Intelligent tutoring system for development of critical reasoning skills

Goh, Ghee Ming

2006

Goh, G. M. (2006). Intelligent tutoring system for development of critical reasoning skills.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2467>

<https://doi.org/10.32657/10356/2467>

Nanyang Technological University

Downloaded on 26 Apr 2023 07:48:30 SGT

Nanyang Technological University

**An Intelligent Tutoring System for
Development of Critical Reasoning Skills**

Goh Ghee Ming

School of Computer Engineering

2005

An Intelligent Tutoring System for Development of Critical Reasoning Skills

Goh Ghee Ming

School of Computer Engineering

A thesis submitted to the
Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy
of
Computer Engineering

April 2005

Acknowledgements

I give my upmost gratitude to my supervisor, Dr Quek Hiok Chai, for his patience and guidance given to me throughout my Ph.D. candidature in School of Computer Engineering, Nanyang Technological University. I would like to thank the Center for Computational Intelligence (C2I) and the lab technician, Mr Tan Swee Huat, for providing the facilities and support to conduct my research. I would also like to show my appreciation to my friend and colleagues in the center; namely, Tung Whye Loon, Ang Kai Keng, Quah Kian Hong, and Ting Chan Wai. I also thank Lim Teck Meng for his assistance in L^AT_EX. Finally, I thank my family for their love and support showered onto me.

Contents

Title Page	iii
Acknowledgments	v
Table of Contents	vii
List of Figures	xiii
List of Tables	xv
Abstract	xvii
 1 Introduction	 1
1.1 Background	1
1.2 Motivation and objective	3
1.3 Proposed solution	5
1.4 Report organisation	6
 I Literature Review	 7
 2 Educational Research	 9
2.1 Three Levels of Education Research	10
2.2 Education and Cognitive Psychology	11
2.2.1 Cognitive psychology	11
2.2.2 Educational psychology	13
2.3 Teaching Strategies and Models	18
2.4 Instructional Design	21
2.5 Summary	21
 3 Intelligent Tutoring System	 23
3.1 History of Intelligent Tutoring System	23
3.2 Types of ITS	28
3.3 Architecture of ITS	33
3.4 Summary	37

II	EpiList I	39
4	EpiList I	41
4.1	Background	41
4.2	Architecture of EpiList I	42
4.2.1	Bottom-up knowledge construction	42
4.2.2	Epistemic Games	45
4.2.3	Thinking Skills and Pedagogical Aspect	46
4.2.4	Software architecture	48
4.3	Rules and Algorithms of EpiList I	50
4.3.1	Intra-scheme Rules	51
	Instructional Sequence of Non-Linear Negative Exemplar Rule (An example of Intra-scheme rules)	52
4.3.2	Inter-scheme Rules	53
	Instructional Sequence of Negative Exemplar Introduction Rule (An example of Inter-scheme rules)	54
4.4	Evaluation	55
4.4.1	Evaluation Procedure	55
4.4.2	Evaluation questions	56
	Generalisation Skill	57
	Comparison Skill	57
	Reasoning Skill	58
4.4.3	Evaluation results and analysis	59
4.4.4	Opinions	62
4.5	Summary	63
5	Formal Representation of EpiList I	64
5.1	Background	64
5.2	Formal Definitions of EpiList I's Rules and Algorithms	66
5.2.1	General Definitions	66
	Models	66
	Terms used	69
	Predicates	71
5.2.2	Intra-scheme Rules	73
	Rule 1: Positive Exemplar Rule	75
	Rule 2: Negative Exemplar Rule	76
	Rule 3: Linear Positive Exemplar Rule	77
	Rule 4: Linear Negative Exemplar Rule	79
	Rule 5: Non-linear Positive Exemplar Rule	80
	Rule 6: Non-linear Negative Exemplar Rule	82
	The Algorithm	85
5.2.3	Inter-scheme Rules	87

	Rule 7: Positive Exemplar Introduction Rule	89
	Rule 8: Negative Exemplar Introduction Rule	90
	Rule 9: Regrouping Rule	92
	Rule 10: Elimination Rule	94
	The Algorithm	95
5.3	Rule Analysis	96
5.3.1	Lemmas	97
5.3.2	Achieving the Target Classification Scheme	97
5.3.3	Worst-case Scenario	99
5.3.4	Complexities	100
	Time Complexity	100
	Space Complexity	100
5.4	Summary	101
6	System Representation of EpiList I	102
6.1	Background	102
6.2	Generic System Representation of an ITS	103
6.2.1	Fundamental of Control Systems	103
	Sophistication Levels of Control System	103
	Classification of Control Systems	106
	Components of Control Systems	107
6.2.2	Representing ITS in terms of Control Theories	109
	Analysis of ITS	109
	Mapping ITS to Control paradigm	111
6.3	System View of EpiList I	115
6.3.1	Classification of EpiList I	115
6.3.2	Mapping of EpiList I to Control Paradigm	117
6.3.3	System Representation of EpiList	121
6.4	System Analysis and Limitations of EpiList I	122
6.4.1	Convergence of the System Steady State	122
	First level control loop	122
	Second level control loop	123
	Open cognitive control loop	125
6.4.2	Limitations of EpiList I	125
	Limitations of Open loop Cognitive Instructions	125
	Limitation of Implicit Cognitive Instruction	126
6.5	Summary	126

III	EpiList II and A-EpiList	129
7	EpiList II	131
7.1	Background	131
7.2	Formal Refinement of Cognitive Skills	132
7.2.1	Generalisation Skills	132
	Simple Generalisation Skills	133
	Difficult Generalisation Skills	134
	Moderate Generalisation Skills	134
7.2.2	Comparison Skills	135
	Comparing sub-skill	135
	Contrasting sub-skill	136
	Types of comparison skills	137
7.3	System Design of EpiList II	139
7.3.1	Student Modeler, H_3 , and Student Representation, $\hat{y}_3$	140
7.3.2	Target Cognitive Skills, r_3	144
7.3.3	Error sensing and Error signal, ε_3	146
7.3.4	Cognitive Rule, P_3	148
7.4	Architecture of EpiList II	151
7.4.1	Pedagogical model of EpiList II	151
7.4.2	Tutorials of EpiList II	152
	Tutorial for Generalisation skill	152
	Tutorial for Comparison Skills	155
	Tutorial for one-to-one Comparing Sub-skill	155
	Tutorial for one-to-group Comparing Sub-skill	158
	Tutorial for Group-to-group Comparing Sub-skill	160
	Tutorial for Contrasting Sub-skills	163
	Tutorial for Reasoning Skills	166
7.4.3	Assessments(Scenarios) of EpiList II	169
	Opportunistic Assessment	169
	Non-opportunistic Assessment	171
7.5	Evaluation of EpiList II	174
7.5.1	Structure of controlled evaluation	174
7.5.2	Result and analysis of EpiList II evaluation	175
	Result of the controlled group evaluation	175
	Result of the EpiList II controlled evaluation	177
7.6	Summary	178
8	A-EpiList	180
8.1	Background	180
8.2	Basics of Adaptive Control System	182
8.3	Design of A-EpiList	183

8.3.1	Identification of A-EpiList	185
8.3.2	Decision of A-EpiList	186
8.3.3	Modification of A-EpiList	187
8.4	Evaluation	189
8.4.1	Evaluation Procedure	190
8.4.2	Evaluation results and analysis	190
8.5	Summary	192
9	Conclusion	193
9.1	Contribution and Significance	195
9.2	Limitations and Future Work	197
	References	199

List of Figures

3.1	History of ITS	27
3.2	Components of an ITS	34
4.1	Student Interface of EpiList I	48
4.2	EpiList I's Software Architecture	50
4.3	Students' Result of the Evaluation Tests	60
4.4	Results of Students' opinions	62
5.1	The Domain Model	67
5.2	The Expert Model	68
5.3	The Student Model	69
5.4	An Example of Correct Classification	70
5.5	An Example of Incorrect Classification	71
5.6	Example of Positive Exemplar Rule	75
5.7	Example of Negative Exemplar Rule	76
5.8	Example of Linear Positive Exemplar Rule	77
5.9	Example of Linear Negative Exemplar Rule	79
5.10	Example of Non-linear Positive Exemplar Rule	80
5.11	Example of Non-linear Negative Exemplar Rule	82
5.12	The Algorithm for Intra-scheme Rules	85
5.13	Cycle between Intra- and Inter-scheme Rules	88
5.14	Example of Positive Exemplar Introduction Rule	89
5.15	Example of Negative Exemplar Introduction Rule	90
5.16	Example of Regrouping Rule	92
5.17	Example of Elimination Rule	94
5.18	The Algorithm for Inter-scheme Rules	95
6.1	Block Diagram of Control System	103
6.2	Sophistication Levels of Control System	104
6.3	A closed loop control system	107
6.4	Block Diagram of ITS	110
6.5	System view of ITS	111

6.6	System View of EpiList	121
6.7	Open loop cognitive control system	125
7.1	System view of EpiList II	139
7.2	An example of UseCounter	143
7.3	An example of WantedValue	145
7.4	An Example of ErrorValue	146
7.5	Value range of ErrorValue	147
7.6	The algorithm of SingleScenario	170
7.7	The algorithm of FullScenario	173
7.8	Result of the Controlled Group Evaluation	176
7.9	Result of the EpiList II Controlled Evaluation	177
8.1	Developments of EpiList	181
8.2	Block diagram of an adaptive system	182
8.3	System view of A-EpiList	184

List of Tables

2.1	Taxonomy of Educational Research	10
2.2	Applicable of primitive cognitive skills	18
3.1	Types of ITS	29
4.1	List of Intra- and Inter-scheme Rules	51
5.1	Formal notation of Positive Exemplar Rule	75
5.2	Formal notation of Negative Exemplar Rule	77
5.3	Formal notation of Linear Positive Exemplar Rule	78
5.4	Formal notation of Linear Negative Exemplar Rule	80
5.5	Formal notation of Non-linear Positive Exemplar Rule	81
5.6	Formal notation of Non-linear Negative Exemplar Rule	84
5.7	Formal notation of Positive Exemplar Introduction Rule	89
5.8	Formal notation of Positive Exemplar Introduction Rule	91
5.9	Formal notation of Regrouping Rule	93
5.10	Formal notation of Elimination Rule	94
6.1	Mapping of ITS to Control Paradigm	111
7.1	Combinations of types and difficulty level of comparison skill	138
7.2	Formal definition of the Cognitive Rules	150
7.3	Formal definition of Generalisation Tutorial	154
7.4	Formal definition of One-to-one Comparing Tutorial	157
7.5	Formal definition of One-to-group Comparing Tutorial	160
7.6	Formal definition of Group-to-group Comparing Tutorial	162
7.7	Formal definition of Group-to-group Contrasting Tutorial	165
7.8	Formal definition of Reasoning Tutorial	168
7.9	Relations between tutorials and cognitive skills	172
8.1	Adaptation Rules	186
8.2	Representation of Intra-scheme Rules	188

8.3 Evaluation results of A-EpiList 190

Thesis advisor: **A/P Chai Quek**

Author: **Goh Ghee Ming**

An Intelligent Tutoring System for Development of Critical Reasoning Skills

Abstract

Research direction in education is gradually moving towards teaching as well as development of cognitive skills or cognitive component of the subject matter. Although current developments in Intelligent Tutoring System (ITS) have successfully incorporated the teaching of specific cognitive skills, only a few address the tutoring of generic cognitive skills. This piece of work attempts to examine the issues related to the development of the teaching of generic cognitive skills in an ITS and proposes pedagogical models and solutions to overcome issues arising from the teaching and learning of such generic skills. These models are based on sound pedagogical theories developed by Professor Collins on epistemic games and forms. EpiList I is an ITS shell (prototype) that develops classification, generalisation and comparison skills, as well as the reasoning processes involved in classifying items. It implicitly develops the generalisation and comparison skills while explicitly imparting classification skill as well as content knowledge.

The formal definition methodology proposed by this piece of work to describe EpiList I uses simple sets and set operations. It attempts to overcome the issue of understandability and readability of an ITS shell. EpiList I inherits the domain and platform independent characteristic of the formal definition, hence allowing the ITS shell to be well specified and understood. Another issue of concern is the general ad-hoc nature of the design and development steps of an ITS; which makes the analysis of the system tedious and difficult to comprehend, especially when applied to the design of an ITS that attempts to model pedagogical instruction to support the learning of cognitive skills. This research proposes a design and analytical methodology appropriate for the teaching of a subset of generic cognitive skills using an ITS framework called EpiList.

The methodology of formal definition and system analysis have been applied to EpiList I. This ITS implicitly teaches the cognitive skills of comparison, generalization etc through demonstration and tutorial guidance. Both field evaluation and system analysis of EpiList I highlight the limitations of implicit skill instruction. EpiList I lacks an explicit representation of the cognitive skills acquired by the student through the tutorial interaction and hence is unable to model them for more effective instructional guidance. These limitations are addressed by EpiList II. EpiList II incorporates a meta-cognitive pedagogical teaching model that attempts to explicitly teach cognitive skills. With the explicit development of cognitive skills in EpiList II, the implicit development of cognitive skills in EpiList I would be remedied. Furthermore, the tutorials of EpiList I are adapted to maximize the teaching of the content for individual students. This adaptation is based on a cognitive competence level derived from the meta-cognitive module in EpiList II, hence giving rise to an adaptive version of cognitive skill instruction in EpiList I. The adaptive ITS system is called A-EpiList. Benchmark results on EpiList II and A-EpiList are promising.

