of" and "has a". Schulz's idea is applied to our proposed work. Finally, Sangkae (2002) studies question-answering system and proposes the method to find an appropriate answer to a question in Thai. This work also suggests the possible forms of questions which can be generated in Thai language. Although this work does not directly involve the tutoring system, we apply possible categories of questions in the question construction process in our tutoring system.

3. Background

The three terms relevant to this work are data, information and knowledge. While data is raw and has no meaning by itself, information is data that has been given meaning by way of relational connection. Finally, knowledge is the appropriate collection of information, such that it is intent is to be useful. Figure 1 illustrates the pyramid of knowledge.

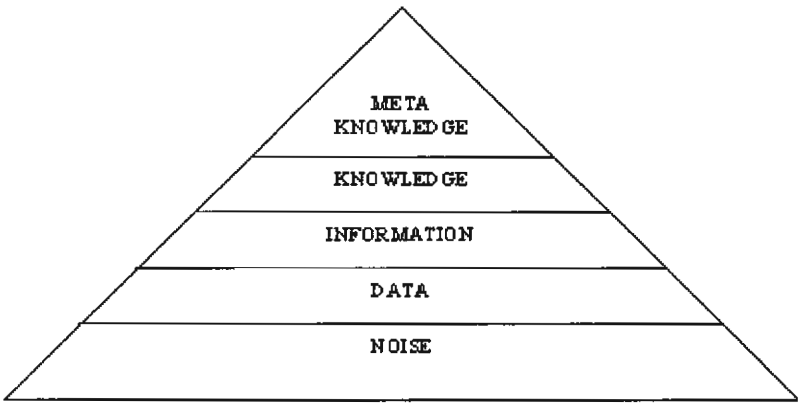


Figure 1 The pyramid of knowledge

Information in this case is an input sentence to our system. Our task considers the process that converts information to become useful knowledgebase in order to be used in the tutoring system.

Knowledge can be classified into following types

- **Declarative knowledge**

This type of knowledge is usually about the object or thing that has a particular name or location. Examples of this knowledge type in our domain are concepts of specific names and their definition.

- **Procedural Knowledge**

This type of knowledge tells us how to perform a given task. Knowing how to carry out a task can be referred to as procedural knowledge. An example of this knowledge type is knowledge about steps to declare a new variable in a program.

- **Strategic Knowledge**

This type of knowledge is comprised of information that is the basis of problem solving, such as action plans to meet specific goals; knowledge of the context in which procedures should be implemented; actions to be taken if a proposed solution fails; and how to respond if necessary information is absent. This type of knowledge is usually in the tutoring system to help students when they cannot solve the problem.

- **Tacit Knowledge**

Knowledge that we unconsciously use without being able to express it by language. This type of knowledge seems not to involve in our specific domain.

4. Semantic Relations and Concepts

According to the definition from Wordnet, semantic relation is the relation between meanings. In our project, we use **"Concept"** to refer to any entity. We use the word **"Relation"** in this project to refer to the semantic relation between concepts. Semantic network is the most popular graphical method for the representation of knowledge. The network comprises of nodes that are connected by arcs, which show the relationship between the nodes. A node can represent an object, concept or an event. An arc can represent the type of relationship between nodes.

Since semantic network is a hierarchy, the various characteristics of the nodes are inherited by any connected nodes at a lower level. In Figure 2.2, all the properties of node "ตัวแปร Integer" can be inherited to "ตัวแปรจำนวนเต็ม". For example, node "ตัวแปรจำนวนเต็ม" can have the relation "ประกอบด้วย (HasA)" to the node "จำนวนเต็มบวกและลบ" as well. That means there is an implied knowledge "ตัวแปรจำนวนเต็มประกอบด้วยจำนวนเต็มบวกและลบ" which is implied from "ตัวแปร integer ประกอบด้วยจำนวนเต็มบวกและลบ" etc.

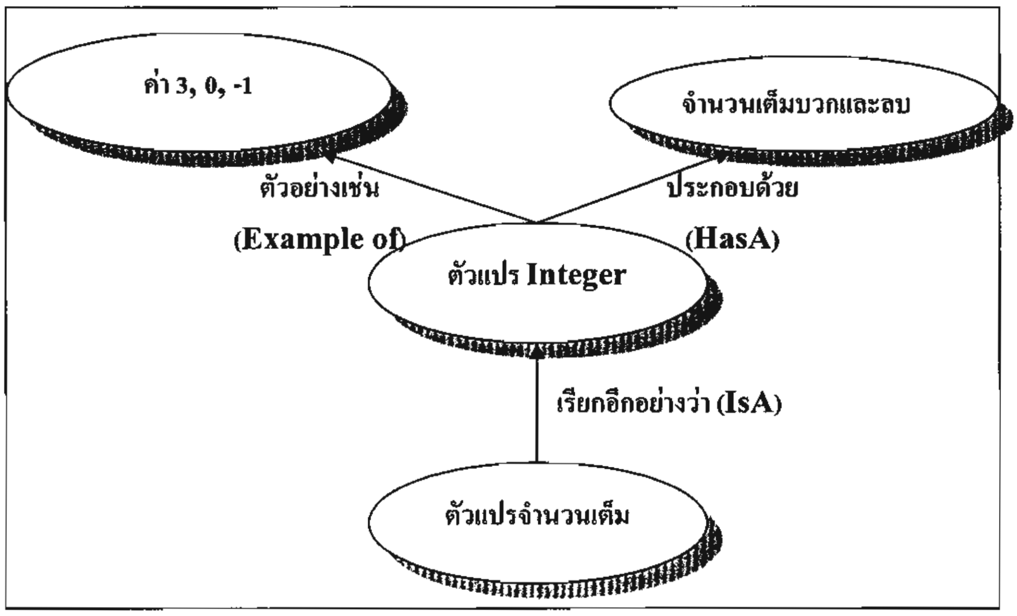


Figure 2 The Semantic Network

5. Knowledge Acquisition System

The term "Knowledge Acquisition" was originally used in the field of artificial intelligence, for the building of expert systems. It refers to the "transfer and transformation of problem-solving expertise" from a knowledge source, or information. Generally, to construct the system, we need the **knowledge engineer** to work with the **expert** in focused domain. The knowledge engineer has the goal of producing software that would perform as a human expert should while an expert is someone who has in-depth knowledge and expertise in a specialized field.

In general, knowledge acquisition process can be decomposed into 4 stages as in Figure 3

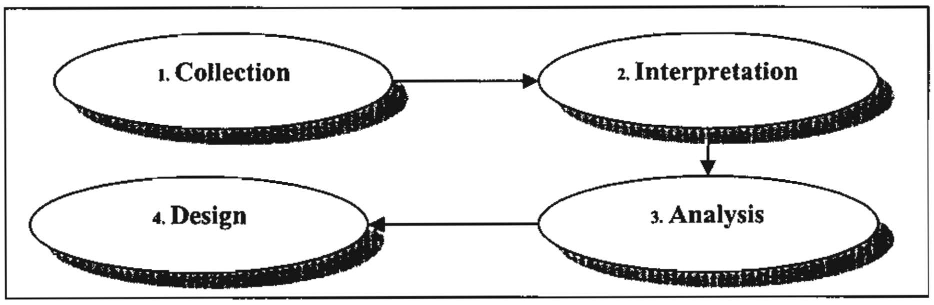


Figure 3 The process in knowledge acquisition

The process in knowledge acquisition can be divided into:

- Collecting
The knowledge engineer typically starts collecting knowledge by conducting interviews with the domain expert.

- Interpretation
At some stage, the knowledge engineer will have collected a satisfactory amount of material and will start working on a more detailed interpretation, organizing and comparing the knowledge obtained from different sources/methods with the knowledge based system objectives.

- Analysis
This interpretation will in turn evolve into a further analysis of the knowledge, now already with the aim of creating a knowledge-representation frame. By now the knowledge engineer will be relating the key pieces of knowledge.

- Design
Once the knowledge engineer has achieved a sound understanding of the key issues involved in the domain and how they relate to each other, he will attempt to design a precise representation of such knowledge. This will then be used to iteratively refine the knowledge collection, interpretation and analysis until an acceptable level of representation is achieved.

Next, we discuss the possible relations between concepts in the domain of C-Programming subject as well as the associations among them. Finally, concept categories will be described. In our knowledge acquisition system, the expert inputs one statement at a time. Mostly, a Thai sentence contains two concepts and one relation as shown in Figure3.1.

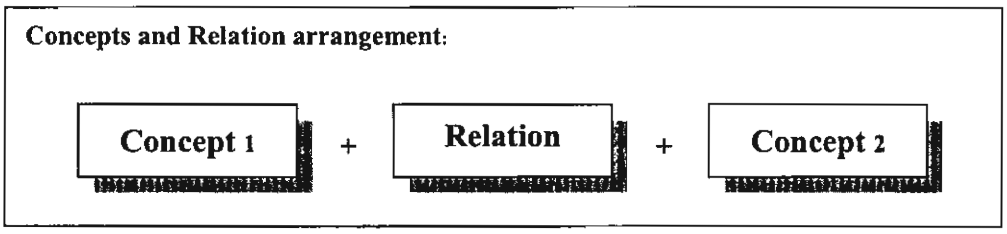


Figure 4 An arrangement example of concepts and relations in a Thai sentence

Next, we are going to discuss possible styles of the associations among relations. The study on the associations among relations is profitable to infer a new knowledge from the existing ones. This is a beneficial technique to the tutoring system to construct many different aspects of questions from a knowledge base.

