model used by the study to represent the knowledge structure of every learning node(concept).

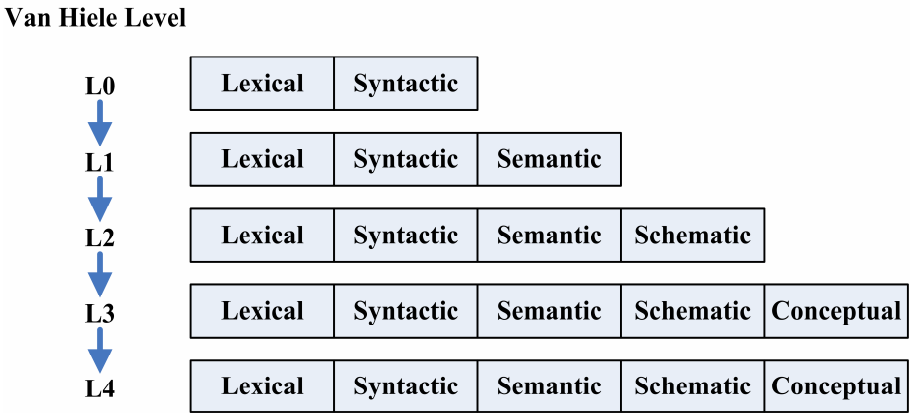

Figure 2. Knowledge structure for each learning node

Using Bayesian Networks in Diagnostic Test

The key to aiding students in navigating through knowledge concepts is two-fold. First, the structural model[16,17,18,19] needs to be defined, which means that nodes and links should be

identified and modeled. Second, we need to keep track of student knowledge regarding each concept node. Bayesian networks can help us meet both of these objectives.

A Bayesian network (BN) [20,21, 22,23] consists of directed acyclic graphs (DAG) and a corresponding set of conditional probability distributions (CPDs). Based on the probabilistic conditional independencies[22, 23] encoded in the DAG, the product of the CPDs is a joint probability distribution. In Bayesian network, conditional probabilities can be defined in terms of unconditional probabilities in the product rule. We defined the product rule and pointed out that it can be written in two forms because of the commutatively of conjunction:

$$P(a \cap b) = P(a \mid b) \cdot P(b)$$

$$P(a \cap b) = P(b \mid a) \cdot P(a)$$

Equating the two right-hand sides and dividing by P(a), we get

$$P(b \mid a) = \frac{P(a \mid b) \cdot P(b)}{p(a)}$$

This equation is known as Bayes' rule (also Bayes' law or Bayes' theorem).This simple equation underlies all modern artificial intelligence systems for probabilistic inference. The more general case of multivalued variables can be written in the P notation as

$$P(Y \mid X) = \frac{P(X \mid Y) \cdot P(Y)}{P(X)}$$

where again this is to be taken as representing a set of equations, each dealing with specific values of the variables.

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 programming concepts, and (3) to keep track of student knowledge regarding each concept.

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 (STUT) and Kun Shan University (KSU). The subjects come from two freshman-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. Since the Java curriculum content will be further structured into levels and categories based on the combined van Hiele model of geometric thought and the Cognitive theory, the computerized diagnostic test has a Java curriculum-based structure. Each test for a particular module is structured in 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

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 edge 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 3. Naturally, Figure 3 depicts a small portion of the entire DAG implemented in the study.

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 i_{th} 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 3, 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)

is shown in Table 1 and Table 2.

All CPDs for the DAG were obtained from the results of the Java Diagnostic Test. We first identified the concept being tested for each question. If the student answered the question correctly, then we considered the concept known. Similarly, if the student answered the question incorrectly, then we considered the concept unknown (not known). The probability of each concept being known, namely, p(ai= known), can then be determined. Moreover, we can also compute p(ai=known, Pi=known), i.e., the probability that the student correctly answers both the concept ai and the prerequisite concepts Pi. From p(ai=known, Pi=known), the desired CPD p(ai=known | Pi=known) can be obtained. Thereby, we can calculate every CPD for the entire Bayesian network.