Chapter 1

Introduction

“Give a man a fish and you feed him for a day.
Teach him how to fish and you feed him for a lifetime.”

– Lao Tzu

1.1 Background

The development of ITS is closely correlated and coupled with the research and needs of the educational field. Initial developments of ITS in the 1980s focus primarily on transferring content knowledge to the students. This is also the primary objective of schools and education during that period.

Nowadays, the schools and teachers are readily accepting the methods of subject teaching and cognitive strategies proposed by educational psychologists (Good & Levin, 2001). The importance of cognitive skills in human learning and reasoning has grown throughout the years. Several tests have shown that teaching cognitive skills to low-graded students will significantly increase the student’s performance, sometimes even higher than that achieved by high-graded students (Mayer, 2001). This fosters the development of ITS that is able to teach cognitive processes and motivates the search for a knowledge representation that is supportive to the development of a computational algorithm that performs such cognitive processes as well as independence

from the subject.

The teaching or development of cognitive skills in ITS already exists in early ITS systems. The best example is the ITS called BUGGY (Barr & Feigenbaum, 1981) (Barr & Feigenbaum, 1981). BUGGY is an ITS that debugs and develops the student's mathematical skills. However, this is only a small number in comparison to the majority that focus on the transfer of factual or procedural knowledge. The major focus of early ITS is to impart the student with the domain knowledge. Such systems includes SCHOLAR (D. H. Sleeman & Brown, 1982) , SOPHIE (Brown, Burton, & Kler, 1982) , and GUIDON (Clancey, 1982) .

Subsequent increasing emphasis on cognitive skills in schools and education have led to ITS focused on cognitive skills. The complexity of cognitive skills has increased dramatically. An example of this incremental in complexity can be observed in ITSs that develop mathematical skills. The complexity has advanced from simple addition and multiplication as in BUGGY to problem-solving of mathematical problems as in PIXIE (D. Sleeman, 1987). The complexity has increased even further in recent years with conceptual learning in mathematics (Cheng, 1999).

In recent years the teaching and development of cognitive skills has become the major focus of ITS research. Although several ITSs have been produced to teach a wide varieties of cognitive skills, most of these skills are only applicable to a particular subject or domain; for example math, science, programming and electric circuits. Such skills belong to the class of *specific cognitive skills*. Another set of cognitive skills is the *generic cognitive skills*. Generic cognitive skills are skills have a wider scope and not restricted to any domain. Examples of generic cognitive skills include classification, reasoning, and critical thinking.

Developing an ITS that focuses on the teaching of cognitive skills involves several issues and difficulties. It not only inherits the issues and limitations of ITS, it also involves the consideration of the issues of translating a pedagogical model that teaches cognitive skills into computational algorithms for the use by the ITS to facilitate such

teaching or instruction. Hence, careful selection of a pedagogical model is necessary together with a systematic translation to design the algorithm to support the teaching of cognitive skills. Other issues inherited include domain and student representations, curriculum design, and student interface.

Furthermore, other issues regarding ITS include the understandability and analysis of ITS have to be addressed. Complaints and concerns raised by teachers and educators indicating that the pedagogical models in the ITS are difficult to comprehend and evaluate. The major cause of this difficulty is the ad-hoc nature of the design and development of ITS, which makes the analysis of such ITS laborious. As a result great design and development efforts are necessary to further improve and enhance such ITS.

1.2 Motivation and objective

The motivation of this work is to design a computational algorithm that would facilitate the teaching and development of generic cognitive skills. The author strongly believes that teaching a person to fish is significantly superior to giving him the fish. Therefore the final outcome of this work is an ITS that is able to support the development of generic cognitive skills. As mentioned above, there are several types and forms of generic cognitive skills. The generic cognitive skill that this work would concentrate on is the classification skill as well as the skills of primitive cognitive and reasoning processes involved. Classification skill is selected as it is one of most basic skills that a child acquires. Furthermore, it is a fundamental skill to support several other higher level cognitive skills and processes. The primitive cognitive processes necessary for classification are generalisation and comparison.

Until recently, only a few developments have successfully incorporated the teaching and development of generic cognitive skills. Developments like TAP (Wong, Quek, & Looi, 1998b) develops inquiry skills, which is a generic cognitive skills, in the stu-

dent. However, inquiry skills belong to a higher level of generic cognitive skills. This work focuses on a more fundamental set of skills. Knowledge structures (Nussbaum, Rosas, Peirano, & Cardenas, 2001) are also used to develop an ITS that exercises the student's generic cognitive skills through completing a knowledge structure. Although their ITS develops a fundamental set of generic cognitive skills, it does not include a pedagogical module. That means the system does not tutor the student on his mistakes; instead the tutoring is provided by a human teacher who would constantly monitor the student's progress in completing a knowledge structure. The work undertaken in this research would carefully examine the teaching processes for generic cognitive skills and attempts to translate these processes into computational algorithms to realise the pedagogical module of the system.

This work addresses the issues in the development of ITS, in particular the issue of teaching generic cognitive skills. The issues include domain knowledge, student modelling and the interface module; together with the pedagogical model, they are the four basic components of an ITS. As mentioned above, the computational algorithms in the pedagogical model requires design and translation from educational theories. Similarly the remaining components, namely domain knowledge, student model, and interface module, also have design and translation considerations. The interface module should be able to provide the means to interact with the student. The student modeler should be able to correctly and precisely capture the student's competency as well as the execution processes of generic cognitive skills. A domain knowledge should provide a representation of the desirable competency of generic cognitive skills. Putting all together, an ITS would be developed that supports the pedagogical model to accurately ascertain the competency and the execution processes of the student's cognitive skills and to tutor him accordingly.

1.3 Proposed solution

To avoid ad-hoc development of ITS, educational theories and models are carefully examined prior to the design of an ITS. Epistemic games are well designed and defined with a complete explanation of forms, structures, moves and targets. The basis of the *List-making* game is the primary format for the ITS developed by this work. The interfaces, moves and domain designs are based on the epistemic list-making game and the proposed ITS is called *EpiList*. The target form or structure of EpiList is to generate lists of items. This action is the process of classification.

The tutorials of EpiList guide the student to complete the lists of items. The tutorials guide the student to identify and correct their mistakes. The mistakes include:

- the incorrect placements of items into lists, and
- the inappropriate generation of lists.

The pedagogical model of EpiList is based on the *inductive and deductive model* for developing thinking skills (Eggen & Kauchak, 1988). These teaching models use a scaffold to tutor the student in the processes of generalisation and comparison. The inductive model is used to rectify the student's misconceptions of items, while the deductive model is used to introduce a new concept to the student. These two teaching models serve as the pedagogical model for EpiList.

Another issue addressed by this work is the understandability of ITS. A proposed solution for easier understanding a system is to 'model' the system (Leitch, Shen, Coghill, & Chantler, 1999). This work proposes a definition or descriptive methodology to formally define and model EpiList. Two approaches would be investigated in this research. The first approach uses simple sets and set operations to define the computational algorithms of EpiList. This approach is able to provide a meta description of the domain knowledge, student model, and the pedagogical model. The

second approach uses system theories to define the whole EpiList system by describing the functionality of the system.

1.4 Report organisation

This thesis consists of 3 parts. Part I covers the literature review and contains 2 chapters. Chapter 2 reviews the research in the educational fields and highlights the psychology and strategies of developing cognitive skills. Chapter 3 presents the history and background knowledge of ITS. This chapter also categorises EpiList in the taxonomy of ITS.

Part II is dedicated to EpiList I. It consists of 3 chapters. Chapter 4 presents EpiList I in detail, including domain representation, pedagogical model, and the prototype. This chapter also covers the evaluation of EpiList I. Chapter 5 presents the formal definition methodology that uses simple sets and set operations. This chapter formally describes EpiList I with this formal methodology. The last chapter, Chapter 6, shows the details of mapping the system representations to ITS systems. It shows the system representations of EpiList I and includes the analysis of EpiList I using system theories.

Part III presents further enhancement to EpiList I, namely EpiList II and A-EpiList. It consists of 2 chapters. Chapter 7 presents the design and development of EpiList II. EpiList II explicitly develops generic cognitive skills while EpiList I implicitly develops such skills. Chapter 8 presents A-EpiList. A-EpiList adds an adaptive level over EpiList II to adapt the instructions to individual student. In both chapters, the formal definition and system representation are used to design, define, represent, and analyse both EpiList II and A-EpiList.

Finally, Chapter 9 concludes this work by addressing the contributions of the research and proposals for future research directions.

Part I

Literature Review

Chapter 2

Educational Research

“The roots of education are bitter,
but the fruit is sweet.”
– Aristotle

Theories and findings derived from educational research is a crucial and contributing factor to ITS. It governs the design of ITS; namely the pedagogical and interface models. Research in the field of education covers a vast area of scope from theories to instructions and applied to topics or subjects ranging from linguistic to problem-solving. Since the focus of this research is on cognitive and thinking skills, the scope of the review in education research is likewise restricted to cognitive skills.

2.1 Three Levels of Education Research

Level	Definition	What it focuses on?	Sample Work
1	Educational and cognitive psychology	How teaching and learning occurs.	Cognitive versus Behavioural.(DeMar, 1988) Bloom's Taxonomy.(Bloom, 1956) Piaget's Cognitive Development Theory.(L. Smith, 2000)
2	Teaching Strategies and Models	The sequence of teaching	Epistemic Forms and Games.(Collins & Ferguson, 1993) Critical Thinking Teaching Models.(Clarke & Biddle, 1993) Implicit versus Explicit Teaching.(Kirsner et al., 1998)
3	Instructional Designs	The structure of instructions	Direct versus Indirect Instruction.(P. L. Smith & Ragan, 1999)

Table 2.1: Taxonomy of Educational Research

Work on research in education can be distinguished into 3 levels as shown in Table 2.1. The first level is Educational Psychology. The second level is Teaching Strategies and Models while the third level is the Instructional Design.

Level 1 focuses on answering the questions of how human beings learn and teach. The research derived from educational psychologists is the most important aspect as it lays the foundation of all educational work including efforts in ITS. Educational

psychologists work very closely and their work is based on the findings and theories from the field of *cognitive psychology*. Cognitive psychology focuses on the the functionality of the brain and its mental processes. Educational psychologists adopt the findings of cognitive psychology in their explanations and definition of learning and teaching. Further discussion on educational and cognitive psychology can be found in Section 2.2

At level 2 are the teaching strategies and models. Based on the theories of learning and teaching, the educationalist formulates several strategies and models to teach and guide the student. Different models and strategies can be designed or developed for a theory or framework defined by a psychologist (Brady, 1985).

At the lowest level, level 3, is Instructional Design. At this level, the educationalist drafts and formulates instructions. These instructions could be guiding the student to perform certain actions, providing information to the student, and inquiry of the student's response according to the requirement of the teaching strategies or models. The design considerations include language, vocabulary, age and competency of the student (P. L. Smith & Ragan, 1999). Hence, several sets of instructions can be drawn from a teaching model for students of different age and competency. Further details will be presented in Section 2.4

2.2 Education and Cognitive Psychology

2.2.1 Cognitive psychology

Psychologists of cognition focus on how the brain works. The functionalities of the brain includes (Kellogg, 1995):

1. Perception,
2. Consciousness,
3. Learning,

4. Memory,
5. Knowledge Representation,
6. Language,
7. Intelligence, and
8. Thinking.

Each function presented above is well-researched and clearly defined by cognitive psychologists. The findings and results from the investigation into cognition also contribute to several different fields of research. In relation to the field of ITS, findings from research in learning and memory also assist in the design and development of the pedagogical instructions; for example the definitions of short-term, mid-term and long-term memory contributing to the psychology of education (Woolfolk, 1995). The instructions employed in an ITS should be tailored and designed in conjunction with the findings of cognitive research to better support and promote the learning process in the brain.

Similarly, findings from the field of knowledge representation support the domain and student representation in an ITS. Different methodologies or styles that a human employs to organise and represent information can be effectively used to represent information in a system for manipulation by computational algorithms (Way, 1991). Metaphors like conceptual map, semantic network and look-up-table proposed in cognitive science and psychology are currently widely used in representing knowledge in ITS.

Since our primary teaching objective is the transfer or development of cognitive skills in the students, it is necessary to take an in-depth view of how cognitive psychologists address the teaching of cognitive skills. The cognitive psychologist view such cognitive skills as *thinking skills* and these are generally associated with 3 levels of proficiency. The first level is the acquisition and organisation of information and

new knowledge. The second level addresses reasoning. The final and highest level involves problem solving and decision making. Each level requires the support from a lower level and in turn supports a higher level. The acquisition of knowledge is necessary in the development of reasoning while problem solving utilises the knowledge and reasoning capabilities to solve problems and undertake decision-making.