Among the sixteen kinds of relations, there are three types of association characteristics among relations: Independent, one-to-one association and one-to-many association.

5.1 Independent
There are some relations that have no relation to other ones. In the question construction, this kind of relation can produce only a question from its origin.

5.2 One to One Association
Of this kind of association, there are two relations relating to each other. Furthermore, there are still sub types of such association which are one-way and two-way association.

5.2.1 One-way association
Figure 5 shows how Relation1 relates to Relation2. This kind of representation means that Relation2 is a subset of Relation1. Also, from this kind of relation, if X is an example of Relation1, then we can imply that X can have the Relation2 as well. But, if the kind of Y is Relation2, we cannot conclude that Y implies the Relation1, anymore; because it is one-way association from Relation1 to Relation2. This method is used in the inference process.

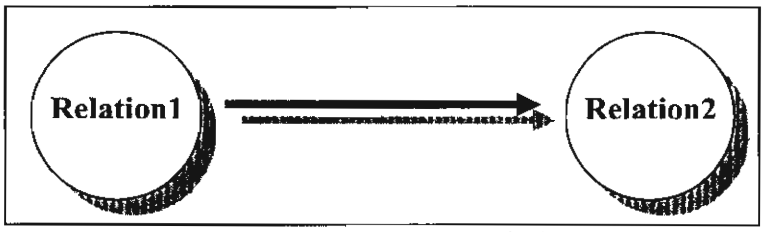


Figure 5 One-way association

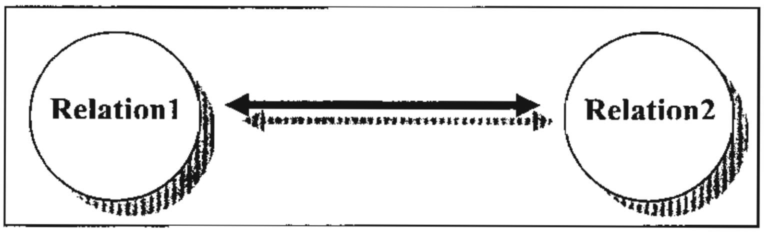


Figure 6 Two-way relation

As illustrated in Figure 6, **Two-way association** is a type of association that two components relate to each other with any aspect of associations. The associations may represent the same or opposite meaning between the two relations. This kind of association helps inferring process since if the input had the property of Relation1, it would have the property of Relation 2 as well.

Also, there exists the **inverse relation**. An obvious example of **the relation** and **its inverse** are the **active voice** and **passive voice.**

For example, in general domain, from the sentence, "The man hits the dog", the relation in this case is "to hit"; while the sentence, "The dog is hit by the man", has the relation between concept "to be hit". We can see that these two sentences have the same meaning but have different position of two nouns ("the man" and "the dog") as well as the aspect of the relation: "to hit" and "to be hit". That is, the relations "to hit" and "to be hit" are inverse relations to each other, as shown in Figure 7.

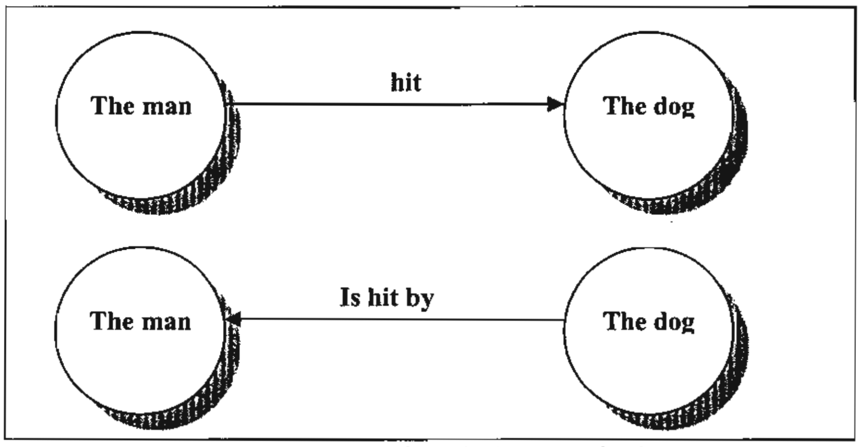


Figure 7 The semantic relation and its inverse

To sum up, the **inverse relation** is a relation that makes the whole sentence have the same meaning but it uses different relation word and the different positions of nouns or concepts, usually, the two nouns or concepts swap the positions.

5.2.2 One-to-Many Association

Figure 8 shows the illustration of this kind of association. In this figure, Relation2, Relation3 and Relation4 are the subsets of Relation1. If X's type were Relation1, X can imply to Relation2, Relation3 and Relation4, but not vice versa.

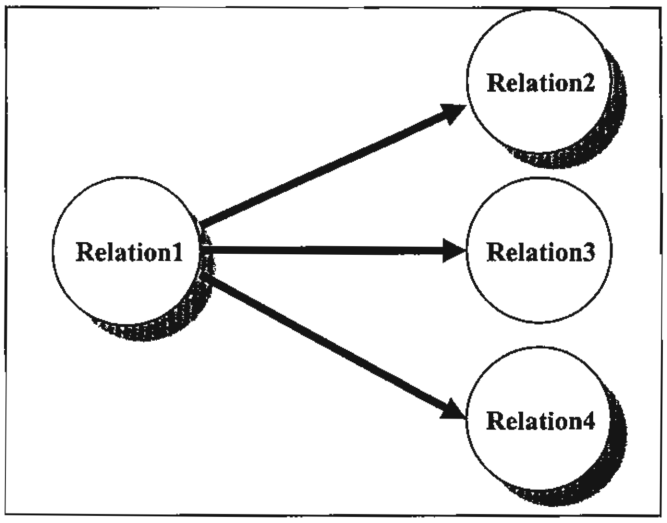


Figure 8 One-to-many association

From the above three kinds of association, we can easily construct the complex nest association as in Figure 9. In this figure, if X is a concept in Relation2, we can imply to Relation1. Furthermore, Relation1 has Relation3 and Relation4 as its subsets; so does Relation2. Relation5 is also a subset of Relation2 as well since it is a subset of Relation4, which belongs to Relation1.

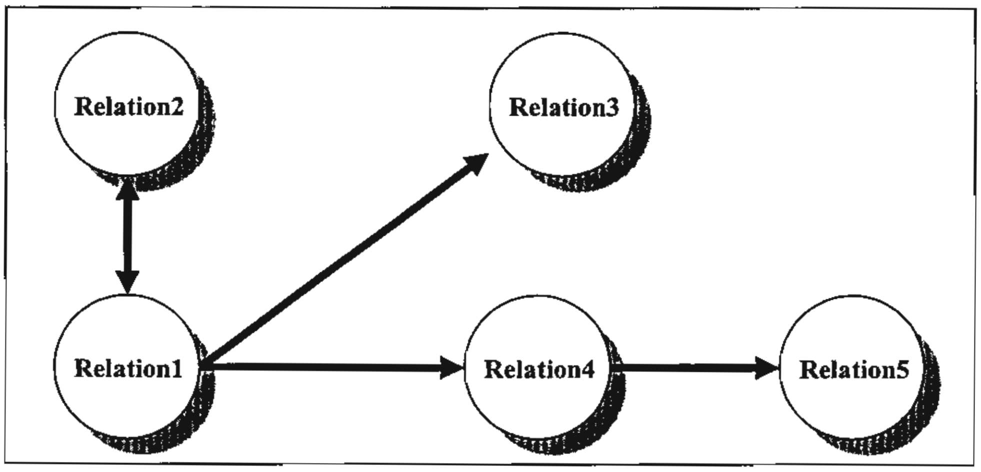


Figure 9 A complex nest association

6. Knowledge Base Format

From a sentence that is consisted of Concept1, Concept2 and Relation, its format is kept in database as follows.