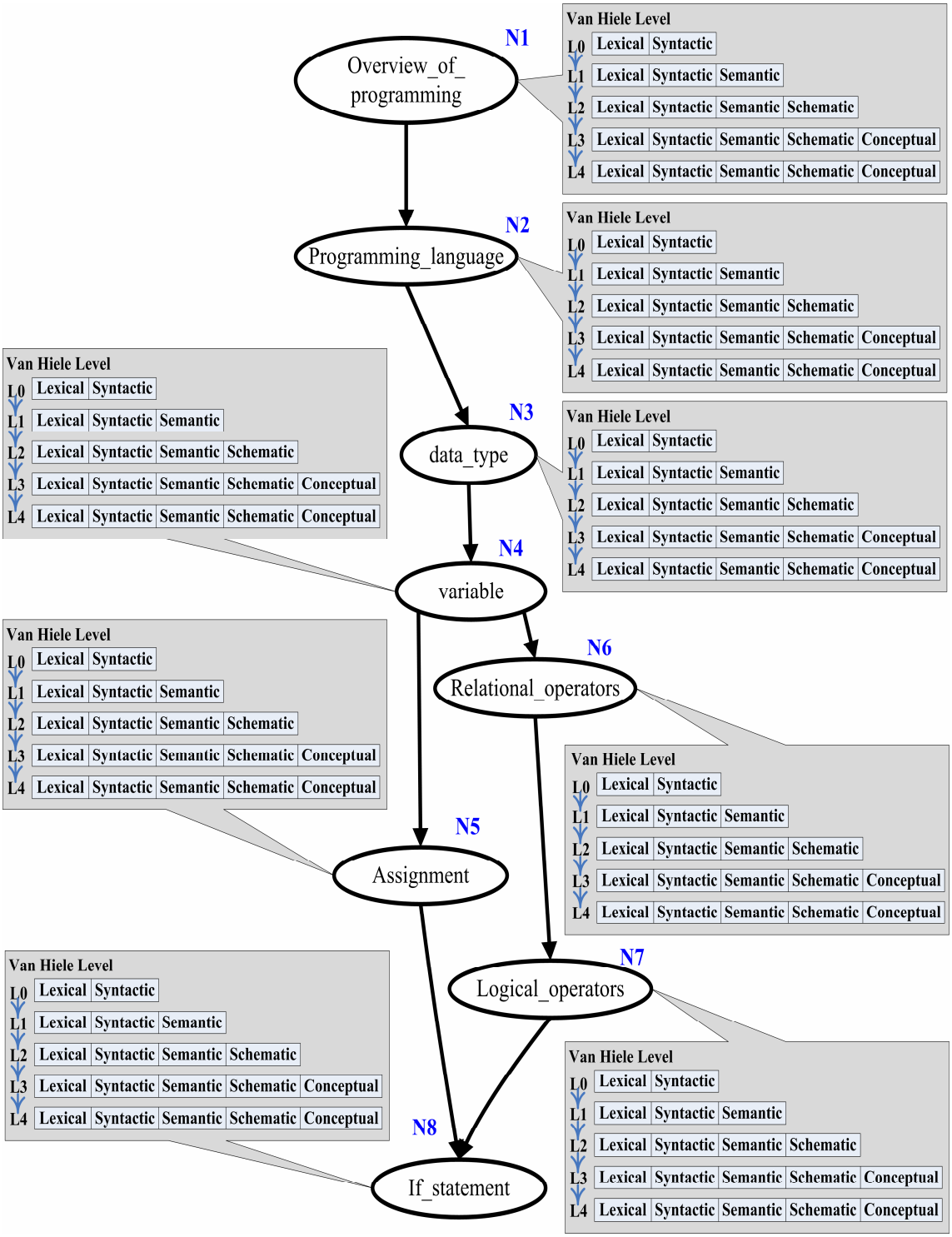

Figure 3 Sub-DAG for the if statement

Table 1. The CPD corresponding to the if-statement nodes within a van Hiele level