Therefore, one could postulate that the development of thinking skills should begin from the most fundamental level; that is the acquisition and organisation of knowledge. Research into knowledge shows that classification of information is a critical aspect in organising information, which is subsequently refined into knowledge (Nixon, 1971). Information is constantly acquired through our five senses of sight, scent, taste, hearing and touch. The acquisition of information requires organisation as well as classification to become useful. Hence the objective of the research reported in this thesis is to guide and teach the student skills involved in classification as well as primitive cognitive and reasoning skills involved.

The primitive cognitive skills involved in performing classification consists primarily of generalisation and comparison. These skills are necessary for the classification of information, but more importantly, they are used in the reasoning processes to identify classification mistakes or misconceptions.

2.2.2 Educational psychology

“Educational psychology is the application of psychology and psychological methods to the study of development, learning, motivation, instruction, assessment, and related issues that influence the interaction of learning and teaching.” (Elliot, Kratochwill, Cook, & Travers, 2000)

That is to say, educational psychology attempts to answer how teaching and learning occurs. It can be broadly grouped into *teaching* and *learning*. Although teaching and learning are intertwined with one another, their theories and principles can be individually examined and defined.

Early educational psychologists debated between cognitive versus behavioural approaches. The cognition camp of psychology postulates that learning occurs through cognition while the behavioral camp postulates that learning occurs through actions. Jean Piaget (1896-1980) (L. Smith, 2000) is one of the few early psychologist that investigated the process of learning from the aspect of active cognition. At the other end of the spectrum, Ivan Pavlov (1849-1936) (DeMar, 1988), the forefather of psychology, focused on the behavioral aspect of learning. As the focus of this research is on the teaching and development of cognitive skills, only the cognitive aspect of educational psychology will be investigated.

Piaget theorised that the thinking process changes radically, though slowly, from birth to maturity and humans are constantly striving to make sense of the world (Goswami, 1998). He defined 4 stages in the cognitive development process; namely:

- (i) The sensory-motor period: 0-2 years,
- (ii) The pre-operation period: 2-7 years,
- (iii) The concrete operation period: 7-11 years, and
- (iv) The formal operation period: 11-maturity.

The first stage is the development of cognitive skills; primarily through the sensory-motor or the senses. During this period of time information is *accommodated* and *assimilated*. The pre-operation period formulates the relationships or schemes among information entities to derive knowledge. This is the period where a child develops the skill to form relationships and inverse relationships between information entities that are used in schemes. The concrete operation period is the crucial phase where the child develops logical and reasoning skills. Finally, during the formal operation period, the second- and higher-order reasoning skills are developed.

Another renowned early psychologist, Lev Semyonovich Vygotsky (1896-1934) (Kolar & D'Ambrosio, 2002), theorised that "Children's knowledge, ideas, attitudes, and

values are developed through interactions with others”. This development, as identified by him, would be greatly affected by culture, language, private-speeches and scaffolding. His theory is known as the “Zone of Proximal Development.” His theory highlighted the importance of *self-guidance* and *scaffolding*.

Another early psychologist, Jerome S. Bruner (1915-Current) (The PSI Cafe, 2001) proposed a “Cognitive Learning Model” that presents his research in education as follows:

Bruner (1966) viewed learning as an ongoing process of developing a cognitive structure for representing and interacting with new information (Craig & Reed, 2004; Bruner, 1966).

He defined 3 modes of representation that break down the classification process. The 3 modes are: *Enactive*, *Iconic*, and *Symbolic* representations. He also assisted in the development of *Discovery Learning*; a teaching model which will be discussed in the next section.

Subsequent psychologists (Elliot et al., 2000) further defined a better and complete model for cognitive development. Bloom’s Taxonomy (Bloom, 1956) is a more abstract definition of cognitive development from information to knowledge to reasoning and, subsequently, analysis. Bloom’s taxonomy consists of 6 levels; they are namely:

(i) Knowledge,	(iv) Analysis,
(ii) Comprehension,	(v) Synthesis, and
(iii) Application,	(vi) Evaluation.

This taxonomy not only covers the reasoning skills, it also covers the hypothesis and self-evaluation which is essential in critical thinking (The PSI Cafe, 2001; Craig & Reed, 2004). Critical thinking is another research area that psychologists actively

investigated in the 70s and 80s.

Critical thinking is the “reasoning and reflective thinking that are focused on deciding what to believe or do.” (Ennis, 1987)

This definition identified the crucial areas of critical thinking to be *clarity, basis, inference, and interaction*. Specifically looking at the inference, the prime objective is to infer the correct conclusion. The inference is categorised as: *deductive inference, inductive inference and inference due to value judgements* (Ennis, 1987) and one of the primary skill necessary for the inference process is generalisation, in particular induction.

“Critical thinking requires deliberate movement through planned steps towards some outcome” (Clarke & Biddle, 1993).

This definition views critical thinking as an endless loop of *Data analysis, Theory building, Theory testing, and Data generation* processes. The data analysis process examines data and develops ideas that describe general patterns found in the data. The theory building process examines the ideas, observations or concepts to propose cause-effect relationships. The theory testing process clarifies and synthesizes ideas to create new ideas. The data generation process generates further data through testing processes or procedures for solving problems.

Modern psychologists place greater research effort and emphasis on the higher order cognitive skills; for example: reasoning, logic, theories, and principle. Psychologist Kurt VanLehn (from Learning Research and Development Center, University of Pittsburgh) extensively examined the acquisition of cognitive skills and learning of principle (Vanlehn, 1996). He defined the learning of principle into 3 phases: learning a single principle, learning multiple principles and practice effects. It explains the acquisition of principles and skills of a learners under different situations.

In recent years, the interest of psychologists has advanced into the area of metacognition. Examples are the strategies for developing metacognitive behaviors (Blakey &

Spence, 1990) and the transfer of meta-cognitive skills to the students (Adkins, 1997). Another metacognitive theory is called cognitive apprenticeship (Collins, Brown, & Holum, 1991).

Metacognition is thinking about thinking, knowing “what we know” and “what we don’t know.” (Blakey & Spence, 1990)

Hence, the first strategy of Blakey and Spence (1990) metacognitive development is to identify what the student knows and what he doesn’t. Subsequent strategies include the teacher articulating the thinking processes; at the same time the student documents the process, and self-regulates and evaluates his own thinking processes. Adkins (1997) defines metacognition into 4 simpler elements; namely, Metamemory, Metacomprehension, Self-regulation, and Schema Training. The teaching of these 4 elements eases the teacher’s task in transferring metacognition to the student. However, these two strategies incorporate the theory of cognitive apprenticeship. The theory of cognitive apprenticeship (Collins et al., 1991) states that metacognition is best transferred by *demonstrating* the skill to a student similarly to the teaching technique in an apprenticeship. According to this theory, a teacher has to articulate and explain the processes of metacognition and also assesses the student on the acquisition of the skill.

From the studies of educational psychologists presented above, the most fundamental process and operation of cognition is the organisation and categorisation of information into knowledge. Semantic maps and formation of these maps is one of the approaches that transform information into knowledge. This is essentially the classification skill (Nixon, 1971). The essential primitive cognitive skills that are involved in classification consist of both generalisation and comparison skills.

These skills; classification, generalisation, and comparison, are not only applicable to the generation of knowledge but are necessary for other cognitive processes like critical thinking, reasoning, and metacognition.

	Organisation of informa- tion	Critical Thinking	Reasoning	Meta- cognition
Classification Skill	✓		✓	✓
Generalisation Skill	✓	✓		✓
Comparison Skill	✓	✓	✓	✓

Table 2.2: Applicable of primitive cognitive skills

Table 2.2 presents the use of the primitive cognitive skills under different cognitive processes. It can be seen that these skills are critical and essential to several other higher-order cognitive skills such as critical thinking, and reasoning. These skills form the basic reasoning skills: classification, generalisation and comparison skills, that are necessary for other cognitive operations. This research effort attempts to develop as well as to teach a set of such skills through interactions and tutorials.

2.3 Teaching Strategies and Models

“A model of teaching is defined as a blueprint which can be used to guide the preparation for and implementation of teaching.” (Brady, 1985)

These models are either in specific or generic forms. Teaching models like *guided discovery*, *reciprocal teaching*, and *inquiry teaching* are a few of the teaching strategies that have generic characteristics.

Before individual strategies are discussed, teaching strategies can be further grouped into implicit and explicit teaching. The former teaching strategy implicitly teaches a

topic when presenting another topic while the latter directly teaches the topic that is presented (Kirsner et al., 1998). Implicit teaching is defined as one that provides a sub-conscious and unintentional learning effect. Such a teaching objective not only includes the topic that is presented to the student but also includes a hidden agenda within the lessons. The advantage of implicit learning and teaching is that it does not have a limit in capacity and it is robust (O'Brien-Malone & Maybery, 1998). On the other hand, explicit teaching is defined as one that provides conscious and intentional learning opportunities (Wynne, 1998). The teaching objective is the topic that is presented to the student, there is no hidden lesson or subliminal objective. The main advantage of explicit learning or teaching is the greater awareness of the student in the topic he is currently learning.

Guided discovery is a strategy that guides the student to discover and learn new information or knowledge. When a student acquires new knowledge freely by himself, he is able to retain the knowledge better when compared to his 'forced' acquisition of knowledge when coerced to do so. The strategy of discovery is to provide an initial guideline or direction and to subsequently provide sufficient materials and information to the student in order to achieve the desired goal (Brady, 1985).

Inquiry teaching is a strategy that instructs the student by guiding the student to answer a question presented by the teacher (Collins & Stevens, 1983). This strategy defines the use of positive and negative exemplars, and hypothesis made by the teacher to guide the student to answer the inquiry or question. A set of epistemic forms and games are defined to guide this inquiry teaching. These epistemic forms and games are models of a specific teaching strategy although this strategy is generic.

However, some teaching strategies are applicable to specific domains. As the central focus of this work is on the teaching of the primitive cognitive skills, the effort is likewise focused specifically on teaching strategies and models that facilitate the teaching of these skills. A good example is the work on the *structural mapping* for the development of cognitive skills in mathematics by Boulton et al (Boulton-Lewis

& Halford, 1991). *Thinking frames* is another strategy proposed to develop thinking skills.

The teaching strategies or models that this work focuses on are the strategies defined by Eggen and Kauchak (Eggen & Kauchak, 1988). These strategies focus on the teaching of concepts and correction of misconceptions through sequences of generalisations and comparisons. Hence, these strategies are incorporated into the prototype of this work as they are on par with the objective of the research.

The prototype of this research would primarily be developed based on the Epistemic Games proposed by Prof. Collins. Epistemic games are developed on the principle of epistemic theory of cognition. Epistemic games are a set of well defined games that guide the student to complete a set of epistemic forms. The *List making game*, a form of the epistemic games, requires the student to form lists of items to answer and understand an inquiry. This process of forming lists is essentially the organisation of items into categories. It is employed in the development of the prototype of this work.

We incorporated into this list making game the teaching strategies defined by Eggen and Kauchak. Specifically, the *inductive and deductive* models are used. The deductive model guides the student with generalisation and comparison processes to deduce his own misconceptions. The inductive model on the other hand guides the student to induce/deduce a new concept from an existing concept through a sequence of reasoning processes involving the use of the generic cognitive skills such as generalisation and comparison. Together, the deductive and inductive models, provide a scaffold to support the student in the important characteristics of a significant classification scheme that can correctly categorise all items under consideration. On top of that, the prototype also incorporates the theory of discovery learning where detailed information is presented upon request by the student. This enables the student to discover and learn facts and information on the items presented.

2.4 Instructional Design

Instructions are the actual words or sentences that a student would read and subsequently carry out the actions instructed. Instructional design is the process of designing a sequence of instructions based on some theory or model. Instructional design also includes the design of Graphics User Interfaces (GUI) . For a single theory or model several levels or types of instructions could be put together to accomplish a similar task. The importance of the psychology in instructional design has grown greatly in recent years (Saskatchewan Education, 1991).

One study on instructional psychology is *indirect and direct* instructions (Addison & Yakimowski, 2003; Saskatchewan Learning, 2004). Indirect instruction guides a student to perform the task with minimal or no information provided. It only guides the student in a series of tasks or actions and the student has to infer the objective of the instruction. Direct instruction, on the other hand, provides sufficient information. It instructs the student on the actions to take and highlights the objectives.

Another psychological theory that supports the role of instruction is the *cognitive load theory* (Wilson & Cole, 1996). It does not define the structural format of the instruction but the difficulty level of the instructions posed to the student. It is important that the instructions do not overwhelm the student with excessive information. Therefore the design of the system should take this into careful consideration to avoid exceeding the student's cognitive load capacity.

2.5 Summary

This chapter examined in depth the field of education, specifically the cognitive aspect of education. Educational research can be classified into 3 categories:

Level 1: psychology,

Level 2: theories and models, and

Level 3: instructional design.

The psychology of education and the key importance of cognitive skills in education and learning were highlighted. Within the set of cognitive skills a subset of primitive skills, classification, generalisation and comparison, are essential and critical for many cognitive processes. Hence, they form the focus of this research.

This chapter also examined teaching theories and models as well as instructional designs required to support and teach this set of primitive cognitive skills. The insights and details at each level of educational research were provided. The models are significant to this work to develop critical primitive cognitive skills were also identified.

