Ontological Support in Modeling Learners' Problem Solving Process

Chun-Hung Lu[1,2], Chia-Wei Wu[2], Shih-Hung Wu[3], Guey-Fa Chiou[1],Wen-Lian Hsu[2]

[1]Dept. of Information and Computer Education, National Taiwan Normal University, Taiwan, R.O.C

[2]Institute of Information Science, Academia Sinica, Taiwan, R.O.C

[3]Dept. Of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung, Taiwan, R.O.C

enrico@iis.sinica.edu.tw

cwwu@iis.sinica.edu.tw

shwu@cyut.edu.tw

gueyfa@ice.ntnu.edu.tw

hsu@iis.sinica.edu.tw

Abstract

This paper presents a new model for simulating procedural knowledge in the problem solving process with our ontological system, InfoMap. The method divides procedural knowledge into two parts: process control and action performer. By adopting InfoMap, we hope to help teachers construct curricula (declarative knowledge) and teaching strategies by capturing students' problem-solving processes (procedural knowledge) dynamically. Using the concept of declarative and procedural knowledge in intelligent tutoring systems, we can accumulate and duplicate the knowledge of the curriculum manager and student.

Keywords

Intelligent Tutoring System (ITS), Ontology, Procedural Knowledge, Student Model

Introduction

Scientists and researchers have carried out extensive studies on mental representations. A person's mental knowledge generally begins with noticing and remembering, and mental representation is called upon to provide knowledge. According to Anderson (Anderson, 1993), there are two essential components of spatial images: declarative knowledge and procedural knowledge. Declarative knowledge collects the factual or conceptual knowledge that a person has. In designing a generic architecture to represent procedural knowledge, the actions defined by domain experts and the control of action flow are two important tasks. Based on our existing ontology tool, InfoMap, we developed the same concept , called "Service-oriented Architecture" , to register all service, and compliance of OWL to describe the composition of services. Self (1999) showed that focusing on the process by which knowledge is constructed is more important than focusing on the target knowledge itself. By using the descriptions of classes, properties, instances and the descriptions of their relationships in ontology, the system can provide more robust function on reasoning.

In this paper, we proposed an ontological representation scheme called Process Map (PM) to represent procedural knowledge. The combination of behavior model (procedural knowledge) and ontology (declarative knowledge) has the advantage of allowing access to existing domain specific glossaries, taxonomies and ontologies from within the processes. If we regarded the repeated processes as reusable components, then we can identify (1) the activity structure from given behavioral models of components; (2) the correlations among these components; and (3) the log of information about a student's operational behavior. Most researchers use declarative knowledge as the sole basis for ontology simulation. However, in Sowa's opinion, a paradigm of declarative knowledge construction has largely failed to produce human-like cognitive processing in computers (Sowa, 1999). To cope with this situation, we have developed an ontology, called InfoMap (Hsu, 2001), based on both declarative and procedural knowledge (see Figure 1).

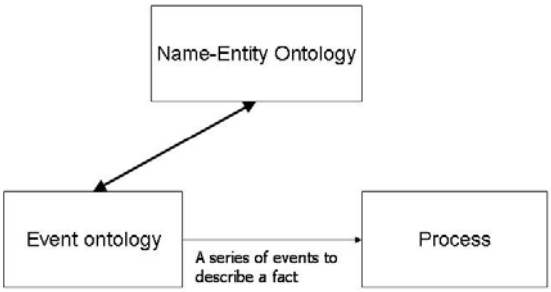

Figure 1 The conceptual architecture of our ontology

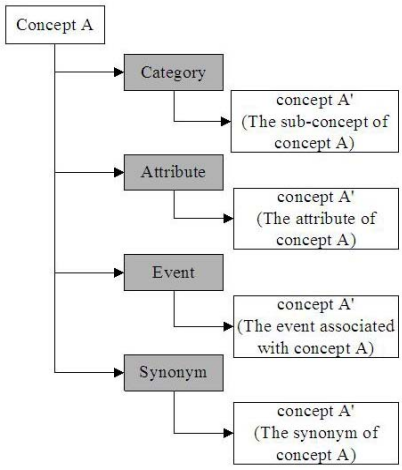

Figure 2. Ontology format of InfoMap

Basic representation

Our ontology is implemented based on InfoMap, which was originally created as a named entity ontology. We have now extended it to include event ontology and process map. We used this ontology to integrate an expert module, a student model, and the curriculum design.

InfoMap

Knowledge representation has long been an obstacle in simulating human understanding. Several strategies have been proposed for natural language understanding; however, many have been confined to illustrations in textbooks rather than actually implemented in large-scale natural language systems. The fact is that different representation schemes are appropriate under different situations.

Our knowledge representation scheme, InfoMap is designed to facilitate both human browsing and computer processing of the domain ontology in a system. The domain ontology is constructed from structured concepts in each specific domain.

