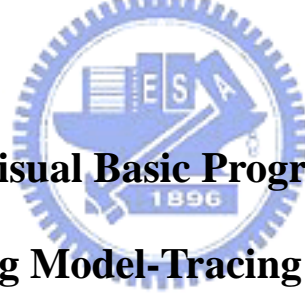


國立交通大學

理學院網路學習學程

碩士論文

應用模式追蹤技術之問題導向式 VB 程式設計教學系統



**A Problem-Based Visual Basic Programming Tutoring System
Using Model-Tracing Approach**

研究生：陳怡靜

指導教授：曾憲雄 博士

中華民國九十四年六月

應用模式追蹤技術之問題導向式 VB 程式設計教學系統
**A Problem-Based Visual Basic Programming Tutoring System
Using Model-Tracing Approach**

研究生：陳怡靜

Student : Yi-Ching Chen

指導教授：曾憲雄 博士

Advisor : Shain-Shyong Tseng

國立交通大學

理學院網路學習學程

碩士論文



National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Degree Program of E-Learning

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

應用模式追蹤技術之問題導向式 VB 程式設計教學系統

研究生：陳怡靜

指導教授：曾憲雄 博士

國立交通大學

理學院網路學習學程

摘要

隨著資訊科技和網際網路的發展，以 e-Learning 支援學生的學習已經變得愈來愈重要。特別是在程式設計的學習上，初學者通常缺乏適當的輔助工具。因此，我們提出應用模式追蹤技術之問題導向式 VB 程式設計教學系統。此模式追蹤技術是在學生解決問題的過程中，透過追蹤其認知行為，挑出學生可能犯錯的部份予以指正，以免一再犯同樣的錯誤，即諺語所謂“前車之鑑”的道理。再者，我們利用框架式知識表示法讓領域專家可以利用知識的誘發，知識事實的驗證，進而達到知識分享與再利用的目的。在這篇論文中，我們提出兩階段的系統架構，第一階段是建構框架式的 VB：包括三個過程，相關知識類別模組過程，框架式知識表示過程，和追蹤可能問題解決路徑的過程。這個階段的目的是在教導學生之前，先架構程式設計的知識。第二階段是應用模式追蹤技術之 VB 教學系統：包括四個過程，問題呈現過程，模組分解成框架過程，框架知識推理過程，和適當的回饋過程。這階段的目的是在學生的學習期間做程式設計知識的推理。搭配雛型系統的設計，其貢獻是提供學生們一套互動式問題導向之 VB 程式設計學習環境。

關鍵詞：模式追蹤，框架式知識，本體論，問題導向式學習，Visual Basic 程式設計教學

A Problem-Based Visual Basic Programming Tutoring System Using Model-Tracing Approach

Student : Yi-Ching Chen

Advisor : Dr. Shian-Shyong Tseng

Degree Program of E-Learning

College of Science

National Chiao Tung University

Abstract

With the development of computer technology and Internet, e-Learning has become more and more important to support student's learning. Novice students usually lack adequate assisting tools especially in programming learning. Therefore, we propose the problem-based Visual Basic programming tutoring system using Model-Tracing approach. The Model-Tracing is based on an idea of tracing students' cognitive behavior during the problem solving process. As the proverb said, "One man's fault is other man's lesson". Besides, we employ the Frame-based knowledge representation which domain experts can do knowledge elicitation, validation of knowledge facts, further knowledge shared and reused. In this thesis, the two-phased system architecture is proposed. ***Phase 1: Frame-based Visual Basic constructing***: This phase includes three processes which consist of Related knowledge class modeling process, Frame-based knowledge representation process and Tracing possible problem-solving path process. The purpose of this phase is to construct programming

knowledge before teaching students. ***Phase 2: Visual Basic Model-Tracing tutoring system:***

This phase includes four processes which consist of Quiz display process, Models decomposition into frames process, Frame-based knowledge reasoning process, and Adaptive feedback process. The purpose of this phase is to reason programming knowledge during student's learning. With designed prototype system, it contributes to construct an interactive Problem-Based VB programming learning environment for students.

Keywords: Model-Tracing, Frame-based knowledge, Ontology, Problem-Based Learning,

Visual Basic programming tutoring.



誌謝

回顧碩士二年來半工半讀的日子，雖然論文的寫作經常是利用業餘時挑燈夜戰，備極辛苦，但即使是在有限的時間內仍期許自己要盡力而為，莫找藉口以輕忽怠惰，唯有學會時間管理和做好學習計劃，才能讓論文的内容和品質達到應有的水平。然而這篇論文得以順利完成，首先要感謝我的指導教授曾憲雄博士，由於教授細心的教導和勉勵，使我的研究架構愈來愈明確，思緒和邏輯也逐漸精進，整個研究的過程中著實獲益良多。

另外感謝論文口試委員：莊祚敏教授、葉耀明教授、曾秋蓉教授的指教，他們對這篇論文提供許多寶貴意見，讓此篇論文顯得更有意義與價值，深表感激。此外，KDE 實驗室的各位學長：蘇俊銘學長和楊哲青學長提供許多意見和指正，特別是翁瑞鋒學長在這篇論文寫作的過程中花費了許多心力與精神，透過和學長持續的研究探討讓此篇論文得以臻至完善。另外同學們平時的協助和互相砥礪，以及助理雅惠提供很多行政支援，真的很感謝他們的幫忙。

這些日子來每週台北-新竹往返上課，總是半夜才到家，讓母親擔心覺得很過意不去，但是揮汗播種、終將歡喜收割，家人默默的支持是我前進的最大動力，使我能無謂艱難順利拿到碩士學位。再者，任職學校同事在二年中，縱使我倦容滿面，仍能無私的包容，和一群可愛的學生，在我這段授課的日子中亦能自律，努力用功，讓我能無牽掛的致力於我的學業。總之，一個人的功成名就其背後是許多人的成全和付出，謹以此時一點點的成就和喜悅與他們分享。最後也要感謝閱讀此篇文章的您，希望它能對您的研究有些許的幫助或啟發，這將是我最大的期盼與貢獻。

Table of Contents

摘要.....	i
Abstract.....	ii
誌謝.....	iv
Table of Contents.....	vi
List of Definitions .	vii
List of Examples .	vii
List of Algorithms .	viii
List of Figures .	ix
List of Tables .	xi
Chapter 1. Introduction .	1
Chapter 2. Related Works .	4
2.1 Traditional Computer Based Tutoring Systems .	4
2.2 Adaptive Computer Based Tutoring Systems .	8
Chapter 3. System Architecture .	19
Chapter 4. Knowledge Constructing .	22
4.1 Frame-based Knowledge Representation .	22
4.2 Programming Model-Tracing .	31
Chapter 5. Knowledge Reasoning .	51
5.1 Reasoning Framework .	51
5.2 Model-Tracing Tutoring Scenarios.....	52
Chapter 6. Prototype System Implementation .	60
Chapter 7. Conclusion and Future work .	66
Bibliography .	68

List of Definitions

Definition 4.1: VB Ontology.....	23
Definition 4.2: Frame-based knowledge.....	26
Definition 4.3: Model-Tracing of Quiz.....	44



List of Examples

Example 4.1: Ontology template for VB.	23
Example 4.2: VB Domain Ontology.	24
Example 4.3: Frames ontology for partial repetition structure of VB.	26
Example 4.4: Frames ontology for partial selection structure of VB.	27
Example 4.5: The sample quiz with independent control loop counters.	42
Example 4.6: Model-Tracing template	44
Example 4.7: The models of possible problem-solving path.	46
Example 4.8: An example of possible errors for sample quiz.	49
Example 5.1: A sample quiz with the partial correct solution.	56



List of Algorithms

Algorithm 4.1: Model-Tracing Construction.....	48
Algorithm 5.1: Knowledge Reasoning.....	52



List of Figures

Figure 2.1: An example for Model-Tracing algebra tutoring.....	10
Figure 2.2: An example for simple filler in frame	16
Figure 3.1: PROBLEM tutoring system framework	19
Figure 4.1: Ontology template of Visual Basic knowledge	23
Figure 4.2: Ontology of partial Visual Basic programs	24
Figure 4.3: Frames Ontology for partial repetition structure of VB.....	26
Figure 4.4: Frames Ontology for partial selection structure of VB.....	27
Figure 4.5: Frame-based Knowledge for Single If/Then structure with	28
Figure 4.6: Rules generated from the Frame.....	29
Figure 4.7: Phase 1—Frame-based Visual Basic Model-Tracing.....	32
Figure 4.8: Related Knowledge Class Modeling.....	34
Figure 4.9: Frame-based knowledge acquisition process	35
Figure 4.10: Required Frames with correct solution	40
Figure 4.11: An example of reusing Frame-based knowledge.....	42
Figure 4.12: The Model-tracing of possible problem-solving path.....	44
Figure 4.13: The models of possible problem-solving path	46
Figure 4.14: An example of possible errors for sample quiz	49
Figure 5.1: Knowledge Reasoning Framework for Bubble Sort quiz	51

Figure 5.2: Phase 2—Visual Basic Model-Tracing Tutoring System..... 56

Figure 5.3: An overview of Visual Basic tutoring contents..... 55

Figure 5.4: An example for the incorrect answer came from the student..... 57

Figure 5.5: Frame-based knowledge with the rules reasoning..... 58

Figure 5.6: An example of Model-Tracing for possible errors 59

Figure 6.1: Prototype tutoring system of PROBLEM 60

Figure 6.2: The flowchart of constructing quizzes for Model-Tracing tutoring..... 62

Figure 6.3: Students login web page 63

Figure 6.4: A quiz web page..... 63

Figure 6.5: Single For structure web page 64

Figure 6.6: Model-Tracing tutoring web page..... 64



List of Tables

Table 4.1: A generic frame for Repetition structure	36
Table 4.2: Frame-based knowledge for For/Next repetition structure	38
Table 4.3: Frame-based knowledge for Nested For/Next repetition structure	39
Table 4.4: Procedure attachment for For/Next repetition structure.....	41
Table 5.1: Ten control structures of Visual Basic programming language	56



Chapter 1. Introduction

As we know, technical advancements bring about changes to teaching methods. In the early days, traditional programming learning assisting tools are used for textbooks, on-line help, e-Books or traditional learning systems, such as CAI, which relied on the student's answer to present a page of text or graphics. With Internet proceeding, a lot of online learning contents are now HTML web pages, internet forums provide on-line Q&A and search engines which can be used for searching related information. E-Learning emphasizes that learning could be any time, any where through any electronic device if supported. Although those teaching contents can assist novice students in learning, it's still difficult for students to find desired information. Students may not know how to ask the right questions about programming. This is not helpful to novice students in programming learning if the system just gives the teaching contents with the medium changing from static textbooks to dynamic presentation. Moreover, the programming learning is a difficult task for many students, and provides feedback to students in a way, that is responsive to individual needs, is another sticky task for many teachers as well. Nevertheless, all of the aforementioned learning methods share a common defect, lack of interaction.

We give an overview over the evolution of instructional system in human learning as well as about Model-Tracing tutoring approach for possible problem-solving. Model-Tracing can enable us to trace every student actions, has brought us to more precise evaluation of student knowledge, and allows students to follow their usual way of thinking about the domain. Some previous tutoring systems have been in possession of remediation capability following this

approach. For instance, cognitive algebra tutor [3], algebraic functions [9] or LISP Tutor [1] consider student's thinking logic and knowledge during a problem solving process. Nevertheless, all of the aforementioned tutoring systems share a common defect, which is difficult to manage huge production rules.

For solving above defects, we propose a **PR**ogramming **Ontology-Based Learning Environment** using **Model-tracing approach (*PROBLEM*)** tutoring system to instruct students the Visual Basic (VB) programming. These teaching contents are specified by Problem-Based Learning (PBL). Previous researches have proposed five processes of PBL including clearly defining the problem, developing possible hypotheses, gathering related information, performing learning actions, and assessing the solution [21][30]. PBL also needs students collaboratively to identify and analyze problems, and generate solutions. The tutoring system focused on providing a platform for performing learning actions, and assessing the solution. There are consisting of two phases in our system: ***Phase 1: Frame-based Visual Basic constructing***: it aims to construct programming knowledge by Model-Tracing approach. There are three processes in first phase: Related knowledge class modeling process, Frame-based knowledge representation process and Tracing possible problem-solving path process. In first phase, an adaptive programming tutoring system is used for Frame-based knowledge representation and Model-Tracing tutoring approach when students learn how to program. Frame is the structure for organizing knowledge. Thus, we propose the Frame-based knowledge for VB programming associated with a given Visual Basic Programming Ontology to support the tutoring system. ***Phase 2: Visual Basic Model-Tracing tutoring system***: it aims to provide a Problem-Based Learning in VB programming by knowledge reasoning. There are four processes in second phase: Quiz display process, Models decomposition into frames process, Frame-based knowledge reasoning process, and Adaptive feedback process. In second phase, students interact with tutoring system via pre-designed quizzes of programming.

The purpose of tutoring system is to integrate Model-Tracing approach with Frame-based knowledge reasoning for guessing student's possible programming errors. Finally, the adaptive learning contents could be provided to students.

We contribute to construct a smarter learning environment for any students interested in programming. Thus, a key consideration for this tutoring system is the guidance and interactive for students by questions whenever he/she deviates from the adaptive programming solutions. Several contributions of our work have been summarized as follows:

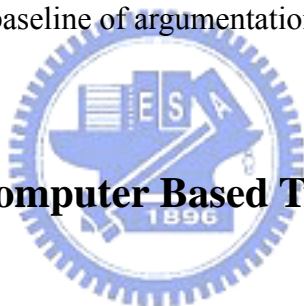
1. **Conduct interactive Problem-Based VB programming learning scenarios.**
2. **Combine Model-Tracing tutoring with Frame-based knowledge reasoning.**
3. **Construct Ontology-based VB programming learning environment.**

This thesis is organized as follows. The next chapter presents related works about the traditional computer tutoring systems and learning background of programming, with the worth of Model-Tracing tutoring system, and literatures on the subject of domain ontology with Frame-based knowledge representation. Chapter 3 gives an overview of the system architecture which consists of two phases. The first phase is knowledge construction, as for technologies and definitions of related concepts, including VB domain ontology, Frame-based knowledge representation and construction, Rule-based knowledge generation, and describing how to construct the Frame-based programming Model-Tracing as shown in Chapter 4. In the second phase, we give an example for practical operation in the tutoring system including knowledge reasoning and tutoring as shown in Chapter 5 in detail. Besides, the prototype implementation of the tutoring system is given in Chapter 6. The conclusion and future work are presented in the final chapter.

Chapter 2. Related Works

This chapter gives an overview over the progression of instruction system in human learning. Section 2.1, we will discuss traditional computer based tutoring systems with Problem-Based Learning (PBL) educational strategy in order to think over their common defect, that is students became passive learning and lack of interaction. Section 2.2, we will discuss certain adaptive computer based tutoring systems using Model-Tracing approach and Frame-based knowledge representation in order to improve the interactive programming learning. Those efforts are the baseline of argumentation for this thesis.

2.1 Traditional Computer Based Tutoring Systems



As we know, most computers based learning systems whatever tutorial systems, programmed instruction, simulations, and mind tools basically follow two pedagogical methods for teaching and learning. The researcher Jonassen (1996) mentioned that *Instructions* see the student as a passive human being, who can absorb a lot of information given by instructors or instructional media and translate it into knowledge. *Construction* see the student as actively constructing knowledge by gathering information from his/her surrounding and combining it with previous knowledge.

Computers are now being used for delivering educational contents and also as a means of communication between students and their teachers or tutors. With current high performance computers, educational contents can be delivered in various formats (i.e., multimedia) ranging

from simple texts to video clips. The CAI (Computer Assisting Instruction) systems were early generation tutors which tended to drill and practice learning. Some of early CAI media are like CD-Title or Web-Title. Conventional ITSs' (Intelligent Tutoring Systems) chief defect is to consider only the final results, i.e. the student's answer. CAT (Computer Assisted Testing) is an example. Through recognizing the deficiencies of traditional CAI systems and conventional ITSs, we find that there is a common feature: these systems all tend towards *instructions*, i.e. *teacher-centered* teaching style.

Previous researches had developed lots of framework or computing system for solving puzzling programming during students' learning [8][15][18][26][33][49][50][60]. Programming learning examines various learning factors, briefly shown as follow: required knowledge of the syntax and semantics of the programming language, composing new and comprehending existing programs, testing and debugging of program, etc. In addition, debugging tools just find the causes of program's fault which students made, so as to display the learning contents. However, previous researches didn't trace learning processes and then students' programming knowledge could not be stored and reused.

A lot of bugs can be prevented following several simple guidelines from the compiler, yet there will still be some fatal errors, such as logic errors, unfound. As we know, conventional debugger or conventional IDEs, such as Microsoft Visual BASIC development environment, have built-in debuggers that allow programs to be preliminary traced [26]. Besides, there are several special tools such as program animators [49][50] which can animate a student's program to provide more functional help than "standard" debuggers. Moreover, the student can even view the animation of the code as each line executes [8]. Friedman's research [24] has gathered some literatures to prove the extensive application using PBL educational theory, such as beginning in medical education, science education, but expanding into other ill-structured and complex disciplines, and easily transferable to other disciplines

like Computer programming and English composition, etc.

These previous researches take into consideration the cognitive skills, such as required problems understanding, analysis and design of the solution require domain, strategic knowledge, and practical knowledge of the programming language. In traditional teaching environment, teachers are used for instructing students by examples. Flowchart of programming is used for interpreting programs when human teacher teaching. Previous research gave an animation mechanism of flowchart as well as tools for simulation designed for use by novice programmers [15]. These problem solving activities are more efficient ways of utilizing aiding tools. The lack of aiding tools cause students' common defects while programming. For example, the student can not make sense of what this help is saying. The student may be misunderstanding the compiler's information since it is not suitable to novice students. The student often makes a mistake about the grammars of syntax structure, don't know how to debug, and even don't know how to ask questions either.

Accordingly, it is worth speaking the problem-based learning in the programming learning more systematically. All this will have a great help for assembling a PBL system. SOLVEIT [26] system also consists of problem formulation and these problem description editor. Traditional topics such as selection and iteration may be taught using object-oriented concepts [8]. And then these aided tools may be similar to common IDE (Integration Development Environment), other program animators or visual systems have been developed to help students in particular programming language learning. A tutorial tool [60] allows the execution of program to be graphically displayed. These might include solving a problem, designing some pseudo code, implementing an algorithm in a programming language, or testing a program.