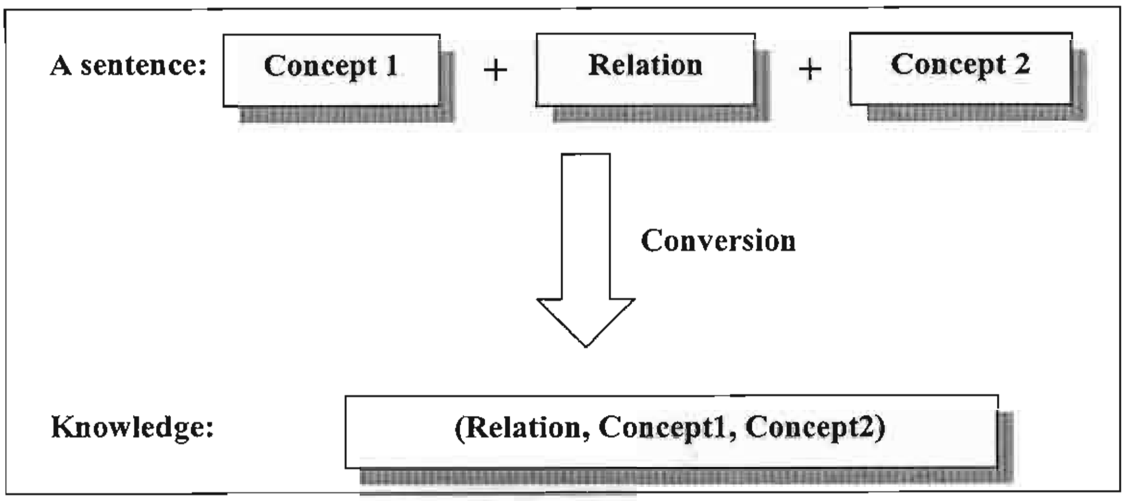


Figure 10 A sentence to Knowledge

Next, we are going to describe the classification of semantic relations as used in this project.

7. Overall Process

To acquire knowledge from input information and later construct questions, there are two main processes as illustrated in Figure 11.

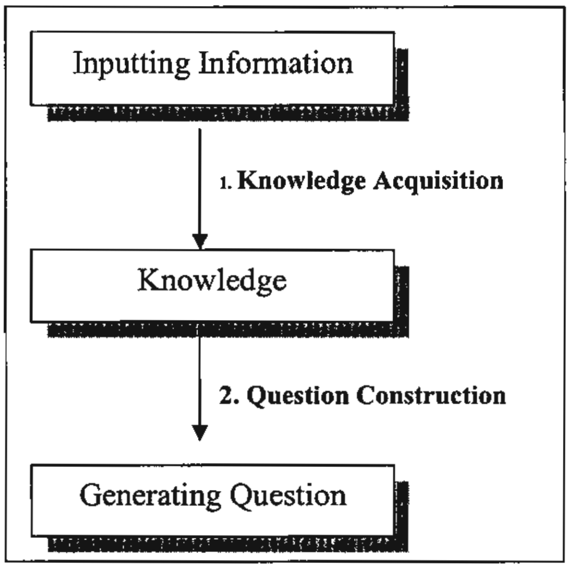


Figure 11 The overall process

The first part is **Knowledge Acquisition** process. This process will acquire the input information from the user, and then extract the knowledge and keep it in knowledge base in order to be used in the next part. The second part is Tutoring system, or **Question Construction** process. Generally, the Tutoring system includes reviewing the lesson, ask questions to the learners, and check the answers. However, this project will focus solely on Question Construction process.

8. Knowledge Structure

The structure to keep the knowledge in the knowledge base is as in table format (relation word and relation category, concept word and concept category and concept-relation). The last table shows the link between question forms and relation categories **Question-Relation table.**

To make the variety of question forms, each relation category must have various kinds of question forms. When a user wants to generate questions, there will be more than one question form from one set of knowledge.

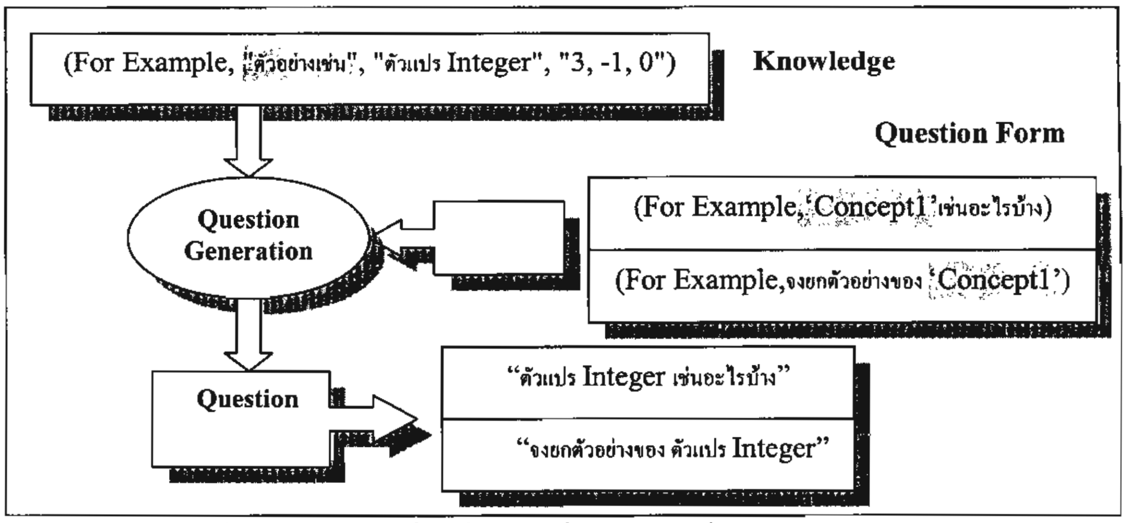

(For Example, "ตัวอย่างเช่น", "ตัวแปร Integer", "3, -1, 0")
Knowledge
Question Form
Question Generation
(For Example, 'Concept1' เช่นอะไรบ้าง)
(For Example, จงยกตัวอย่างของ 'Concept1')
Question
"ตัวแปร Integer เช่นอะไรบ้าง"
"จงยกตัวอย่างของ ตัวแปร Integer"

Figure 12 The simple question generation process

Figure 12 shows the simple process to match knowledge to question forms. From the knowledge ("For Example", "ตัวอย่างเช่น", "ตัวแปร Integer", "3, -1, 0") in which the Concept1 is "ตัวแปร Integer", in table **Concept-Relation** and the question forms in the table **Question-Relation,** we can generate two questions by replacing the word **'Concept1'** in question forms with "ตัวแปร Integer". Next, we are going to discuss the knowledge acquisition process in detail.

9. Knowledge Acquisition Process

The input of the system is in the form of sentence, which is assumed to contain two concepts and one relation. Figure 4.3 shows the processes in Knowledge Acquisition System part.

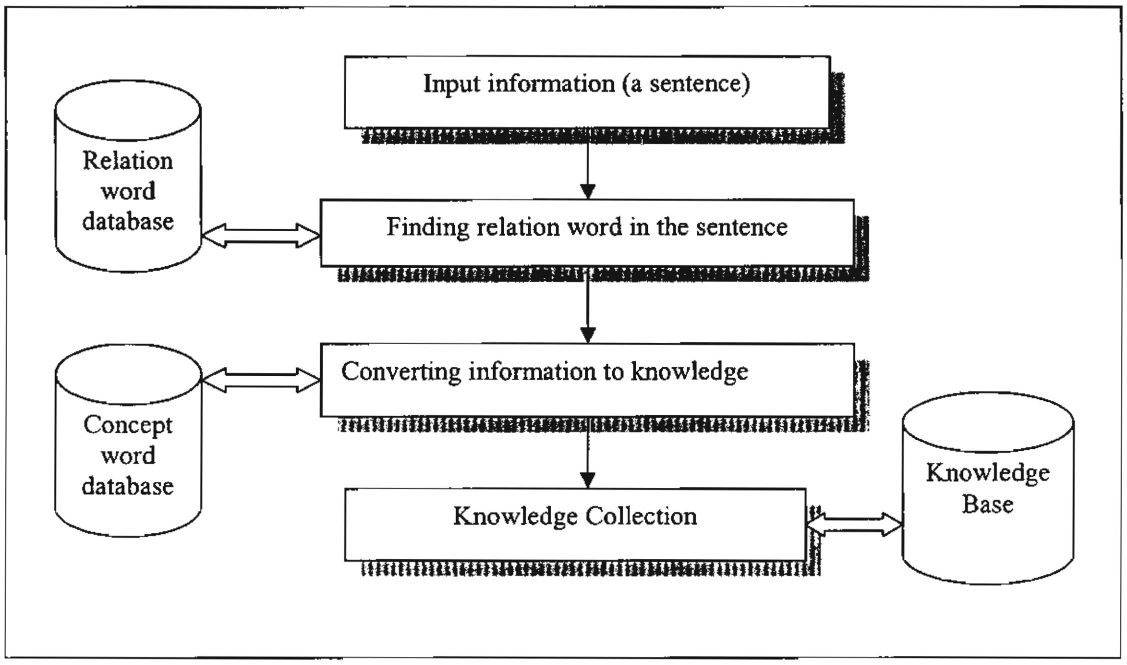

Input information (a sentence)
Relation word database
Finding relation word in the sentence
Concept word database
Converting information to knowledge
Knowledge Base
Knowledge Collection

Figure 13 The processes in Knowledge Acquisition System

The knowledge acquisition system proceeds as follows. First, the user input information in the form of individual sentences. We assume that there will be only two concepts and one relation in a sentence and that the alignment of concepts and a relation will be in order as Concept1+Relation+Concept2. Next, the system will query all relation words in the Relation table. Then, it will compare the relation words in the table to the input sentence. If there is a relation word in that sentence, the process will put that relation word and its category in the query result. Sometimes, there can be many results (more than one kind of relation category) since in different categories can have the same relation word. In this case, the process will report them to the user to make decision that which category the relation in the sentence should belong to that means the system is semi-automatic; the process and the user must work together.