Examples of concept structures range from simple concepts, such as a word, a phrase, or an event, to more complex concepts, such as a sentence, a paragraph, a script (a collection of related events), a story, or the passive tense of English, and so forth. Each concept is associated with a structure (a sub-map) describing the relationships of this concept to its related concepts. The system can store a large number of events, syntactic or semantic structures, and scripts. Given a natural language sentence, the system tries to match it to a sub-map or decompose it into several events within InfoMap.

We represent InfoMap as a tree hierarchy (Figure 2). There are two types of nodes: concept nodes and function nodes. The basic function nodes are: category, attribute, synonym, and event. They are used to label the relationships between two concept nodes.

Process Map

Process Map (PM) is a way to represent procedural knowledge, which can be treated as a series of processes connected by junctions and links. The direction of the flow of each instance is decided by the preconditions of each process. The steps of problem solving may also be recorded, as long as the processes are clearly defined. This is useful for people who want to detect the errors or track the history of a procedure. In this section, we introduce how to represent a PM and how to use a PM to describe procedural knowledge. In Section 4, we introduce the prototype of the process engine system.

Structure of the Process Map

Before describing how to express a PM, we present our idea of PM in Figure 2. It uses basic subtraction and is adapted from (Brown and Burton, 1978). Suppose we have a problem (p1):

  T3 T2 T1
- B3 B2 B1

Figure 3 represents the procedure for solving problem p1. The grey boxes are composite processes that can be further decomposed. For example, process C is a composite process that can be decomposed into processes F and G. Process G can be further decomposed into processes H, I and J. Actually, processes K, L and M, N, O can also be represented as composite processes, which are not shown here because of space limitation. In process I, we use a different method to show the same composite idea. A white box indicates an atomic process with or without preconditions or effects. The junction “Or” indicates a one-to-many relationship and a temporal constraint between the processes connecting them (Chen-Burger, Tate and Robertson, 2002). The details are discussed later.

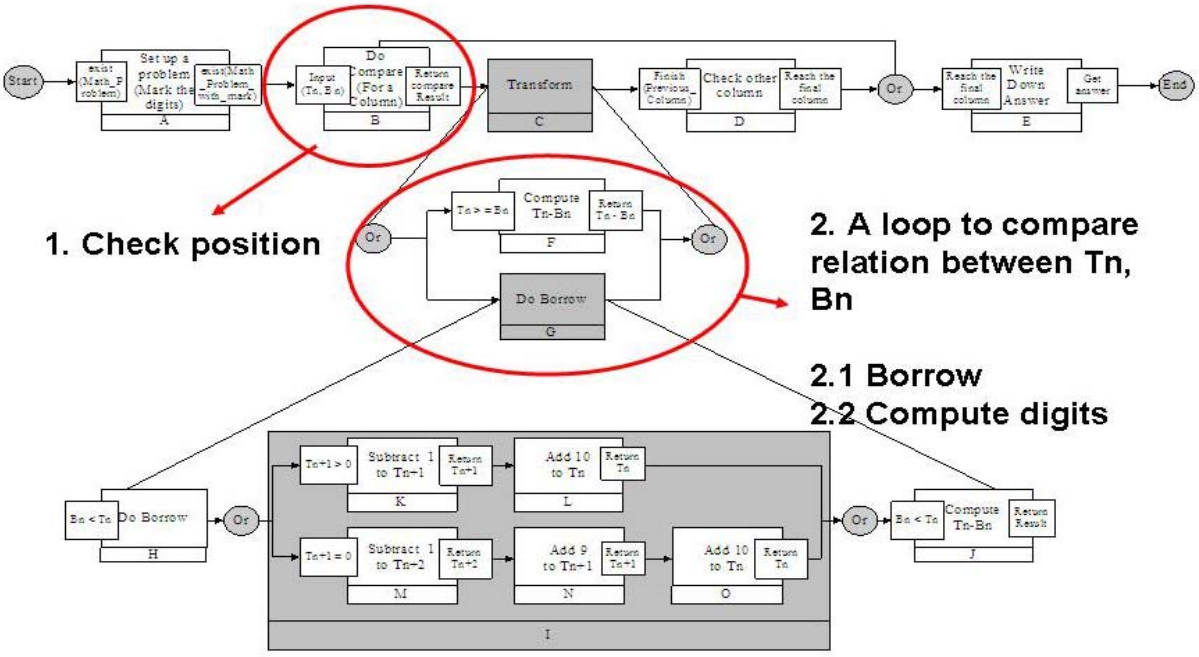

Figure 3. Visualized Process Map

As shown in Figure 3, PM is XML-format knowledge representation of the procedural knowledge in Figure 2. The original idea of PM was derived from a business process modeling language called FBPML, and a DAML-based web service ontology called DAML-S. FBPML is a visual and conceptual language that captures and describes the business processes of an organization. Any such procedure may be expressed using this language (Kuo, 2002). We borrow the concepts of junctions and links from FBPML, because they play an important role in decisions about the flow of the processes. The main structure of PM comes from the ontology for process models described in DAML-S. The main idea of process ontology of DAML-S is process decomposition. Process can be categorized as “Atomic”, “Composite”, or “Simple” (DAML Services Coalition, 2002). Here, we borrow the first two categories. A simple procedure can be represented as a single atomic process, while a complicated procedure can be represented as a composite process (or several composite processes). The latter can be further decomposed into many composite processes or atomic processes. The advantage of this model is that it presents different views of the same procedure (i.e. either a higher view or a detailed view of the same procedure). The procedures can be represented in a more structured way. Figure 4 shows the part of the process map that represents the procedural knowledge in Figure 3.