In other words, the above mentioned tutoring system provided merely a learning tool for students. Although they are all useful for different student's level, there is no real interaction during the learning process. This is an important factor especially for programming. Due to the change of instructional manner we think over some advantages and disadvantages for students' studying and future understand about the development of computer supported learning. We find that some systems tend towards *student-centered* learning style [43]. Students take responsibility for learning, rather than passively receiving it. Students use resources to construct their own knowledge, based on needs. Students "learn how to learn" by developing problem-solving skills, critical thinking. Therefore, PBL just is consistent with the methodological view of the theories of *constructivism* and *situated learning*. The student can act on problems, and reflectively thinking about "what do we know", "what do we need to know", and "how do we find in this process". Previous research like CROCODILE [41] cooperative systems focused on the activities of a group. They emphasized that an activity-oriented approach to representing knowledge is the most suitable for the support of learner-centered learning in a collaborative setting.

Problem-based learning means to solve mainly ill-structured problems more than well-structured problems and authentic learning contexts. The discipline way of PBL has already applied to some medical student's learning. PBL poses meaningful, contextualized, authentic situations, and provides resources, guidance, and instruction to students as they develop subject knowledge and problem-solving skills [39]. If we just create problems with manual filling by experts, not all entries are completely accurate, lack of intuition and time consuming task. Therefore, gathering more knowledge utilizes knowledge acquisition, representation and extraction methods. However, the acquisition of the VB programming knowledge from domain experts is a very hard for knowledge engineer. They have considered that to acquire those expertises which can use a lot of knowledge acquisition techniques.

We can briefly say that concepts usage through the appearing of the question. Some researches made good use of PBL in collaborative learning. PBL is normally conducted within small discussion groups of students facilitated by a faculty tutor [7]. Additionally, CHALLENGE [52] provided an authoring tool, in which students can even create their own scenarios for problem solving. The use of computers in education is currently gaining momentum along with the increasing popularity of the Internet. At the same time, the Internet allows the interaction between students and their teachers, tutors, or other students without being restricted by their geographical locations or time. Such additional information processing power and interactivity allow developing a computer based education system both in terms of pedagogical and technological aspects. In other words, the contents of e-Learning may be suitable for matching PBL issues or combining student-centered teaching with e-Learning [43]. Thus, problem-based learning is more suitable for programming learning than traditional learning. It is adapted for situating teaching in certain domains, such as medical course, programming course, etc. But using PBL is only changing another manner of learning. As for coding a program, students still don't know how to program with the syntax of VB programming language. The student real needs are adaptive assisting in programming process. Accordingly, previous researches have proposed another learning strategy, named Model-Tracing, which will gain the most profit from interaction as shown in next section.

2.2 Adaptive Computer Based Tutoring Systems

Model-Tracing has enabled the requirements by students. Model-Tracing comes from the facts that the diagnostic program merely traces model's execution and compares it to the student's activity to achieve the goal of interpretation of student's behavior. This learning system has enabled us to trace every student's action and that has brought us to more precise

evaluation of student's knowledge; especially the most activities are emphasized rather than contents in situated learning. The basic idea is to model possible problem solving which is used for situation learning theory and assumes that all of the student's significant mental states are available to the diagnostic program [6]. One of these states should correspond to the state generated by the student. Besides, the Model-Tracing can be a set of intelligent diagnostic technique which traces students' cognitive behavior, such as cognitive algebra tutor [3], algebraic functions [9] or program errors [1] during a problem solving process, and follow student's reasoning (or program logic), and not only consider final result, but also evaluate student's knowledge (or programming process) the same as LISP Tutor [1]. This tutor is used for teaching students in LISP programming with Model-Tracing methodology for tutoring is based on the ACT theory of skill acquisition. Student's problem-solving behavior can be interpreted and tutored by tracing their solution through the production rules [2]. In other words, Model-Tracing approach is well suitable to construct programming learning.

However, conventional Model-Tracing has lots of disadvantages. McArthur et al. has criticized that the designed learning contents in Anderson's LISP tutor may cause each error matching an incorrect rule, which will always present the same message to the student. Thus, several different tutoring systems make good use of Model-Tracing approach to improve these defects. Heffernan & Koedinger (2001) have proposed the Ms. Lindquist Tutor [32] which is used for teaching students in Algebra arithmetic. Model-tracing does not only trace the student's actions, but also carry on a running conversation completed with probing questions, worked examples, positive and negative feedback, follow-up questions in embedded sub-dialogs, and requests for explanation as to why something is correct. SlideTutor [16] is another system which is based on the Cognitive Tutor (CT) approach to teach visual classification problem-solving in Pathology. Ms. Lindquist has multiple different tutorial strategies which are better than the traditional tutoring system [7][32].

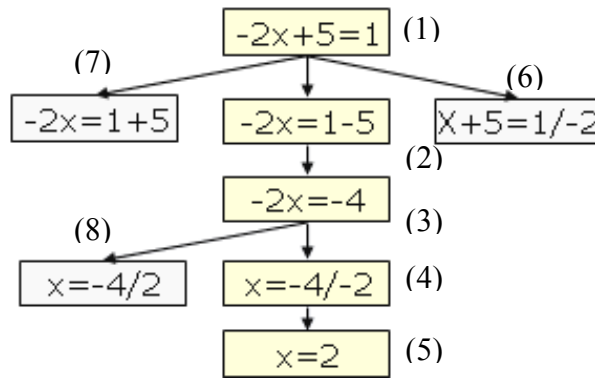


Figure 2.1: An example for Model-Tracing algebra tutoring.

At a given state of problem solving, rules which model the problem solving activities within the domain are selected and fired to predict the next state, such as Figure 2.1, the system traces the learning path and find those possible errors, i.e. node 6, node 7, or node 8. This knowledge came from domain experts. They predefined those facts and buggy rules which are fired by the goal of matching student's new state. The student is assumed to be using the rule that predicted the new state and the student model updated to reflect this. Any deviation between the student's solution and the systems can be acted upon and suitable action taken [50].

In sum, the learning activities play a fundamental role in determining learning outcomes and they determine how students engage with the various learning contents. Deek (1997) have considered how to combine PBL, web-based distributed education and a problem solving environment; moreover, Friedman. et al. (2002) have examined the design and implementation of problem solving tools used in programming instruction. Those tools are complementary with the fundamental theories of problem-based learning (PBL), the pedagogy and practices of distributed education environments [24]. Camp. (1996) indicated that over 30 years ago, Canada's McMaster University's School of Medicine began a program of instruction that was "*student-centered [and] problem-based, [in which] small group learning*

took shape". That is the core of problem-based learning. They proved the importance of PBL.

Model-Tracing provides reflectively thinking to the student. The student is given a problem and the tutor monitors the student input character by character. The tutor generates all the possible next characters using both correct and buggy production rules [46][50]. The aim of this tool is to help students think more deeply about the problem in question. It is one of the few specific tools available to support the teaching and learning of programming, and provides help in this important area of problem analysis and formulation, planning and designing a solution, and testing and delivering that solution, that is provided through all problem solving stages [26]. According to above viewpoints, the advantages of Model-Tracing are following usual ways of thinking about programming, like (1) experimenting by trial and error, and (2) testing hypotheses. Thus, we can use it well for programming instruction. However, we can clearly observe that most tutoring systems using Model-Tracing approach have to construct lots of production rules to inference their collectively facts or conditions. Another challenge for knowledge engineer is to learn how to reuse and manage these rules.

Not only those techniques are useful information for students, but also students need adequate knowledge to think how to solve puzzles out, for example, concerning program structure, logic and simple algorithms, object-oriented concepts, etc. All this is necessary to be a good programmer. Programming paradigms can be classified as procedural and nonprocedural. The distinguishing feature of the procedural paradigm is that the programmer must specify exactly how a problem solution must be coded. Traditional programming languages belong to the procedural language, such as FORTRAN, PASCAL, C, and LISP; make use of data stored in the form of arrays, records or structures. This method fails in representing complex relationships especially based on heuristics. The goal of nonprocedural programming is to have the programmer specify what the goal is and then let the system

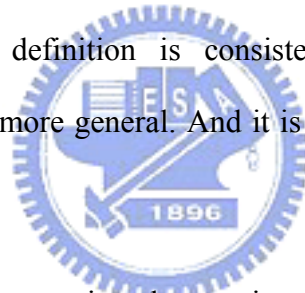
determine how to accomplish it. AI programs are used for knowledge structures to represent objects, facts, rules, relationships and procedures, such as SMALLTALK, JAVA, or VB, which belong to the Object-Oriented language [27][55]. An algorithm is a method of solving a problem in a finite number of steps. The terms algorithmic programming, procedural programming, and conventional programming are often used interchangeably to mean non-AI-type programs [27].

LISP Tutor [1][2][13][14] is used for teaching students in LISP programming. The acquisition of a complex skill like writing LISP code can be decomposed into the learning of an underlying set of production rules. Students in this study worked with the CMU Lisp Tutor [13] which is an instructional program that assists students as they complete LISP, Pascal and Prolog [2] programming exercises. The tutor is able to provide feedback in problem solving because it contains knowledge that allows it to generate solution step-by-step along with the student. The tutor contains a set of approximately 500 production rules for generating Lisp code collectively called the ideal student model. Each of these productions is an if-then rule that specifies a programming goal and actions to take to satisfy the goal.

We take a real-world problem into account the encoding information with hundreds or thousands of rules. The numbers of rules are difficult management. Especially for non-procedural programming languages, the tutor will construct huge rules to teach students how to program. Those rules will expend great endeavors and too complex to achieve the goal of teaching knowledge. The following section we will introduce how to implement the object-orient programming language using the Frame-based knowledge representation.

Regarding early programming language, a frame may be analogous to a record structure in a high-level language such as Pascal or an atom with its property list in LISP. In previous paper, in order to develop this knowledge base, the use of an ontology-based acquisition tool

is used for bring together knowledge of experts. The word Ontology is taken from philosophy where it means a systematic explanation of existence. In the Artificial Intelligent field, the most commonly used definition of ontology is from Tom Gruber [28], who announced that “*An ontology is an explicit specification of a conceptualization*”. This definition is most referenced in the certain literatures. Besides, later this definition is slightly modified by some researchers [58] for future understanding the ontology, who explained that ontology is defined as a formal specification of a shared conceptualization, which refers to an abstract model of some phenomenon in the world. The formal refers to the fact that the ontology should be machine-readable. The most advantage of ontology can enable the knowledge sharing and reuse. In the context of knowledge sharing, that is, ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general. And it is certainly a different sense of the word than its use in philosophy.



Some researches have mentioned certain ontology-based knowledge systems implemented in Jess which is Java expert system shell developed by Ernest Friedman-Hill and Protégé-2000 [45], which is a tool developed by KSL of Stanford University that can be used for constructing ontology. The subject of ontology is the study of the categories of things that exist or may exist in some domain, such as the role of ontology in information systems, as sketched in [29], is refined for knowledge. Ontology may be viewed as a catalogue of objects sorted according to the properties of the objects, e.g., concrete and abstract, existent and non-existent, real and ideal. A more complex ontology is a taxonomy in which classes of objects inherit properties from more general ones in a hierarchy [52][34].

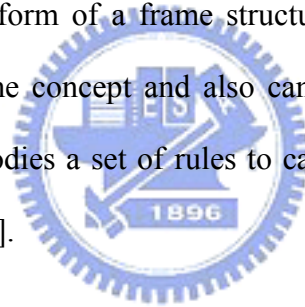
Frame, a notion originally introduced by Marvin Minsky. The knowledge representation by rules and frames is used. A frame consists of a series of slots, each of which represents a

standard property or attribute of the element represented by the frame. Its features include there is no limit on the number of slots in a frame, slots support fields or attributes, complex data structures can be easily represented, and incomplete knowledge can be handled to some extent conveniently. Frame-based ontology relations between objects can be expressed by stating restrictions or dependencies between classes in terms of frames. For instance, OKBC protocol [60] is a formal specification of frames for the development of Frame-based systems. An axiomatically ontology is a system of logical relations between objects with the full expressive power of a logical theory. And then Ontology in combination with logical formalisms provides a semantically-logical language in which logical relationships between objects can be expressed. This introduces the possibility to perform reasoning with objects and makes the ontology a tool for generating new objects. In addition, automated reasoning can be used for constraining domain specific semantics of possible future facts. Swartout and Tate's defined as "*Ontologies provide the necessary armature around which knowledge bases should be built*" [48]. Formally, ontology is the statement of a logical theory.

Besides, we will illustrate related applications about Frame-based knowledge which is suitable for implementing the data structure of program. EMC domain [55] was used for depicting the complex relationships, the validity of representing knowledge and manage huge amount of knowledge. That author of paper pointed out lots of demerits about predicate logic as well. We may take advantage of frame to define the same slots corresponding to fill in different slots value, in which can express situation difference. In 1989, a previous research has used for the Frame-based knowledge constructing their knowledge-based software development environment, called LISP-PAL [58], which is designed to assist interactively beginning programmers by supplying them with experienced expert programmer knowledge. Frames are popular because Frame-based modeling is similar to object-based modeling and intuitive for many users. The AKO (A Kind Of) is the relationship linking many frames and

slots. The advantages of Frame-based knowledge are shown in the following paragraph which refers to previous literatures.

Minsky (1975) proposed organizing knowledge into chunks called *frames*. These frames are supposed to capture the essence of concepts or stereotypical situations, for example being in a living room or going out for dinner, by clustering all relevant information for these situations together. Firstly, Fikes and Kehler (1985) offer a concise way to express knowledge in an *object-oriented* way. Secondly, by using only a fragment of first order logic, Frame-based systems may offer more efficient means for reasoning [42]. The complex relationships between parameters and existing modules in the electromagnetic environment can be easily stored and accessed using frame base system [34]. The Frame-based technique concept is represented to the form of a frame structure. Each slot of the frame contains a variant explanation of the same concept and also can be linked to other frames. A system using this technique also embodies a set of rules to calculate the most appropriate slot to be presented to a specific user [50].



Frames allow to describe domain knowledge in hierarchies of classes. Frames consist of slots, which can be symbol, numerical, linguistic variable, date and time. Frame can be joined with rules and procedures, processing determined events. Furthermore, ontology of the abstract inference rules is elicited. New domain facts, in combination with the ontology, can be used as an elicitation tool for more domain facts. These new facts instantiate implemented general knowledge rules, resulting in executable inference rules that can instantly deduce all consequences of the new facts. This implies that validation of new knowledge can be done instantly, and its impact on existing knowledge is directly accessible. With an ontology-based knowledge system, elicitation and validation can be carried out in new and more efficient ways. Future research will have to reveal to what extent this method of acquisition is realistic for domain experts.

The entities of Frame may be physical entities or conceptual entities, and the filler can be attribute-value pair, pointer to another frame, or simple data, e.g. a student called Kevin who is 17 years old and majors in computer. If we use a frame to describe that person, it should be as below. In fact, Gender is restricted in this diagram, i.e. <Gender (restrict (OR male female))>.

```
(Student
  <Name "Kevin">
  <Gender "male">
  <Age "20">
  <Major "Computer"
  ... )
```

Figure 2.2: An example for simple filler in frame

The main distinguishing characteristic of frame base presentation is that it can express component behaviors and interconnections [54]. Some software tools are developed by knowledge base [23], for instance, the development of KnEd has focused on the creation of an interface for knowledge base, and a Frame-based knowledge representation language which uses slot-and-filler structures [22], such as KM [23], CODE4-KR [17] and CYC-L [56]. Its basic representational entity is a frame. While knowledge base grows larger, these tools would be essential to help users managing complex knowledge. A Frame-based knowledge representation scheme is proposed to describe documents from different domains [34], such as electromagnetic environment [54].

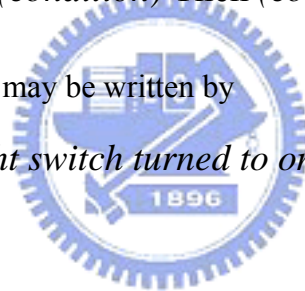
Frame-based knowledge clusters together information about *stereotyped situations and objects* such as “Person”, “Project” or “Organization”, etc. Some complex relationships between syntactic programs focus on high relevant knowledge representation and similar to object-based modeling like *inheritance*. That is describing domain knowledge in *hierarchies of classes* like taxonomy. According to this standpoint, the program implemented by frames

could be shared and reused. The domain experts can do knowledge elicitation and validation of knowledge facts. Frames may also contain procedures attached to the slots, called procedural attachments. There are generally three types: If-needed type is a procedure to be executed when a filler value is needed but none is initially present or the default value is not suitable. Defaults correspond to the expectations of a situation that we build up based on experience. When we encounter a new situation, the closest frame can be modified for the situation. We use default value when no more situation specific knowledge is available. If-removal type is run whenever a value is to be removed from a slot [27]. Frames have also been incorporated into production rule based systems with frames defining the objects that occur in the rules [50].

If (*condition*) Then (*conclusion*)

For example, a production rule may be written by

If *light switch turned to on* Then *light comes on*



To sum up, in this thesis, we have examined some related works for studying all advantages and disadvantages of traditional computer based tutoring systems, and computer assisting tutoring systems for programming are used to find the possible solutions for the puzzles of student's programming learning. We generalize a conclusion of learning motivations, for instance, required information, trial and error learning, or learning by doing, and lack of interaction, etc. We take the above standpoints into account to propose a set of systematic programming scenarios.

We summarize the aspects of all this mentioned in the following two main reasons:

- Traditional computer based assisting systems usually lack of interaction.
- Previous computer based tutoring systems using Model-Tracing approach are difficult to manage huge production rules.

In briefly, we have come up solutions of quizzes to instruct programming learning using Model-Tracing approach in order to increase interaction, with which knowledge could be clustered into Frames. Our goal is to produce the programming tutoring system against those chief shortcomings in programming. We will specify in detail that our system architecture consists of two phases and the problem-based tutoring system. Overall detailed illustrations will be presented in the next chapter.

Chapter 3. System Architecture

By previous literatures, we have known that giving an absolutely faultless learning environment will be a difficult task for teachers and engineers. Our research will announce a VB learning environment which is different from previous researches. Based on above mentioned, there are two reasons: (1) Traditional computer based assisting systems usually lack of interaction, (2) Previous computer based tutoring systems using Model-Tracing approach are difficult to manage huge production rules. Based on above reasons, we propose a tutoring system to combine Model-Tracing tutoring with Frame-based knowledge reasoning. The tutoring system framework is shown in Fig.3.1.