CP Item	Known	NRLN	P(NRLN)	CP Item Number	NRLN Number	P(NRLN | CP_Item)
N2L0	Y	N2L1	0.310344828	48	16	0.333333333
N2L0	N	N2L1		10	2	0.2
N2L1	Y	N2L2	0.448275862	18	10	0.555555556
N2L1	N	N2L2		40	16	0.4
N2L3	Y	N3L0	0.310344828	6	4	0.666666667
N2L3	N	N3L0		52	14	0.269230769
N3L0	Y	N3L1	0.275862069	18	10	0.555555556
N3L0	N	N3L1		40	6	0.15
N4L0	Y	N4L1	0.137931034	8	2	0.25
N4L0	N	N4L1		50	6	0.12
N4L1	Y	N4L2	0.137931034	8	4	0.5
N4L1	N	N4L2		50	4	0.08
N5L1	Y	N5L2	0.448275862	18	16	0.888888889
N5L1	N	N5L2		40	10	0.25
N5L2	Y	N5L3	0.344827586	26	16	0.615384615
N5L2	N	N5L3		32	4	0.125
N4L3	Y	N6L0	0.724137931	8	6	0.75
N4L3	N	N6L0		50	36	0.72
N6L0	Y	N6L1	0.655172414	42	32	0.761904762
N6L0	N	N6L1		16	6	0.375
N8L1	Y	N8L2	0.342926863	18	14	0.777777778
N8L1	N	N8L2		40	6	0.15
N8L2	Y	N8L3	0.412382567	20	18	0.9
N8L2	N	N8L3		38	6	0.157894737

The Categorical Sequence of the van Hiele Model

Figure 4 is a graphical representation of the results generated in Table 1. For example, the first piece of graph in Figure 4 indicates that students who mastered the prerequisite (N2L1) of N2L2

have better chance to pass N2L2 (cp(.N2L2| N2L1)= 0.555555556). The cp value is much higher than cp value for students who did not master the prerequisite (N2L1) of N2L2 but passing N2L2 test (cp= 0.4). The Bayesian network data validates the statement that instruction in this sequence promoted the acquisition of the next higher category in a van Hiele level in computer programming learning. This further testifies that Bayesian networks are useful for both inferential exploration of previously undetermined relationships among variables as well as descriptions of these relationships upon discovery.

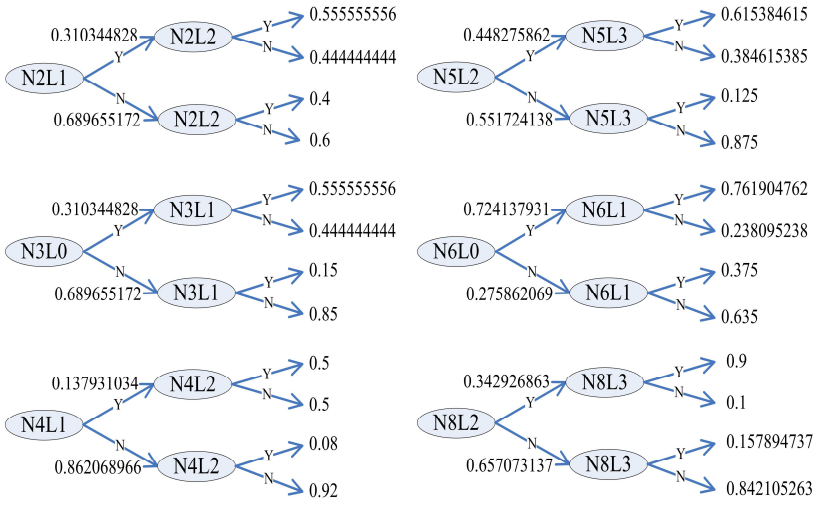

Figure 4. The DAG representation of Figure 2

The researcher found that instruction in this sequence promoted the acquisition of the next higher level in computer programming learning. Common programming mistakes committed by students also revealed their faulty explanation stances if these five levels of thought were not modeled properly. The five sequential instructional steps will be integrated into the proposed combined model to serve as a guiding framework to develop a rich and flexible web-based environment for Teaching and Learning Computer Programming languages.