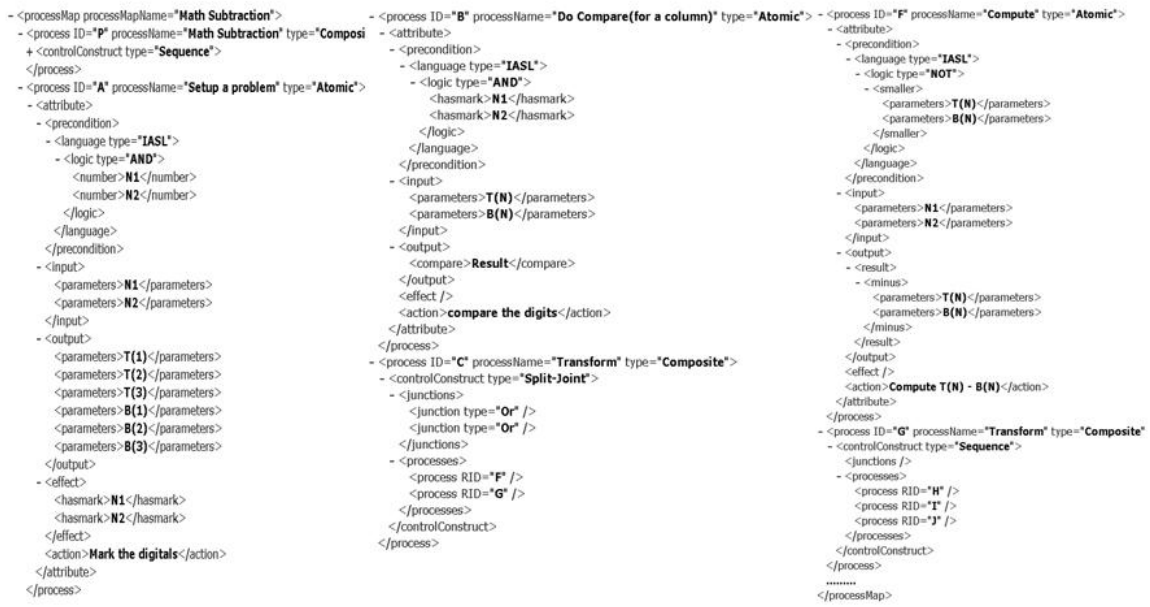

Figure 4: A Math Subtraction Process Map

Representation of a Process

In PM, a process can be categorized as atomic or composite with different parameters. An "atomic" process has three properties: ID, processName, type; and five attributes: precondition, input, output, effect, action.

Properties: An ID defines the ID of a process and must be unique for all processes in PM. A processName is the name of a process, which has nothing to do with the execution of the process, but is only used for human interpretation. A type indicates whether the process is atomic or composite.

Attributes: A precondition gives the conditions controlling the execution of the process. Procedural knowledge can be represented by a series of "If-Then" rules. The condition of the process is very important, because it determines whether a process will be executed or not. In Figure 3, the preconditions of process A stipulate that both subtrahend and minuend must be a number in order to execute the process of setting up a problem. In PM some basic precondition terms are provided such as "check" if a variable is a number, compare two different variables, etc. They can be composed into a logical form (i.e., "And", "Or" and "Not" can be used to form more complicated preconditions).

An output indicates the execution results of the process. Sometimes it may also provide suitable information about the preconditions of the next process. An effect indicates the additional effects (or the state changes of the object), which are produced by the process, but does not belong to the output. In process A, the problem marked by an ellipse is an effect of process A. For example, suppose we want to describe a reservation process. The output of the process ConfirmReservation will be a ReservationNumber. The effect will be a HaveFlightSeat status. An action indicates the action that will be executed in this process. We will discuss this more in Section "Logical Meaning of Junctions".

The properties of the composite process are the same as those of an atomic process, namely: ID, processName, and type. There is an additional controlConstruct attribute that indicates different compositions of the structures of the processes. The idea comes from DAML-S. Two composition methods are used here: Sequence and Split-Joint. A Sequence is the simplest control construct. The processes in a sequence are executed sequentially. Split-Joint is more complicated than Sequence. The processes in a Split-Joint can be executed in parallel when more than one precondition of the process are satisfied. At which point, all these processes will be triggered and executed. We can say a Split-Joint is a decision point in which the direction of the flow can be different as long as different data (information) is provided. Inside the controlConstruct tag, junction and process tags represent the type of the junction and the IDs of the connecting processes respectively. For example, in Figure 3, process C is a Split-Joint composite process composed of processes F and G with Or-Or junctions. We now give a more detailed explanation about the logical meaning of junctions, such as And-And, And-Or, Or-And, and Or-Or. It is represented as:

```
<controlConstruct type="Split-Joint">
        <junctions>
            <junction type="Or"/>
            <junction type="Or"/>
        </junctions>
        <processes>
            <process RID="F"/>
            <process RID="G"/>
        </processes>
</controlConstruct>
```