In the case where there is no relation matching to the sentence or there is a match but it seems incorrect in the user's thought; the user can add a new relation word and its category. The new relation word can always be added into Relation table. Afterwards, the information from users will be converted into knowledge format. After the relation word is found in a sentence, the next step is to find the concepts in the sentence.

Concept1 is a group of words occurring before the relation word. **Concept2** is a group of word after the relation word. Also, the unknown concept word's category can be added depending on the user's decision.

Figure 14 shows the conversion between information, a sentence to knowledge. What contained in knowledge are **Relation-Category**, which determines the category of relation of the sentence, **Relation-Word**, which is a relation word of the sentence, **Concept1** and **Concept2**, which are concepts contained in the sentence.

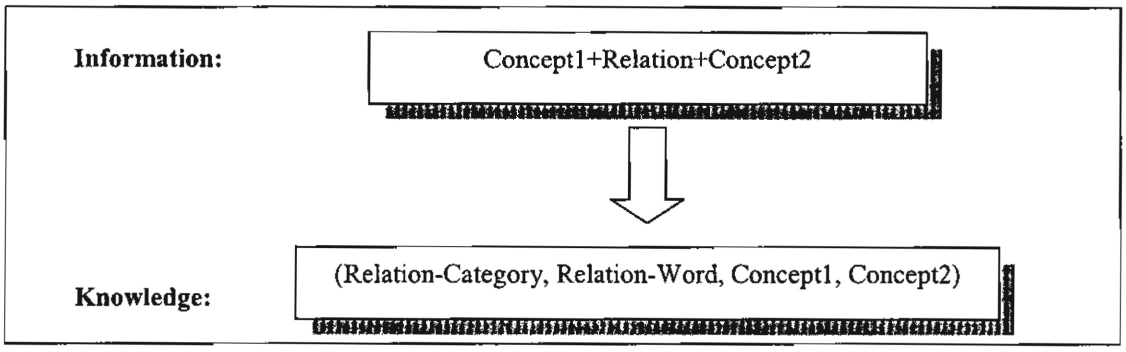

Information:
Concept1+Relation+Concept2
Knowledge:
(Relation-Category, Relation-Word, Concept1, Concept2)

Figure 14 Conversions from Information to Knowledge

Thus, the knowledge collection process simply puts new knowledge to Knowledge Base. If the new knowledge has already existed in Knowledge Base, the process will let the user know and decide what to do.

In the Knowledge Acquisition process, if there is an unknown relation word in the input sentence, the process will ask the user to add it and define its category immediately; otherwise the process will not go on. But, for the new concept found, the process will automatically add the concept in the concept database and assign its type as UNKNOWN unless the user wants to define its category in that time. The processing to a new relation word and a new concept are different because there usually are new concepts while the new relation words are rarely found.

10. Question Construction Process

To generate questions, the user can specify both relation category of the concept and relation to limit the scope of generation. For each relation category, there will be various question forms, which are collected in table "Question-Relation".

There can be four kinds of question forms in However, this project focuses only on generating questions of the types: Yes-No, Wh-word and alternative. The following questions are two kinds of question forms. And, the processes in the question generation is shown in Figure 15

Yes-No question: "จริงหรือไม่ที่ภาษาCจัดเป็นภาษาระดับกลาง"

Wh- question: "อะไรคือตัวแปร integer"

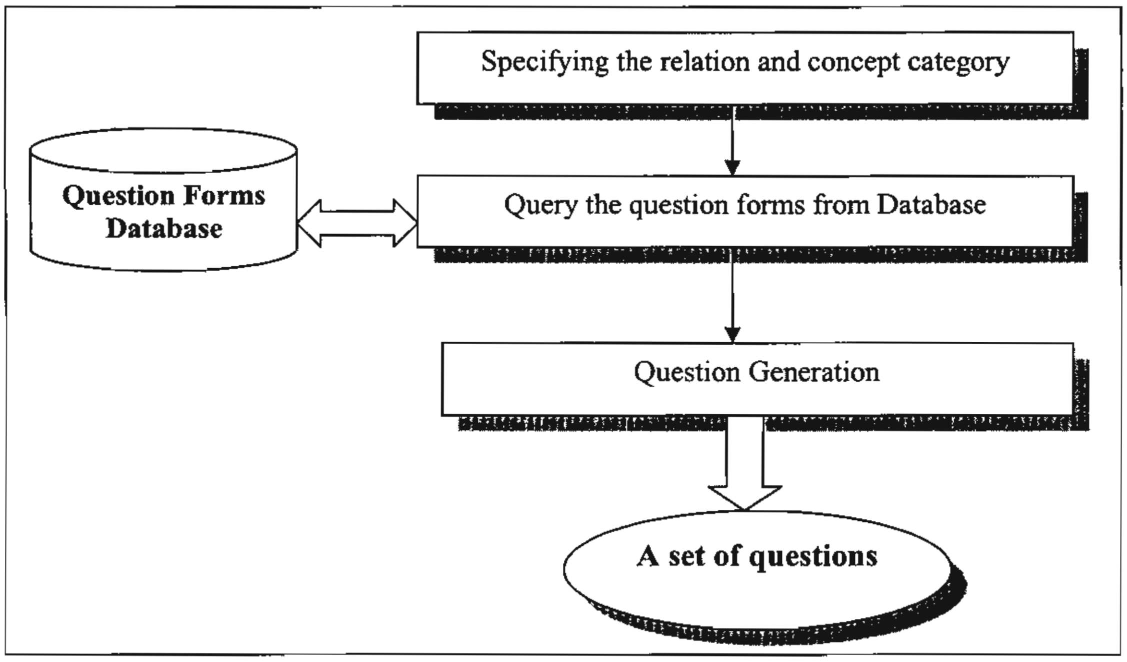
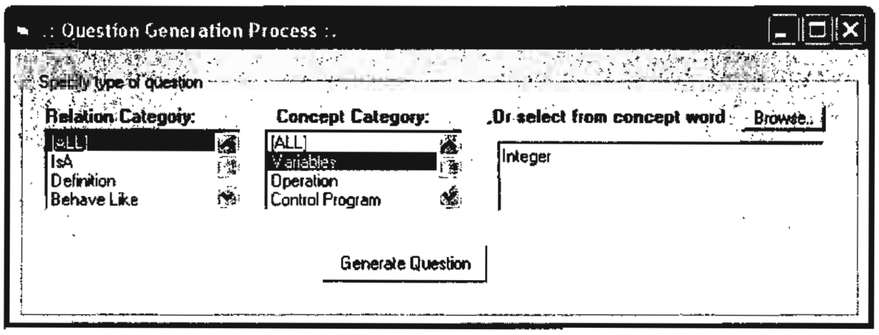

Specifying the relation and concept category
Question Forms Database
Query the question forms from Database
Question Generation
A set of questions

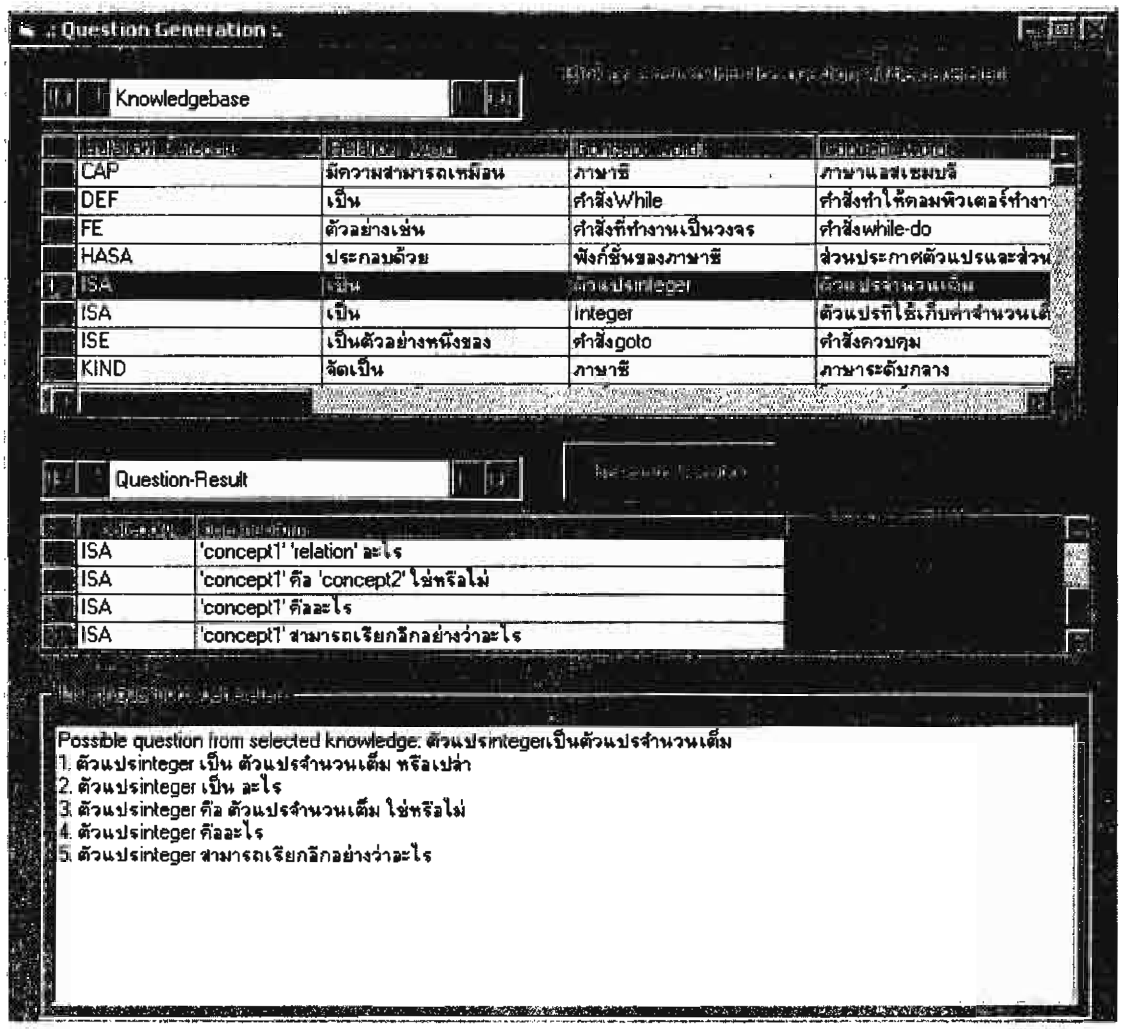
Figure 15 The processes in Question-Construction process