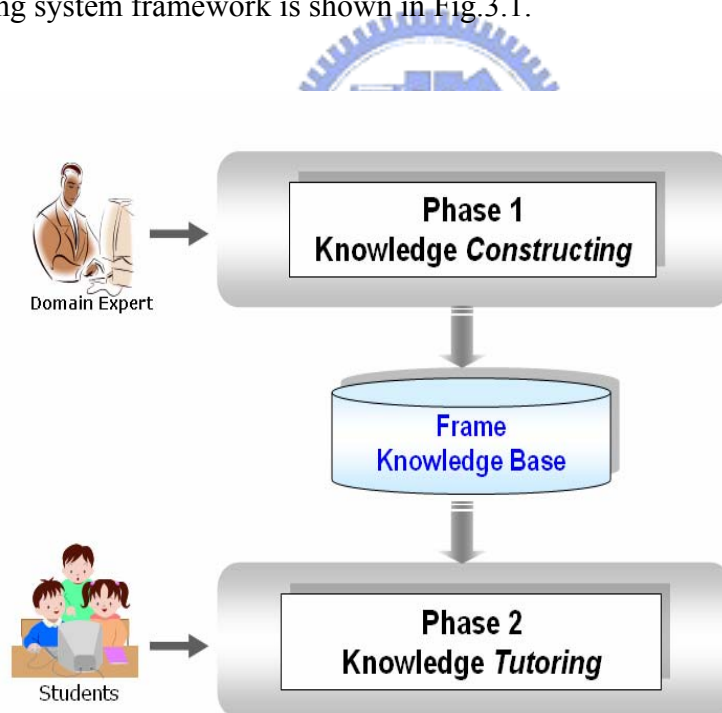


Figure 3.1: *PROBLEM* tutoring system framework

This idea is inspired by generalizing a conclusion from a collection of summarizing above mentions in related puzzles of the students learning how to program. In this thesis, we

propose a **PR**ogramming **O**ntology-**B**ased **L**earning **E**nvironment using **M**odel-tracing approach (**PROBLEM**) tutoring system. This chapter introduces a description of our system architecture. The skeleton of research will address to match both teaching and learning requirements. This led us to identify our system architecture. Our system consists of two phases: **Phase 1: Frame-based Visual Basic constructing**: it aims to construct programming knowledge by Frames. There are three processes in first phase: Related knowledge class modeling process, Frame-based knowledge representation process and Tracing possible problem-solving path process. In first phase, an adaptive programming tutoring system is used for Frame-based knowledge representation and Model-Tracing tutoring approach when students learn how to program. Frame is the structure for organizing knowledge. These instructional contents about programming are specified by Problem-Based Learning (PBL). Thus, we propose the Frame-based knowledge for VB programming and associate with a given Visual Basic Programming Ontology to support the tutoring system. **Phase 2: Visual Basic Model-Tracing tutoring system**: it aims to provide a Problem-Based Learning in VB programming by knowledge reasoning. There are four processes in second phase: Quiz display process, Models decomposition into frames process, Frame-based knowledge reasoning process, and Adaptive feedback process. In second phase, students interact with tutoring system via pre-designed quizzes of programming. The purpose of tutoring system is to integrate Model-Tracing approach with Frame-based knowledge reasoning for guessing student's possible programming errors. Finally, the adaptive learning contents could be provided to students. The detailed illustrations will be presented in the next chapter.

The Frame knowledge base is an intermediary in our system architecture. The professor Edward Feigenbaum said that the basic concept of expert system is an intelligent computer program that uses knowledge and inference procedures to solve difficult enough to require significant human expertise for their solutions, i.e. a program that emulates the decision making ability of a human expert. Moreover, knowledge base has a number of advantages,

such as the expertise is permanent, the knowledge of multiple experts can be made available to work simultaneously and continuously on a problem at any time, increased availability, reduced cost, increased reliability, explanation and intelligent tutor, etc [27]. Due to the expertise of knowledge base also associates with experienced teachers whom have known where the students make possible mistakes. It corresponds to our work needs.

Since the knowledge base provides the foundation in facts when knowledge inference, we are trying to piece up programming knowledge with data gathered from various resources provider such as teachers or programmers. In order to develop this programming knowledge base, we have considered that a Frame-based knowledge representation is used for bring together knowledge of experts from the features of program including concepts, syntactical and semantic issues. We aim to aid students in Visual Basic programming learning.

We will mention that knowledge construction and reasoning with necessary knowledge models. Thus, preface to our **PROBLEM** system in Chapter 6, we have to consider aspects of several knowledge first as follows, *(1) Problem-Based Learning (PBL), (2) Domain Ontology, (3) Frame-based knowledge representation, and (4) Programming Model-Tracing technique.* Our research must describe what of perspectives for the baseline of devising the tutoring system.

Based on PBL, the following chapter about programming knowledge construction and reasoning, we will use quizzes for example to explain them as follows:

Quiz1: Input five students' scores and arrange them from high to low using Bubble Sort method.

Quiz2: Input a 9X9 multiplication table.

Chapter 4. Knowledge Constructing

The literatures of above theories have already been discussed in Chapter 2, so we will introduce the knowledge models of our tutoring system in detail including the definition of those related theories, the educational strategy and information techniques integrated into the instructional needs, etc. The following sections explain our four processes of the first phase; we will describe the Frame ontology, Frame constructing and programming Model-Tracing constructing.



4.1 Frame-based Knowledge Representation

The domain ontology includes some axioms and experiences about VB programming knowledge. In this thesis, we do not mention how to build the ontology but just to discuss the definition of ontology and employ it in VB domain. The ontology is a tree-like structure consisting of classes and concepts. The VB Ontology in our research refers to Gruber's definition in whom defined ontology as "a specification of a conceptualization" [28], and is defined by:

Definition 4.1: VB Ontology.

$O := (KC, R, C, H, Root)$, A VB ontology is a 6-tuple symbols, which consist of:

- KC:** (knowledge class) denotes the nodes of tree structures.
- R:** (Relations) denotes the interactions between classes, or class and concepts between classes, two kinds of relation are “*a-problem-of*” and “*is-a*”.
- “a-problem-of” :** (subset-of) denotes the type of general class which is called sub-class. I.e. the relationship between super-class and sub-class.
- “is-a” :** (instance-of) denotes the type of the specific class which is called concepts.
- C:** (concepts) denotes knowledge concepts into tree structures.
- H:** (A class hierarchy) which is a directed relation $H \subset KC \times KC$ also called taxonomy.
- Root:** (a kind of KC) denotes top-level class into tree structures.

Example 4.1: Ontology template for VB.

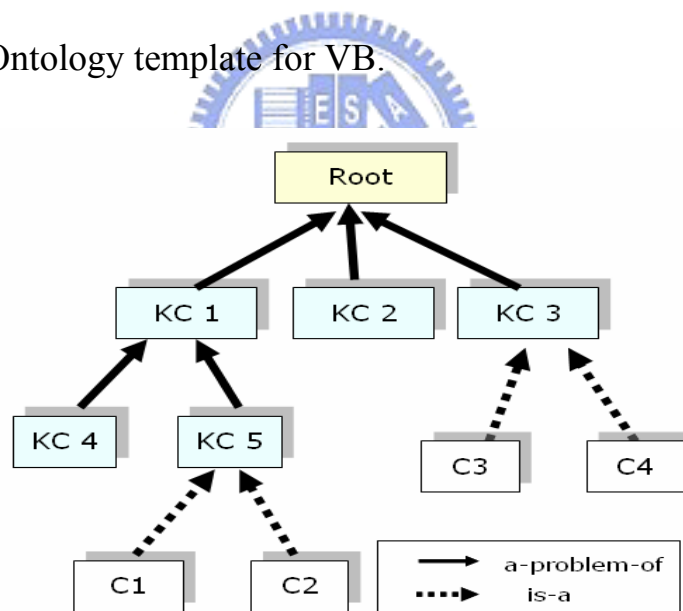


Figure 4.1: Ontology template of Visual Basic knowledge

In this example, we employed the ontology to manage those quizzes of VB domain. The classes (super-classes and sub-classes) are organized in a structured hierarchy, using generalization/specialization links (Relations) to produce taxonomy. This ontology template is shown in Figure 4.1. The purpose of tutoring system is to gather teachers’ intelligent knowledge, in order to instruct students in learning VB programming language.

Example 4.2: VB Domain Ontology.

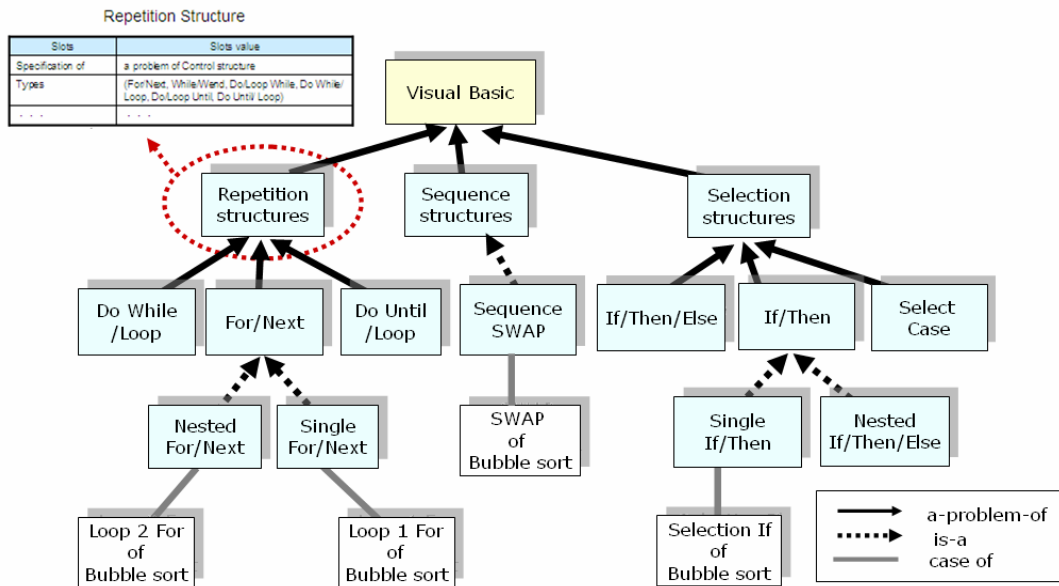


Figure 4.2: Ontology of partial Visual Basic programs

In this example, as shown in Figure 4.2, each node of ontology stands for a frame in our work. We focus on the hierarchical relationships between VB-related problems where two types of relationships are used in VB problem ontology. The first one is the “a problem of” relation (i.e., subset-of, a type of common problem) to express relationship between classes. the “*Repetition structure*” class means one common type of problem of the “*For/Next*” class, or called parent class of “*For/Next*” class. The “*Nested For/Next*” class means subset of the “*For/Next*”. The second one is “*is a*” relation (i.e., *instance-of*), which could be used to describing the concept taxonomy in the ontology hierarchy.

Figure 2.1 is the ontology of partial VB program which is given by domain experts before experiment; moreover, the example is also constructed by the above definition of the ontology, and using them when the Model-Tracing is in progress. One type of schema that has been used in many AI applications is the frame [27][40]. We have known from previous researches that the Frame-based knowledge is well suitable for implementing the programming language which contains some complex relationships between syntactic

programs, such as language PROLOG [54]. Those related knowledge classes are stored in a knowledge base by frame format. This Frame-based knowledge representation could be explained structural knowledge, especially clustering information about *stereotyped situations and objects* such as “Person”, “Project” or “Organization”. As we know, the structure of program, such as For, If, all have the stereotyped attributes and fixed pattern of behavior, just suitable for frame representation. Different programs have focused on high relevant knowledge as well. That is similar to object-based modeling such as *inheritance*, i.e. to describe domain knowledge in *hierarchies of classes* like taxonomy.

The main distinguishing characteristic of frame base representation is that it can express component behaviors and interconnections [54], as the **specification-of** slot can fill in a slot value of hierarchical relation of frames. There are two types: **Is-a** relation is used to describe the specific knowledge taxonomy in the frame hierarchy and **A-problem-of** relation is used to describe the generic knowledge taxonomy in hierarchical frame systems and inheritance. A frame is basically a group of slots and fillers that define a stereotypical object and generally design to represent either generic or specific knowledge [27].

Thus, a frame can represent a data structure of program, such as For/Next repetition structure in VB program. The definition of the frame for VB domain refers to the explanations of the frame which go through the literature of Giarratano & Riley [27], as shown in Definition 4.2. Frame slots may hold rules, graphics, comments, debugging information, and questions for users, hypotheses concerning a situation, or other frames.

Definition 4.2: Frame-based knowledge.

$F = (Slots, Slots\ value, name, specification-of, procedural\ attachments)$, A frame of VB program is a 5-tuple symbol which consists of:

Slots: denotes the fields of the frame corresponding to the knowledge class.

Slots value: denotes the fillers of matched slots of the frame, and it may be values or procedural attachments, etc., sometimes with restrictions and constraints, as

name: denotes a field of the frame name. (Required).

specification-of: denotes the hierarchical relations in a slot value of the frame. (Required). There are two types which belong to the slot, as follows:

Is-a: denotes a relation describing the specific knowledge taxonomy in the frame hierarchy.

A-problem-of: denotes a relation describing the generic knowledge taxonomy in hierarchical frame systems and inheritance.

procedural attachments: denotes the slot value which may also contain procedures attached to the slots. There are three types as follows:

If-needed: denotes the slot value is needed but none is initially present or the default value is not suitable.

If-added: denotes the slot value is to be added to a slot.

If-removal: denotes the slot value is to be removed from a slot if a value is obsolete.

Example 4.3: Frames ontology for partial repetition structure of VB.

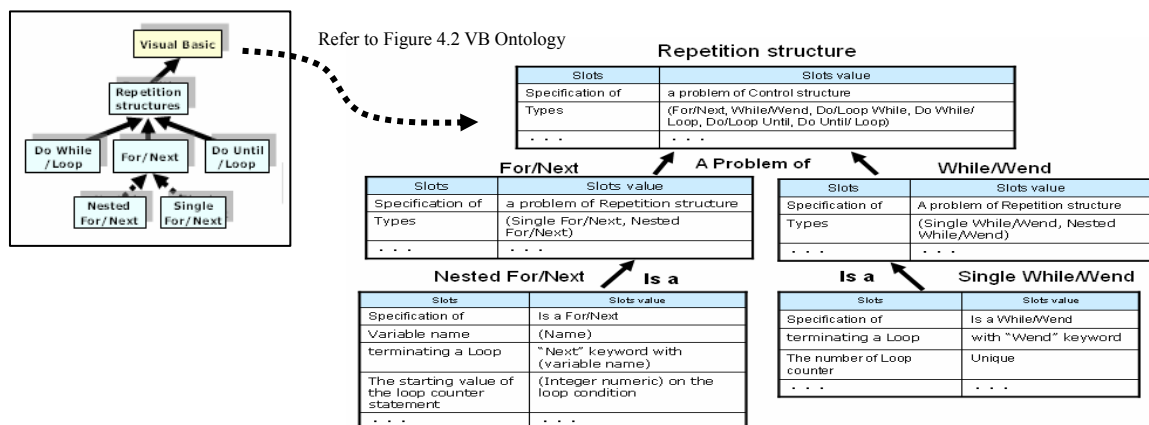


Figure 4.3: Frames Ontology for partial repetition structure of VB

In this example, those frames allow describing domain knowledge in hierarchies of classes. As for the Repetition structure, partial frames as shown in Example 4.3, there are six types of the repetition structures including “For/Next”, “While/Wend”, “Do/Loop While”, “Do While/Loop”, “Do Loop/Until”, and “Do Until/Loop”. However, the modeling of Frame-based knowledge hierarchical can be constructed looking like the tree shape and the object-based hierarchy, i.e. the child frame inherits its parent frame; for instance, in VB program, a *For/Next* repetition structure generates two children frames including *Single For/Next* and *Nested For/Next*, and a *While/Wend* repetition structure also has two children frames comprising *Single While/Wend* and *Nested While/Wend* both, as shown in Figure 4.3. In the frame hierarchy, for instance, we called the “For/Next” structure is a problem of “Repetition structure”, and the “Nested For/Next” structure is a “For/Next” structure.

Frames are generally designed to represent either generic or specific knowledge. Figure 4.3 and Figure 4.4 illustrate a generic frame for the concepts of Repetition structure and Selection structure in Visual Basic programs. The Frame-based knowledge framework refers to the VB Ontology skeleton. The frames and rules knowledge representation are used to reason those facts of programming as shown in above.

Example 4.4: Frames ontology for partial selection structure of VB.

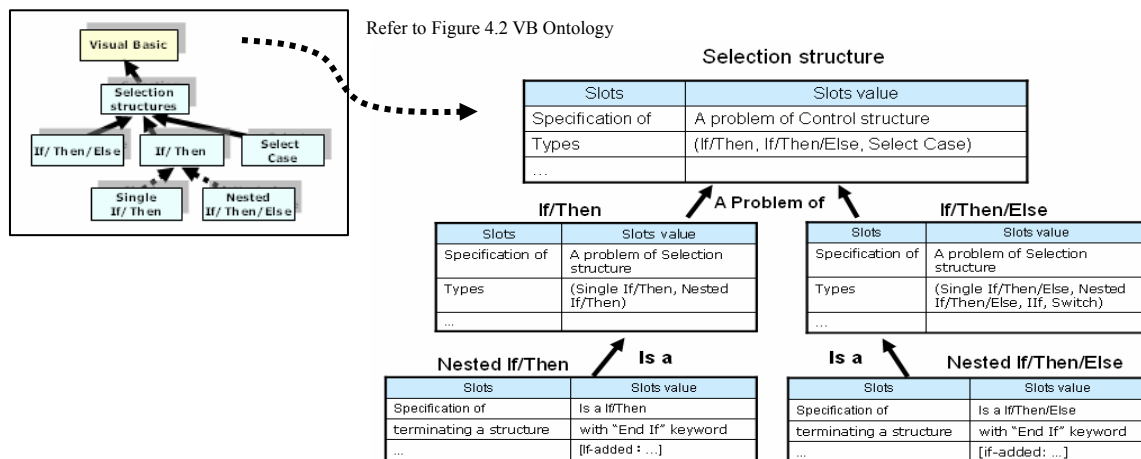


Figure 4.4: Frames Ontology for partial selection structure of VB

In this example, in VB programming language, there are three types of the selection structures including “If/Then”, “If/Then/Else”, and “Select/Case”. However, there are two types of If/Then structure including “Single If/Then” and “Nested If/Then”. There are four types of If/Then/Else structure including “Single If/Then/Else”, “Nested If/Then/Else”, “IIf”, and “Switch”. In the frame hierarchy, for instance, we called the “If/Then” structure is a problem of “Selection structure”, and the “Nested If/Then” structure is the “If/Then” structure. The domain expert must fill in the slot value via the slots of frame. The related works already mentioned that frames consist of slots which can be symbol, numerical, linguistic variable, date and time. Frame can be joined with rules and procedures, processing determined events; for instance, Figure 4.5 is shown in Single If/Then selection structure which inherited the If/Then parent frame. In Figure 4.5, the attributes of slots extract from the If/Then selection structure of VB programming language with procedure attachment.