Chapter 3

Intelligent Tutoring System

“Study history, study history.
In history lies all the secrets of statecraft.”
– Winston Churchill

3.1 History of Intelligent Tutoring System

ITS was first introduced by Sleeman and Brown in 1982 (D. H. Sleeman & Brown, 1982). It was a descendent of Computer Aided Instruction (CAI) and Intelligent Computer Aided Instruction (ICAI). CAI and ICAI were research fields highly topical in the 60s and 70s.

“The goal of CAI research is to build instructional programs that incorporate well-prepared material in lessons that are optimized for each student.” (Barr & Feigenbaum, 1981).

It was a simple program that presented information in a structured sequence that is optimal for individual learners. However programs of this type were quite rigid and difficult to adapt to different learners. Hence, ‘intelligence’ was added to the CAI such that it became “sensitive to student’s strengths, weaknesses, and preferred style of learning.” (Barr & Feigenbaum, 1981). Early ICAI programs include SCHOLAR, WHY (Stevens, Collins, & Goldin, 1982), SOPHIE, WEST (Burton & Kler, 1982)

, WUMPUS (Goldstein, 1982) , GUIDON, BUGGY and EXCHECK (R. L. Smith, Graves, Blaine, & Marinov, 1975) . The early ICAI can be generally grouped into 4 forms:

- Problem-solving monitors,
- Coaches,
- Laboratory instructors, and
- Consultants (D. H. Sleeman & Brown, 1982).

They share a common goal to have the learner gain knowledge and expertise that has been implicitly programmed within. D. H. Sleeman and Brown (1982) introduced a change in approach from “instruction” to “tutoring” that subsequently resulted in the popularly used term of ITS. This change also brought about a shift towards tutoring a learner instead of just instructing. To tutor a person, one has to be well versed in knowledge and instructional methods and be able to constantly monitor and adapt to the reaction and performance of the learner in order to maximise the learning outcome.

In the mid-80s, ITS was somewhat forgotten with the rapid enhancement of computer speed and capabilities. This was also the time when multi-media was blooming and much research effort was focused on the use and design of the multi-media to enhance user friendliness and attractiveness of the program. This heralded the beginning of research work in Human-Computer Interface (HCI). Nevertheless, there was still good work done and some outstanding ITS were developed in the mid- and late-80s, such as PIXIE (D. Sleeman, 1987) and ObjectLogo (Drescher, 1987). These developments also introduced new concepts in ITS; namely: *Microworld* and *ITS Shells*. Definitions of microworld and ITS shells will be elaborated in Section 3.2. ObjectLogo is an example of a microworld, it is a modification from the traditional program LOGO (Papert, 1987).

ObjectLogo makes use of the idea of “turtle” to teach object-oriented programming. PIXIE is a diagnostic program that attempts to identify error sources within a domain-independent environment. PIXIE is the pioneer of the concept and the development of an ITS shell. An ITS shell is a system without a specified domain. Any subject or topic can be readily embedded as its domain knowledge as long as the subject conforms to the knowledge representation method employed by the ITS shell.

Research in ITS once again blossomed in the 90s. With the maturity of research in multi-media, focus has gradually been shifted back into algorithms and computational intelligence of systems to fully utilize tools in multi-media. The research directions of the 90s and onwards are as follows:

- Student Modeling Analysis,
- Curriculum Planning,
- Distance Learning/Collaboration Learning, and
- Skills Training/Teaching.

Student analysis attempts to correctly determine the amount of knowledge comprehended by students. Several advancements and improvements on the accuracy of such analysis have been accomplished since the first ITS. The study of student models will be discussed in the section on the architecture of ITS.

‘Curriculum Planning’ is a form of ITS that evolves from ITS shells. A Curriculum Planner allows greater freedom and flexibility in the design of an ITS system. It allows the planning of domain knowledge, interface, tutorial strategies and instructions. Nkambou R., from Department of information of Quebec University, contributed greatly in this research direction with his curriculum modeling approach CREAM (Nkambou, 1997). The “Global Curriculum Planner” (Peachey & McCalla, 1986) is another technique proposed by Peachey and McCalla that systematically combines separate components to derive the curriculum using a strip-like approach (Wong,

Quek, & Looi, 1998a).

With the growth of the World Wide Web, several research efforts attempt to make effective utilisation of the web for learning. This involves research fields in distance learning, collaborative learning and distributed learning. Current developments related to this efforts include DiscoverNet (Belkada, Cristea, & Okamoto, 2001) and INSPIRE (Grigoriadou, Papanikolaou, Kornilakis, & Magoulas, 2001).

Another significant research effort during the 90s was training and the teaching of skills. The difference between ‘training’ and ‘teaching’ is that skill training provides scenarios that were otherwise difficult to create in real life for the student to allow them to practise or hone their skills. While the latter attempts to teach students skills. Skills training requires the student to possess certain competency of a skill in order to practise that skill. Skills training programs are also known as the Computer-Based Training (CBT) programs. CBT programs developed in the 90s include Ego (Ng, Butler, & Kay, 1995), and OFMspertII (Mitchell, Chappell, Gray, Quinn, & Thurman, 2000).

Skills teaching is a highly topical research area in ITS. Due to the complexity and differences between knowledge and skills, skills teaching in ITS has brought life back to several research areas like domain knowledge, student analysis, tutoring strategies and instructional design. Since the focus is now on the teaching of skills, every aspect of an ITS has to be redesigned and redeveloped with respect to this. Turbinia-Vyasa (Vasandani, Govindaraj, & Mitchell, 1989; Vasandani & Govindaraj, 1990, 1991, 1995) is an ITS that teaches diagnostic skill in a complex domain. The developers of this program had to redefine the knowledge representation, student model, HCI and tutoring strategies. It shows the complexities involved in developing an ITS to teach skills. Another example of such an ITS would be the Circuit Exerciser (Yoshikawa, Shintani, & Ohba, 1992). This ITS is capable of analysing the students’ skill competence level in terms of trouble-shooting an electrical circuit.

Figure 3.1: History of ITS

Fig. 3.1 shows a summary of the development of ITS. It shows the development of ITS in terms of content or skill teaching over time. It can be seen that the focus of ITS is moving towards skill instruction. Teaching of content to students is inadequate; they also require skills to make good use of the knowledge. The acquisition of skills enables the students to perform and complete the task independently. The skills taught by ITS includes:

- Mathematical/Science analytical skills [*LED*(Cheng, 1999), *FITS*(Nwana, 1991)]
- Operational Skills [*OFMspertII*(Mitchell et al., 2000)]
- Programming Skills [*EGO*(Ng et al., 1995)]
- Complex Domain Diagnostic skills [*Turbinia-Vyasa*(Vasandani et al., 1989)]

- Circuit Trouble-shooting skills [*SOPHIE*(Brown et al., 1982), *CIE*(Yoshikawa et al., 1992)]
- Cognitive Skills [*EpiComp*(Ang, 1999), *TAP*(Wong et al., 1998b)]

3.2 Types of ITS

The initial characteristics and structure of early ITS is very similar to the ICAI and it comes in five forms (D. H. Sleeman & Brown, 1982) as described as follows:

- (i) Articulate expert systems,
- (ii) Mixed initiative dialogues,
- (iii) Diagnostic tutors,
- (iv) Computer coaches, and
- (v) Microworlds.

The principal research direction of ICAI is to transfer content knowledge to students. Categorization of ICAI is also based on the method employed during tutorial sessions to transfer knowledge to students. With the advancement of ITS and the growing importance of computational intelligence, categories of ITS have subsequently changed to reflect the manner the computational algorithm tutors the student. The categories are as follows (McFarland & Parker, 1990):

- (i) Problem-solving monitors,
- (ii) Coaches,
- (iii) Laboratory instructors, and
- (iv) Consultants.

	Types of ICAI					Types of ITS			
	Articulate Expert Systems	Mixed Initiative Dialogues	Diagnostic Tutors	Computer Coaches	Micro-worlds	Problem-Solving Monitors	Coaches	Laboratory Instructors	Consultants
Declarative Knowledge	SCHOLAR (1970) WHY (1978)	SCHOLAR (1970) WHY (1978)		GUIDON (1979)					
Procedural or Strategic Knowledge			EXCHECK (1977) BUGGY (1978)	WUMPUS (1977) WEST (1979)	LOGO (1987)				
ITS Shell						PIXIE (1987)	EGO (1993)		TAP (1998)
Skill Training (CBT)								OFMspert II (2000)	
Skill Teaching			SOPHIE (1976) [Problem-solving Skill]			EpiList (2001) [Cognitive Skills]	Turbinia-Vyasa (1995) [Diagnostic Skills]	Circuit Exerciser (1992) [Trouble-shooting Skills]	TAP (1998) [Inquiry Skills]

Table 3.1: Types of ITS

Table 3.1 presents the categorization of different ITS systems in terms of the types of knowledge that they attempt to teach. As seen in the table, both ITS and ICAI employ coaching techniques. “A computer coach, similar to a human coach, guides the student during the game or tutorial instruction and at the same time monitoring the student’s performance” (McFarland & Parker, 1990). The main purpose of a computer coach is to provide students with appropriate advice in order to accomplish the task set forth by the systems. The major issue in the use of computer coaches is the timing of interruptions to the students and what advice should be given. Early examples of computer coaches are WEST and WUMPUS. Both systems are implemented in a form of a game, they coach the student on the strategies that would enable them to win the game. Recent coach based systems include EGO and Turbinia-Vyasa. They coach the student on the types of skills that are required to perform programming or diagnostic tasks.

Articulate expert systems and consultants are similar, whereby the system emulates an expert in answering the students’ query related to the expert’s knowledge. SCHOLAR and GUIDON can be considered as articulate expert systems, although they are more appropriately considered to be another variant of ITS. Students using SCHOLAR and GUIDON are able to query the system and in turn the system provides answers to them. TAP is an example of a consultant based technique. It uses the inquiry teaching methodology to communicate and teach students

A diagnostic tutor is able to diagnose the student’s “bug” or conceptual error and tutor accordingly to rectify the error. The emphasis of such ITS is on the identification of the student’s error and their rectification through the tutorial. Usually, the tutor for such a system guides the students to uncover the errors for themselves as in BUGGY. BUGGY is a diagnostic tutor for mathematics. It debugs the conceptual mistakes made during the solving of a mathematical problem and repeatedly presents the mistake until the students are able to comprehend and recognize the mistake. BUGGY ensures that the student has comprehended the mistakes by probing them

to reproduce the incorrect scenarios. In several aspects, a diagnostic tutor is similar to a problem-solving monitor. In a problem-solving monitor, the process of solving a problem is monitored to analyze and determine the “bug” or error in the problem-solving process. However, tutoring in a problem-solving monitor may not be the same as that in a diagnostic tutor. A good example of a problem-solving monitor is the EXCHECK. It monitors the proving sequence and provides hints and guidance when the student makes a mistake. Current development such as PIXIE and EpiList are also problem-solving monitor systems. PIXIE analyzes the solving process of mathematical problems for mistakes and highlights these mistakes to the student. Similarly, EpiList monitors the classification process of the student and highlights the mistakes.

Mixed-initiative dialogue systems teach students by engaging them in a conversation. It is a two-way communication where both the system and student can ask questions and answer questions posed to each other. SCHOLAR, GUIDON, and WHY are ITSs that employ mixed-initiative dialogues. They are able to ‘decode’ students’ questions and provide answers to them accordingly. Similarly, they can comprehend the answer provided by the student and determine if the student has correctly answered the query or if tutorials are required. This category of ICAI is not grouped under ITS mainly due to its generic definition; mixed-initiative dialogues describe the form of interface that is used to communicate with students. It does not describe the nature or method of transferring knowledge to the students. Hence, mixed-initiative dialogues are not generally found in description of ITS.

Microworlds forms the final group of ICAI under our consideration. “A microworld in a sense is a little world, a little slice of reality.” (Papert, 1987) This means that within a microworld there are limitations to the interface. The moves of students are strictly and completely defined. In a microworld, the student can only perform pre-defined actions or moves; this greatly limits the scope of possible errors. An early example of microworld is the LOGO program which employed mixed-initiative

dialogues; hence, it is not included in the discussion of ITS.

A new category of ITS that is not found in ICAI is the Laboratory Instructor. Laboratory Instructors are systems that attempt to transfer experimental knowledge and expertise to students. They relate factual knowledge to experimental knowledge by providing several scenarios to examine the students' understanding of factual information. SOPHIE is a program that belongs to such a category. SOPHIE analyzes an electric circuit that the student inputs to the system. It troubleshoots the circuits for errors. Laboratory Instructors are popular as CBT. Skills training programs like OFMspertII and Circuit Exercisers belong to this category. They provide the training or experimental environment for the students to practise their skills and tutors them when mistakes are made.