The Level Sequence of the van Hiele Model

Table 2 shows the CPD corresponding to the if-statement nodes across van Hiele levels. Figure 5 is a graphical representation of the results generated in Table 2. For example, the second piece of graph in Figure 5 indicates that students who mastered the prerequisite (N2L3) of N3L0 have

Table 2 The CPD corresponding to the if-statement nodes across van Hiele levels

CP Item	Known	NRLN	P(NRLN)	CP Item Number	NRLN Number	P(NRLN | CP_Item)
N1L0	Y	N1L0	0.655172414	58	38	0.655172414
N1L0	Y	N2L0	0.827586207	38	34	0.894736842
N1L0	N	N2L0		20	14	0.7
N2L2	Y	N2L3	0.103448276	26	2	0.076923077
N2L2	N	N2L3		32	4	0.125
N2L3	Y	N3L0	0.310344828	6	4	0.666666667
N2L3	N	N3L0		52	14	0.269230769
N4L2	Y	N4L3	0.137931034	8	0	0
N4L2	N	N4L3		50	8	0.16
N4L3	Y	N5L0	0.310344828	8	2	0.25
N4L3	N	N5L0		50	16	0.32
N4L3	Y	N6L0	0.724137931	8	6	0.75
N4L3	N	N6L0		50	36	0.72
N6L2	Y	N6L3	0.275862069	32	12	0.375
N6L2	N	N6L3		26	4	0.153846154
N6L3	Y	N7L0	0.517241379	16	14	0.875
N6L3	N	N7L0		42	16	0.380952381
N7L2	Y	N7L3	0.103448276	0	0	0
N7L2	N	N7L3		58	6	0.103448276
N5L2	Y	N5L3	0.344827586	26	16	0.615384615
N5L2	N	N5L3		32	4	0.125
N5L3, N7L3	Y,Y	N8L0	0.264301757	4	2	0.5
N5L3, N7L3	N,Y	N8L0		2	0	0
N5L3, N7L3	Y,N	N8L0		16	6	0.375
N5L3, N7L3	N,N	N8L0		36	8	0.222222222

better chance to pass N3L0 (cp(.N3L0| N2L3)= 0.666666667). The cp value is much higher than cp value for students who did not master the prerequisite (N2L3) of N3L0 but have passed N3L0 test (cp= 0.269230769). Every piece of graph in Figure 5 will testify a level sequence case appeared in Figure 3. The Bayesian network data again can validate the statement that instruction in this level sequence promoted the acquisition of the next higher level knowledge in a van Hiele model for computer programming learning.

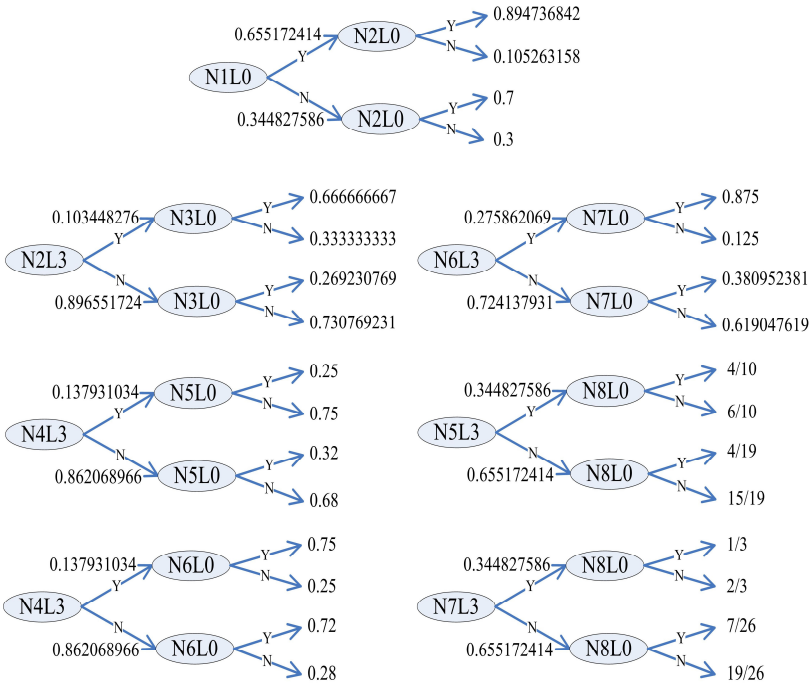

Figure 5 The DAG representation of Table 2

From the structure of the knowledge base and the local distributions learned from the data (Table 1 and Table 2), a BN is automatically built to infer if each topic is learned and infer the next topic to be learned.