Slots	Slots value
name	Single If/Then
Specification of	Is a If/Then
Variable name	(Name) [If-added: {if the number of control variable more than one then call procedure Add_Variable_Name() }]
The data type of variable	(Data type)
The action of execution	structure either (performs an action if a condition is True) or (skips the action if the condition is False)
The number of control variable	(Integer numeric)
The number of statement inside body structure	(Integer numeric)
The statement of control structure written	(Single line, multi-line)
terminating the structure	[If-added : {if statement written on a single line then (without "End If" keyword) else (with "End If" keyword) }]

Slots	Slots value
Variable name	(Name)
The data type of variable	(Data type)

Figure 4.5: Frame-based knowledge for *Single If/Then* structure with Procedure Add_Variable_Name()

All facts are created by the slots with related slot value about knowledge class in Frames. The facts/rules generation process is used for generating the facts/rules of knowledge classes which extracted from domain experts. The rules determine the actions to be performed when a set of conditions are met. The facts constitute the knowledge base. The typical If-then rules, often called *production rules*, have been by far the most widely used method of representing domain knowledge in expert systems. The knowledge of an expert system may be represented in a number of ways. It can be encapsulated in rules and objects. One common method of representing knowledge is in the form of **IF condition(s) THEN conclusion(s)** type—rules [27]. The *condition(s)* denotes the facts which come from the slots of the frame. The *conclusion(s)* denotes the facts to be executed when the condition is true. In a rule-based system, the inference engine determines which rule antecedents are satisfied by the facts.

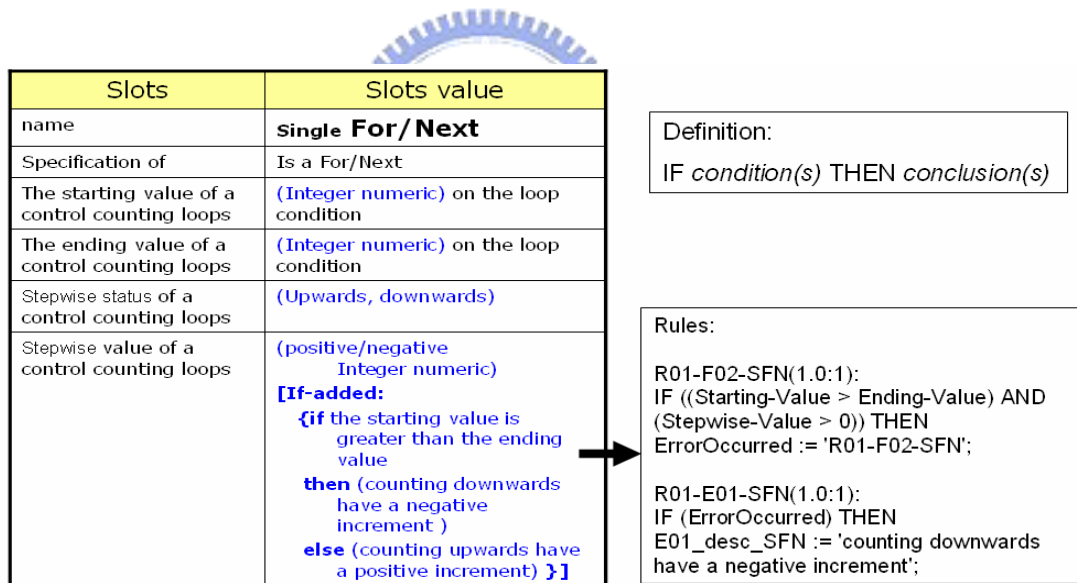


Figure 4.6: Rules generated from the Frame

However, the rules are also generated by the slot value when the slots are filled in the incorrect value, i.e. the slots value of the frame will trigger a matching rule, as shown in Figure 4.6. Sometimes the slot value is decided on other slots value, for instance, the Stepwise value of a control counting loops is referred to the Starting value and the Ending value of a control counting loops. The system has to define the error fact and trigger rule in the DRAMA

system previously. Once the filled value is incorrect, the rule will be triggered by the frame.

Two general methods of inferences are commonly used as the problem-solving strategies of expert systems: forward chaining and backward chaining. Both inference cases are sought between what is currently known and a goal. The control strategy determines the order in which the rules will be compared and also determines which rule to be fired, in case multiple rules can be fired. According to the reasoning of frames, we still need rule-based to assist in inference using the forward chaining. Our rule-based implementation is used to the DRAMA rule-based knowledge base which is a Java application. The further DRAMA system introduction and rule-based knowledge construction will be presented in Chapter 6. In this section, we only focus on how to inference using those rules.

In other words, ontology composes the necessary reference knowledge for making two or more related quizzes. Thus, we emphasize on the infrastructure role of ontology for knowledge sharing among related learning contents of VB program. In such cases, we expect to reuse parts of the knowledge built in knowledge base. Moreover, following the skeleton we will illustrate knowledge models usage in next section. In this section, the knowledge models are talking about various models which are Ontology, Model-Tracing, Frame-based and Rule-based knowledge representation. We represent the knowledge additional concepts associated with the domain ontology. Because ontology is concerned with the specification of concepts, it is usually related to a knowledge base. The most well known ontology construction guidelines were developed by Gruber [28], to encourage the development of more re-usable ontology. Some knowledge such as problem-solving method is seldom represented as ontology but is often stored in a knowledge base. The previous research has ever mentioned the distinction between “Ontology” and “ontologies”, in general, the whole computer science community for the latter term [29]. For the purpose of tutoring system needs a formalized representation of knowledge. Therefore, we have to find a proper way to give

expression to the domain knowledge of VB using manifest Ontology. This definition of both model-tracing and algorithm of problem-solving path is shown in the following sections, which we will discuss how to construct the programming using the Model-Tracing technique. But all of the functional tutoring process about the VB programs will be described in detail in Chapter 5.

4.2 Programming Model-Tracing

In this section, we explain mainly model tracing process for teachers in programming teaching, that is, problem decomposition process which is used for decomposing the problems into sub-problems. We can recognize what kind of knowledge elements are needed by tracing the models in order to well define the learning path of students. Model tracing is based on an idea of analyzing student cognitive process by reconstructing, step by step, process of making conclusions during a problem solving. This diagnostic technique is called Model-Tracing. Previous researches have been built for algebra word problem solving and algebra equation solving [14]. Our purpose focuses on how to recommend some adaptation learning concepts or objects to students. To make this kind of tracing work, both experts' correct knowledge and students' incorrect knowledge should be available.

■ Phase 1: Frame-based Visual Basic Constructing

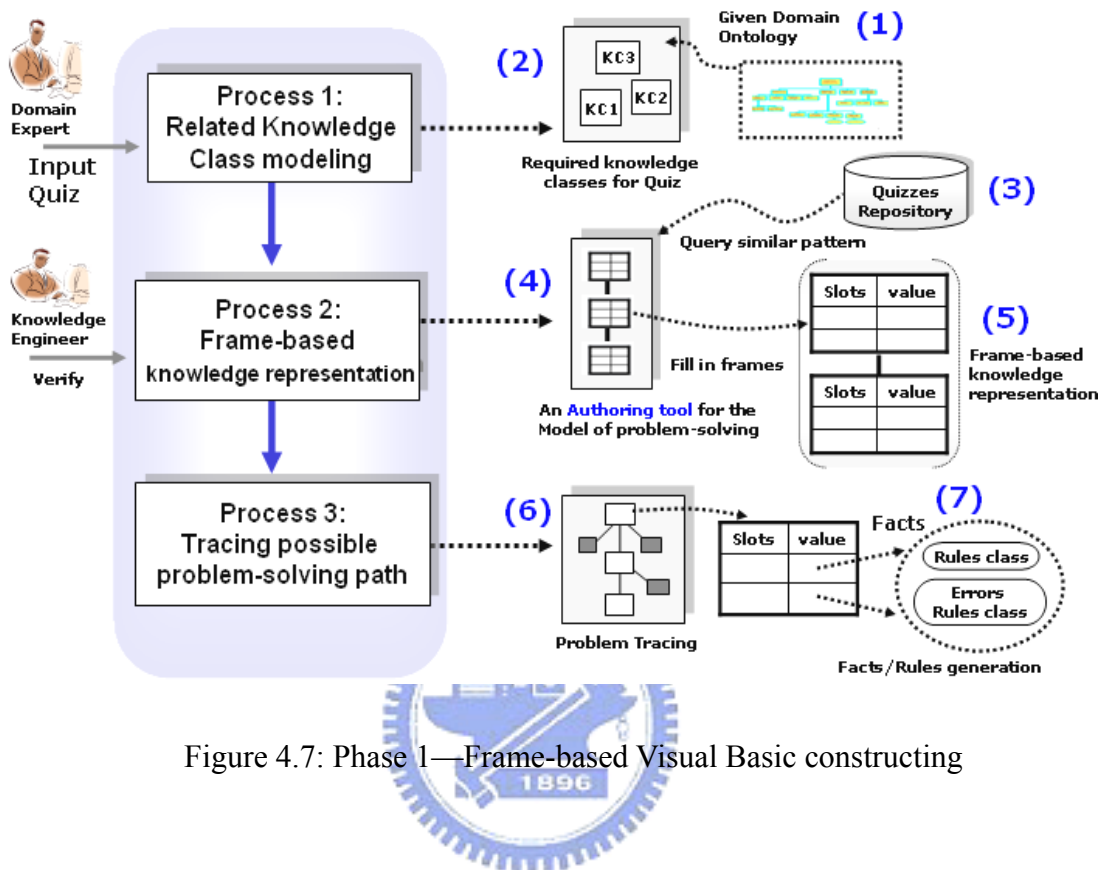
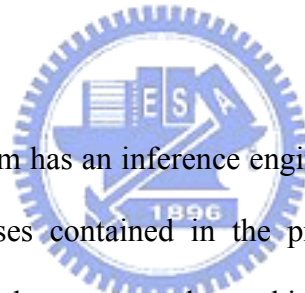


Figure 4.7: Phase 1—Frame-based Visual Basic constructing

In general, ontology representation is appropriate for knowledge modeling. There are researches to expand the domain knowledge acquisition bottleneck for Intelligent Tutoring Systems (ITS) [44]; moreover, previous researches have attempted to develop an authoring system that acquires knowledge for procedural as well as nonprocedural domains [46]. Those researches proved that have existed some difficulty in getting the domain knowledge. We propose an authoring tool for the model of problem-solving with the help of a given VB ontology, and which is the basis of domain experts constructing Model-Tracing tutoring system. This phase includes three main processes which consist of process 1: related Knowledge Class modeling, process 2: Frame-based knowledge representation and process 3: tracing possible problem-solving path.

The first process is used for allowing teachers input and describe the quiz of programming, find related knowledge classes in order to analyze the components of quiz by decomposing the whole problems into sub-problems, including two sub-processes: (1) The VB ontology is given by domain experts. (2) Any of teachers can input the VB quiz. And that decomposes related knowledge classes from a quiz. The second process is used to form Frame-based knowledge structure and filling in attributes, including three sub-processes: (3) Check whether there are similar quizzes or not. (4) Assemble divided KCs into the model of possible problem-solving, and that add/modify/delete the frames by knowledge engineer (KE), if necessary. (5) Fill in the Frames with correct or incorrect concepts And the last process is used to verify facts of frames to trace possible problem solving path and generate rules, including two sub-processes: (6) Tracing possible problem-solving path. (7) Generate rules or erroneous rules for inference.



The common expert system has an inference engine in order to make decisions based on known facts and the rule bases contained in the problem modeling component and the Frame-based knowledge. We demonstrate the architecture of Model-Tracing using either Frame-based or Rule-based knowledge representation. This tracing technique has enabled us to pick up every student's actions and that has brought us to more precise evaluation of students' knowledge, we demonstrate the architecture of phase 1 is shown in Figure 4.7. All detailed description of what we define the functionality of each process will be depicted in the next paragraph.

Process 1: Related Knowledge Class modeling

In this process, we aim to let teachers input or upgrade their quiz of Visual Basic. Therefore, the purpose of this process is to gather required Knowledge Class (KC) when quiz

is created, and must refer to a given domain ontology. And required KCs have to be arranged orderly by requests of the quiz, as shown in Figure 4.8. So the production of the modeling will be useful to fill in the attributes of each KC at the next process. Besides, it must be confirmed whether the quiz is similar to one or more of predefined others in quizzes repository when teachers want to create a lot of quizzes. Because the program of quiz may be executed many times at different troubleshooting during student's learning. If the quiz has found similarly in quizzes repository, the teacher may upgrade the learning contents of quiz in order to more adaptable to programming learning, or create a new quiz directly. We call above sub-process under the name of Patterns Finder.

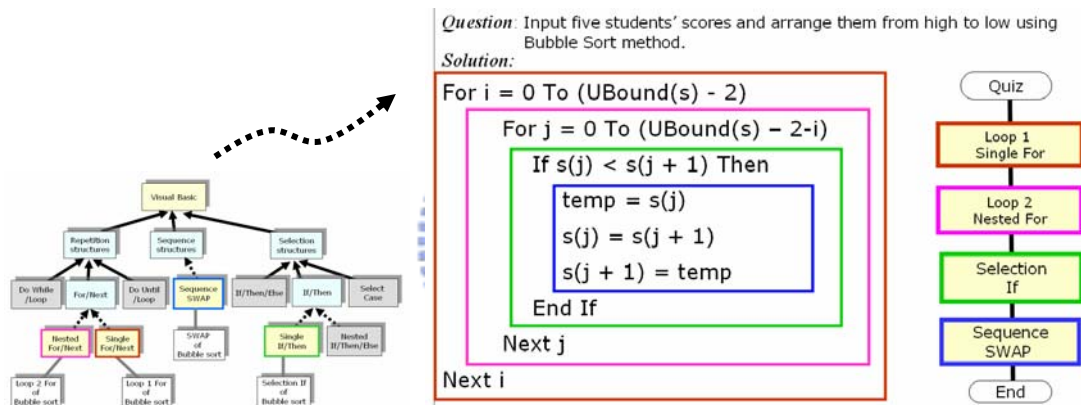


Figure 4.8: Related Knowledge Class Modeling

The content of quiz is another consideration. As for learning contents display, it takes into consideration most programmers' experiences; that is, they develop skills included learning the syntax and semantics of a programming language. All of these skills are essential to carry out the tasks of program development. These are cognitive tasks related to language and require knowledge of the syntax and semantics of the programming language [2][32]. In a word, we must set up an environment such as an authoring tool if possible, which can make the quiz easily for teachers. In following process, we just mention this Frames modeling method, but the tool will remain at the future work.

Process 2: Frame-based knowledge representation

In this process, we aim to construct well all relevant Frames before inference. According to the previous definition of the frame and the meaning of the current quiz may structure the hierarchical models for possible solution. And then domain expert fills in the value of slots, such as the stepwise value of the For/Next repetition structure. Following that situation of quiz, we can insert into the procedural attachment of the relevant slots, for example using if-needed or if-added inducting a procedure call. Hence, the slots value stands for the facts and the rules generation. That rule-based knowledge must be constructed by teachers and stored into knowledge base. Those knowledge acquisition processes are shown as below.

■ Frame-based Knowledge Acquisition process:

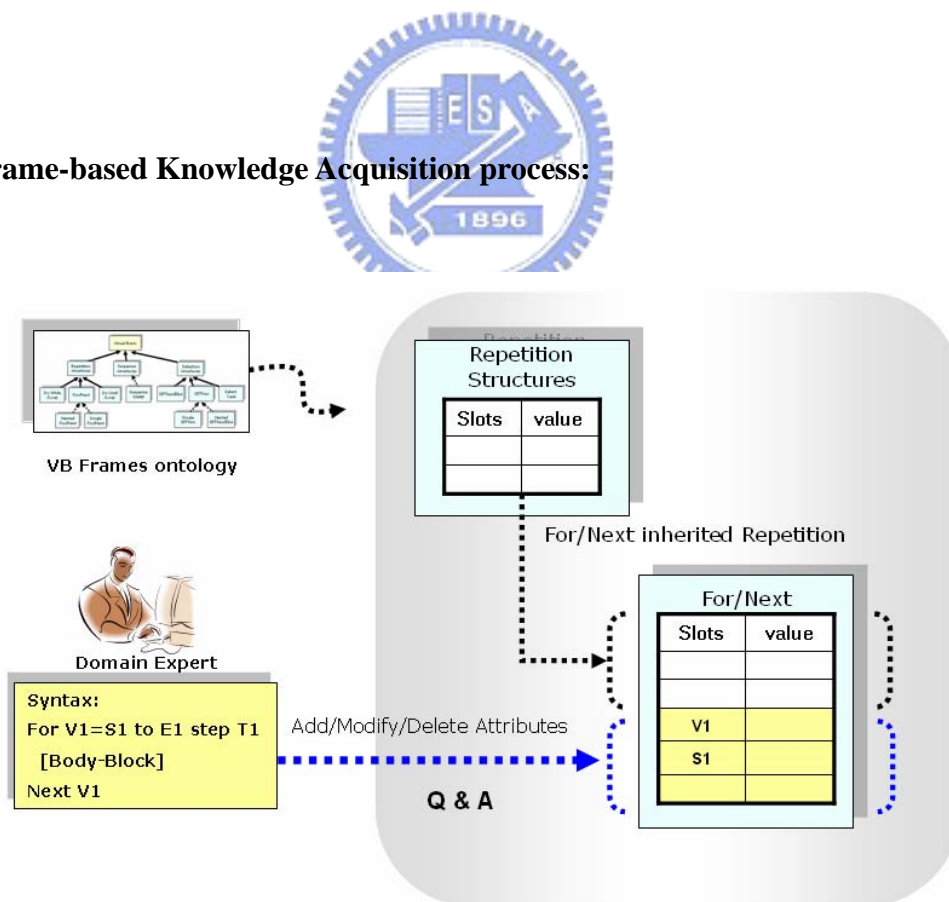


Figure 4.9: Frame-based knowledge acquisition process

Knowledge acquisition is about extracting knowledge from sources of expertise. The above attributes of For/Next come from interviewing the domain expert and construct following the syntactical of For/Next repetition structure in VB language. The main goal of knowledge acquisition is to create a knowledge base; the quality of the knowledge base will depend upon the skills of the knowledge engineer as this person plays a major part in the process of knowledge acquisition. As we know, knowledge is hard-earned, and therefore we have to use it well.

Table 4.1 presents an example for Frame-based knowledge representation in VB programming language. Teachers should fill in those slots value with predefined keywords in accordance with the given slots of the Frame. So that these value of slots will be used for the facts of inference engine. This description is as tabled below.