In subsequent research efforts, more categories for ITS have been defined. The first group is the ITS shell. An ITS shell does not have a specified domain knowledge; it is domain-independent. The algorithms and artificial intelligence exhibited by such ITS is based on the knowledge representation method used. Therefore such ITSs can be used to tutor many different types of subject or topic if they are able to represent the expert knowledge of the subject. Only the domain is independent; the Human-Computer Interface (HCI), strategies, instructional sequences, student and pedagogical models are static and remain hard-coded in the system. PIXIE and EpiList examples of ITS shells that have been developed. The next category is Computer-Based Training (CBT). It is similar to the Laboratory Instructors with respect to the objective; to teach students experimental knowledge. CBT is geared towards the teaching of procedural skills. A prerequisite of Laboratory Instructors is the student possesses certain knowledge of the factual knowledge whereas CBT has no such requirement. WITS Tutor (Lefkowitz, Farrell, & Yoo, 1993) and OFMspertII are ITS that fall into this category.

As mentioned above, categorisation of ITS is based on tutoring approaches employed by the computational algorithm of the system. However both ITS shell and

CBT do not explicitly describe the tutoring approach they employ. Hence, ITS shell and CBT cannot be used independently to categorise an ITS; they have to be paired with the original categories of ITS; i.e. Problem-solving Monitors, Couches, Laboratory Instructors and Consultants. For example, PIXIE is an ITS shell that can be grouped under the class of Problem-solving Monitors.

From last column of Table 3.1, each category of ITS reflects the tutoring approach can be employed to teach cognitive skills. For different types of cognitive skills, the best approach has to be selected to be implemented in the ITS. Cognitive skills to be developed by this work are fundamental generic skills; namely, the classification, generalisation, and comparison skills. This work uses the approach of Problem-solving Monitoring to develop these skills. Epistemic games guide the student to complete an epistemic form that answers an inquiry. This is essentially a problem-solving process and this approach is most suitable for the tutorials in epistemic games.

3.3 Architecture of ITS

An ITS is composed of the following models (Burns & Parlett, 1991; Wenger, 1987):

- Domain model
- Student model
- Pedagogical model
- Interface

Figure 3.2: Components of an ITS

Fig. 3.2 depicts the architecture and organisation of models in an ITS. The issues and problems of developing an ITS can be divided into components. The domain model contains the factual and procedural expert knowledge and skills that are used by system diagnostics, student modelling, curriculum and tutorial planning. The student model reflects the moves of students. It represents the knowledge that students comprehend. The student model allows the analysis of errors and misconception in the student's understanding. The pedagogical model holds the strategies and instructional sequences used to instruct students. According to the errors and analytical results derived from the student model, different strategies from the pedagogical model are used.

The research on the domain model focuses mainly on knowledge representations and the usage of these representations. Most of the research effort on knowledge representation is mainly by cognitive scientists. There are several knowledge representation methods (Way, 1991), however not all of them can adequately represent the knowledge for the use in the ITS. The semantic net representation method is used by SCHOLAR, GUIDON and WHY. Different ITSs use and teach different types of knowledge and skills; hence the suitability of a particular representation method in an

ITS is critical to the development of the ITS (Vasandani & Govindaraj, 1995; Faria, Vale, Ramos, & Marques, 2000).

A suitable domain representation method has to be designed for this work to facilitate the teaching and developing of cognitive skills. This representation method should sufficiently represent cognitive skills to allow the analysis of cognitive competency. The complexity in designing the domain knowledge is increased in the ITS shell where the domain is independent from the subject matter. Nevertheless, a domain model exists in an ITS shell but it is generically defined in terms of the format and structure. This generic definition allows different topics and contents to be used in the ITS shell as long as the information conforms to the format and structure.

Since the beginning of ITS, intensive research has been conducted on student modelling. Ideally, it is the student model that provides a correct level of students' knowledge or skills. Several efforts have been made to derive a student model that can accurately analyse and determine the state of students' knowledge. The most commonly used method is the overlay method (Goldstein, 1982). The differential method was also introduced (Brown et al., 1982) around the same time. These methods make several assumptions and have limitations when used to analyse student models. For example, the overlay method can only be used to determine mistakes the students make but not how the mistakes were made. This is still an important research area and has drawn much effort. (Maciejewski & Kang, 1994; Nour, Abed, & Hegazi, 1995).

The student model of this work poses issues in the development of an ITS that teaches and develops cognitive skills. The student model should correctly capture the cognitive competency of the student in order for the system to precisely and correctly analyse the students' cognitive processes and tutor accordingly. With the restrictions and limitations of *microworld*, the monitoring and capturing of student's knowledge is much simpler (Drescher, 1987). In this work, the epistemic game is incorporated to form a microworld that enables the generation of a student model that would

correctly reflect the student's competency in cognitive skills.

Research on the pedagogical model can be grouped into 2 general directions. The first is on instructional sequences. This is related to the research on education. The other is on tutoring strategy. Tutoring strategies determine the usage of instructional sequences. These strategies can be based on the correct or incorrect answers made and steps taken by students. Tutoring strategies also decide on the moment or timing to interrupt the students to tutor them. Rule-based strategies have been used for several years. However the number of rules defined are getting larger and more complex. In the recent research efforts, there are attempts to incorporate neural network (Stathacopoulou, Magoulas, & Grigoriadou, 1999) and fuzzy logic (Nkambou, 1997) in decision-making of the strategies.

The pedagogical model is an important component of the whole system. Although this model is tightly coupled with the research and theories on education; it is often designed and incorporated independently without sufficient support from educational theories. This makes the tutorials of the system technical and difficult to evaluate by educational experts (Reeves, 1997). On top of that, the theories and models proposed by educationalists were targeted at the human teacher and student environments. Hence the translation of these theories and models to computational algorithm and ITS's tutorials is difficult and tedious (Kinshuk, Tretiakov, Hong, & Patel, 2001). However, educationalists are beginning to recognize the use of computers in education (Williams, 2000) and are proposing theories and models that are suitable for ITS. This work will examine the theories and models specifically to the teaching and development of cognitive skills and attempt to propose a computational algorithm that would develop the targeted cognitive skills.

The last model is the interface model. This model provides all the interactions necessary to capture the student's knowledge level and to tutor the student. It is closely coupled with research efforts on *instructional design*. There are 2 issues in the The design of the interface model. The first is the design of the User Interface

(UI) that can effectively capture the student's knowledge. The other is the design of the instructions. The vague and imprecise models of instruction (P. L. Smith & Ragan, 1999) elevate the difficulties in designing the interface model. The UI of this work will be designed based on the structures defined by epistemic games while the instructions will be carefully designed and enhanced through evaluations of the system by educationalists.

3.4 Summary

This chapter presented a brief history, categorisations and architecture of ITS. It highlighted the different research directions that come from ITS research efforts. One of the directions is cognitive skill instruction in ITS which is also the research focus of this work. The chapter continued to closely examine the existing ITS that incorporate cognitive skills developing or teaching instructions.

This chapter subsequently presented the categories of ITS. It elaborated on the characteristics and properties of each category and categorises some past ITS developments. This chapter also examined the capabilities in teaching and developing cognitive skills of ITSs in each category as well as the category of the ITS that is to be developed by this work. Lastly, this chapter presented the components of an ITS. It examined the issues and problems of each component involved in the development of an ITS that incorporates cognitive skills instructions.

Part II

EpiList I

Chapter 4

EpiList I

“To learn without thinking is fruitless;
To think without learning is dangerous.”
– *Confucius*

4.1 Background

EpiList I is a prototype developed for teaching generic skills in classification, generalisation and comparison. These skills are not specific to any subject domain and are generic cognitive skills that can be applied to many domains.

Peter (Cheng, 1999) presented a system that supports conceptual learning in the domain of mathematics and science. This system emphasises the method or concept rather than the content of math or science (Cheng, 1999). Although the cognitive skills acquired by the student include problem solving and reasoning skills, the skills are only applicable to mathematics and science. Nevertheless, it demonstrated the capability of ITS to teach cognitive skills. Nussbaum, Rosas, Peirano and Cardenas (Nussbaum et al., 2001) defined a set of knowledge structures for teachers to generate domain knowledge that is used in their ITS. This ITS focuses purely on the teaching of cognitive skills based on knowledge structures. The cognitive skills that this ITS develops are generic in nature but without a pedagogical model for the

development of these skills. The system requires human intervention to tutor the student.

Their work is very similar to EpiList I (the system described here). However there is no attempt to formally represent or define the ITS. EpiList I goes beyond simple description. The rules and tutorials in EpiList I are explicitly represented in terms of simple sets and set operations. The mathematical representation provides a clear and structured understanding and a platform independent of the rules and tutorials. The simple set representation is also used to define the knowledge, expert and student models. This sufficiently describes EpiList I so that it can be readily comprehended. Hence, EpiList I is not only platform independent, it is also *domain independent*.

4.2 Architecture of EpiList I

4.2.1 Bottom-up knowledge construction

One of the methods of knowledge representation is representation schemes or conceptual graphs (Way, 1991). A conceptual graph is a semantic network with nodes and links. “The nodes represent objects, concepts or situation, while the links represent relations between them”. A conceptual graph is an aid to human memory, understanding and recall of several related items or information. The development of conceptual graphs, as described by Goswami (Goswami, 1998), begins with the determination of the “basic-level” categories followed by superordinate categories. Nixon (Nixon, 1971) described the formation of conceptual graphs in greater details. The process requires classification skills that are separated into matching, cross-classification, serialisation, equivalence and hierarchical tasks. From these methods, the formalisation of a conceptual graph or semantic network can be grouped into top-down and bottom-up approaches. Hence, a learning environment where a top-down or bottom-up approach is used is known as a top-down or bottom-up learning environment respectively.

In a top-down learning environment the main topic is first defined. Then the topic is split into more detailed sub-topics and subsequently the sub-topic is further split into sub-topics; this goes on until individual items are reached. The multiple concept compare and contrast game (Ang, 1999) is an example of a top-down ITS. The categories are predefined and made known to the students. The students are to group a list of given items into their respective categories.

In a bottom-up learning environment the set of individual items is first given. Based on these items, categories have to be inferred to group these items together. These categories are subsequently grouped together to form a classification scheme. The bottom-up approach can result in several semantic networks from a similar set of items. Every semantic network can be used as a classification scheme for a given set of items. Hence, rules and criteria are needed to determine the significance and suitability of these semantic networks. The following are the criteria for a significant and suitable semantic network:

- **Completeness** - A complete semantic network must be able to correctly classify all items under consideration. Otherwise the semantic network is deemed as incomplete. If a student's knowledge consists of an incomplete semantic network, it is crucial to correct that semantic network as the incomplete semantic network might misguide and confuse the student.
- **Uniqueness** - Every item under consideration should only be classified into one category. When an item can be classified into two or more categories in a semantic network, the network is non unique. Although several unusual items exist, for example platypus and dolphins, it would compromise the uniqueness of a semantic network. The focus of this system is to teach the student the basic structure of the network. Therefore, in order to avoid confusion to the student, whenever such unusual items are encountered, the semantic network containing this unusual category is considered non-unique and hence not a significant and

suitable network. In the situation where a unique semantic network does not exist, the current set of items would be discarded and a new set of items is compiled till a unique semantic network is attainable.

- Teaching goals - Teaching goals are defined as the knowledge or information that is to be acquired by students. Even with the criteria set to determine the significance of semantic networks, there could be more than one semantic network that are significant. These significant semantic networks are to be taught to the student, however only one is taught at a time so as not to overload the student cognitive capabilities. Therefore, the semantic network that is selected to be taught to the student is the suitable network that is set in the teaching goal.

Examples of complete semantic networks include grouping human beings according to 'male' and 'female' and grouping animals into 'warm-blooded' and 'cold-blooded'. These two examples are also unique in their nature. However, it is possible to have semantic network that is complete but non-unique. For example, to group animals according to their habitat of land, water and air is complete but non-unique. This is because some animals can live on both land and water. However the knowledge of animals' habitat is crucial to some semantic networks. When a semantic network is set along the scope of 'Reptile', 'Mammal' and 'Amphibian', the knowledge of the fact that an amphibian lives in both water and land is critical. A semantic network set along this scope is complete and unique. Only when a semantic network is both complete and unique, it is deemed significant. A significant semantic network however might not be in line with current teaching goals. This makes the semantic network unsuitable. Hence, a suitable semantic network has to be both significant and match the teaching goals.

4.2.2 Epistemic Games

Epistemic forms or epistemic games were first introduced by Collins & Ferguson (Collins & Ferguson, 1993). Epistemic Forms are target structures that guide inquiry. Both productive inquiry and constructionism entail the structuring of prior knowledge and the production of new knowledge. Epistemic forms act as formalised structures for representing prior or new knowledge, and enable relationships between concepts to be visualised and graphically understood. Epistemic Games, on the other hand, are general-purpose strategies for analyzing phenomena in order to fill in a particular epistemic form. These strategies are not just inquiry strategies and methods; they involve a complex set of rules, strategies and moves associated with a particular representation.

Epistemic Games can broadly be divided into three categories. They are: structural, functional and process analyses. Structural Analysis Games involve the breaking down of an object into subsets or constituents and describing the relationships among the constituents. They consist of the List-Making Game, Primitive Element Game, Stage Model Game and the Compare-and-Contrast Game. Functional Analysis Games are games for analyzing functional and causal aspects of a system. They include Multi-Causal Analysis, Form and Functional Analysis. Process Analysis Games are games for describing the dynamic behaviour of the system. Trends Analysis falls into this category.