Integer

Figure 16 The input interface of Question Construction process

The user can choose the relation category, the concept category as well as to browse from the concept word in the knowledge. After submitting the request, the program will use this information in the next process. The program retrieves the question forms corresponding to the user's specification. The question form kept in database can be divided into two categories: **Yes-No** and **Wh-question form.**



[illegible]	[illegible]	[illegible]	[illegible]
CAP	มีความสามารถเหมือน	ภาษาซี	ภาษาแอสเซมบลี
DEF	เป็น	คำสั่งWhile	คำสั่งทำให้คอมพิวเตอร์ทำงา
FE	ตัวอย่างเช่น	คำสั่งที่ทำงานเป็นวงจร	คำสั่งwhile-do
HASA	ประกอบด้วย	ฟังก์ชั่นของภาษาซี	ส่วนประกาศตัวแปรและส่วน
ISA	เป็น	ตัวแปรinteger	ตัวแปรจำนวนเต็ม
ISA	เป็น	Integer	ตัวแปรที่ใช้เก็บค่าจำนวนเต
ISE	เป็นตัวอย่างหนึ่งของ	คำสั่งgoto	คำสั่งควบคุม
KIND	จัดเป็น	ภาษาซี	ภาษาระดับกลาง

[illegible]	[illegible]
ISA	'concept1' 'relation' อะไร
ISA	'concept1' คือ 'concept2' ใช่หรือไม่
ISA	'concept1' คืออะไร
ISA	'concept1' สามารถเรียกอีกอย่างว่าอะไร

Possible question from selected knowledge: ตัวแปรintegerเป็นตัวแปรจำนวนเต็ม
1. ตัวแปรinteger เป็น ตัวแปรจำนวนเต็ม หรือเปล่า
2. ตัวแปรinteger เป็น อะไร
3. ตัวแปรinteger คือ ตัวแปรจำนวนเต็ม ใช่หรือไม่
4. ตัวแปรinteger คืออะไร
5. ตัวแปรinteger สามารถเรียกอีกอย่างว่าอะไร

Figure 17-- The result of question generation

11. Experimental Results and Conclusion

For our knowledge acquisition system, with these sixteen kinds of the relation categories, almost all information from the focused domain of C-programming subject can be categorized into the suitable classes. However, if the system cannot recognize the relation word in the sentence, it is the users' task to decide which category of that sentence should belong to. Therefore, the user to input information to the system should be one who has an expertise in this domain

In the tutoring system, to generate questions from existing knowledge, there are two kinds of question form which are: Yes-No and Wh-question. the user just specify the type of knowledge to generate question. The results are various forms of questions which can be used for tutoring purpose.

REFERENCES

1. K. L. Mcgraw and K. Harbison-Briggs, 1989, **Knowledge Acquisition: Principles and Guidelined,** Prentice-Hall International Edition.

2. J. Allen, 1995, **Natural Language Understanding**, the Benjamin/Cummings Publishing Company.

3. J. Kriz, **Knowledge-Based systems in industry: Introduction,** Artificial Intelligence Group, Brown Boveri Research Center, Barder-Battwil, Switzerland.

4. S. Sangkae, 2002, **Question-Answering in Thai Language using Syntactic-Semantic Analysis**, King Mongkut's University of Technology Thonburi.

5. M. Witbrock, D. Baxter, J. Curtis, D. Schneider, R. Kahlert, P. Miraglia, P. Wagner, K. Pantom, G. Matthews and A. Vizedom, 2002, **An Interactiver Dialgoue System for Knowledge Acquisition in Cyc**, Cycorp.

6. N. Muansuwan and B. Sirinaovakul, **Intelligent Dialogue-Based Tutoring for the Subject of Computer Programming**, Computer Engineering Department, King Mongkut's University of Technology, Thailand.

7. S. Schulz, M. Romacker, G. Faggioli and U. Halm, 2002, **From Knowledge Import to Knowledge Finishing: Automatic Acquisition and Semi-Automatic Refinement of Medical Knowledge**, Freiburg, Germany.

8. **Knowledge Acquisition** [Online], Available: http://www.epistemics.co.uk/Notes/63-0-0.htm

9. W. Akinyele, D. Allnutt, R. London, D. Radley and Shazia Siddiqi, **Questions on Knowledge Acquisition** [Online], Available: http://www.scism.sbu.ac.uk/inmandw/tutorials/kaqu/g2.htm

10. A. Harris, M. Korsakova, M. Wakefield, M. Baxter and D. Farinha, **Knowledge Based Systems: Acquisition, Design and Interfaces** [Online], Available: http://www.bogo.co.uk/harrisaa/

(manuscript)
Role of natural language processing and tutoring strategies in a dialogue-based tutoring system

Nuttanart Muansuwan and Booncharoen Sirinaovakul
Computer Engineering Department, King Mongkut's University of Technology Thonburi
91 Pracha-u-thid Rd. Bangmod, Thungkru,
Bangkok 10140 Thailand

Abstract— This research presents a practical dialogue-based tutoring system for the subject of computer programming. The tutoring system focuses on helping learners learn by solving programming exercises and it is dialogue-based since it processes the inputs which are natural language questions from learners and generate natural language explanation as response to questions from learners, explanation of solutions and tutoring feedback. The goals of this tutoring system are to mimic human tutors by making the conversation between the learner and the system sound natural and to ensure that the student functions as an active learner and an active conversational partner. In generating the explanation, the tutoring techniques as well as natural language techniques are incorporated to make the generated text cohesive and aggregated. The system is equipped with remediation model to fix the incorrect answers or mistakes made by learners. It provides the sub-dialogues (feedback) to refine vague answers, fill in missing concepts or redirect learners to relevant concepts. All of these researched methods are employed to create the intelligent tutoring system which is practical and allows for collaborative learning.

Keywords: Intelligent Tutoring System, Natural Language Processing, Dialogue-based System, Remediation

1. Introduction

Computer-based learning environments have been available since 30 years ago which can serve as individualized interactive learning support due to the advantage that it overcomes the restrictions in terms of time, space, and distance of learners. With the development of WWW, e-learning has been introduced in various forms. The components of e-learning systems vary from *HyperTextbook*[1] which basically provides contents with or without simple feedback to computer tutor or *Intelligent Tutoring System (ITS)* which incorporates artificial intelligence technology and allows interaction between learner and tutoring system (Han and Kim, 2001; Di Eugenio, 2001). We have developed an ITS for the subject of Computer Programming for Engineers, which is a required course for all engineering students at King Mongkut's University of Technology Thonburi (KMUTT), Thailand. We apply the dialogue-based approach to build intelligent tutoring system which possesses capabilities close to those of human tutors.

2. Related Work

Intelligent tutoring systems designed for programming targeting specific programming languages are quite a few (Brusilovsky, 1995). Most of these systems support on-the-fly

[1] HyperTextbook is initially a combination of a "custom database" with versatile tools for organizing and presenting the content. The database contains Web pages with content and the tools include means for students to customize the content to their personal tastes and needs (Brewer, 2000).