Table 4.1: A generic frame for Repetition structure

Slots	Slots value
Name	Repetition structure
Specification of	A problem of VB structures
Types	(For/Next, While/Wend, Do/Loop While, Do While/ Loop, Do/Loop Until, Do Until/ Loop)
The number of executing the loop body	(at least once, Dependence on condition)
Stepwise position of a control counting loops	(on the loop condition, inside the loop body)
The position of initial value of the loop counter statement	on the loop condition, before the loop body
The body of control structure	(required, optional)
The position of control condition	(at the beginning of the loop before the body of loop, at the ending of the loop after the body of loop)

Regarding early programming language, a frame may be analogous to a record structure in a high-level language such as Pascal or an atom with its property list in LISP. Due to OOP proceeding, a frame may be similar to a structure or an object, such as repetition structure of VB, or a string object of JAVA. The slots and slot fillers of a frame are corresponding to the fields and values of a structure. A frame is basically a group of slots and fillers (slots value) that defines a stereotypical object [27]. The utility of frames lies in hierarchical frame systems and inheritance. It looks like Frame-based knowledge ontology as shown in Figure 4.3. By using frames in the slot fillers and inheritance, very powerful knowledge representation systems can be built. In particular, Frame-based expert system is very useful for representing causal knowledge because their information is organized by cause and effect. By contrast, rule-based expert systems generally rely on unorganized knowledge that is not causal [27]. Table 4.2 explains the slots 1 to 8 are the For/Next attributes inheriting the slots of repetition structure. The slots 9 to 14 are new attributes for the For/Next slots as follows.

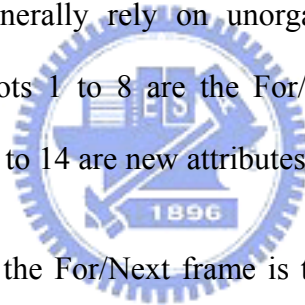


Table 4.2 specifies how the For/Next frame is to be used. The *a-problem-of* and *is-a* relations are used in Example 4.3 and Example 4.4 to show how these frames are hierarchically related. Table 4.1 is a *generic frame* whereas that of Table 4.2 is a specific frame because it is an instance of *For/Next* frame. We have adopted the convention here that the *a-kind-of (a-problem-of)* relation is generic and *is-a* relation is specific. Frame systems are designed so more generic frames at the top of the hierarchy. It is assumed that frames can be customized for specific cases by modifying the default cases and creating more specific frames. For instance, Table 4.1, we give a generic frame for repetition structure of the VB program which can generate six specific frames. Frames attempts to model real-world objects by using generic knowledge for the majority of an object's attributes and specific knowledge for special cases.

In above “Stepwise value of a control counting loops” field, we have given a condition using “If-added” type to determine which statement comes into existence. The if-added type is run for procedures to be executed when a value is to be added to a slot. Slot fillers may also contain relations, as in the specialization of slots [27]. The following table is shown in the one type of repetition structure, i.e. For/Next structure inherits the parent frame which is Repetition structure. Therefore, the slots of For/Next came from both parent frame and itself frame. Table 4.2 explains the slots 1 to 8 are the For/Next attributes inheriting the slots of repetition structure. The slots 9 to 14 are new attributes for the For/Next slots as follows.

Table 4.2: Frame-based knowledge for *For/Next* repetition structure (Table 4.1 inheritance)

Slots	Slots value
1. Name	For/Next
2. Specification of	A problem of Repetition structure
3. Types	(Single For/Next, Nested For/Next)
4. The number of execution the loop body	at least once and Dependence on condition
5. Stepwise position of a control counting loops	on the loop condition
6. Position of initial value of a control counting loops	on the loop condition
7. Body of control structure	optional
8. Position of control condition	at beginning of the loop before the body of loop
9. Starting value of a control counting loops	(Integer numeric) on the loop condition
10. Ending value of a control counting loops	(Integer numeric) on the loop condition
11. Stepwise status of a control counting loops	(Upwards, downwards)
12. Stepwise value of a control counting loops	(positive/negative Integer numeric) [If-added: {if starting value is greater than the ending value value then (counting downwards have a negative increment) else (counting upwards have a positive increment) }] }}
13. Position of initial value of the loop counter statement	On the loop condition
14. Terminating a Loop	“Next” keyword with (variable name)

Table 4.3: Frame-based knowledge for *Nested For/Next* repetition structure (Table 4.2 inheritance)

Slots	Slots value
1. Name	Nested For/Next
2. Specification of	Is a For/Next
3. The number of executing the loop body	at least once and Dependence on condition
4. Stepwise position of a control counting loops	on the loop condition
5. Position of initial value of a control counting loops	on the loop condition
6. Body of control structure	optional
7. Position of control condition	beginning of the loop before the body of loop
8. Starting value of a control counting loops	(Integer numeric) on the loop condition
9. Ending value of a control counting loops	(Integer numeric) on the loop condition
10. Stepwise status of a control counting loops	(Upwards, downwards)
11. Stepwise value of a control counting loops	(positive/negative Integer numeric) [If-added: {if starting value is greater than the ending value then (counting downwards have a negative increment) else (counting upwards have a positive increment) }]
12. Position of initial value of the loop counter statement	On the loop condition
13. Terminating a Loop	“Next” keyword with (loop counter name)
14. The number of loop counter	(integer numeric) [If-added: call procedure change_counter_number()]
15. The number of remained loop counter	[If-added: call procedure loop_counter_minus() [If-added: {if terminating exist then {if the number of loop counter more than one then (to other Nested For/Next) else (to other Single For/Next) }]
16. Body of control structure including return control	If-added: {if the number of remained loop counter more than one and The number of loop counter is not equal the number of original loop counter then (return to previous Nested For/Next) }]
17. Body of control structure including other contents	[If-added: {if True then (to other Frame) }]
18. Loop counter name	(Name)
19. The number of variable	[If-needed: (Integer numeric) { if the number of variable more than zero then call Procedure Add_Variable_Name() }]
20. The data type of loop counter	Restricted only (Integer) type
21. The relationship between control loop counters	(dependent/independent) [If-added : {if Inside value of loop counter is dependent on outside loop counter then call Procedure Relationship_Loop_Counter() }]

Table 4.3 explains the slots 1 to 14 are the Nested For/Next attributes inheriting For/Next slots. The slots 15 to 21 are new attributes for the Nested For/Next slots. We attempt listing possible attributes of the Nested For/Next to treat them as slots of the field, and the slot value of the field must be filled in by given possible value or condition. Those setting value came from domain experts. Student answers the slots value of structure according to quiz requirements during learning, and then system will map the correct slots value with student's answer to pick up possible errors. Table 4.3 is shown in one type of For/Next repetition structure. Those slots of attributes in Nested For/Next child frame consist of two parts: the one is new attributes of itself, the other is to inherit attributes of For/Next parent frame. Figure 4.10 explains each required KC for this quiz is corresponding to each pre-defined frame. All attributes of a KC must be filled in frame with correct slots value.

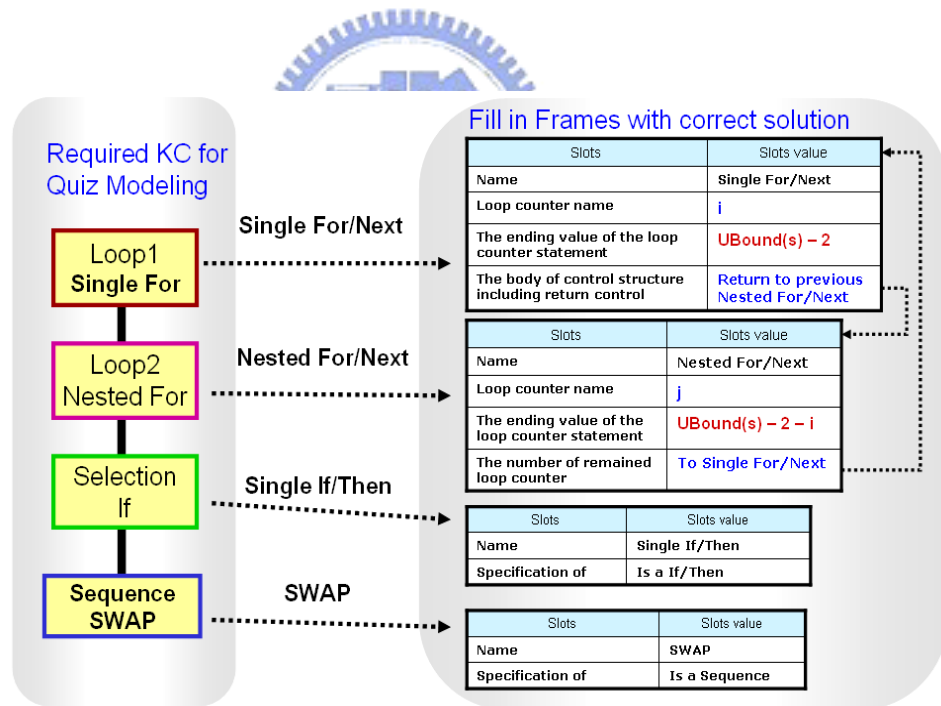


Figure 4.10: Required Frames with correct solution

In Nested For/Next frame, the domain expert has to determine the relationship between control loop counters. There are two possible relations: Dependence and Independence. This slot is given a “If-added” procedure type to be executed as follows:

[If-added :

{ **If** Inside value of loop counter is *dependent* on outside loop counter
Then call Procedure **Relationship_Loop_Counter()** }

Table 4.4: Procedure attachment for *For/Next* repetition structure

Procedure Relationship_Loop_Counter()

Slots	value	Instance of Quiz
Outside variable name	(V1) (String)	i
Outside starting value	(S1) (Integer numeric)	0
Outside ending value	(E1) (Integer numeric)	UBound(s) - 2
Outside stepwise value	(T1) (Integer numeric)	1
Inside variable name	(V2) (String)	j
Inside starting value	(S2) (Integer numeric)	0
Inside ending value	(E2) (Integer numeric)	UBound(s) - 2 - i
Inside stepwise value	(T2) (Integer numeric)	1
Relationship of S1,S2,V1,V2	(R1) Irrelevant/relevant [If-needed: {if relevant then call Procedure expression_R1() }]	irrelevant
Relationship of S1,E2,V1,V2	(R2) Irrelevant/relevant [If-needed: {if relevant then call Procedure expression_R2() }]	Irrelevant
Relationship of E1,S2,V1,V2	(R3) Irrelevant/relevant [If-needed: {if relevant then call Procedure expression_R3() }]	Irrelevant
Relationship of E1,E2,V1,V2	(R4) Irrelevant/relevant [If-needed: {if relevant then call Procedure expression_R4() }]	E2=E1-V1

When “If” condition is true, i.e. inside value of loop counter dependences on outside loop counter, we have to consider Table 4.4 should be added to fill the source code of the quiz in the frame. On the contrary, we consider another quiz with control loop counter independence as shown in the following Example.

Example 4.5: The sample quiz with independent control loop counters.

Quiz 2: Input a 9x9 multiplication table.

Solution:

```
For i = 1 To 9
  For j = 1 To 9
    M(i , j) = i * j
  Next j
Next i
```

For this quiz, the inside loop condition and the outside loop condition is irrelevant. Therefore, for the structure of the program, domain experts have to define relation between slots except that derive slots value from lots of attributes. Organization of frames matches the learning situation.

Hence, in Figure 4.11, we give an example for acquiring knowledge of both Bubble sort and Selection sort using the Frame-based knowledge of the elementary programming, (e.g. the program of two dimension (2D) arrays). Both sorting programs need the Nested For/Next structure. Reusing already modeling frames can save time and effort. Besides, the mostly contribution is knowledge sharing.

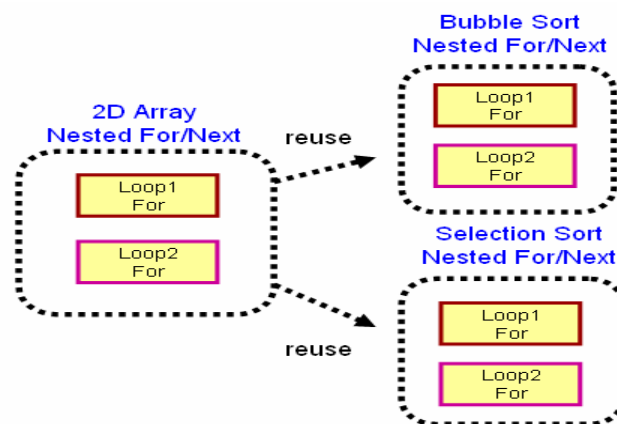


Figure 4.11: An example of reusing Frame-based knowledge

In this thesis, we employ the Frame-based knowledge representation, which needs the knowledge engineers create some frames about the domain, where experts go straight into putting their knowledge into the slots value of frame. The correct or incorrect value matches the student's answers is corresponding to the features of Model-Tracing. Moreover, we still need a translator tool which can transfer the Frame-based knowledge into a knowledge base. Here we do not mention how to construct the translator. We focus on how to use these values of slots in the facts of inference and commonly are used as the problem-solving basis; however the translator tool will remain at the future work. How to define and fill in a frame is an important consideration for our work, so we will give an example for interpreting this whole process later.

Process 3: Tracing possible problem-solving path

In this process, we aim to trace the possible path of problem-solving models in order to find the incorrect programs while students may make, and then fill in the facts and rules. These important cognitive outcomes of this stage included intellectual skills, demonstrated by the ability to apply knowledge, and the diagnostic analysis of errors. In this thesis, we do not mention how to construct out of this VB Ontology but applied to the model tracing of VB programming language. A definition of a Model-Tracing of quiz is defined as follows:

Definition 4.3: Model-Tracing of Quiz.

$M = (Quiz, C, Ec, H, End)$, a model of quiz of VB program is a 4-tuple symbol which consists of:

- Quiz:** denotes the root of solution path in structures of VB program.
- C:** denotes the concepts of matched knowledge class of VB ontology into the correct solution path.
- Ec:** denotes the concepts of matched or mismatched knowledge class of VB ontology into the incorrect solution path.
- H:** a model hierarchy which is a directed relation $H \subset C \times Ec$.
- End:** denotes ending of program structures.

Example 4.6: Model-Tracing template

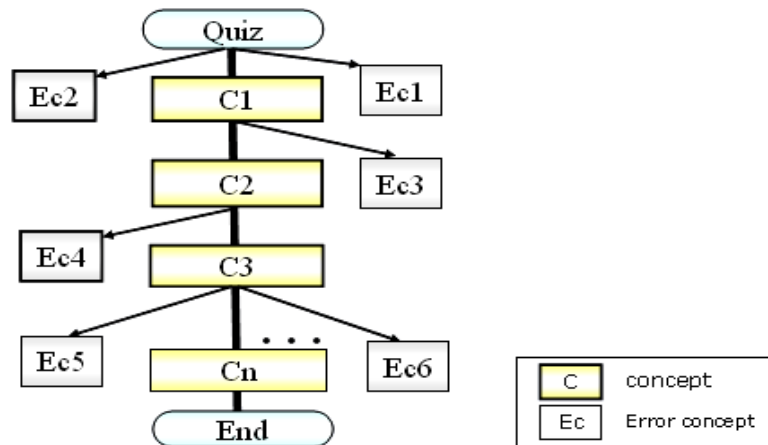
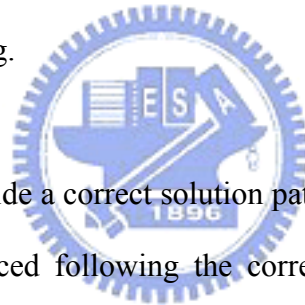


Figure 4.12: The Model-tracing of possible problem-solving path

In this example, in Figure 4.12, the node C stands for assembling concepts of the quiz and the node Ec means possible error concepts of the quiz. The student learns a quiz which will cause the correct or incorrect learning path. Hence, the teacher has to construct model-tracing of each quiz previously.

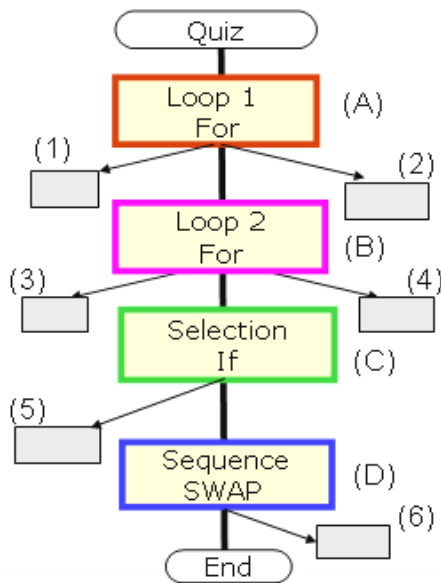
Mostly tutoring systems just focus on either Model-Tracing approach or Frame-based knowledge representation, but we combine both in this thesis. The key point is to give the student for a help while learning programming. We have known from literatures that the

ontology teaching systems are used for employing the concepts of the model tracing as well [46]. For example, previous researches were like LISP Intelligent Tutoring System (LISPITS) [14] which teaches how to program using the LISP programming language, algebraic functions [3], and Demonstr8 [9] which is an authoring tool for building Model-Tracing tutors for arithmetic, etc. Those tutoring systems aim to perform an exhaustive the problem-solving procedure which is used for obtaining the correct solution. That's exactly what we expected. We explain how to develop the models of quiz, i.e. the source code should be partitioned off into several parts. Previous Figure 4.2 graphically has depicted the VB Ontology citation, which is used for constructing the domain ontology and is the key reference for tracing the student's source code. Previous Figure 4.1 is shown in Model-Tracing templates; hence, we divided the practical modeling operation into two actions. The first action is Quiz modeling and the other is Problem tracing.



This process aims to provide a correct solution path for the specific quiz. However, those incorrect concepts will be traced following the correct concepts. In general, according to Definition 4.3, we provide such a method of quiz modeling for teachers in this stage that they can generate their quiz in a way. In the beginning of constructing the quiz, the source code will be divided into some components according to the syntax of VB programs. The system has to list the model of possible problem-solving of quiz. Figure 4.13 is an example for Bubble Sort program. In this process, we will illustrate modeling method with the sample program which is given by domain experts.

Example 4.7: The models of possible problem-solving path.



The correct solution path could be expressed by node “(A)-(B)-(C)-(D)”

The following nodes stand for certain of possible errors:

- Node (1) and (2) denote *For loop 1* error.
- Node (3) and (4) denote *Nested For loop 2* error.
- Node (5) denotes *If selection* error.
- Node (6) denotes *SWAP sequence* error.

Figure 4.13: The models of possible problem-solving path

In the example shown in Figure 4.13, the system scans the source code from top to bottom. Whereas the characters of the source code are grouped into collections that correspond to reserved words, symbolic names, and other code components from the syntactic structure of the VB language’s grammar. Accordingly, we can clearly partition the program that following the VB Ontology relationship. The modeling path is a sequence of these characters. The Bubble Sort program (i.e., Figure 4.13) has described the practical modeling process. That is the first node “*Loop 1 For*” can be defined by tracing the “For” keyword of source code. And then the KE keeps tracing the source code of quiz until the terminal “*Next*” keyword appeared. The second node “*Loop 2 For*” is also the same operational steps to extract them. But the terminal keyword of “*Selection If*” is “*End If*” identifier. And the last node “*Sequence SWAP*” consists of three assignment statements. This process aims to trace the models of possible problem-solving path for filling in related slots of frames with the source code. While teachers decide to teach programming using quizzes, through quiz modeling and ontology mapping, solutions of the quiz will be generated. Figure 4.13 is one of

among the correct solutions. We list an example of possible problem-solving path during Model-Tracing, This correct path could be expressed by node “(A)-(B)-(C)-(D)”. We will find related errors following the modeling path. The domain experts have various possible pre-designed errors. In Figure 4.13, those nodes “(A)-(B)-(C)-(D)” means knowledge class including a lot of errors, such as, the nodes (1), (2) and (3), (4) mean two errors of “*Loop 1 For*” and “*Loop 2 For*”, respectively. The nodes (5) and (6) mean an error of “*Select If*” and “*Sequence SWAP*”, respectively.