The list-making game is an example of the bottom-up learning environment. It requires the student to create lists from a set of items. The student has to determine the categories of the lists and subsequently to arrange items into the lists. However, there are no remedial methods for incorrect categorisation of items. The compare and contrast game, on the other hand, uses comparison and contrast to indicate the inclusion or exclusion of an item in a category. Hence, it provides the remedy to incorrect categorisation in the list making game. *EpiList I* was developed as a list-making game together with the integration of a compare and contrast game. Therefore

the target form of EpiList I is a collection of lists of items instead of a collection of similarities and differences as in the compare and contrast game. The lists are created according to the significant and suitable classification scheme mentioned above.

This remedial process using the compare and contrast game can also be applied to several other games such as the Stage Model and Multi-Causal Analysis games. In the Stage Model game, the target form is a stage transition table associated with the rationales for the transitions. The reasons for transitions are applied to respective domains to derive their stage transition tables in a Stage Model game. Similarities and differences exist between the stages and domains; hence, the compare and contrast game can also be used to highlight these similarities and differences to guide the student to derive and comprehend the target form in the Stage Model game. Similarly in the Multi-Causal Analysis games and several other Epistemic Games, similarities and differences can likewise be observed and used with the compare and contrast game to highlight mistakes or derive the target forms.

4.2.3 Thinking Skills and Pedagogical Aspect

Thinking is referred to as “the process by which the human mind manages information to understand established ideas, to create new ideas, or to solve problems” (Clarke & Biddle, 1993). Eggen and Kauchak (Eggen & Kauchak, 1988) are more specific and state that thinking skills consist of *generalisations, explanations, predictions, hypothesis and comparison skills*. We add classification as a form of thinking skill as it helps to better understand a category and create new categories. The cognitive skills that are developed in EpiList I include the following: classification, generalisation, and comparison. These skills are component parts of thinking skills. EpiList I is designed as an ITS that attempts to develop generic cognitive skills for knowledge construction.

Educational experts and psychologist around the world define several teaching strategies or models. Eggen and Kauchak derived a set of teaching strategies that

utilise generalisation and comparison skills in their models (Eggen & Kauchak, 1988). EpiList I incorporated both Inductive and Deductive Models of their teaching strategies to tutor the student.

The *inductive* model begins with a display of examples and non-examples to the student. The student subsequently performs comparison to induce the similarities and differences between the examples and non-examples. EpiList I incorporated this model to highlight the student's incorrect grouping. In EpiList I, instead of examples, the student induces similarities and differences of concepts or common properties they have generalised from groups of items. Thereby ensuring that the student comprehends the similarities and differences between groups of items according to their common properties. Furthermore, EpiList I generalises items that were incorrectly categorised to derive a common property among these items and subsequently guides the student to identify the similarities or differences against a common property of a correctly categorised items or a group of such items. When two common properties are similar, it allows the student to infer that the incorrectly categorised items should be grouped together with the correctly categorised items. On the other hand, if the two common properties are different, it allow the student to infer that the incorrectly categorised items should not be grouped together with the correctly categorised items.

The *deductive* model begins with the presentation of a known or previously taught common property which is accompanied by examples. Subsequently, a new but related common property is introduced and the student is required to provide new examples having the new common property. EpiList I incorporated this model in the introduction of categories from the significant and suitable classification scheme. However, there is a minor change to this model for use within EpiList I. In the second phase, examples of the new common property are provided and the student has to deduce this new common property. Through this deductive model, the student comprehends the characteristic of the newly introduced common property and its relationship with the known common property. As the knowledge of the pool of common properties in-

creases, the student is able to understand the classification scheme that is significant and suitable. This model is used in the “Negative Introduction Rule” which will be discussed later.

4.2.4 Software architecture

A prototype was built for EpiList I. The interface module provides the user interface of the list making game to the student. Through this interface, the student selects the categories for their classification scheme and arranges a list of items into the categories. Fig. 4.1 shows the student interface of EpiList I. The left side of the

Figure 4.1: Student Interface of EpiList I

interface displays the items that are used in this instant of the game. On the right

are the categories that the student has selected to form his classification scheme. The student can edit his classification scheme using the “Remove” and the “Add New Group” buttons. The “Remove” button removes that particular category from the classification scheme while the “Add New Group” button enables the student to add a category to the scheme. The inter-scheme rules of EpiList I guide the student to add or remove categories to accomplish the migration from his classification scheme to the target classification scheme. The student performs classification by dragging and dropping the items into the classification scheme that he has created. Clicking on the “Check” button initiates the checking sequence and the instructional process. This interface will create a student model that reflects the student’s classification scheme and categorisation.

The checking module checks the student model against the domain and expert model. The domain model provides the correct categorisation of all items. This essentially employs an overlay approach. Checking of the student model against the domain model ensures that all categorisation are correct, thereby obtaining a complete classification scheme. The expert model reflects the classification scheme that is significant and suitable; this scheme is the target scheme of the game. The checking of the student model against the expert model will ensure that the student recognises the significant and suitable classification scheme. The checking model determines the type of mistakes the student has made and fires the rules of EpiList I accordingly to remedy the wrong classifications.

Finally, the tutorial module provides the interface for the tutorials on the errors found in the checking module. When different rules are fired, different instructional sequences are signaled to the tutorial module. The tutorial module accordingly interacts with the students to tutor them on their mistakes. Fig. 4.2 illustrates the overall architecture of EpiList I. The image and text database stores information that is used by the interface and tutorial modules. The taxonomy database stores the domain and expert models. The rule database stores the rules and algorithms that are used in

Figure 4.2: EpiList I's Software Architecture

the checking module to determine which tutorial is to be used.

4.3 Rules and Algorithms of EpiList I

The rules of EpiList I determine which tutoring strategy and instructional sequence to use in different situations. These rules integrate the teaching methods of *inductive and deductive models*. The rules and algorithms are organised into two categories: *Intra- and Inter-schemes* (as shown in Table 4.1). The former focuses on the incorrect grouping of items while the latter focuses on the migration of a classification scheme to another that is more significant and suitable for the current teaching context.

Intra-scheme Rules	Inter-scheme Rules
Positive Exemplar Rule	Positive Introduction Rule
Negative Exemplar Rule	Negative Introduction Rule
Linear Positive Exemplar Rule	Regrouping Rule
Linear Negative Exemplar Rule	Regrouping Rule
Non-linear Positive Exemplar Rule	
Non-linear Negative Exemplar Rule	

Table 4.1: List of Intra- and Inter-scheme Rules

4.3.1 Intra-scheme Rules

The intra-scheme rules are used to guide student to identify their incorrect categorisations within a given classification scheme. It consists of six rules to handle different situations where mistakes are made. They are 1) Positive Exemplar, 2) Negative Exemplar, 3) Linear Positive Exemplar, 4) Linear Negative Exemplar, 5) Non-linear Positive Exemplar and 6) Non-linear Negative Exemplar Rules.

When a mistake is encountered in a situation where there is no correct categorisation, examples would be used to highlight the mistakes. *Positive or negative examples* can be used. A positive example is compared against the mistake to show similarity; hence, suggesting that the incorrectly categorised item belongs to the same category as the example. Instructions of this form are labeled as '*Positive Exemplar*'. Similarly, '*Negative Exemplar*' is the contrasting of *negative examples* with the mistake to suggest that the incorrectly categorised item does not belong to the category it is grouped under.

Under the situations where some correct categorisations can be found in the student's classifications, these correct answers can be used to great advantage to tutor the student. Since the items are correctly categorised, it shows that student has un-

derstood the category unify these items. Hence, it is more efficient to tutor with these correctly categorised items. This form of tutoring is labeled as '*Linear*'. If the correct categorisations used in the tutorial belong to the same category as the mistake, then the tutorial to be used is the Linear Positive Exemplar Rule. The Linear Positive Exemplar Rule uses the correct categorisations as the positive examples to compare against the mistakes to show similarities; hence suggesting that the mistake should be categorised together with the examples. If the correct answers made by the student are categorised in a group where the mistake is found, the Linear Negative Exemplar Rule is fired to use the correct categorisations as the negative examples.

When there is more than one mistake and some correctly classified elements are present, it would be more efficient to gather these mistakes and tutor the student simultaneously. This is labeled as '*Non-linear*'. Hence, under a non-linear condition, generalisation occurs twice, once for the correct answers and another for the mistakes. Similar to linear tutoring, non-linear tutoring can be tagged as positive and negative exemplars according to the category where the correct categorisations are found.

Instructional Sequence of Non-Linear Negative Exemplar Rule (An example of Intra-scheme rules)

This rule is activated when there is more than one wrongly categorised item and some correctly categorised items in the category where the mistakes are found. To tutor the student on his mistakes, the items that have been correctly categorised are first generalised to identify a common property and subsequently the mistakes are generalised together to obtain another common property. Lastly, these two properties are contrasted against each other to demonstrate the difference between the two properties. The student can then infer that the two groups of items are of different properties and should not be grouped together under the same category.

Example 1 is a dialogue session between EpiList I and the student. For this example, consider that a student had correctly grouped 'Whale' and 'Dolphin' under

the category ‘Live in water’. However, he had also incorrectly grouped ‘Tiger’ and ‘Bear’ in the same category. This situation triggers the Non-linear Negative Exemplar Rule. Example 1 shows the interaction of the rule and student that highlights this mistake.

Example 1

System: (Displays Whale, Dolphin and a list of categories.)
 You have correctly grouped these animals. Which of the following categories do you use to group them?

Student: (Selects ‘Live in water’.)

System: (Display Tiger and Bear)
 You have incorrectly grouped these animals. Using the category ‘Live in water’ that you have chosen just now, do these animals also belong to the category ‘Live in water’?

Student: (Selects ‘Yes’)

System: (Display the detail of Tiger and Bear)
 Wrong. Read the details of these animals and select again.

Student: (Selects ‘No’)

System: (Displays Whale, Dolphin, Tiger, and Bear)
 Therefore, can these animals be grouped together under the category ‘Live in water’?

Student: (Selects ‘No’)

System: Good. Now correct your mistakes.

In this example, one of the steps is to display the details of the animals. As EpiList I is designed as a generic framework that develops generic cognitive skills, the display of details has to be customised for students of varying capabilities. For example, a young student below the age of 10 may receive the details in point form, while older student may receive them in paragraph form.

4.3.2 Inter-scheme Rules

Inter-scheme rules focus on guiding students to migrate from their classification scheme to a significant and suitable classification scheme. The migration is accom-

plished by drawing the student to recognise the importance of the target classification scheme. It involves the student in adding categories of a target scheme while removing the old categories from his existing scheme. There are four rules in the set of inter-scheme rules; namely 1) Positive Exemplar Introduction, 2) Negative Exemplar Introduction, 3) Regrouping and 4) Elimination Rules.

The Positive Exemplar Introduction Rule directly introduces a new category of the target scheme to the student and instructs him to add that category. The Negative Exemplar Rule indirectly introduces a new category of the target scheme based on the *deductive teaching model*. It introduces a new category of the target scheme by comparing it with another category that the student has selected. The Regrouping Rule focuses on guiding the student to move items from groups in their classification scheme to the categories of the target scheme. This rule uses items that are correctly grouped in a category of the target scheme as positive examples to compare with the items that are correctly grouped in categories not belonging to the target scheme; hence, suggesting that the items should be categorised under the target scheme. Lastly, the Elimination Rule instructs the student to remove insignificant categories of their classification scheme.

Instructional Sequence of Negative Exemplar Introduction Rule (An example of Inter-scheme rules)

This rule is activated to introduce an unselected category of the target classification scheme. The condition of this rule is that the student had selected only some but not all categories from the target scheme. Firstly, the items from a selected category of the target scheme are generalised together. Then items from an unselected category are also generalised together. Lastly, the two common properties are compared with each other to demonstrate the higher-order relationship between them. This indicates that the two categories belong to the same superordinate class and should be selected.

Example 2 is another dialogue session between EpiList I and the student. For this example, consider that a student creates a multiple-lists form with the categories ‘Live on land’, ‘Live in water’ and ‘Cold-blooded’. Assume that the expert model for this example is a classification schema with ‘Cold-blooded’ and ‘Warm-blooded’ groupings. The ‘Negative Exemplar Introduction Rule’ will be triggered to introduce the category ‘Warm-blooded’ to the student. Example 2 shows the interaction of EpiList I with the student to introduce the category.

Example 2

- System: (Displays all the cold-blooded animals and a list of categories.)
Which of the following categories do you use to group these animals?
- Student: (Selects ‘Cold-blooded’.)
- System: (Display all the warm-blooded animals and a list of categories)
Now we shall examine this set of animal. Which of the following categories do these animals belong to?
- Student: (Selects ‘Warm-blooded’)
- System: (Display cold-blooded animals in one group and warm-blooded animals in another group)
Do the category “Cold-blooded” (on the left) and the category “Warm-blooded” belong to the same higher-order category “Body Temperature”?
- Student: (Selects ‘Yes’)
- System: Good. “Warm-blooded” is an important category. Now add this category to your classification scheme.