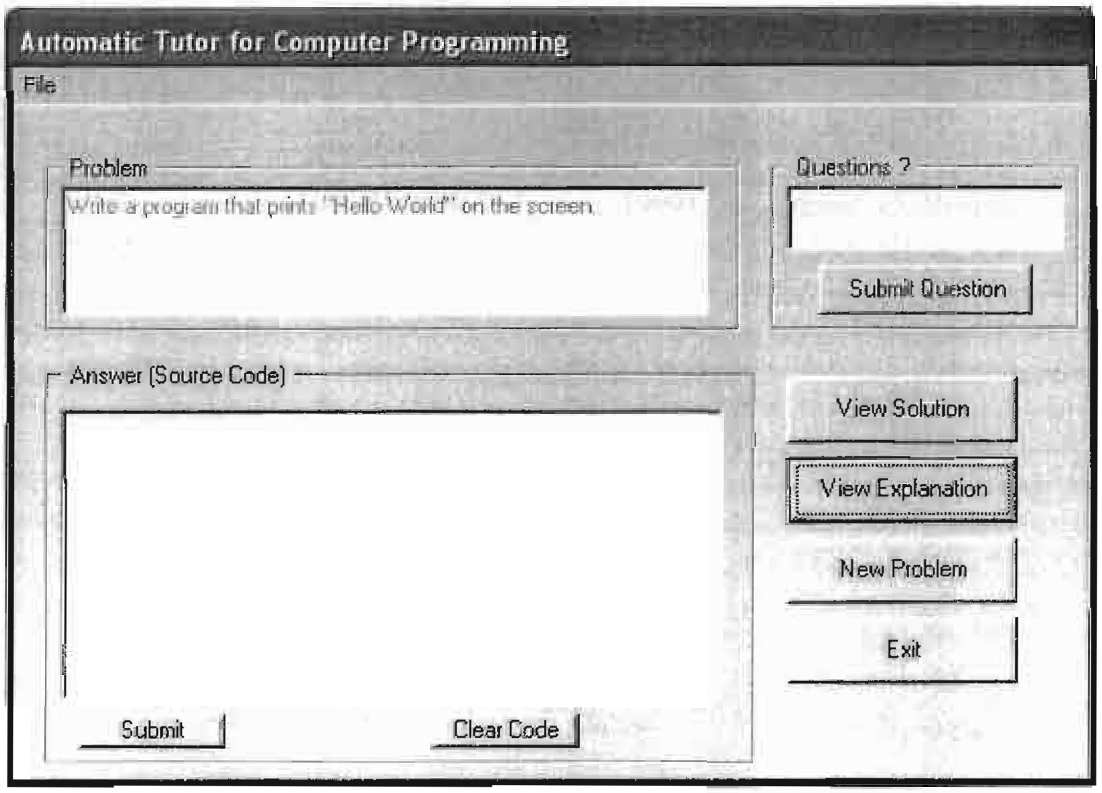
planning of the programs, meaning that there is no clear distinction between the traditional phrases of programming of understanding, planning, implementing and debugging. However, BRIDGE system is an exception in the sense that it helps students construct a solution description in natural language (via menus), followed by successive rewrites in more accurate forms, ultimately resulting in a working program (Bonar and Cunningham, 1988). Another exception is found with DISCOVER (Ramadhan et al., 2001), a model tracing system that supports a restricted form of pseudocode and provides a rich environment for code execution and testing.

Among these systems, Duke Programming Tutor and PROPL are dialogue-based systems. Duke Programming Tutor helps students correct syntax errors in simple Pascal programs via a speech interface. PROPL focuses on pseudocode and program design. A number of dialogue-based educational system which have emerged recently are for other domains that use more advance models of dialogue management. There is also mounting evidence that suggests students participating in such dialogue system do correlate with learning gains (Core et al., 2003)

3. Overview and User Interface

The dialogue-based tutoring system helps learner in solving problems in computer programming exercises. The learner is given a problem statement. If it is a programming problem, then the learner will have to write source code and submit the source code to be compiled. The tutoring part comes to play a role when i) there are errors in students' answers and ii) when students ask questions. For errors in programming exercise, this could also mean that the source code compiles successfully, but the learner did not use the right method as specified in the problem statement.

The following figure illustrates the designed user interface which reflects how learners interact with the tutoring system.



Write a program that prints "Hello World" on the screen.

Figure 1: Designed User Interface

The tutoring part essentially consists of two processing systems: 1) **language generation** for 'View Explanation' and for generating answers to students' questions and 2) **input processing**. The details of language generation and input processing will be discussed in the following sections.

4. Processing Inputs to Dialog-Based ITS

Currently, our ITS is being developed to be used by students in international program of Computer Engineering at KMUTT. Therefore, the system is designed to support English as the language for user interface. A crucial problem in practical processing of natural language inputs from learners is what to do with ungrammatical and/or incomplete inputs (Muansuwan, 2003). According to our initial survey of questions that students are likely to ask tutors, we found that English questions by non-native speakers of English tend to be ungrammatical and in shorten forms. For example, students' questions related to Fibonacci numbers include "Why recursive?" or "Why recursion?". However, since grammaticality is not an issue here, we want to be able to process such inputs.

To solve the problem of ungrammatical or incomplete inputs, we will apply unification grammar in the quick parser, which maps input strings into Quasi-Logical Forms (QLF). QLF is the semantic representation in the form of function/argument and is widely used in language processing tasks, as a transfer level in spoken language translation, for example (Rayner and Carter, 1997). Our quick parser focuses on deriving predicate and arguments (i.e. noun phrases) from input strings and leaves out function words such as auxiliaries, articles, and some prepositions. Furthermore, the question words such as *'why'* are the key words that tell the system to find explanation of the argument that follows *why*.

5. Natural Language Generation in Dialog-Based ITS

Natural language generation is involved in our dialog-based ITS in two forms. The first, which is the explanation of the solution which will show when the learner selects to 'View explanation', is a rather fixed format text, although it will be different from explanation found in textbook in the sense that it is already preprocessed to include tutorial techniques and organized with cognitive goals in mind (Joyce and Weil, 1996; Reigeluth, 1999). The second, challenging form of language generation is when the system generates answers as response to students' questions. The architecture for this response generation is as shown below (this architecture is revised from CIRCISM-Tutor v. 3 (Freedman, 1996)).

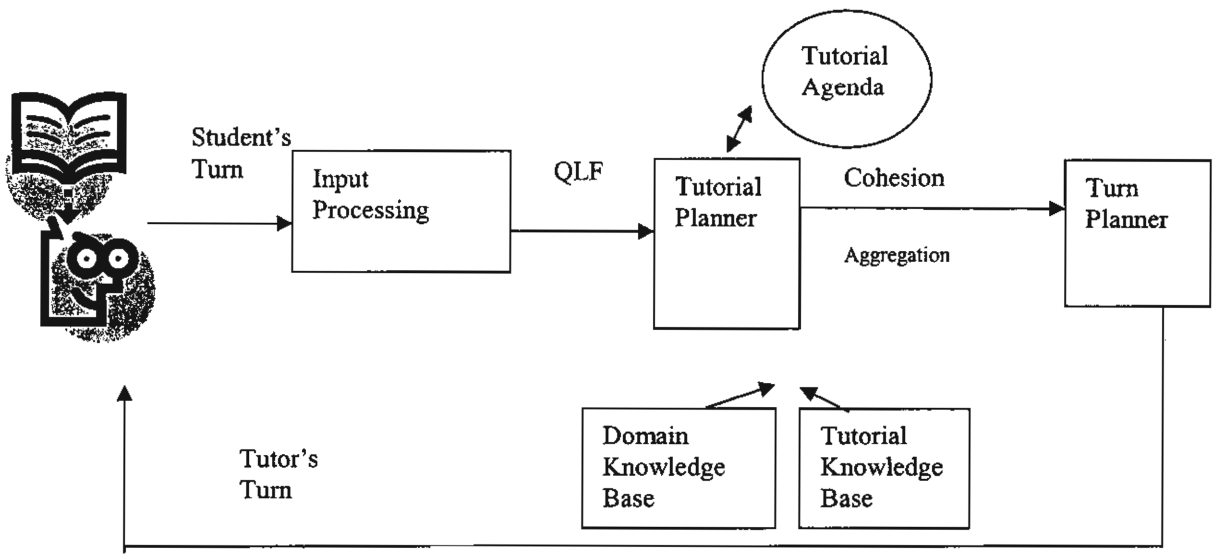


Figure 2: Architecture for Response Generation in ITS

Student's turn and tutor's turn in this case refer to dialogue turns. Based on the problem statement, students can ask tutors about what they do not understand. The tutoring module processes the input from the student and represents it in Quasi-Logical forms as mentioned above. The *domain knowledge base* is the content of the subject of computer programming. By analyzing a corpus of human-to-human tutorial dialogues, we acquired the tutoring techniques and patterns and store them in the *tutorial knowledge base.* In tutorial dialogues of our ITS system, we extract the learning content from the domain knowledge base and "put it in the mouth of" the tutor agent by using various tutoring techniques and patterns. Note that the learning content in the domain knowledge base is acquired from textbooks of the subject of computer programming, but the learning content text is processed, i.e. simplified to have straightforward sentential structures and contain simple words. This processing is necessary for effective tutoring, especially for students whose English is not their first language.