Causal Bayesian Networks

A causal Bayesian network is a Bayesian network where the directed arcs of the graph are interpreted as representing causal relations in some real domain. The directed arcs do not have to be interpreted as representing causal relations; however in practice knowledge about causal relations is very often used as a guide in drawing Bayesian network graphs, thus resulting in causal Bayesian networks.

In Figure 5, there are two possible children nodes, N5L0 and N6L0, for students to learn once they have passed the N4L3 test. By comparing the two cp's values along the directed arcs, one can suggest the next reasonable learning path for the student. Looking up Table 2, it can be observed that

$P(N5L0 \mid N4L3) = 0.25$ and
$P(N6L0 \mid N4L3) = 0.75$,
and since $0.75 > 0.25$,

therefore, the next learning node to be dispatched is N6L0 because the relationship between N4L3 and N6L0 is more solid than that of N4L3 and N5L0.

Bayesian Networks for Diagnostic Applications

N8L0 has two parent nodes, N5L3 and N7L3. If a student fails to pass N8L0, we are in an embarrassing state to determine the failure of passing which node (N5L3 or N7L3) has strong causal relationship to interpret the failure of N8L0.

$$P(\overline{N5L3} \mid \overline{N8L0}) = \frac{P(\overline{N8L0} \mid \overline{N5L3})P(\overline{N5L3})}{P(\overline{N8L0})}$$

$$= \frac{15/19 * 0.655172414}{0.735698243} = 0.703061866 \quad \text{and,}$$

$$P(\overline{N7L3} \mid \overline{N8L0}) = \frac{P(\overline{N8L0} \mid \overline{N7L3})P(\overline{N7L3})}{P(\overline{N8L0})}$$

$$= \frac{19/26 * 0.655172414}{0.896551724} = 0.534023669$$

Since $P(\overline{N5L3} \mid \overline{N8L0}) = 0.703061866$ and, $P(\overline{N7L3} \mid \overline{N8L0}) = 0.534023669$, the BN can draw a conclusion to state that there is a greater possibility that the failure to master N5L3 result in the failure of N8L0 learning.

The DAG (directed acyclic graph) of Java Curriculum Content

Once the training data have been further analyzed and examined, a complete Java curriculum

content and instruction sequence can be determined. Figure 6 is a DAG representation of the entire Java Curriculum Content.