4.4 Evaluation

4.4.1 Evaluation Procedure

The animal kingdom is used as the domain knowledge for the prototype of EpiList I. Students from the Methodist Girls School (MGS), Singapore participated in the eval-

uation of the system. The students were consolidated from four primary 4 classes and four primary 5 classes. The primary 4 classes consist of 150 ten years old students and the primary 5 classes consist of 150 eleven years old students. Although MGS is one of the top ten schools in Singapore according the *Primary School Leaving Examination* (PSLE) results of the year 2003, the students evaluating EpiList I are well distributed across the four classes at each level consisting of one high-graded class, two average-graded classes and one low-graded class.

The evaluation was performed in 3 stages. The first stage was a pre-evaluation test. This was a written test that consisted of 20 questions that specifically assessed the students' classification, generalisation and comparison skills. The objective of this stage is to determine the efficacy of the students' cognitive skills before using EpiList I. The second stage involved a hands-on session for the student to use EpiList I. The students were given a quick demonstration of the interface for the system before they were given 30 minutes to use EpiList I. The students were instructed to go through EpiList I at least twice. The time given to the student was enough to go through the system at least twice. This was also used to test the efficiency of EpiList I in cognitive skills development within the given time. The third stage was another written test. The post-evaluation test consisted of 28 questions that also specifically assessed the students' cognitive skills. Together with the pre-evaluation test, this result was used to measure changes in the students' skills. The final written test also prompted for the students' opinions regarding EpiList I.

4.4.2 Evaluation questions

The pre- and post-evaluation tests consisted of questions that specifically test the student's competency level in the set of cognitive skills; namely: classification, generalisation and comparison.

Generalisation Skill

The generalisation skill has three levels of difficulty; namely: *simple, moderate and difficult*. Simple generalization is the generalization of animals belonging to the same family. Since the animals are from the same family, the common properties among them are easily identifiable. The questions in the test set use animals like ‘Tiger’, ‘Lion’ and ‘Cat’ to assess the student’s abilities to perform simple generalisation. The generalised or common properties of these three animals include properties such as ‘warm-blooded’, ‘skin covered with hair’, and ‘breathe through lungs’.

Moderate Generalization requires the student to generalize animals from 2 different families. Although the animals can be from different families, there are still some slight similarities among them. For example, as used in the evaluation test, consider the animals ‘Cobra’, ‘Lizard’, ‘Batfish’, and ‘Catfish’, these animals are from two different families, namely reptile and fish. A few common properties that can be generalised from this group of animals include ‘cold-blooded’ and ‘skin covered with scales’.

Difficult generalization consists of generalization of greatly varied animals. The common property of such a generalisation is not easily identifiable. The difficult generalization set in the evaluation test consists of ‘Cat’, ‘Snake’, ‘Turtle’, and ‘Dolphin’. The only common property of these animals is ‘breathe through lungs’.

Comparison Skill

During the comparison exercise, students identify the similarities and differences between two groups of animals. In the evaluation test, the student is required to compare ‘Catfish’ and ‘Whale’ with ‘Crocodile’ and ‘Angelfish’ to identify that they are the same and against ‘Frog’ and ‘Lizard’ to identify the differences. A more complicated form of assessing the comparison skill during the test requires the student to form two pairs of groups of animals from the four groups. From the four groups of

animals, the student has to first generalise to obtain their common properties. Subsequently compare these common properties to identify the similarities and differences and reorganise the four groups of animals into two pairs of groupings with the same common properties.

Discrimination is a simplified version of comparison. The difference between discrimination and comparison is that discrimination involves only one animal against one or a group of animals. Discrimination usually appears in the form of identifying the odd-one-out. The odd-one-out question that appears in the evaluation test uses 'Tiger', 'Deer', 'Bat', and 'Kangaroo'. The odd-one-out animal for these animals is the 'Bat' as it is the only mammal that is able to fly. Another form of discrimination is the determination of an animal belonging to a group of animals. Using the evaluation test question as an example, the student has to determine if 'Angelfish' belongs to 'Frog', 'Dolphin', 'Shark', and 'Slowworm'.

Reasoning Skill

Reasoning skills are critical skills for classification. The reasoning involved in EpiList I comes in two forms, the first form is simple reasoning skill and the other form is the reasoning processes used in the tutorials of EpiList I. Simple reasoning skill refers to the ability to make use of the families of animal kingdom to determine the properties of an animal. This reasoning is the simplest form of Syllogistic Reasoning (Kellogg, 1995). An example of simple reasoning, as used in the evaluation test, is to deduce that 'Angelfish' is cold-blooded since all fish are cold-blooded.

The reasoning methods used in the tutorials (or rules) of EpiList I refer to the generalisation and comparison sequences that highlight the mistakes in classification. Although the teaching objective of EpiList I is the development of the generic cognitive skills, the reasoning processes in the tutorial are also important for the student to acquire and apply to everyday life. As mentioned in Section 4.3, EpiList I rules are grouped into 'positive' and 'negative' approaches. Hence, the positive reasoning

processes are the reasoning methods of positive approach rules that employ positive examples, while the negative reasoning processes belong to the negative approach rules that employ the negative examples (or counter-examples). Since these reasoning processes are acquired through the interactions with EpiList I, only the post-evaluation test assesses for these skills. The test questions require the student to explicitly describe the reasoning processes through a series of fill-in-the-blank questions.

4.4.3 Evaluation results and analysis

The following sections present a summary of the full technical report (Goh, 2003) of EpiList I's evaluation. Fig. 4.3 is the bar chart of the students' result for the pre- and post-evaluation tests. The chart is organised in terms of different cognitive skills along the horizontal axis while the vertical axis is the percentage of students that have correctly answered the questions for the respective skills. Each cognitive skill consists of four readings obtained from the primary 4 pre-evaluation, primary 4 post-evaluation, primary 5 pre-evaluation and primary 5 post-evaluation tests.

From the pre-evaluation tests, as shown in Fig. 4.3, 91% of primary 4 and 94% of primary 5 students were proficient with the simple generalisation while 61% of primary 4 and 76% of primary 5 students could perform the moderate generalisation. However, only a small number of students, 13% for primary 4 and 50% of primary 5 students, were able to correctly perform difficult generalisation. After using EpiList I, an increase to 80% and 88% of primary 4 and primary 5 students respectively could perform moderate generalisation, and 49% and 63% for difficult generalisation. For comparison skill, initially on average 60% of the students are able to correctly perform this skill. This number increases to 76% after the interactions with EpiList I. The pre-evaluation test shows that more students, 80% on average, are able to perform discrimination. The post-evaluation shows that almost all the students, 97% on average, are capable of discrimination.

a) Improvement in skills

b) Decrease in Performance Gap

Figure 4.3: Students' Result of the Evaluation Tests

Focusing on Fig. 4.3a, the improvement on moderate generalisation skill is 19% for primary 4 and 12% for primary 5 students. Skill improvement measures the improvement of students' skills after using EpiList I and it is the difference of the post-

and pre-evaluation test results. A greater improvement on the difficult generalisation can be observed for primary 4 students with 36% skill improvement whereas there is only 13% reported/observed for primary 5 students. Similarly, the skill improvement for comparison skill is 21% for primary 4 and 12% for primary 5 students as compared to 15% and 8% for discrimination skill.

The improvements of various skills observed from the evaluation field results of Fig. 4.3a, strongly affirms that EpiList I is able to develop the generalisation and comparison skills of the student. On average, with 30 minutes of interaction with EpiList I, there are 23% and 12% improvement for primary 4 and primary 5 students respectively. The result also shows that EpiList I has greater impact on primary 4 students and significantly reduces the performance gap. Performance gap is the difference in skills between primary 4 and 5 students. It is determined by the difference between the primary 5 and the primary 4 results; hence, there will be pre- and post-evaluation performance gap.

Fig. 4.3b shows the values of performance gaps of the student. The results in the chart show that the performance gaps for difficult generalization are 37% and 14% for pre- and post-evaluation tests respectively. There is an improvement of 23% after using EpiList I. This shows that the system is very effective in developing the generalisation skill of primary 4 students. Similarly for comparison and discrimination skills, the performance gaps, on average, have improved from 13% to 5%.

The pre-evaluation test shows that 83% of primary 4 and 93% of primary 5 students are able to reason syllogistically. For the last two cognitive skills, positive and negative reasoning processes, the post-evaluation test shows that 16% of the students have acquired positive reasoning process while only 4.5% of the students have acquired negative reasoning process. 11.5% more students are able to acquire the positive than the negative reasoning processes. This reinforces the argument that the ability to observe similarity is an inborn basic skill for humans; hence understanding the positive reasoning process is easier. However, the low number of students

that have understood the positive and negative reasoning processes respectively is due to the inefficiency of the implicit acquisition of the reasoning processes employed in EpiList I. Each step of the reasoning processes is explicitly described in the instructions of the tutorials but the student has to implicitly acquire them during the interaction with EpiList I. Due to this implicit acquisition, only a very small number of students have acquired the reasoning processes.

4.4.4 Opinions

Figure 4.4: Results of Students' opinions

Fig. 4.4 shows the results of the questions prompting for the student's opinion. Nearly half of the students felt that EpiList I slightly improved their generalisation and comparison skills. 16.5% of the students felt that EpiList I helped to improve their skills, while 6.5% feel that it greatly improved their skills.

Almost the same distribution of primary 5 students can be observed from the second opinion as the first opinion. However, 5% more primary 4 students felt that EpiList I would greatly improve their skills and 22% more felt that it was a very useful tool to use. This reflects the higher efficiency of EpiList I for primary 4 students. In another words, primary 4 students are more suitable for skills development; therefore,

it is best to teach cognitive skills as early as possible.

4.5 Summary

EpiList I is an ITS that was developed from epistemic games. It is based on a bottom-up learning environment. It systematically teaches the skills of classification to the student. With this skill, the student is able to construct or organise knowledge by means of a semantic network. Tutorials of EpiList I implicitly impart the skills of generalisation and comparison to the student. EpiList I demonstrated that an ITS can be designed to teach cognitive skills with a systematic approach. It also shows that pedagogical theories from the educational field can be translated and incorporated as rules into an ITS that is independent of the domain.

EpiList I presently only conveys the basic human reasoning skills of classification, generalisation and comparison. However, it can form the basis or foundation for the future development of ITSs that teach higher forms of basic human reasoning. The mechanisms of EpiList I are formally expressed in terms of sets and set operations. At the same time, issues on the convergence of instruction to the target scheme was also examined. This resulted in a formal definition of EpiList I that is clear, precise and concise. It also enables mathematical analysis of EpiList I and makes EpiList I domain independent. These will be reported in the following chapter, Chapter 5.

Chapter 5

Formal Representation of EpiList I

“Unto you it is given to know the mystery of the kingdom of God;
but unto them that are without, all these things are done in parables.”

– Mark 4:11

5.1 Background

From the Webster dictionary, the following terms are defined as follows:

- Classification is “an act to arrange or organise according to classes or categories”.
- Generalisation is “the act of bringing individuals or particulars under a common genus or class”.
- Comparison is “the act to examine in order to note the similarities or differences”.

However, in the field of educational research, they have different connotations or definitions. Classification skill is referred to as Concept Learning and it consists of generalisation and discrimination skills. To further complicate matters, concept learning has two aspects that are very different to explicate; namely cognitive and behavioural aspects. From the cognitive aspect, “generalisation is a cognitive process by which one induces generalities from particulars” (P. L. Smith & Ragan, 1999);

whereas, discrimination is the process of identifying of differences between examples and non-examples of a concept. From the behavioural perspective, Pavlov's definition of generalisation is to respond similarly to a new situation as one would respond to a similar situation that he is familiar with (Hergenhahn & Olson, 1997), and discrimination is defined as responding differently to a new situation that is similar but not identical to the situation that he is familiar with.

From the various points above, the definitions and explanations of cognitive processes have numerous and differing connotations that result in confusion. To avoid verbose definitions of cognitive processes in EpiList I which may likewise result in ambiguous and incorrect understanding, a more concise and appropriate approach of explaining EpiList I is required. There are other ways of defining an ITS such as pseudo-code as in PIXIE (D. H. Sleeman & Brown, 1982) and logical expression as in Ego(Ng et al., 1995); however, these forms are inadequate to describe cognitive processes. This research proposes a formal approach that will clearly define the notion of classification, generalisation and comparison skills used in EpiList I. This method uses simple sets and set operations to define the representation of knowledge as well as the rules that guide the construction of relationships amongst the domain items in EpiList I. It concisely defines the cognitive processes: classification, generalisation and comparison, in the instructional sequences of EpiList I. Furthermore, it provides a clear description of the domain knowledge, instructional sequences and rules of EpiList I. Such a concise approach allows the cognitive reasoning process to be tracked, analysed and understood by the system and user. This formal approach also allows the analysis of execution sequence, time and space complexities of the system. Furthermore, the notions of simple sets and set operations are generic and independent of domain; therefore, the formal definition of EpiList I is domain and platform independent.

5.2 Formal Definitions of EpiList I's Rules and Algorithms

The rules and algorithms are organised into *intra- and inter-scheme* as shown in Section 4.3. The following sections present the formal definitions of the rules and algorithms.