The scheme of Model-Tracing is to analyze the obtained source codes from students. But the correct program have finished by teachers previously. The Model-Tracing algorithm is used to create those quizzes by teachers and find possible errors which students may make mistakes. Finally, the model can be used to trace the students’ actions and provide appropriate feedback. We gather the entire quizzes which were modeled into the quizzes repository. Considering if the other teacher has to input and upgrade the similar quiz, which is where reusing. Once students learn this quiz, the system can compare their answers with teachers’ answers. For the purpose of tracing, it is soon detecting many faults in syntax and semantics of coding program of the student. In short, making mistakes is also a kind of learning. As the old word said, “One man’s fault is other man’s lesson”. The whole process is used to construct the elements of inference via the quiz modeling partition and the Frame-based knowledge representation with ontology helping, problem decomposition and facts/rules generation. In short, based on the Model-Tracing tutoring skeleton with ontology help, and facts diagnosis and rules inference, we could easily provide more adaptively learning contents to the students.

We will illustrate in detail using an example in Chapter 4. This Problem-based Model-Tracing constructing algorithm is as follows:

Algorithm 4.1: Model-Tracing Construction

Given three inputs: The Quiz of VB, the Ontology of VB, and the Quizzes Repository.

Output: The model of possible problem-solving path.

Step 1: Decompose related concepts of knowledge classes from a quiz.

Step 2: Domain expert verify whether the node of concepts has existed or not.

Step 3: If the node of concepts does not exist, the knowledge engineer try to add the node first.

Step 4: Teacher edits the program of quiz.

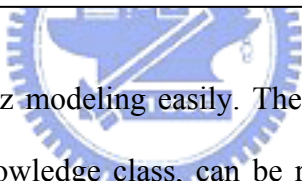
Step 5: Teacher edits the related descriptions of quiz.

Step 6: Select the included knowledge classes of program. For example, “For/Next”, “If”, etc.

Step 7: System partitions the program according to VB Ontology relationship.

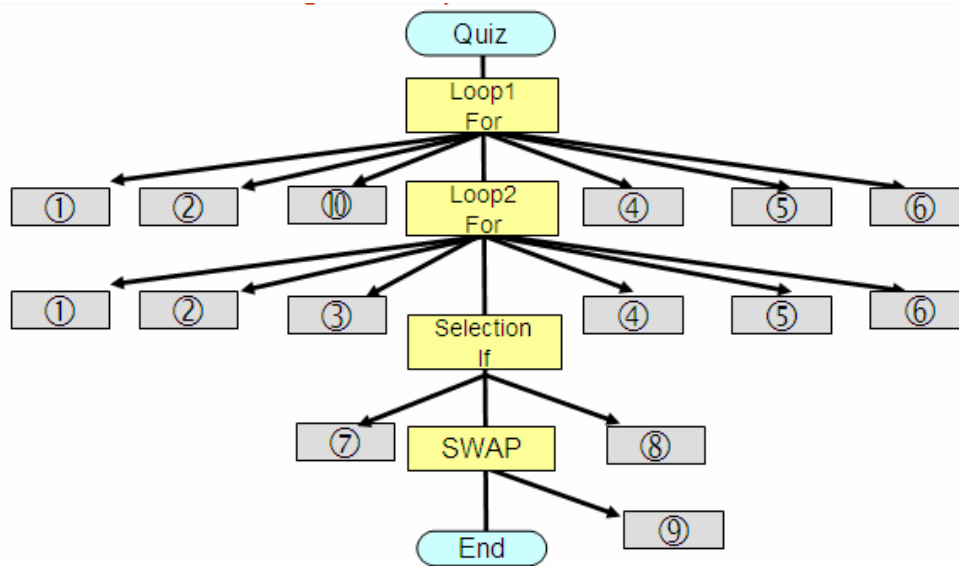
Step 8: Arrange for those divided concepts to be shown in solution path.

Step 9: Domain expert define the possible errors for this quiz through the correct solution path.



The ontology aids the quiz modeling easily. The certain components of previous quiz, such as “Nested For/Next“ knowledge class, can be reused and be integrated with existing programs to compose new programs, especially for the same possible errors. As we know, the problem solving skills are essential to understand the fundamentals of computing, and should be learned while learning programming. Problem formulation, planning and design are prerequisite for coding and testing. Some difficulties or incorrectness at these earlier stages will lead to errors in the final stages. This is firstly consideration for us.

Example 4.8: An example of possible errors for sample quiz



- ① Wrong variable name in Next statement
- ② Not terminating a For/Next Loop with "Next" keyword.
- ③ In nested For/Next loops, using the same control variable name for more than one loop
- ④ using the same control variable name which gives a value causing an infinite loop in For/Next Loop when the value is between the starting and ending value
- ⑤ counting downwards have a positive increment in a For/Next loop, when the starting value is greater than the ending value
- ⑥ counting upwards have a negative increment in a For/Next loop, when the starting value is less than the ending value.
- ⑦ When written on a single line, normally expects only one statement in its body.
- ⑧ the End If is accidentally omitted.
- ⑨ Needed at least three variables

Figure 4.14: An example of possible errors for sample quiz

In this example, as shown in Figure 4.14, the number one of errors is “Wrong variable name in Next statement” which means a kind of “Nested For/Next” problems. If programs follow a wrong path, the explanation should be presented. In order to let the system provide more adaptive learning contents for the student, the solutions of various version of each quiz are given by teachers who are the domain knowledge provider. We consider modeling method orderly for the program of quiz.

In Figure 4.14, overall errors have been collected by experienced teachers; in this case, the knowledge base does a great of favor to store that knowledge. Every error can be trace by the filled value of the frame. The certain errors in loop two may be the same as loop one. Therefore, we can reuse those created slots of frame for reasoning. In next chapter, we will use an example to explain the whole knowledge reasoning, including reasoning framework, algorithm, Model-Tracing tutoring, and PBL tutoring system for programming improvement.



Chapter 5. Knowledge Reasoning

In this chapter, we aim to illustrate the process of knowledge reasoning for our work. Section 5.1, we will introduce the reasoning framework using a sample quiz modeling, and comprehend necessary knowledge classes (KCs). Accordingly, the arguments of each KC of the quiz must be derived by the slots of frame, i.e. to explain relationship between the models, Frames and Rules. Section 5.2, we give a tutoring scenario to describe the whole tutoring system how to teach students in programming learning using Model-Tracing techniques.



5.1 Reasoning Framework

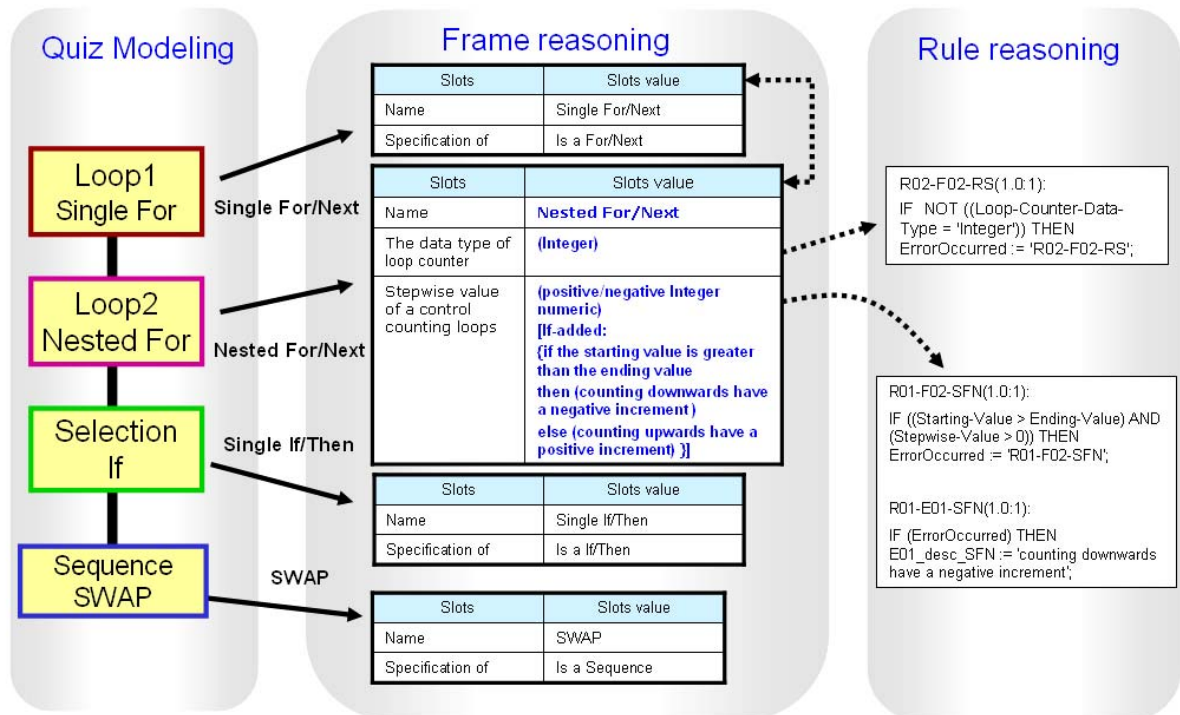


Figure 5.1: Knowledge Reasoning Framework for *Bubble Sort* quiz

Algorithm 5.1: Knowledge Reasoning.

Given three inputs: The source code of quiz, Frames, and Rule classes.

Output: Error facts, and hints, guidance, or questions.

Step 1: Find the keywords of KC from the source code of quiz, e.g. For, If.

Step 2: List required Frames following the found keywords.

Step 3: Fill in the slots value of Frames with source code. Each slots value means a fact.

Step 4: Finish overall filled value of Frames of KC about the quiz.

Step 5: Start inference corresponding those facts with Rule class.

Step 6: Load rule class. Once rule is fired, setting facts.

Step 7: Response error facts if occurred.

Step 8: Transfer error facts into information or matching teaching contents.

Step 9: Present hints, guidance, or questions to students.

Step10: Repeat from step 5 to step 9 until no rule is fired.




Figure 5.1 shows in the process of knowledge construction from quiz modeling, via Frame-based knowledge reasoning framework, and Rule-based knowledge inference. Finally, the system fired the rules to generate error facts. The tutoring system will show the adaptive hints, guidance or questions to students according to those error facts.

5.2 Model-Tracing Tutoring Scenarios

We will introduce a programming tutoring system, which provides an environment for students to learn about the knowledge of programming. It not only may be an Editor to provide students coding their programs but also can be an Answer to ensure that the student's final answers are correct. At the starting, we take advantage of computer to display the quiz of

VB for students, whom already login the web pages system with problem-based learning diagrams. In general, the purpose of coding a program is to solve the real-world problem, such that programming instruction will present several situated learning described by quizzes. Briefly, we hope the student interacts with problems. The student must focus on solving possible problem via tracing programs, trial and error, step by step, and finally finished the answer of the quiz. This cognitive process is called learning path as well. In other words, our tutoring system supports the syntax structure aiding instruction as yet. In this section, we will introduce the process of Model-Tracing tutoring is described in, and give an example to explain the tutoring scenarios later.

In second phase, an expert's knowledge is specified by the problem based manner, and presented a quiz to students. The student understands the description of quiz and types source code through on-line editor. The system has to decompose the student's source code and fill in related slots of frames, and then system trace the slots value of frame to find the possible errors of the program which came by way of mapping the student's answer with the teacher's solution. If the answer does not match the correct solution, the rule is fired. At first, the system will load the rule class, set facts, and inference by DRAMA system. To set facts, return messages to the tutoring system and show messages to the student. The student accepts suggestions and amends the incorrect programs. If the message is a question, the system will show the question. If the message is the hints or guidance, the system will go on tracing the student's programs until the student finishes the quiz.

In short, the purpose of tutoring system it to integrate Model-Tracing approach for guessing possible errors where students make mistakes with Frame-based knowledge to reason the Facts corresponding to rule-based knowledge support, and using Ontology-based to manage this domain knowledge and a number of quizzes, and finally given adaptive learning contents to students. Figure 5.2 shows the tutoring scenarios.

■ Phase 2: Visual Basic Model-Tracing Tutoring System

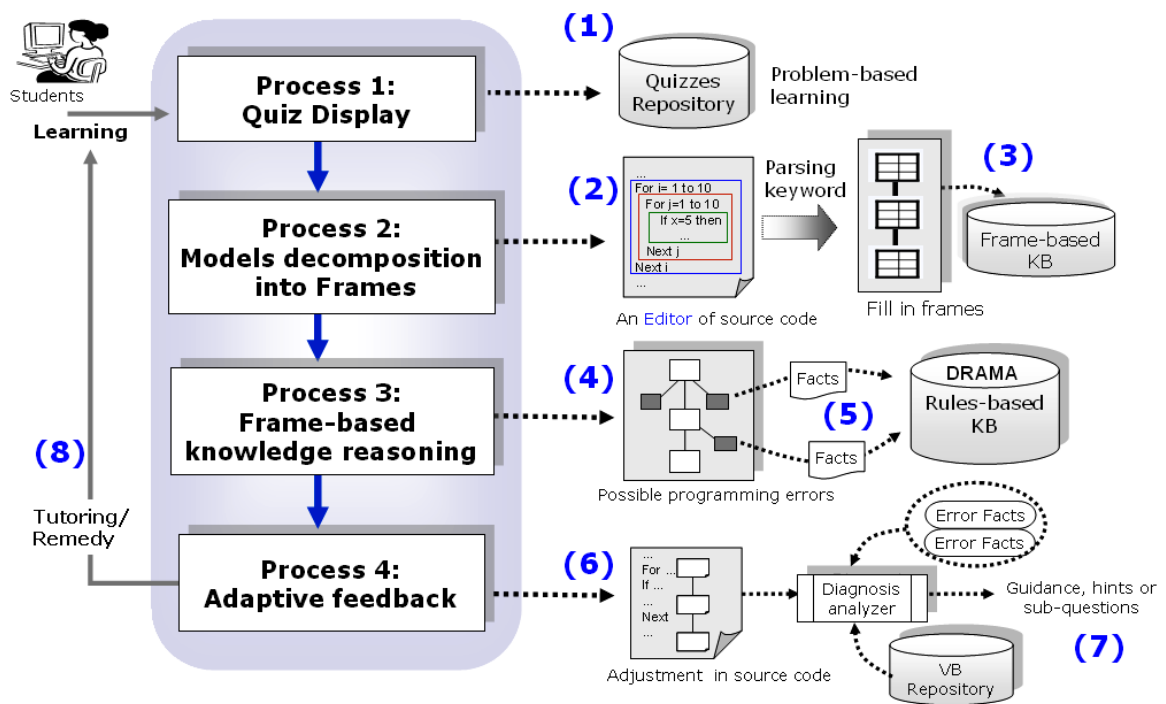


Figure 5.2: Phase 2—Visual Basic Model-Tracing Tutoring System

We describe briefly the above processes as follows:

Process 1:

- (1) Pre-designed quizzes of programming have gathered in repository to get ready for students. Students login web pages tutoring system to learn programming by quiz display.

Process 2:

- (2) Parse student's partial answer and partitioning source code into models following syntactic structure. Each model matches a frame, fills in slots value, and then is collected in Frame-based knowledge base.

Process 3:

- (3) Decompose problems and trace possible programming errors.
- (4) Map facts from each frame to trigger rules.

Process 4:

- (5) Analyze error facts to find corresponding learning contents from VB repository.
- (6) Give adaptive feedback for the adjustment of student's source code.

A main consideration of this phase is to provide the adaptive programming solutions for students. Seriously speaking, the whole family of cognitive tutors started with the LISP tutor. Later on, cognitive tutors for geometry and algebra were developed using the same Model-Tracing methodology, including support for various types of problems and skills [3]. Model-Tracing process consists of two steps. Firstly, constructing the models of program must partition teacher's source code into several Frame-based knowledge components, and corresponding to slots value for the correct solution of program. Secondly, the system should trace the student's answer to contrast the teacher's correct solution. Any mismatch of slots value means where errors come up. Accordingly, the system can give some suggestions for the student. In second phase, the Problem-Based Learning Tutoring System consists of four processes for students in VB programming learning.

Process 1: Quiz Display

In the first process, the system will display the quiz for students. The learning contents are restricted to the following research scope.

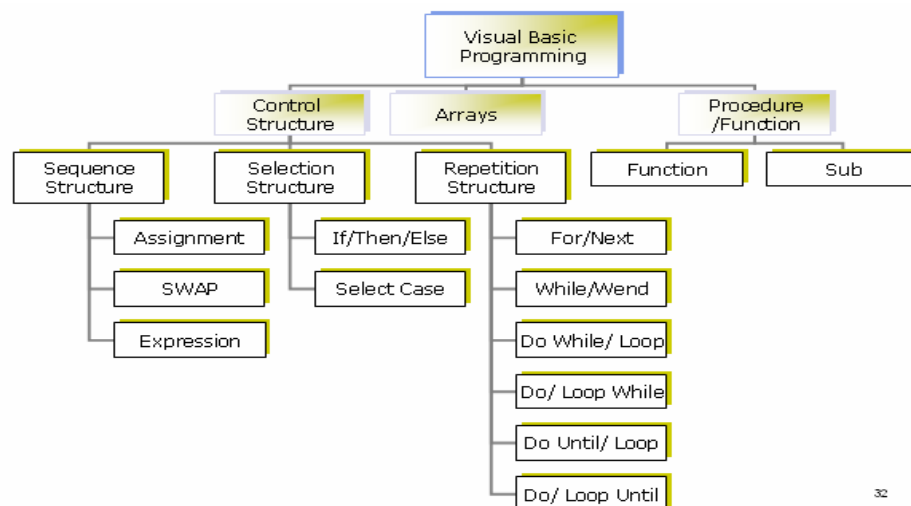


Figure 5.3: An overview of Visual Basic tutoring contents

Table 5.1: Ten control structures of Visual Basic programming language

Structure	Description/ classes
The sequence structure	Executing VB statements one after the other in the order in which they are written. Anywhere a single action may be placed, we may place several actions in sequence.
Three types of selection structures	A single-selection structure: <i>If/Then</i> A double-selection structure: <i>If/Then/Else</i> Multiple-selection structure: <i>Select Case</i>
Six types of repetition structures	<i>While/Wend</i> <i>Do While/Loop</i> <i>Do /Loop While</i> <i>Do Until/Loop</i> <i>Do /Loop Until</i> <i>For/Next</i>
Each program case is formed by combining as many of each type of control structure as is appropriate for the program implements.	