Logical Meaning of Junctions

Processes can be connected with junctions. A sequence is the simplest connection type. It does not have any logical value and only represents a sequential order of the connected processes. The process will be interrupted if preconditions cannot be satisfied. A split is a point at which one process can be split into more processes. A joint is a point at which two processes can be joined together. Joints are always paired in PM. The topologies can be divided into four different types: Or-Or, Or-And, And-Or, and And-And (Chen-Burger, Tate and Robertson, 2002). In Figure 5 (a), an And-And junction is composed from And_Split and And_Joint junctions.

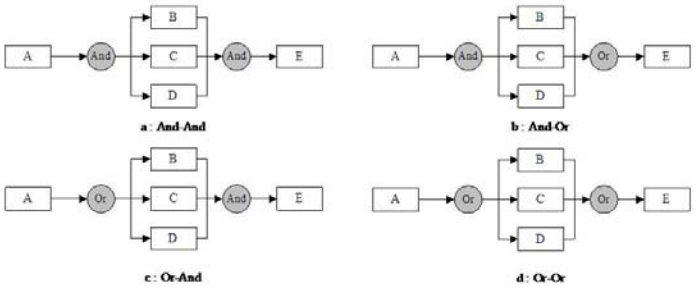

Figure 5. The different topologies of junctions

An And-And (And_Split and And_Joint) junction (Figure 5 (a)) indicates that when process A finishes, then processes B, C and D must start. After processes B, C, and D finish process E can start. The And-And combination has the strictest

restriction in a PM . The combination of Or-Or, as shown in Figure 4, (d) means that after process A finishes, at least one of the processes B, C, or D will be executed. Process E will not be started unless one of the triggered processes finishes. As process A finishes, suppose processes B and D are started. After process B or D finish, then process E can start. It does not need to wait for both processes B and D to finish. Thus, the Or-Or junction is a looser constraint than the And-And junction.

And-Or means that when process A is finished, processes B, C and D must start and be executed. If process B, C or D finishes, process E can start. It is different from an And-And junction in which all the processes B, C and D must finish before process E can start. The Or-And junction indicates that at least one of the processes B, C or D will start and will be executed after process A finishes. Process E will not start unless all of the triggered processes are finished. The triggered processes may be a combination of some processes (B, C, and D) Because of the Or_Split junction, it is not necessary to trigger all the preceding processes. Thus, it has more flexibility than the And_Split junction. These different combinations: “And-And”, “And-Or”, “Or-Or”, and “Or-And” can be used to represent and describe a complicated PM.

Based on these junctions, we can represent a complicated PM. A junction can be used to represent concurrent processes, which may not be all executed (Or-Or) or need to be executed concurrently (And-And) processes. It can also be used to represent a decision point that can be used to determine different flows of a PM. For example, as shown in Figure 3, an Or-Or junction is used to compare T(N) and B(N) in problem p1. This may produce different execution results for a PM based on different input data. Each instance may have a different flow as long as different data is provided. If we want to apply this to a problem-solving procedure, we can define suitable preconditions for different processes. To monitor the flow for different instances (students), we can extract some useful information to form a student model by tracking the flow. We will explain this in the session “The Teacher/Curriculum Manager Model”.

Actions

As mentioned earlier, PM is used to describe a procedure. Actions are the actual execution behaviors in a PM, and can be provided by experts, teachers or other teaching systems. Because actions can be stored in a repository, relevant information about the action should also be provided such as the purpose of the action and the input parameters of the action. When we describe a PM, this information will be used to find suitable actions for a particular purpose. A registry and a searching mechanism should be provided in the action repository. The advantage of the action repository is that actions can be reused and experts’ experiences can be shared. Action repository is also flexible in that, teachers can design a new PM to satisfy a new instructional goal by reorganizing the actions.

System architecture

In this section, we introduce the architecture of our system, which is comprised of 3 layers, as shown in Figure 6a. Our architecture addresses the interaction using the same representation scheme, InfoMap. Based on this representation , modules in this architecture form a dynamical cycle (shown in Figure 6b). More details will be described later.