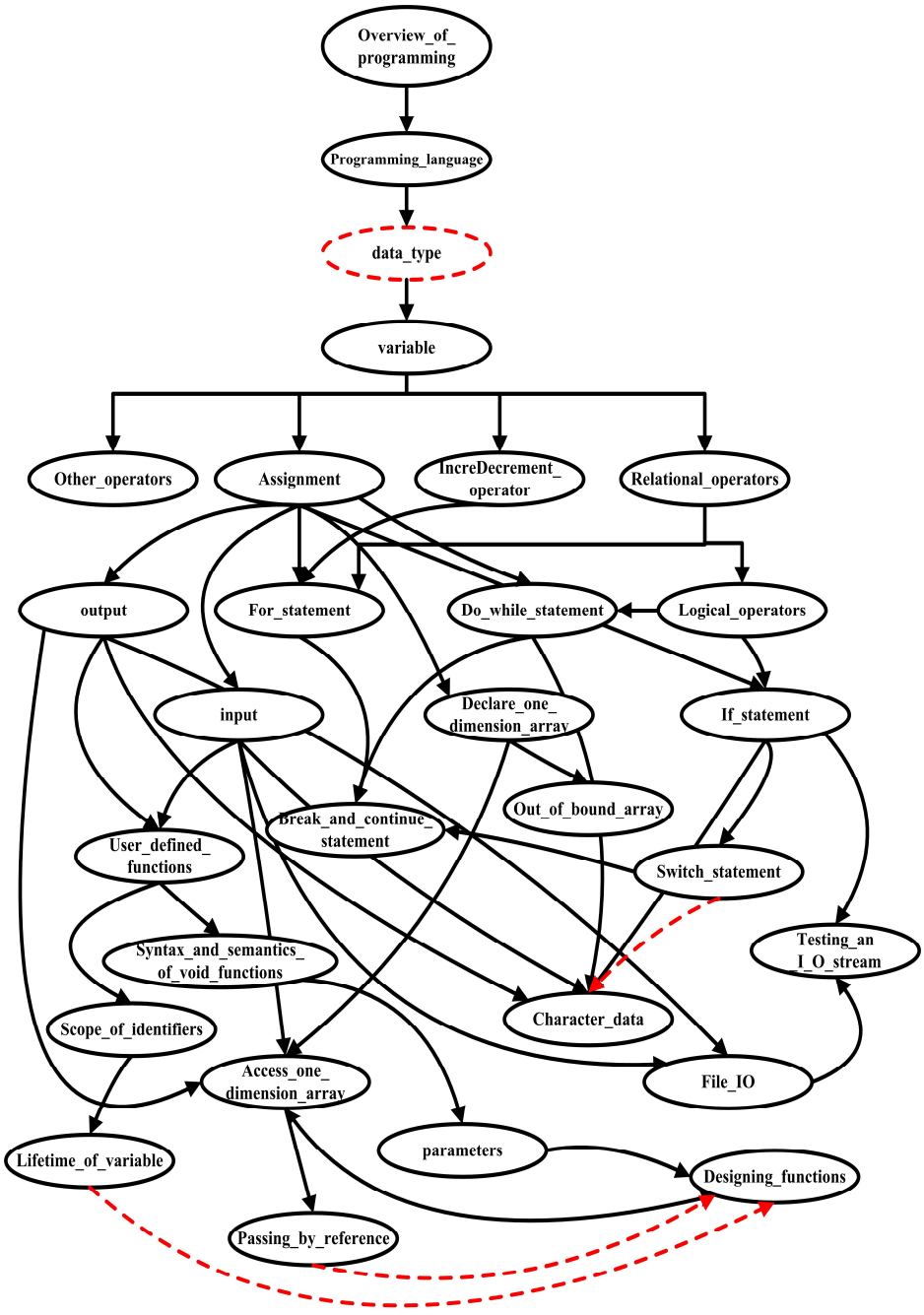

Figure 6. The directed acyclic graph of Java Curriculum Content

A diagnostic test constructed based on the suggested curriculum structure will have good chance

to be employed as the base to build an intelligent tutoring system for learning Java language.

A Practical Model for Applications

To help engineering educators wisely utilize the information described in this paper, we suggest the following approach be taken to design sound curriculum content and sequence:

1. Hold an expert roundtable discussion to roughly determine a set of knowledge concepts required for a course.
2. Manually construct the course DAG similar to Figure 6 with the aid of the course textbook.
3. Develop a diagnostic test to have test questions which cover every cognitive category for every level of understanding in the entire curriculum structure as shown in Figure 3.
4. Extensively conduct the test and collect sufficient Bayesian training data.
5. Analyze and use the Bayesian training data to trim the unrelated content and adjust the logical sequence for learning. Once the process is completed, a new course DAG similar to Figure 6 will be produced.
6. Group the related knowledge concepts into chapters according to their sequences appearing on the course DAG.

The approach formulated in this paper encompasses the van Hiele Model, the cognitive model, and Bayesian network, which has proven to be a successful model for educators and content experts to design a pedagogically sound curriculum content and sequence for students to intelligently navigate in any online course material.