The following example initiates the knowledge elicitation process. We use a Bubble Sort quiz to explain the overall tutoring scenarios. Example 5.1 is a Bubble Sort quiz description. The possible solution is created by domain experts. Considering the following sample problem description and the satiable solution:

Example 5.1: A sample quiz with the partial correct solution

```

Quiz 1:   Input student's scores and arrange them from high to low using Bubble Sort method.
Solution:
    For i = 0 To (UBound(s)-2)
        For j = 0 To (UBound(s)-2-i)
            If s(j) < s(j+1) Then
                temp = s(j)
                s(j) = s(j+1)
                s(j+1) = temp
            End If
        Next j
    Next i

```

In this section, we have known the flow control from source code of quiz to the models partition; moreover, the frames have be filled in the value of attributes. But we can not finish our work without Frame-based knowledge reasoning and rule-based knowledge assisting.

Process 2: Models decomposition into Frames

In above chapter, we have discussed the ontology aid the quiz modeling easily. We suppose the incorrect answer as shown in Figure 5.4, which came from the student. This section aims to partition student's answer into models. In order to trace possible errors of each model, it is necessary to fill the attributes of models in related Frames. These templates of frames have been constructed previously with slots and fillers (slots value). According to needs of various quizzes, the filler should be distinct from others. That is the same attributes have different values to stand for the state of frame in the quiz. Frame-based knowledge representation can be used for presenting the stereotyped situation, objects, etc. Therefore, Frames are suitable for implementing the programming domain knowledge.

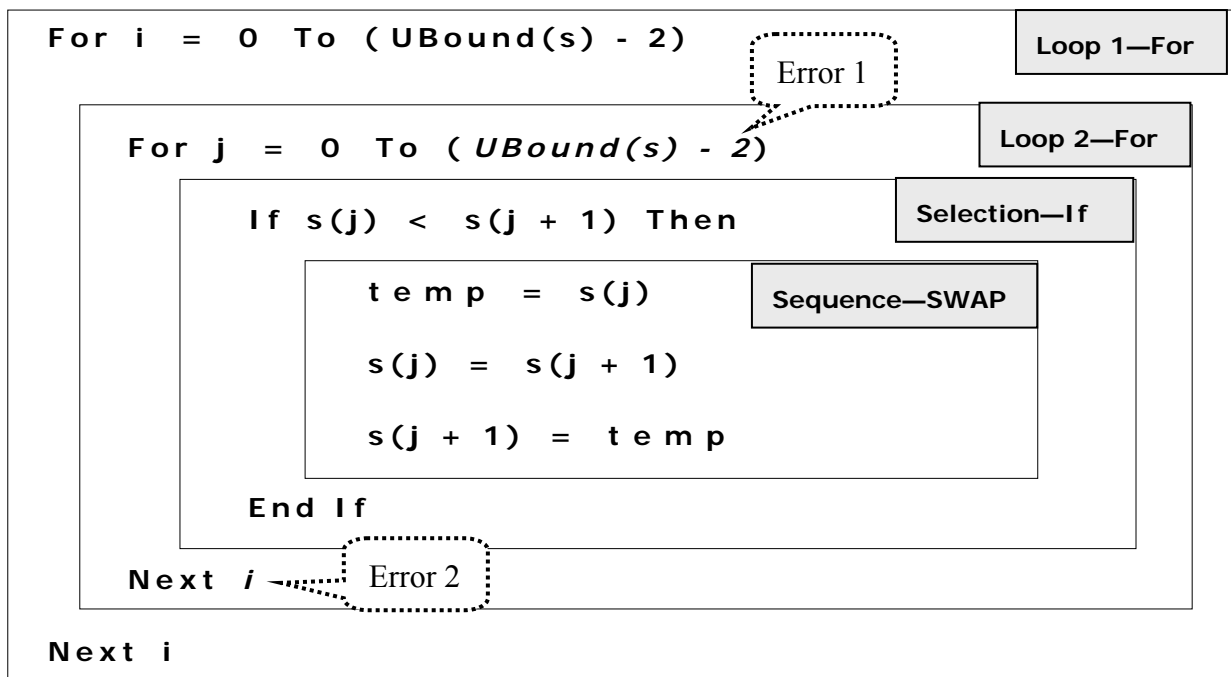


Figure 5.4: An example for the incorrect answer came from the student

Process 3: Frame-based knowledge reasoning

Therefore, we have to find the possible errors of above student’s program. And then the tutoring system will model the program and fill in each frame with attributes as shown in Figure 5.5, which is the outcome of the quiz modeling and the related frames are presented for tracing possible programming errors.

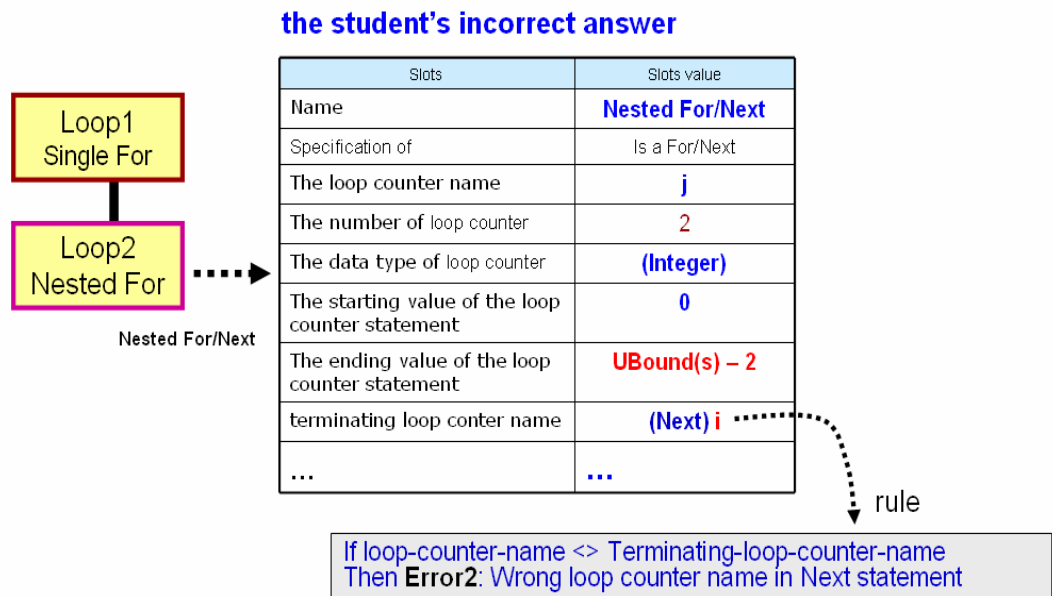


Figure 5.5: Frame-based knowledge with the rules reasoning

As for our system, the problem domain is Visual Basic programming learning. A good programmer depends on experienced skills in programming. That is why the knowledge engineer extracts knowledge from human experts and then translates their knowledge into a Knowledge Base, which contains all the collected knowledge related to the problem to be solved. We give an example of Model-Tracing for possible errors as shown in Figure 5.6. Those incorrect slots value (i.e. student’s answer) will be found during Model-Tracing process according to the teacher’s correct solution.

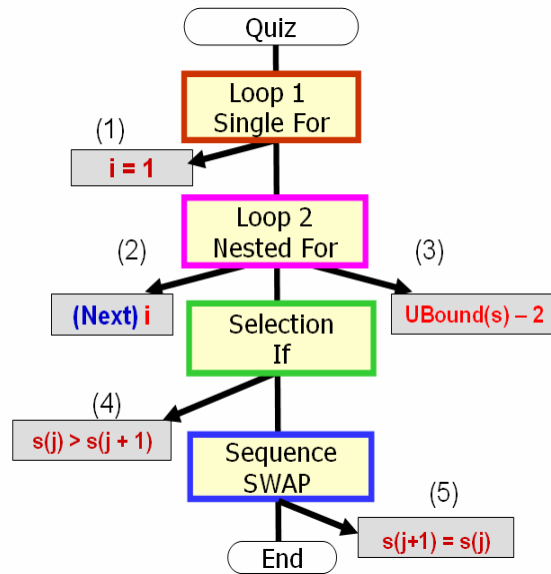


Figure 5.6: An example of Model-Tracing for possible errors

Process 4: Adaptive feedbacks

This process provides clear advices and explanations for students to amend their VB programs according to syntactic errors, logic errors or misconceptions while they learn how to program. Those feedbacks came from mapping the keywords of source codes to find corresponding questions, learning contents, etc., eventually, the system will aid learning with correct and useful programming knowledge.

In this thesis, we take advantage of the Frames to organize the solution of a quiz. The quiz's answer will be divided by the keywords of program when the student has finished and submitted the work of programming to the system. Those frames may be used for inference. In addition, we also implement the rule-based knowledge from DRAMA rule-based knowledge base system which was published by KDE [20]. The detailed illustration will be described in Chapter 6.

Chapter 6. Prototype System Implementation

In this chapter, we have implemented the Model-Tracing tutoring approach in a prototype system, called *PROBLEM* (*PR*ogramming *Ontology-Based Learning Environment* using *Model-tracing* approach) tutoring system to instruct students the Visual Basic programming. It will be useful and adaptable conducted interactive Problem-Based VB programming learning scenarios. We have presented practicable to develop a framework of the tutoring system to simulate a programming learning environment. The prototype consists of tutoring and authoring services. The system design is similar to an expert system. The system traces the student's learning processes. Figure 6.1 is a tutoring platform of the prototype system.

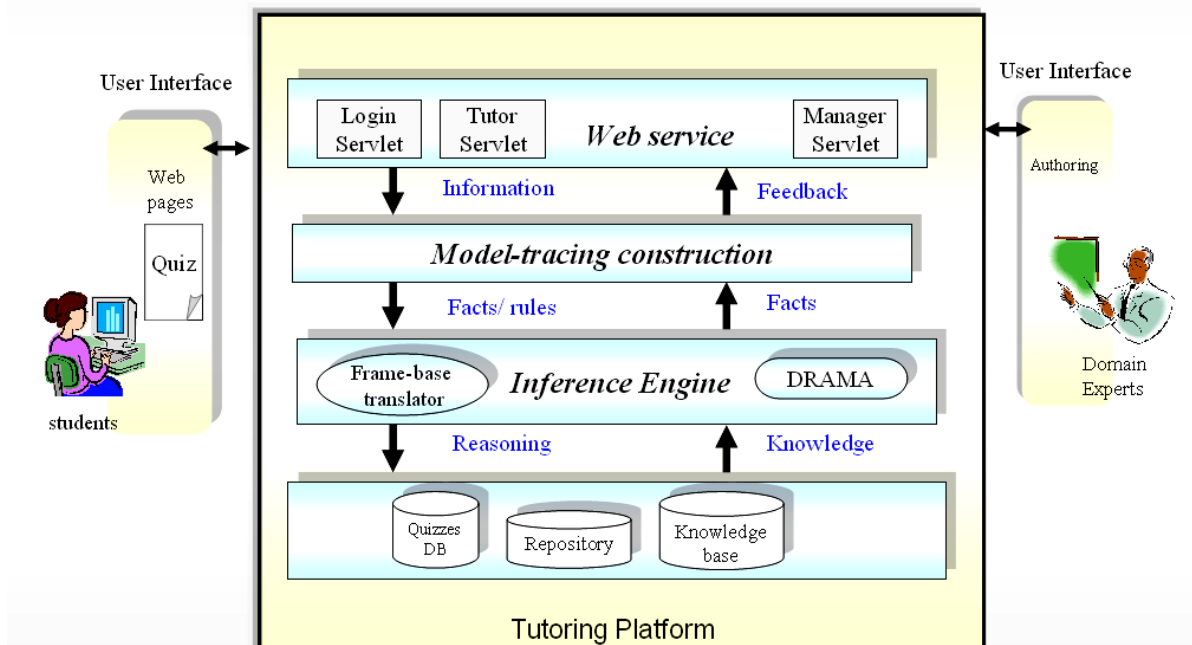


Figure 6.1: Prototype tutoring system of PROBLEM

This tutoring system is implemented by Java Server Pages (JSP). All users' data and quizzes data base are stored in MySQL 4.0.15 system. Overall tutoring system is executed on Apache Tomcat 5.0.29 system. DRAMA knowledge base is used as the foundation of our tutorial system. DRAMA, a NORM based rule base platform, is a product of Coretech Inc. [20], Taiwan, which is developed in cooperation with Knowledge and Data Engineering Laboratory (KDB Lab.) of National Chiao Tung University, Taiwan. DRAMA is implemented using Java, and it includes DRAMA Server, DRAMA Console, DRAMA Knowledge Extractor, and DRAMA Rule Editor. DRAMA Server is implemented to manage rule base, which is used to contain and process knowledge, and provide rule base services. NORM-modeled knowledge can be contained in DRAMA Server and inferred according to user given facts [36].



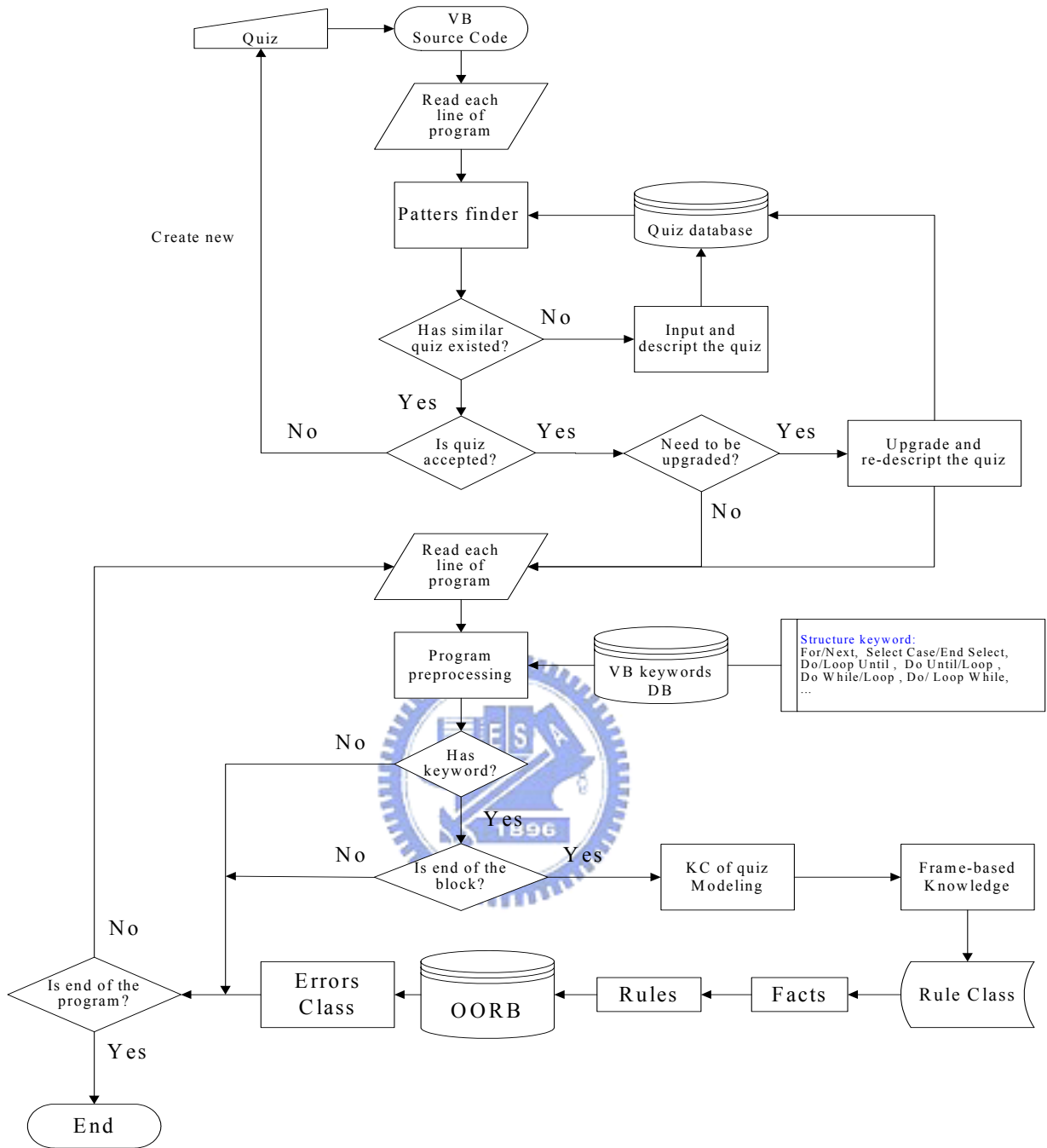


Figure 6.2: The flowchart of constructing quizzes for Model-Tracing tutoring

Six actions are available for teachers when they attempt to solve the possible problems of quiz. Including (1) adding nodes (2) modifying nodes (3) deleting nodes (4) adding links (5) modifying links (4) deleting links.

We define the sequences of elementary components of quiz that is student actions (steps), we called them Model. The system will mark the model with keywords when tracing source codes. Every quiz is described with certain nodes and links to represent the syntax of problem-based quiz. The procedures respond to events, situations and production rules. We describe briefly a student's learning activity who learns the programming in the PROBLEM tutoring system, as shown below:

Step1: The student key in his Id and password to login the teaching web site. The system chooses a quiz for the student automatically.



Figure 6.3: Students login web page

Step 2: The tutor gives some suggestions. The student clicks on the help button, and then the system displays required structures of this program for the student.

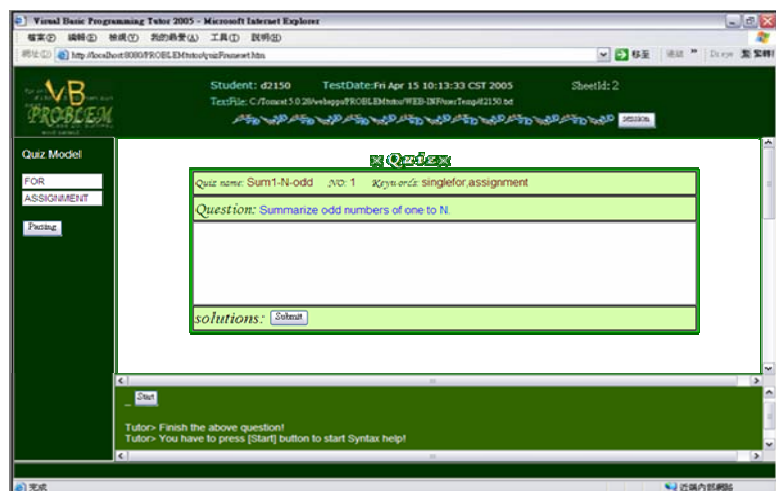


Figure 6.4: A quiz web page

Step 3: The student has to input the source code into those textbox of Single For structure. Sometimes the system may be give another related question to induct the student tends to the correct answer. The tutor keeps on giving guidance to remind the student till finishing the quiz.