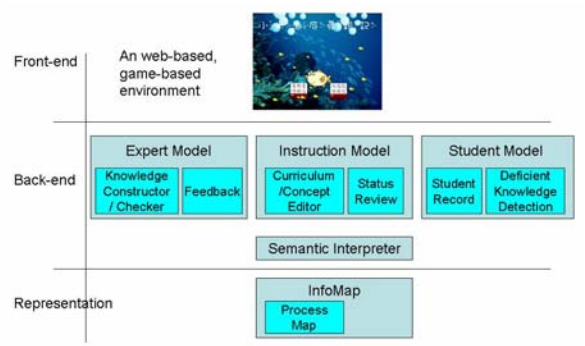

Figure 6a. Ontology format of InfoMap

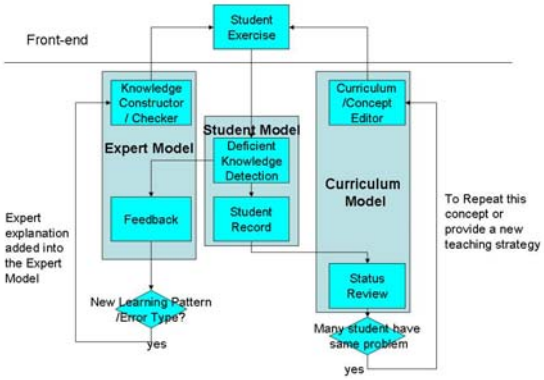

Figure6b. the cycle in our system

Front-end, a game-based environment

Playing computer games is a popular leisure activity for children. Macfarlane (McFarlane et al, 2002) reported that 85% of parents interviewed said that they thought that their children could learn something from playing computer games. They identified skill development in areas such as decision making, design, strategy, cooperation, and problem solving. Because of the belief that, enriching educational games with ITSs can help students learn effectively, we use the movie "Finding Nemo" to develop our ITSs for teaching Arithmetic to elementary school students. When a student logs in, the system will load the student's profile and information about the current teaching session.

In our system, one game stage is designed for one learning session and combined by a curriculum's sequence of sub-stages, which contains several smaller game units. Every stage will load some questions from the current learning session. When the student answers all the questions and achieves a 70% correct answer rate, the next stage will be shown with another contextual learning session. If the student's correct answer rate is below 60%, the system will provide another easier, contextual learning session. The front-end processing flow is shown in Figure 7. By using this design, computer games can be tightly integrated with arithmetic in a curriculum.

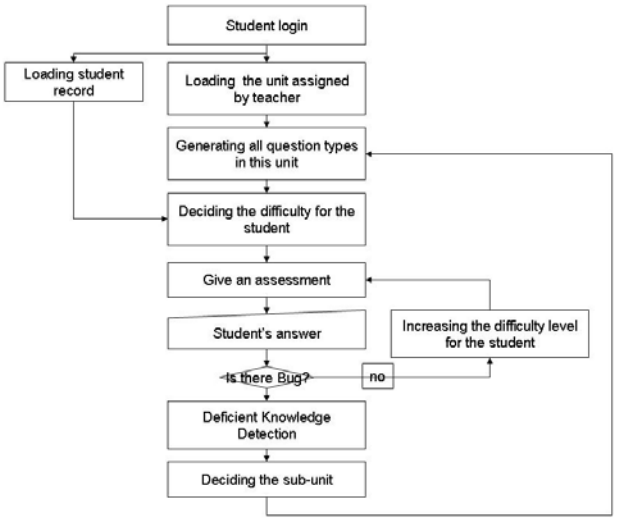

Figure 7. the processes of front-end

The Expert model

We can view an expert model as a repository for storing and organizing information. It should include knowledge that a teacher wants students to learn. With sensitivity analysis, a component or system can be examined to see how responsive its behavior is to differences in the information provided (Gaschnig et al, 1983). This could be particularly relevant when evaluating ITSs that offer individualized instruction. The sensitivity of an ITS towards different learner characteristics might indicate whether additional teaching expertise needs to be incorporated into the system. In the initial phase the domain expert determines what actions will be used. The system designer develops actions and registers them in a repository. At the same time, the expert constructs an exemplification to help teachers use these actions. After a complete course of interaction, the expert collects the teachers' comments to revise the actions.

In the problem (p1), we can use T3*100 + T2*10 + T1 to represent augends, B3*100 + B2*10 + B1 to represent addends, and X,Y,Z to represent answers (the sum). If n represents position, the arithmetic handles the relations between Tn and Bn. The expert knowledge of the arithmetic represented by InfoMap is shown in Figure 8.