The *tutorial planner* and *turn planner* collaborate in generating tutoring texts back to the student. The tutorial planner receives the input from the domain knowledge base and tutorial knowledge base, chooses pedagogical goals, and maintains an agenda for the tutor agent. The turn planner accumulates what the tutor is going to say, considers the issues of cohesion and aggregation (Halliday and Hasan, 1976) in dialogue turns, and issues the feedback text to the student. For example, our domain knowledge base may contain the following content:

Iterative solution of Fibonacci number

- calculate f_0 and f_1
- calculate f_2 from f_0 and f_1
- calculate f_3 from f_2 and f_1

Figure 3: Content from textbook in the Domain Knowledge Base

However, when a human tutor explains the above content to students, the tutor uses the following form of explanation.

First you calculate f_0 and f_1, then calculate f_2 from these, then f_3 from f_2 and f_1, and so on.

Figure 4: Dialogue turn by human tutor to explain the content from Figure 3

From the above dialogue turn by the human tutor, we can see that there are at least 3 techniques involved for cohesion and aggregation. The first technique is the issue of addressing. By calling the learner *'you'*, the tutor reduces the gap between learner and tutor and makes the tutoring less formal. The second issue is pronominalization. The tutor uses the demonstrative pronoun 'these' to refer to f_0 and f_1. In other words, the tutor employs pronominalization to avoid redundancy, when the reference is clear. Finally, discourse markers such as *then, and, and so on*, are used to aggregate the whole explanation.

6. Natural language use and program design

For new learners of programming, the fundamental aspect of program design is the problem decomposition. The approach which programming learners can use is, according to Soloway et al. (1988), to lay the components of the solution on the table, meaning to decompose the problem and identify the solution components, which include:

- **goals** – the objective as indicate by the problem statement
- **schemas** – the methods by which goals are achieved
- **objects** – the data items that will be needed in a solution

Students who learn how to program must identify these components in order to create a working program. However, for new learners of programming, these components are not realized in a distinct planning phase but rather "on the fly", namely intermixed with the implementation phase. This, together with shallow programming knowledge, result in students' difficulties in programming problem solving. For an ITS which aims at helping learners during the course of program design, the system must help promote learner's cognitive mastery planning with the following steps:

- provide students with program design interface
- provide assistance during the process of program design
- check students' knowledge in the course of program design

7. Tutoring Strategies

The learning process is comprised of three steps: 1) learning of factual knowledge and proposition, 2) acquiring procedural skills and 3) acquiring problem solving skills. Accordingly, three types of special knowledge are used to achieve effective human-tutor training including: knowledge of the subject matter, knowledge of teaching strategy and methods, and knowledge of the student. While knowledge of the student is beyond the scope of this paper, teaching strategy is closely tied with use of natural language processing in tutoring systems. In fact, applying dialogue-based approach is obviously the advantage of the system since it mimics the human tutor's behaviour.

As to measure the learner's performance, *cognitive diagnosis* is the process which allows a tutor to assess the cognitive state of the student whose performance has been observed (Ohlsson, 1987). In particular, if the learners give the wrong answer, the ITS should "remediate" the wrong answer and stimulate the learners to figure out the correct answer by themselves instead of providing the correct answer right away. Remediation is the technique that human tutors naturally use all the time to help learners to understand the exercise and to solve it correctly after failure. Direct remediation is widely studied and applied in intelligent tutoring systems (Woolf and Hall, 1995). There have been studies on two remedial approaches: recall and articulation.

Recall

- Remind or recall an already experienced learned process or schema

Articulation

- Learners reach the solution themselves, by knowledge construction. For example, if identification or recognition of a concept is lacking, the system will ask the students to enumerate the attributes of this concept.

Based on the study of human tutoring techniques, when the learner does not give ideal answers, a sub-dialogue is used with the goal of drawing out a better answer from the student. To improve student answers, we aim at refining vague answers, completing incomplete answers, and redirecting to concepts which are more relevant. Some of these tutoring strategies include:

I. **Context-free pump**: Example: "Can you tell me more about that?"
II. **Refer to problem statement**: Example: "You should go back to look at the problem statement."
III. **Elicit requisite observation**: Example: "Does this program generate a sequence?"
IV. **Hint delivery**: Example: "Think about what allows us to repeat something over and over."
V. **Hypothetical example**: Example: "How much does count go up each time we see a new value?"

To summarize, the main difference between tutoring and teaching is the capability to remediate what the learner misunderstands or the mistake that s/he makes. Therefore, for an intelligent tutoring system to be different from other e-learning software, it should incorporate this tutor's capability. In our system, remediation is expressed in the form of natural language feedback to learners using the sub-dialogues mentioned above

8. Concluding Remarks

This paper presented a practical dialogue-based intelligent tutoring system for the subject of computer programming. By 'practical' we mean that we intend to use straightforward processing techniques and, meanwhile, aim for the system that mimics human tutors, with the model of human tutors' capabilities.

Acknowledgement:
This research project is supported by the research grant from the Thailand Research Fund

References:

Bonar, J. G., and Cunningham, R. (1988). Bridge: Tutoring the programming process. In Psotka, J.; Massey, L.D.; and Mutter, S.A., eds., *Intelligent Tutoring Systems: Lessons Learned*, 1988, pp. 409-434.

Brewer, J. H. (2000). Available at: http://musr.physics.ubc.ca/~htb/HTBdef.html.

Brusilovsky, P. (1995). Intelligent learning environments for programming. In *Proceedings of AI-ED'95, 7th World Conference on Artificial Intelligence in Education*, pp. 1-8.

Core, M.G.; More, J.D.; and Zinn, C. (2003). The role of initiative in tutorial discourse. In *10th Conference of the European Chapter of the Association for Computational Linguistics.*

Di Eugenio, B. (2001). Natural-Language Processing for Compute-Supported Instruction. *Intelligence*, Winter 2001, pp. 23-32.

Freedman, R. (1996). Using Tutoring Patterns to Generate More Cohesive Text in an Intelligent Tutoring System. In *Proceeding of ICLS' 96*, Evanston, IL, pp. 75-82.

Halliday, M. A. K. and Hasan, R. (1976). *Cohesion in English*. London: Longman.

Han, S. G. and Kim, Y. G. (2001). Intelligent Dialogue System for Plane Euclidean Geometry Learning. In *Proceedings of International Conference on Consumer Electronics 2001.* Available at: www.icce2001.org/cd/pdf/p08/KR042.pdf.

Joyce, B. and Weil, M. (1996). Models of Teaching (5th ed.). Boston: Allyn & Beacon.

Muansuwan, N. (2003). Natural Language Processing for Intelligent Web Robot. In *Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business and Applications*, Rio de Janeiro, Brazil.

Ramadhan, H.A.; Deek, F; and Shihab, K. (2001). Incorporating software visualization in the design of intelligent diagnosis systems for user programming. *Artificial Intelligent Review* 16:61-84.

Rayner, M. and Carter, D.M. (1997). Hybrid Language Processing in the Spoken Language Translator. In *Proceedings of ICASSP-97*, Munich.

Reigeluth, C. M. (Ed.). (1999). Instructional Design Theories and Models: A New Paradigm of Instructional Theory (2nd ed.). Mahwah, NJ: Lawrence Erlbaum Publishers.

Soloway, E.; Spohrer, J.C.; and Littman, D. (1988). E unum pluribus: Generating alternative designs. In Mayer, R.E., ed., *Teaching and Learning Computer Programming*, pp. 137-152.

4th Global Congress on Engineering Education
Bangkok, Thailand, 5-9 July 2004

Intelligent dialogue-based tutoring for the subject of computer programming

N. Muansuwan & B. Sirinaovakul

King Mongkut's University of Technology Thonburi
Bangkok, Thailand

ABSTRACT: In this paper, the authors present a designed architecture for a practical dialogue-based tutoring system for the subject of computer programming. The tutoring system focuses on helping learners learn by solving programming exercises and it is dialogue-based, since it processes the inputs that are natural language questions from learners and generates natural language explanations as a response to questions from learners and the explanation of solutions. In generating an explanation, tutoring techniques, as well as natural language techniques, are incorporated to make the generated text cohesive and aggregated. All of these methods have been employed in order to create an intelligent system that is practical and allows for collaborative learning.

INTRODUCTION

E-learning has the advantage that it overcomes certain restrictions in terms of time, space and the distance between learners. With the development of the World Wide Web (WWW), e-learning has been introduced in various forms. The components of e-learning systems vary from a HyperTextbook, which basically provides contents with or without simple feedback, to an Intelligent Tutoring System (ITS), which allows for interaction between a learner and tutoring system [1][2].

It should be noted that a HyperTextbook is initially a combination of a *custom database* with versatile tools for organising and presenting the content. This database contains Web pages with contents and the tools include the means for students to customise the content to their personal tastes and needs [3].

At the Department of Computer Engineering at King Mongkut's University of Technology Thonburi (KMUTT), Bangkok, Thailand, the authors are developing an ITS for the subject of *Computer Programming for Engineers*, which is a required course for all engineering students at the KMUTT. The authors applied the dialogue-based approach as the tutor agent and focused on helping students learn by solving exercise problems.

OVERVIEW AND USER INTERFACE

A dialogue-based tutoring system helps a learner to solve problems in computer programming exercises. The learner is given a problem statement. If it is a programming problem, then the learner will have to write the source code and submit the source code to be compiled. The tutoring part comes into play when the following occurs:

- There are errors in students' answers;
- When students ask questions.