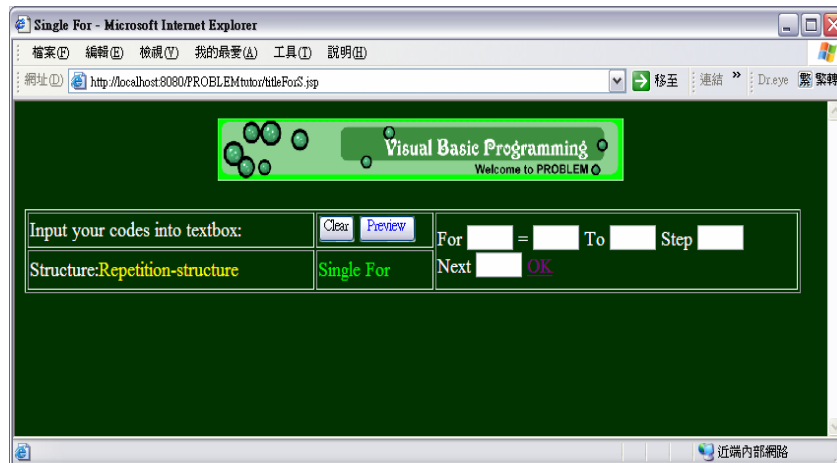


Figure 6. 5: Single For structure web page

Step 4: The system will compare the teacher's correct answer with the student's incorrect answer for picking up the mismatching slots to provide questions and guidance for students whenever the student deviates from the adaptive programming solution.

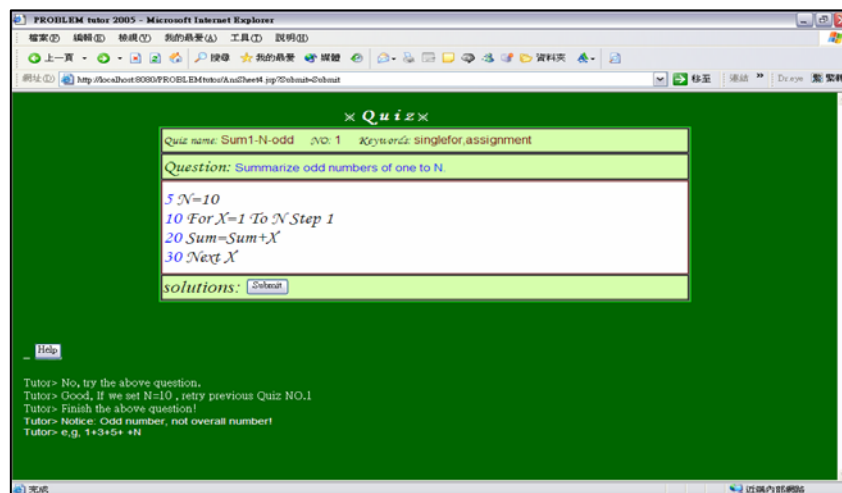


Figure 6.6: Model-Tracing tutoring web page

By comparing the correct answer with the incorrect answer, we can pick up the student's errors and then to take the right remedial steps to inform the student corrects those

shortcomings. Above the diagnosis of frames are according to the rule-based knowledge reasoning as shown in below. The following is that the rule “R01-FN” is fired to generate “ErrorOccurred” fact by DRAMA inference engine.

Example 6.1: Using Bubble Sort quiz for example

■ **Facts:**

F12-FN-Outside-ending-value = 'UBound(s)-2'
 F13-FN-Inside-ending-value = 'UBound(s)-2'
 F14-FN-Outside-variable-name = 'i'
 ErrorOccurred = 'false'
 E01_FN = false
 E01_desc_FN = 'null'

■ **Rules:**

R01-FN:

IF NOT ((F13-FN-Inside-ending-value = (F12-FN-Outside-ending-value - F14-FN-Outside-variable-name))) THEN ErrorOccurred := *R01-FN*;

R02-E01-FN:

IF (E01_FN = true) THEN E01_desc_FN := *'The ending value of inside loop must be to subtract the variable name of outside from the ending value of outside'*;



Once the error rule is fired, the tutoring system must decide to set fact while the error fact occurred. The following condition expression in the tutoring system decided the fact setting.

If ErrorOccurred='R01-FN' ***Then*** Set Facts E01_FN=true

The rules given by KE following the fact of frame, the inference engine will use the facts to trigger the rule inference.

Chapter 7. Conclusion and Future work

In this thesis, we propose an interactive Problem-based Visual Basic Programming Tutoring System using Model-Tracing approach. In conclusion, the tutoring system is based on the following techniques: *Model-Tracing* tutoring skeleton, *Problem-Based Learning (PBL)* environment, *Ontology-based* VB domain concept taxonomy, and *Frame-based* knowledge representation.

All this are used for constructing *Visual Basic (VB)* programming language knowledge which could be further shared and reused. Our *PROBLEM* tutoring system framework is a knowledge system which the elicitation and the validation of domain facts can be done. *Frame knowledge base* is used for foundation of this tutoring system. For this foundation, we proposed the two-phased system architecture for programming knowledge constructing and reasoning. Each of two-phased system is based on a problem-solving principle using *PBL* strategy. In our research, programming learning contents are presented to both teachers and students by quizzes. Any quizzes employ *Ontology* to manage required knowledge classes of VB domain. Overall our tutoring system takes advantage of *Model-Tracing* methodology skeleton. We have three contributions as follows:

- (1) Conduct interactive Problem-Based VB programming learning scenarios.
- (2) Combine Model-Tracing tutoring with Frame-based knowledge reasoning.
- (3) Construct Ontology-based VB programming learning environment.

Our work provides interactive and conducting programming learning for students. At present, the *PROBLEM* tutoring system can only deal with tracing VB programs syntactically.

Thus, we give the following suggestions:

- Develop adaptively quizzes based on student model for asking advanced questions in programming learning. Let each student receives an individualized practice session (learning contents, hints, guidance, etc.), i.e. a well mechanism of asking questions.
- Enable further reusing Knowledge Class Hierarchy of models constructed, e.g. Binary Search uses the Objects of Bubble Sort.

We hope deeply the related researches are going from strength to strength to support students learning in future.



Bibliography

- [1] Anderson J. R. and Reiser B. J (1985). The Lisp Tutor. *Byte*, 10, 159-75.
- [2] Anderson, J. R. & Pelletier, R. (1991). A Development System for Model-Tracing Tutors. In *Proceedings of the International Conference of the Learning Sciences*, 1-8. Evanston, IL.
- [3] Anderson, J.R., Corbett, A.T., Koedinger, K.R. & Pelletier, R. (1995). Cognitive tutors: lessons learned. *Journal of the Learning Sciences*, 4(2), 167–207.
- [4] Andrey Gavrilov. (2001). *The toolkit for development of hybrid expert systems*. IEEE.
- [5] Ani Amižić. (2001). *Model tracing – a Diagnostic Technique in Intelligent Tutoring Systems*. diploma thesis, Faculty of Natural Sciences, Mathematics and Education, University of Split.
- [6] Ani Amižić, Slavomir Stankov, Marko Rosić. (2002). *Model Tracing – A Diagnostic Technique in Intelligent Tutoring Systems*. Retrieved May 1, 2005, from web site: http://mapmf.pmfst.hr/~ani/znanstveni_radovi/CEEPUS_2002.pdf
- [7] Aspy, D.N., Aspy, C. B., and Quimby, P.M. (1993). What doctors can teach teachers about problem-based learning. *Educational Leadership*, 50(7), 22-24.
- [8] Becker, B.W. (1999, August). *Karel the Robot as an Example Object*, OOPSLA '99 Workshop Proposal. Retrieved April 10, 2005, from web site: <http://www.cs.uwaterloo.ca/~bwbecker/papers/oopsla99/>
- [9] Blessing, S.B. (1997). *A programming by demonstration authoring tool for model-tracing tutors*. *Int. J. Artificial Intelligence in Education*, 8, 233–261.
- [10] Boose, J. H. (1984). Personal Construct Theory and the Transfer of Human Expertise, *AAAI84*, 27-33.
- [11] Bratko, I. (1991). PROLOG: Programming for Artificial Intelligence.

Addison-Wesley Publishing.

- [12] Burns, H.L. and Capps, C.G. (1988). Foundations of Intelligent Tutoring Systems: An Introduction. *Foundations of Intelligent Tutoring Systems*, Polson M.C. and Richardson J.J. (Eds).
- [13] Corbett, A. T. & Anderson, J. R. (1991). *Feedback Control and Learning to Program with the CMU Lisp Tutor*. AERA Annual Meeting.
- [14] Corbett, A. T. & Anderson, J. R. (1992). LISP Intelligent Tutoring System: Research in Skill Acquisition. J. Larkin, R. Chabay & C. Sheftic (eds). *Computer assisted instruction and intelligent tutoring systems: Establishing communication and collaboration*. Hillsdale, NJ: Erlbaum.
- [15] Crews, T., & Ziegler, U. (n.d.). The Flowchart Interpreter for Introductory Programming Courses. Retrieved April 4, 2005, from web site: <http://fie.engrng.pitt.edu/fie98/papers/1107.pdf>
- [16] Crowley, R., Medvedeva, O. & Jukic, D. (2003, July). *SlideTutor: A model-tracing Intelligent Tutoring System for teaching microscopic diagnosis*. 11th International Conference on Artificial Intelligence in Education (AIED 2003), Sydney, Australia.
- [17] D. Lenat and R. Guha. (1990). *Building Large Knowledge Based Systems*. Addison-Wesley, Reading, MA.
- [18] Deek, F. P. (1999). A FRAMEWORK FOR AN AUTOMATED PROBLEM SOLVING AND PROGRAM DEVELOPMENT ENVIRONMENT. *Society for Design and Process Science (SDPS)*, 3(3), 1-13.
- [19] Deitel, H. M., Deitel, P. J. & Nieto, T. R. (1999). *Visual Basic 6 How to program*. NJ: Prentice Hall publishing.
- [20] DRAMA, Coretech Inc, <http://www.ctknow.com.tw/dtr/>, 2005.
- [21] Ellen Weber. (2002). Five-Phases To PBL: MITA (Multiple Intelligence Teaching Approach) Model For Redesigned Higher Education Classes. Retrieved June 10, 2005, from web site: <http://www.newhorizons.org/strategies/mi/weber3.htm>
- [22] E. Rich and K. Knight. (1991). *Artificial Intelligence*. McGraw-Hill, New York, NY,

2nd edition.

- [23] Erik floyd eilerts, B. A. (1994). KnEd: An Interface for a Frame-Based Knowledge Representation System. Master's thesis. The University of Texas, Austin, Texas, U.S.A.
- [24] Friedman, R. S. & Deek, F. P. (2002, November). *THE INTEGRATION OF PROBLEM-BASED LEARNING AND PROBLEMSOLVING TOOLS TO SUPPORT DISTRIBUTED EDUCATION ENVIRONMENTS*. 32nd ASEE/IEEE Frontiers in Education Conference. Boston, MA.
- [25] Gagne, R.M. (1985). *The Conditions of Learning*. New York: Holt, Rinehart and Winston.
- [26] Garner, S. (2003, June). *Learning Resources and Tools to Aid Novice learners Learn Programming*. Proceedings of the Informing Science + Information Technology Education (InSITE) Joint Conference. Pori, Finland. Retrieved April 4, 2005, from web site:
<http://proceedings.informingscience.org/IS2003Proceedings/docs/036Garne.pdf>
- [27] Giarratano, J. C. & Riley, G. (1998). *Expert system: principles and programming*. Boston: PWS publishing.
- [28] Gruber, T. R. (1995). Towards Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human and Computer Studies*, 43(5/6), 907-928.
- [29] Guarino, N. (1998). *Formal Ontology in Information Systems*. Proceedings of FOIS'98, Trento, Italy . Amsterdam, IOS Press, pp. 3-15.
- [30] Hayes, J. R. (1980). *The complete problem solver*. Philadelphia: Franklin Institute Press.
- [31] Heffernan, N. T. (2001). *Intelligent tutoring systems have forgotten the tutor: Adding a cognitive model of human tutors*. Dissertation. Computer Science Department, Carnegie Mellon University, PA. Retrieved December 28, 2004, from web site:
<http://gs260.sp.cs.cmu.edu/diss>
- [32] Heffernan, N. T. & Koedinger, K. R. (2001). Building a 3rd Generation ITS for

Symbolization: Adding a Tutorial Model with Multiple Tutorial Strategies.
Computer Science Department, Carnegie Mellon University, PA.

- [33] Hsieh, S. J., Hsieh, P. Y., & Zhang, D. (2003, November). *WEB-BASED SIMULATIONS AND INTELLIGENT TUTORING SYSTEM FOR PROGRAMMABLE LOGIC CONTROLLER*. 33rd ASEE/IEEE Frontiers in Education Conference, Boulder, CO.
- [34] Lam, S. W. & Srihari, S. N.F. (1991). Frame-Based Knowledge Representation for Multi-domain Document Layout Analysis. *IEEE*. 1859-1864.
- [35] Lebbink. Henk-Jan., Witteman. Cilia L.M. & Ch. Meyer. John-Jules. (2004). Ontology-Based Knowledge Acquisition for Knowledge Systems. Retrieved December 28, 2004, from web site:
http://www.agent.ai/doc/upload/200407/lebb02_1.pdf
- [36] Lin, Y. T. (2004). Design and Implementation of a New Object-oriented Rule Base Platform. PhD thesis. College of Electrical Engineering & Computer Science, National Chiao-Tung University, Taiwan.
- [37] Liu, J. L. (n.d.). *Ontology-based DNS Model-Tracing Tutoring System*. Unpublished Ph.D. thesis, College of Electrical Engineering & Computer Science, National Chiao-Tung University, Taiwan.
- [38] Mayer, R.E. (1981). The psychology of how novice learners learn computer programming. *ACM Computing Surveys*, 3 (1), pp. 121-141.
- [39] Mayo, P., Donnelly, M. B., Nash, P. P., and Schwartz, R. W. (1993). Student Perceptions of Tutor Effectiveness in problem based surgery clerkship. *Teaching and Learning in Medicine*, 5(4), 227-233.
- [40] Minsky, M. (1975). A Framework for Representing Knowledge. In Patrick Winston (ed.), *The Psychology of Computer Vision*. (pp. 211-217). McGraw-Hill.
- [41] Miao, Y., Holst, S., Holmer, T., Fleschutz, J., & Zentel, P. (2000, May). *An Activity-Oriented Approach to Visually Structured Knowledge Representation for Problem-Based Learning in Virtual Learning Environments*. Proceedings of the 5th International Conference on the Design of Cooperative Systems (COOP'2000). Sophia Antipolis, France, 303-318.

- [42] MITECS. (1999). *MITECS: Frame-Based Systems*. Published by Massachusetts Institute of Technology. Retrieved March 26, 2005, from web site: <http://www.cs.umbc.edu/771/current/papers/nebel.html>
- [43] Motschnig-Pitrik, R. & Holzinger, A. (2002). Student-Centered Teaching Meets New Media: Concept and Case Study. *Educational Technology & Society* 5 (4).
- [44] Murray, T. (1997). Expanding the knowledge acquisition bottleneck for intelligent tutoring systems. *Int. J. Artificial Intelligence in Education*, 8, 222–232
- [45] Natasha. Friedman Noy, Ray W. Ferguson, and Mark. A. Musen. (2000). *The knowledge model of Protege-2000: Combining interoperability and flexibility*. In the Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000).
- [46] Pramuditha Suraweera, Antonija Mitrovic, Brent Martin (2004). The Use of Ontologies in ITS Domain Knowledge Authoring.
- [47] Rogalski, J. & R. Samurcay, (1990). Acquisition of programming knowledge and skills. *Psychology of Programming*. in J.-M. Hoc, T.R.G. Green, R. Samurcay, D. Gilmore (Eds.), pp. 157-174, London: Academic Press.
- [48] Rogalski, J. & R. Samurcay, (1993). Task analysis and cognitive model as a framework to analyze environments for learning programming. in E. Lemut, B. du Boulay, G. Dettori (Eds.), *Cognitive Models and Intelligent environments for Learning programming*. pp. 6-19, Berlin: Springer-Verlag.
- [49] Rowe, G., & Thorburn, G. (2000). VINCE - an on-line tool for teaching introductory programming. *British Journal of Education Technology*, 31(4), 359-370.
- [50] Smith, S. (1998). *Intelligent Tutoring System*. Retrieved March 20, 2005, from web site: <http://www.cs.mdx.ac.uk/staffpages/serengul/table.of.contents.htm>
- [51] Smith, P. A., & Webb, G. I. (2000). The Efficacy of a Low-Level Program Visualisation Tool for Teaching Programming Concepts to Novice C Programmers. *Journal of Educational Computing Research*, 2(2-2000), 187-215.
- [52] Soshnikov, D. (2002, Sept.). *An Architecture of Distributed Frame Hierarchy for Knowledge Sharing and Reuse in Computer Networks*. *Artificial Intelligence*

Systems, (ICAIS 2002). IEEE International Conference, 115 – 119.

- [53] Sowa, J. F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole Pub Co.
- [54] Stewart, T. & Bartrum, P. (2002). *CHALLENGE—An Authoring Tool for Problem-Based Scenarios, Delivered Alone or Over the WWW*. Proceedings of the International Conference on Computers in Education (ICCE'02), IEEE.
- [55] Surekha, M., Patki, A.B., Radha, G., SUDHA, A.V., SEKHAR, G.S., & SHANTHI, P. (1990). Frame based knowledge systems for EMC analysis. *IEEE*.
- [56] T. Lethbridge. (1994). Practical techniques for organizing and measuring knowledge. PhD thesis in progress at the University of Ottawa.
- [57] TURBAN, E. (1993). *EXPERT SYSTEMS AND APPLIED ARTIFICIAL INTELLIGENCE*, pp 117-139,
<http://www.scism.sbu.ac.uk/inmandw/review/knowacq/review/rev11656.html>
- [58] Uehara, S., Yamamoto, R. & Ogawa, T. (1989). LISP-PAL: An Approach to Natural Language Consultation in a Programming Environment. *IEEE*.
- [59] Uschold. M., King. M., Moralee. S., & Zorgios. Y. (1998). The Enterprise Ontology. *The Knowledge Engineering Review*, 13(1):31-89. Special Issue on Putting Ontologies to Use.
- [60] Vinay K. Chaudhri and Adam Farquhar. (1998). *Open Knowledge Base Connectivity 2.0.3 (Proposed)*. Technical report, SRI International.
- [61] Wild, M., & Quinn, C. (1997). Implications of educational theory for the design of instructional multimedia. *British Journal of Educational Technology*, 29(1), 73-82.