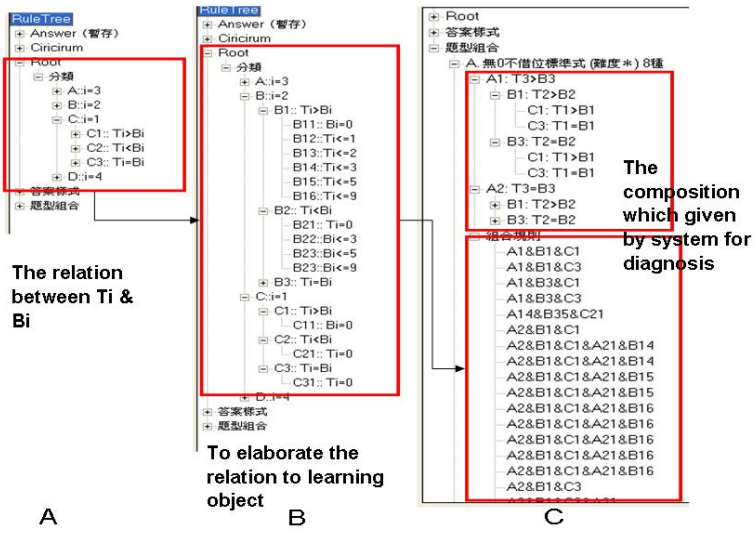

Figure 8. the UI of knowledge constructor

The Teacher/Curriculum Manager Model

The teacher uses curriculum manager (shown in Figure 9) to arrange learning modules (the lesson plan), where each module may include one or more learning objects to help students learn. Each learning object has its own teaching strategy. The teaching strategy and the curriculum can both be represented in InfoMap. The curriculum map gives teachers a more comprehensive understanding of what they should be prepared to teach. It can eliminate sequencing errors, and enable teachers to develop lessons that are truly interdisciplinary (Martin, 1994). Similar to an outline or a flowchart, we describe curriculum map by PM. Every element in the curriculum map can be regarded as a composite process that can be further divided into more detailed processes. Finally, we represent the procedural network of subtraction (Brown and Burton, 1978) by Process Map (as shown in Figure 3). Process Map can be used to represent teaching strategies in the curriculum manager and can also be arranged as post-conditions with error types in Process Map. After teachers have collected students' problem-solving procedures and error types, they can update new learning maps for students. Curriculum manager creates a sustained cycle: "curriculum design, teaching strategies design, recording (student's behavior), error analysis, and feedback on teaching strategies", which can help other teachers create good learning maps for their students.

Based on a student's diagnostic description in the session "The Student Model", we provide an interface for the teacher to understand the achievements and status of the student. When the teacher compares the overlay information of the student, he can regulate the error distribution to repeat some concepts that most students misunderstood.

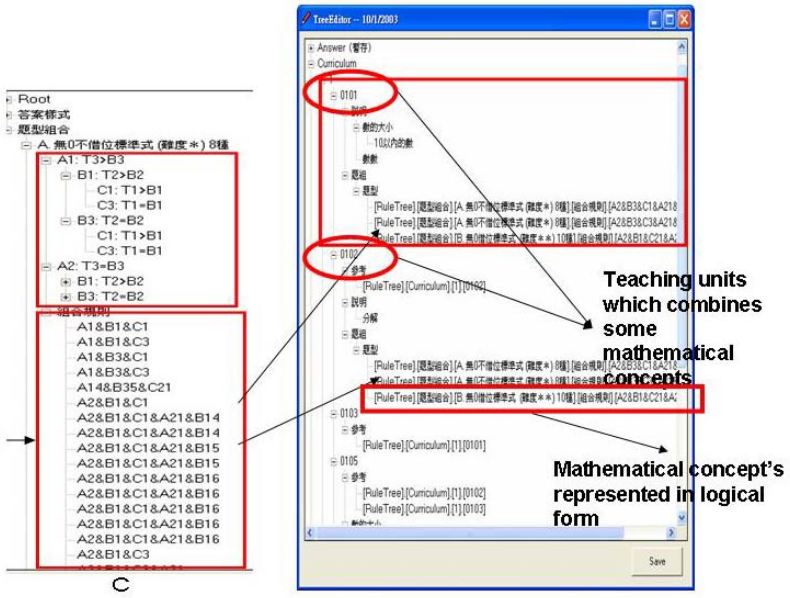

Figure 9 the semantic interpreter between Expert model and Curriculum Model

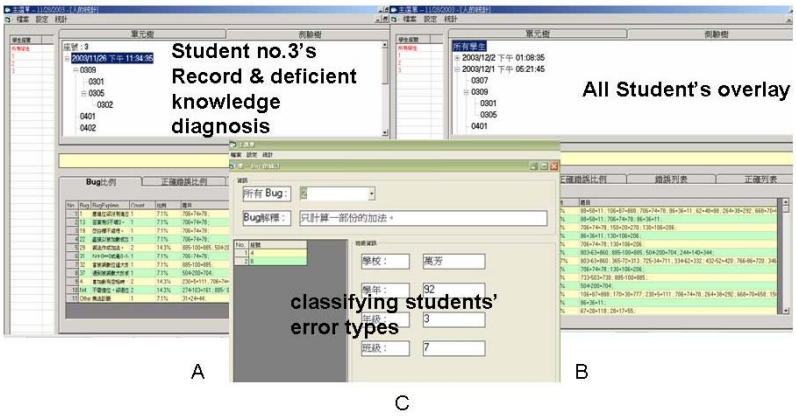

Figure 10. Student status review module

The Student Model Using Deficient Knowledge Detection

We have proposed a process called Identification, Simulation, Interaction and Mapping Schema (ISIM) for student modeling (Tu and Hsu, 2002). Our model is designed to detect not only a student's incorrect answers, but also the underlying cognitive reasons for such errors. In this paper, we extend the process to a student's deficient knowledge detection. The deficient knowledge identification is based on a buggy model & curriculum concept mapping. For this part of the study, we collaborated with Dr. Hue Chih-Wei of the Department & Graduate Institute of Psychology, National Taiwan University in (Hue , 2002).