For errors in a programming exercise, this could also mean that the source code compiles successfully, but that the learner did not use the right method, as specified in the problem statement.

Figure 1 illustrates the designed user interface and reflects how learners interact with the tutoring system.

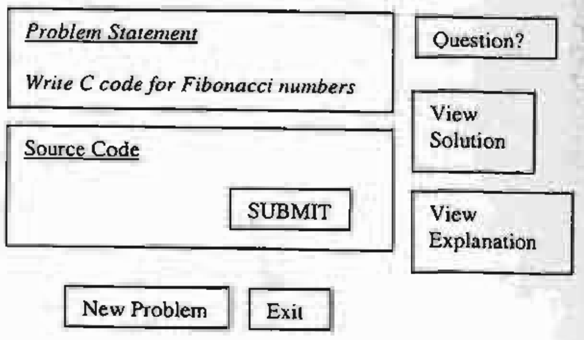


Figure 1: The designed user interface.

The tutoring part essentially consists of two processing systems, namely:

- *Language generation* for 'View Explanation' and for generating answers to students' questions;
- *Input processing*.

The details of language generation and input processing are discussed in more detail in the following sections.

NG INPUTS FOR THE DIALOGUE-BASED ITS

he ITS is being developed to be used by students in ional programme of Computer Engineering at the 'herefore, the system is designed to support English iage for the user interface.

problem in the practical processing of natural iputs from learners is what to do with ungrammatical omplete inputs [4]. According to the initial survey is that students are likely to ask tutors, it was found h questions by non-native speakers of English tend rammatical and in shortened forms. For example, juestions related to Fibonacci numbers include *why* or *why recursion?* However, since grammaticality is ue here, the focus is more on how to process such

) solve the problem of ungrammatical or incomplete : authors have applied unification grammar in the ser, which maps input strings into quasi-logical forms LF is the semantic representation in the form of rgument and is widely used in language processing example, as a transfer level in spoken language n [5].

:k parser focuses on deriving predications and s (ie noun phrases) from input strings and leaves out words such as auxiliaries, articles and some ons. Furthermore, question words, such as *why* are the is that tell the system to find an explanation for the that follows *why*.

AL LANGUAGE GENERATION IN A DIALOGUE-ITS

language generation is involved in the dialogue-based sented in this paper in two forms. The first is the ion of the solution that will appear when the learner he *view explanation* option; this is a rather fixed format hough it will be different from the explanation found in ook in the sense that it is already pre-processed to include tutorial techniques and organised with cognitive goals in mind [6][7]. The second form involves the challenging element of language generation and occurs when the system generates answers in response to students' questions. The architecture for this response generation is shown in Figure 2. It should be noted that this architecture is revised from CIRCISM-Tutor v.3 [8].

In this case, the *student's turn* and *tutor's turn* refer to dialogue turns. Based on the problem statement, students can ask tutors about what they do not understand. The tutoring module processes the input from the student and represents it in quasi-logical forms as mentioned above.

The *domain knowledge base* is the content of the subject of computer programming. By analysing a corpus of human-to-human tutorial dialogues, the tutoring techniques and patterns are acquired and stored in the *tutorial knowledge base.*

In tutorial dialogues of this ITS system, the authors extracted the learning content from the domain knowledge base and *put it in the mouth of* the tutor agent by using various tutoring techniques and patterns. It should be noted that the learning content in the domain knowledge base has been acquired from the textbooks of the subject of computer programming, but that the learning content text is processed, ie simplified to have straightforward sentential structures and to contain simple wording. This processing is necessary for effective tutoring to take place, especially for those students whose English is not their first language.

The *tutorial planner* and *turn planner* collaborate in generating tutoring texts back to the student. The tutorial planner receives the input from the domain knowledge base and tutorial knowledge base, chooses the pedagogical goals, and maintains an agenda for the tutor agent. The turn planner accumulates what the tutor is going to say, and considers the issues of cohesion and aggregation in dialogue turns [9]. Furthermore, it also issues the feedback text to the student. For example, the domain knowledge base may contain the content shown in Figure 3.

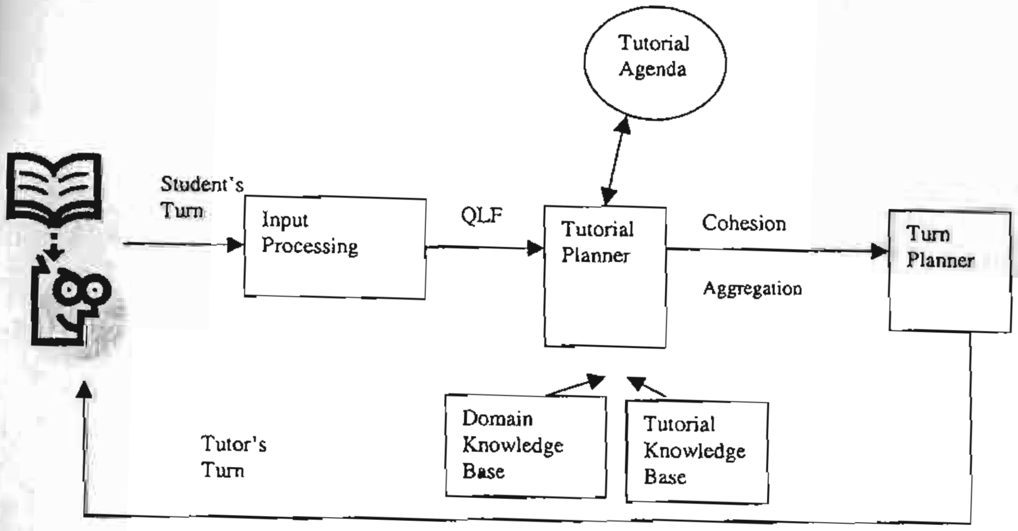


Figure 2: The architecture for the response generation in the ITS.

Iterative solution of a Fibonacci number

- calculate f_0 and f_1
- calculate f_2 from f_0 and f_1
- calculate f_3 from f_2 and f_1

Figure 3: Content from the textbook in the domain knowledge base.

However, when a human tutor explains the above content to students, the tutor uses the form of explanation in Figure 4.

First, you calculate f_0 and f_1, then calculate f_2 from these, then f_3 from f_2 and f_1, and so on.

Figure 4: Dialogue turn by human tutor to explain the content from Figure 3.

From the above dialogue turn by the human tutor, it can be seen that there are at least three techniques involved regarding cohesion and aggregation. The first technique involves the issue of addressing. By calling the learner *you*, the tutor reduces the gap between learner and tutor and makes the tutoring less formal.

The second issue is pronominalisation. The tutor utilises the demonstrative pronoun *these* to refer to f_0 and f_1. In other words, the tutor employs pronominalisation to avoid redundancy when the reference is clear. Finally, discourse markers such as *then*, *and*, and *so on*, are used to aggregate the whole explanation.

CONCLUDING REMARKS AND FUTURE WORK

In this paper, the authors presented the designed architecture of a practical dialogue-based intelligent tutoring system for the subject of computer programming. By *practical*, the authors mean that they intend to utilise straightforward processing techniques and, at the same time, aim for a system that can mimic human tutors to a certain level.

The authors are now at the initial stage of this project's development and, in the future, the authors intend to have this system take on more initiatives compared to the current system, which is rather responsive in nature.

ACKNOWLEDGEMENT

This research project was supported by research grants from King Mongkut's University of Technology Thonburi, Bangkok, Thailand, and from the Thailand Research Fund

REFERENCES

1. Han, S.G. and Kim, Y.G., Intelligent dialogue system for plane Euclidean geometry learning. *Proc. Inter. Conf. on Consumer Electronics* (2001), www.icce2001.org/cd/pdf/p08/KR042.pdf
2. Di Eugenio, B., Natural-language processing for computer-supported instruction. *Intelligence*, Winter, 23-32 (2001).
3. Brewer, J.H., What is a HyperTextBook Anyway? (2000), http://musr.physics.ubc.ca/~htb/HTBdef.html.
4. Muansuwan, N., Natural language processing for intelligent Web robot. *Proc. Inter. Conf. on Computer Science, Software Engng., Info. Technology, e-Business and Applications*, Rio de Janeiro, Brazil (2003).
5. Rayner, M. and Carter, D.M., Hybrid language processing in the spoken language translator. *Proc. ICASSP-97*, Munich, Germany (1997).
6. Joyce, B. and Weil, M., *Models of Teaching* (5th edn). Boston: Allyn & Bacon (1996).
7. Reigeluth, C.M. (Ed.), *Instructional Design Theories and Models: A New Paradigm of Instructional Theory* (2nd edn). Mahwah: Lawrence Erlbaum Publishers (1999).
8. Freedman, R., Using tutoring patterns to generate more cohesive text in an intelligent tutoring system. *Proc. ICLS '96*, Evanston, USA, 75-82 (1996).
9. Halliday, M.A.K. and Hasan, R., *Cohesion in English*. London: Longman (1976).