Conclusions and Future Work

This paper discusses a new architecture of designing a van Hiele-based intelligent tutoring system for computer programming using Bayesian technology. The main focus is centering on the explicit curriculum structure and the way to use Bayesian training data for diagnostic and recommending 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, by utilizing Bayesian network techniques in modeling the programming bugs, this study will help us to design a complete Java curriculum content and instruction sequence.

The system provides remote access to take the diagnostic test and based on the overall picture of the test result, the system will provides the learner with intelligent navigation support, recommendation, and integrates the features of an electronic hypermedia textbook with intelligent tutoring tactics. The system can propose learning goals and guide users by generating reading sequences for them.

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. Allen Tucker. (2003). A Model Curriculum for K-12 Computer Science. Final Report of the ACM K-12 Education Task Force Curriculum Committee. ACM.
2. Bransford, J.D., Brown, A.L., and Cocking, R.R.(2000). How People Learn: Brain, Mind, Experience, and School. Washington, D.C.:National Academy Press.
3. Resnick, M. (1995). New paradigms for computing, new paradigms for thinking. In A. diSessa, Hoyles, C., & Noss, R. (Eds.), Computers and Exploratory Learning (pp. 31-43). New York: Springer-Verlag.
4. Papert, S. (1980). Mindstorms: children, computers, and powerful ideas, Basic Books.
5. Ebrahimi, A. (2008) Empirical study of errors by novice programmers and design of visual plan construct language (VPCL) Polytechnic University, Brooklyn, New York.
6. Kanemune, S., Nakatani, T., Mitarai, R., Fukui, S., and Kuno, Y. (2004). Dolittle - Experiences in Teaching Programming at K12 Schools. The Second International Conference on Creating, Connecting and Collaborating through Computing, IEEE, pp. 177-184.
7. Kelleher, C. & Pausch, R. (2005). Lowering the Barriers to Programming: a survey of programming environments and languages for novice programmers, ACM Computing Surveys.

8. 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 (ICALT2006), pp. 114-116.
9. van Hiele, P. M. (1986). Structured and insight: a theory of mathematics education. Orlando, Fl: Academic Press.
10. 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.
11. 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.
12. 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.
13. Chen, J. (2005).Designing a Web-based van Hiele Model for Teaching and Learning Computer Programming to Promote Collaborative Learning, The 5th IEEE International Conference on Advanced Learning Technologies (ICALT2005), pp. 163-166.
14. Bonar, J. & Soloway, E. (2001). Uncovering Principles of Novice Programming Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, pp. 10 – 13.
15. 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.
16. E. Millán, M.Trella, J.L. Pérez-de-la-Cruz and R. Conejo. (2000). Using Bayesian Networks in Computerized Adaptive Tests. In Manuel Ortega and José Bravo (eds.): Computers and Education in the 21st Century, Springer Netherlands, pp. 217-228.
17. Millán, E., & Pérez-de-la-Cruz, J. L. (2002). A Bayesian Diagnostic Algorithm for Student Modeling and its Evaluation. User Modeling and User Adapted Interaction, 12, 281-330.
18. Millán, E., Pérez-de-la-Cruz, J. L., & Suárez, E. (2000). An Adaptive Bayesian Network for Multilevel Student Modelling. In Lecture Notes in Computer Science 1839. Proceedings of 3rd International Conference on Intelligent Tutoring Systems ITS´2000 (pp. 534-543). Berlin Heidelberg: Springer Verlag.
19. Fenton, N. E. and Neil, M. (2001), Making Decisions: Using Bayesian Nets and MCDA. Knowledge-Based Systems, Vol. 14(7), pp. 307-325.
20. Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA.
21. Wong, S.K.M. and Butz, C.J. (2001). Constructing the Dependency Structure of a Multi-Agent Probabilistic Network, IEEE Transactions on Knowledge and Data Engineering, 13(3): pp. 395-415.
22. Butz, C.J., Hua, S. and Maguire, R. (2006). A web-based bayesian intelligent tutoring system for computer programming, International Journal on Web Intelligence and Agent Systems, 4, pp. 61–81.

23. Wong, S.K.M., Butz, C.J, and Wu, D. (2000). On the Implication Problem for Probabilistic Conditional Independency, IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 30(6), pp.785-805.