Deficient knowledge identification is divided into 4 steps using Process Map. The 4 steps included : 1) student's behavior mapping/recording, 2)error analysis, 3) curriculum mapping , 4) deficient knowledge detection & suggestion.

In the first step, we adapt the procedural knowledge description reported by Brown and Burton (Brown and Burton, 1978, shown in Figure 11, you also can see this represented by Process Map in Figure 3). Brown and Burton proved that even many of the poor students were very consistent in applying a procedure to solve problems. By using Process Map to represent the procedure, we recorded the information about students' operations, concurrent session in curriculum.

In the error analysis step, Hue conducted an experiment, in which 2,590 students from 10 different schools participated in the test. Error analysis was preceded by two phases. In the first phase, the errors were divided into three groups:

1. Carelessness
2. Systematic and predictable errors
3. Random errors

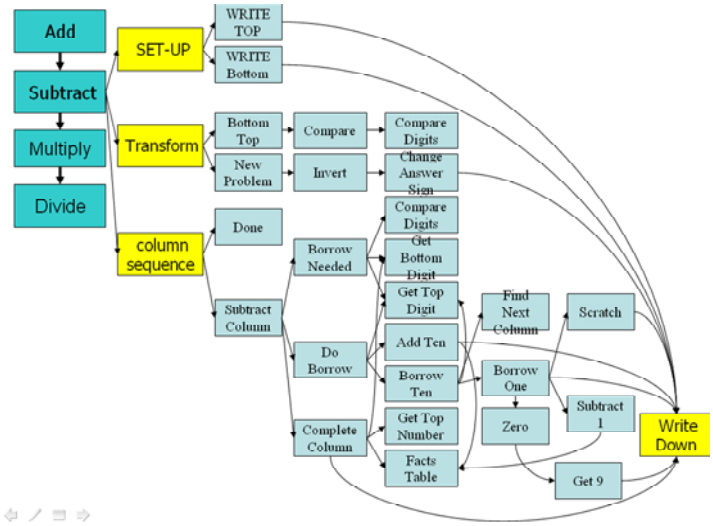

Figure 11. The procedural network of subtraction report by Brown & Burton

The results of the second phase can be roughly summarized as:

(1) 31 types of addition errors.

(2) 51 subtraction errors. Among them, there were 11 local errors not reported by Brown and Burton (Brown and Burton, 1978; VanLehn, 1990). Also, there were 57 subtraction errors reported by Brown and Burton(Brown and Burton, 1978) that do not apply to Taiwanese students.

The systematic errors described by Brown and Burton and our findings were translated into a logical format. For example, we translated the error "**0－N = 0/after/borrow**" into the rule representation as "**T2=1, T1＜B1, X=T3－B3, Y=0, and Z=T1＋10－B1".** When we finished the translations, we obtained the student's error data from the previous experiment to test the correctness of the logical representations. After finishing the test, we concluded 40 categories and added some error descriptions which discussed with five mathematical teachers to explain why the student made the mistakes.

In the third step, we use semantic information, which includes error type explanations and logical representations to relate to the Curriculum module. The Curriculum module provides information about the current session including the main concepts & contextual sessions. The system traced the past record of the student. By comparing with the record , the system could find out in which component the same error happened.

Finally, the system combines the results of the above three steps (procedural information, buggy model, current session & contextual sessions) into an ontological rule description; the system will provide some suggestions for a student's deficient knowledge. The final testing user interface is shown in Figure 12.

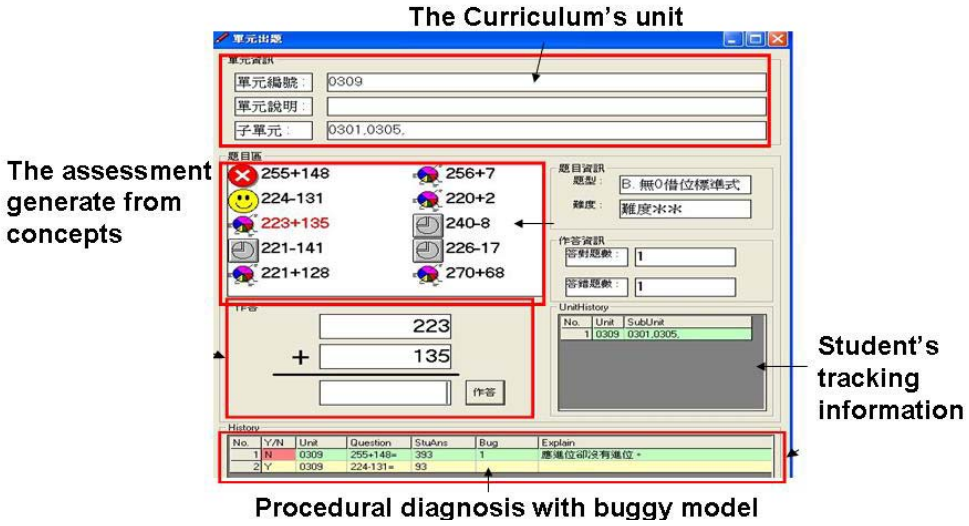

Figure 12 The integrated testing UI

Conclusions

This paper describes domain knowledge and student knowledge representation in our ITS based on InfoMap with semantic inferences. Based on ontological engineering, we can store, compare, merge, reuse all knowledge in repository by the context. By integrating the concepts of "registration", "reusable", "modeling" in InfoMap, we can provide a highly elastic architecture to implement ITS. Because there are many different learning styles, our ITS collects students' error types continually. In the future, various data mining techniques will be applied to semi-automatically identify (or cluster) the error types. In the experimental process, we found that using computer games in education can enhance the motivation of students.

By using Process Map in ITS, we hope to help a teacher accomplish the following tasks and accumulate teaching experience by observing the teaching strategies of experts and other teachers.

1. Develop teaching strategies with a personal style.
2. Observe students' learning maps.

3. Detect and classify students' error types and design appropriate teaching strategies.
4. Exchange teaching strategies and students' error types with other teachers.

The ITS implemented by InfoMap can also help students in the following way. If the student has any systematic and predictable misconceptions, the system could determine the underlying reasons for such errors based on experts' opinions. The Process Map can then record students' problem solving behavior, which could provide more feedback to teachers.

Acknowledgement

We would like to thank Dr. Hu Chih-Wei of the National Taiwan University for his earlier collaboration with us in computational scaffolding, which inspired this result. We would also like to thank the National Science Council for their generous support under Grant NSC93-2524-S-001-002.

References

Anderson, J. R., Rules of the Mind. Hillsdale, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ , 1993

Brown, J.S., & Burton, R.R. "Diagnostic models for procedural bugs in basic mathematical skills.", In Cognitive Science, 1978, pp. 155-192.

Burns, H. L. and Capps, C. G., "Foundations of Intelligent Tutoring Systems: An Introduction. Foundations of Intelligent Tutoring Systems." , Lawrence Erlbaum Associates, Hillsdale, NJ., 1988.

Carpenter, T. P., "Learning to add and subtract: An exercise in problem solving. In E. A. Silver (Ed.)", Teaching and learning mathematical problem solving: Multiple research perspectives, Hillsdale, NJ: Erlbaum., 1985, pp. 17-40

C. W. Hue C. H. Kao, M. Lo, L. Y. Tu and W. L. Hsu , "NTUs: An Intelligent Tutorial System Fosters Number Concepts Through Computational Scaffolding," Proceedings of ICCE'02, Auckland, (2002).

H.L. Wang, W.K. Shih, C.N. Hsu, Y.S. Chen, Y.L. Wang, and W.L. Hsu, "Personal Navigating Agent", Proceedings of AGENT'99, 1999.

H.L. Wang, S.H. Wu, I.C. Wang, C.L. Sung, W.L. Hsu and W.K. Shih, "Semantic Search on Internet Tabular Information Extraction for Answering Queries", Proceedings of the Ninth International Conference on Information and Knowledge Management, Washington DC, 2000.

Gaschnig, J., Klahr, P., Pople, H., Shortliffe, E. & Terry, A., " Evaluation of expert systems: Issues and case studies." In F. Hayes-Roth, D.A. Waterman, & D.B. Lenat (Eds.), Building expert systems. Reading, Massachusetts: Addison-Wesley, 1983.

L.Y., Tu, W. L. Hsu and S. H., Wu, "A Cognitive Student Model – An Ontological Approach," ICCE'02, Auckland, 2002.

Martin, D. J., "Concept Mapping as an aid to lesson planning: A longitudinal study.", Journal of Elementary Science Education, 6(2), 1994., pp. 11-30

Mcfarlane A., Sparrowhawk A. and Heald Y., "Report on the educational use of computer games. Teacher Evaluating Educational Multimedia report. Retrieved from http://www.teem.org.uk/publications/teem_gamesined_full.pdf

Roger. T. Hartley., "Representation of Procedural Knowledge.", Department of Computer Science Technical Report, New Mexico State University, February, 1998.

Self, J. The defining characteristics of intelligent tutoring systems research: ITSs care, precisely. International Journal of Artificial Intelligence in Education(10), 1999.

Sowa, J.F., “Knowledge Representation: Logical, Philosophical, and Computational Foundations.” , Brooks Cole Publishing, Pacific Grove, CA, 1999

W.L Hsu, “Elementary school Math. tutoring agent”, Proceedings of Agent Technology Workshop, 1997.

W.L Hsu, Y.S. Chen and Y.K. Wang, “Natural language agents – An agent society on the Internet” , Proceedings of PRIMA’99, 1999.

W.L. Hsu, Y. S. Chen and S. H. Wu, “Event Identification Based on the Information Map - INFOMAP, ” NLPKE’01, 2001, Tucson