5.2.1 General Definitions

Models

3 models are used in this game. They are the domain, expert and student models and are defined as follows:

- Domain Model (Dm): This model consists of l number of groups and o number of items; where l is the total number of classification categories and o is the total number of items used in the game. The groups in the model are the properties possessed by the items, which can be used as concept groups or categories to categorise the items.

The concept groups or categories are denoted as $G_{D_1}, G_{D_2}, \dots, G_{D_t}, \dots, G_{D_l}$ and $\{e_1, e_2, \dots, e_i, \dots, e_o\} \in Dm$ where e_i is an item in the game.

This model presents the correct categorisation of all items in their respective groups. The groups are the properties of all items used in the game. Fig. 5.1 shows the graphical representation of the domain model of the game. The *Groups*, noted as $\{G_{D_1}, G_{D_2}, G_{D_3}, G_{D_4}, G_{D_5}\}$, in the diagram are the properties of the items used in this example. When an item is labelled within a group, for example $e_1 \in G_{D_1}$, the item e_1 possesses the property G_{D_1} , hence $G_{D_1} \in prop(e_1)$ and $prop(e_1) = G_{D_1}, G_{D_4}$ where $prop(e_1)$ is a predicate that returns the properties of e_1 . This model completely but not uniquely classifies all items ; therefore, it is neither significant nor suitable.

Figure 5.1: The Domain Model

- Expert Model (Em): This model consists of m number of groups and q number of items; where m is the number of groups that are to be taught to the student and q is the number of items in the *playlist*. *Playlist* is a set of p number of items that will be used throughout a particular scenario of a game.

The concept groups in the expert model are denoted as $G_{E_1}, G_{E_2}, \dots, G_{E_t}, \dots, G_{E_m}$ and $\{e_1, e_2, \dots, e_i, \dots, e_q\} \in Em$ where $q < o$ and $q = p$.

The expert model uses a small group of properties extracted from the domain model to form a classification scheme. This classification scheme is the target classification scheme to be acquired by students. Hence, $G_{E_t} = G_{D_t}$ and $Em \subseteq Dm$. Fig. 5.2 shows the graphical representation of the expert model for the same scenario as Fig. 5.1. The *playlist* of this scenario is $\{e_1, e_2, e_3, \dots, e_{10}\}$. The groups in the figure, G_{E_4} and G_{E_5} , are the categories of the target classification scheme that are to be taught to the student. The items are items of the playlist and are correctly classified in their respective categories. Therefore, this model is both significant and suitable.

Figure 5.2: The Expert Model

- Student Model (*Sm*): This model consists of n number of groups and r number of items; where n is the number of groups that the student has selected and r is the number of items in the *playlist* or *lesser*. This model reflects the students' moves in which the items are placed according to the student's knowledge in the classification scheme created by the student.

The concept groups in this model, which the student has selected, are denoted as $G_{S_1}, G_{S_2}, \dots, G_{S_t}, \dots, G_{S_n}$ and $\{e_1, e_2, \dots, e_i, \dots, e_r\} \in Sm$ where $r < o$ and $r \leq p$, where o is the number of items in the domain model and p is the number of items in the *playlist*.

The student extracts properties from the domain model to form his or her classification scheme. Therefore, $G_{S_t} = G_{D_t} = G_{E_t}$.

Fig. 5.3 shows the graphical representation of the student model of the student playing the scenario presented in Fig. 5.1 and 5.2. The groups in the figure are the categories of the student's classification effort and the location of the items reflects the categorisation of items performed by the student. The scheme created was insignificant and unsuitable as it differs from the expert model,

shown in Fig. 5.2, that represents the current instructional goal. Furthermore, the items could be incorrectly categorised. Hence, this model may not be a proper subset of the domain model as the expert model is. On the other hand, when this model is similar to the significant and suitable scheme presented in the expert model ($Sm = Em$), the student is deemed to have comprehended the classification scheme currently being taught.

Figure 5.3: The Student Model

Terms used

- *Playlist* is the set of items that is used in the game.

Therefore, $playlist = \{e_1, e_2, \dots, e_i, e_j, e_k, \dots, e_p\}$ where p is the number of items in the playlist.

$$p = |playlist| = |Em| \text{ and } Playlist \subseteq Dm$$

- Let $G_{D_t}, G_{E_u}, G_{S_v}$ be a category found in Dm, Em and Sm respectively.

For notational purpose, if $t = u = v$, then G_{D_t}, G_{E_u} and G_{S_v} belong to the same category; hence, have the same property.

$$\text{Hence, } G_{D_t} = G_{E_t} = G_{S_t}.$$

- Referring to Fig. 5.4, if an item, e_i , is placed in Sm under the group G_{S_i} , this entry

is correct if and only if the item is a member of the group G_{D_t} in Dm .

Therefore, $(e_i \in G_{S_t}) \wedge (e_i \in G_{D_t}) \leftrightarrow e_i$ is correctly categorised in G_{S_t}

Figure 5.4: An Example of Correct Classification

There are two form of errors: namely, *False Acceptance*(FA) and *False Rejection*(FR). FA is classifying an *incorrect* item into a category. This item is ‘falsely accepted’ into the category. On the other hand, FR is not classifying a *correct* item into a category. This item is ‘falsely rejected’ from the category. Referring to Fig. 5.5 as an example, the item e_i is incorrectly grouped. With respect to group G_{S_t} , the error is a FR. with respect to group G_{S_u} , the error is a FA. Hence, FA and FR are not independent. A FA in one category also implies a FR in another category.

Figure 5.5: An Example of Incorrect Classification

FA: $(e_i \in G_{S_u}) \wedge (e_i \notin G_{D_u}) \leftrightarrow e_i$ is wrongly categorised in G_{S_u} .

FR: $(e_i \in G_{S_u}) \wedge (e_i \in G_{D_t})$ where $t \neq u \leftrightarrow e_i$ is wrongly categorised in G_{S_u} .

Predicates

- $prop(A)$: This returns the set of properties that is possessed by A . A can be a model (Dm , Em or Sm) or item. Since the groups of the model represents the properties, the set returned is a set of groups.

Examples:

$prop(Dm) = \{G_{D_1}, G_{D_2}, \dots, G_{D_i}, \dots, G_{D_o}\}$ where o is the number of groups in domain model. Using the Dm in Fig. 5.1 as an example, $prop(Dm) = \{G_{D_1}, G_{D_2}, G_{D_3}, G_{D_4}, G_{D_5}\}$.

$prop(Em) = \{G_{E_1}, G_{E_2}, \dots, G_{E_t}, \dots, G_{E_m}\}$ where m is the number of groups in expert model. Therefore, For Fig. 5.2, $prop(Em) = \{G_{E_4}, G_{E_5}\}$ and $prop(Em) \subseteq prop(Dm)$, since $G_{E_4} = G_{D_4}$ and $G_{E_5} = G_{D_5}$.

$prop(e_i) = \{G_{D_1}, G_{D_2}, \dots, G_{D_u}\}$ where $u \leq o$ and u is the number of properties in Dm possessed by e_i .

- *Group(model)*: This predicate will return a set of concept groups that is currently found in the model.

For example, from Fig. 5.2, $Group(Em) = \{G_{E_1}, G_{E_2}\}$

Therefore, $Group(Em) \subseteq Group(Dm)$ and $Group(Sm) \subseteq Group(Dm)$

- *BestExample(item)*: This returns the best example for an item. Best example of an item is the item most similar to the item and is used as example for the item. For example, $BestExample(e_1) = e_{11}$.

The following predicates are actions for students to perform or carry out. They describe the type of interfaces needed and the expected actions from the student.

- *ObtainProperties(item)*: This predicate defines what the student is required to identify of the properties of an item presented to him.
- *GeneraliseProperties*($\{e_1, e_2, \dots, e_i\}$): This predicate defines that the student is required to derive the common property of the items in the set. The return value is the set of common properties present.

Hence, $GeneraliseProperties(\{e_1, e_2, \dots, e_i\}) = prop(e_1) \cap prop(e_2) \cap \dots \cap prop(e_i) = \{G_{D_t}, \dots\} \cap \{G_{D_u}, \dots\} \cap \dots \cap \{G_{D_v}, \dots\}$

- *Compare(PropertySetA, PropertySetB)*: This is used to instruct the student to compare 2 sets of properties together to identify the similar properties among them. It returns a set of properties common to both sets of properties.

Hence, $Compare(PropertySetA, PropertySetB) = PropertySetA \cap PropertySetB$.

If $PropertySetA = \{G_{D_1}, G_{D_2}, G_{D_3}\}$ and $PropertySetB = \{G_{D_2}, G_{D_4}, G_{D_6}\}$, then $Compare(PropertySetA, PropertySetB) = \{G_{D_2}\}$.

- *Contrast(PropertySetA, PropertySetB)*: This is used to instruct the student to contrast 2 sets of properties against each other to identify the different properties between them. It returns a set of properties present in one but not the

other.

Hence,

$$\text{Contrast}(\text{PropertySet}A, \text{PropertySet}B) = (\text{PropertySet}A \cap \text{PropertySet}B)^{-1}$$

Using the example shown in the *Compare()* predicate,

$$\text{Contrast}(\text{PropertySet}A, \text{PropertySet}B) = \{G_{D_1}, G_{D_3}, G_{D_4}, G_{D_6}\}.$$

- *ShowImportance(Property)*: This predicate shows the student that the property is an important property and instructs him or her to include it in *Sm*.
- *ShowRedundancy(Property)*: This predicate shows the student that the property is a redundant property and instructs him or her to remove it from *Sm*.

5.2.2 Intra-scheme Rules

This section focuses on the set of Intra-scheme rules. These rules are used to instruct students on how to identify their mistakes within a classification scheme. These mistakes are based on the incorrect classification of items in the lists. There are six intra-scheme rules that are used to handle various types of mistakes made.

When highlighting the mistakes to the student, examples of similar concepts are used to show similarity hence indirectly hinting that the misclassified item belongs to the same concept. Instruction of this form uses *positive examples* to compare with the mistakes; hence it is labeled as a ‘*Positive Exemplar Approach*’. Similarly, a ‘*Negative Exemplar Approach*’ is the contrasting of *negative examples* with the mistakes. Negative examples are correctly placed examples of the category in which the mistakes have been found and are used to hint to the student that they belong to different categories and hence should not be grouped together.

Under situations where some correct categorisations can be found in the *Sm*, these correct answers would be used to great advantage to tutor the student. Since the items are correctly grouped, it shows that the student has understood the concept uniting these items. Hence, it is more effective to tutor using these correctly grouped

items. This form of tutoring is labeled '*Linear*'. Linear tutoring is used to highlight *one* mistake made by the student. Hence the number of items in the tutorial is directly proportional to the number of correctly grouped items. Linear tutoring can be tagged with either positive or negative exemplars depending on the categories of the correct answers and the mistake. If the correct answers made by the student, which are used by the tutorials, have the same property as the mistake; then the tutorial uses the linear positive exemplar rule. If the correct answers made by the student were to be grouped in a category where the mistake was found or wrongly classified to, it would be the linear negative exemplar rule.

When there is more than one mistake and these mistakes belong to the same category, it is more efficient to gather these mistakes and tutor the student simultaneously. This is labeled as '*Non-linear*'. Hence, under a non-linear condition, generalisation occurs twice, once for the correct answers and once for the mistakes. The number of items used in the tutorial varies non-linearly depending on the number of mistakes made and correct categorisation. Similar to linear tutoring, non-linear tutoring can be performed with positive and negative exemplars.

The following sections describe the intra-scheme rules in detail. The rules are described in the following sequence of *Positive*, *Negative*, *Linear Positive*, *Linear Negative*, *Non-Linear Positive* and *Non-linear Negative Exemplars Rules*. To assist in the explanation of the rules, each rule is explained with respect to an example. The *playlist* of all the examples is defined by the list: $playlist = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$. Fig. 5.6 is the example of the 'Positive Exemplar Rule'. In this example, the domain model(Dm) presented is a simplified version of the complete Dm in Fig. 5.1. This simplified Dm only presents items that is required by the rule to determine the student's mistake and tutor accordingly. Hence, the simplified Dm is overlaid with the student model(Sm) to derive the error in the student's actions. The Sm reflects the moves or actions of the student. The Sm in Fig. 5.6 shows that the student has selected G_{S_1} , G_{S_2} , and G_{S_3} as his classification scheme and grouped e_1 in

G_{S_3} .

Rule 1: Positive Exemplar Rule

Figure 5.6: Example of Positive Exemplar Rule

	Semantic of notation	Formal notation	Actions based on example of Fig. 5.6
<i>If</i>	Misclassification exists	① $(\exists e_i \in G_{S_t} \text{ s.t. } e_i \notin G_{D_t})$	$e_i = e_1, t = 3$
	and	$\wedge$	
	No correct classification present	② $(\forall e_j \in G_{S_v}, e_j \notin G_{D_v}) \text{ where } i \neq j$	$e_j = \emptyset$
<i>Then</i>	Obtain the properties of the best example of the mistake	③ $C = ObtainProperties(BestExample(e_i))$,	$BestExample(e_1) = e_{11}$ $C = \{G_{D_1}\}$
	Compare the derived properties against the properties of the mistake	④ $Compare(C, Prop(e_i))$	$Prop(e_1) = \{G_{S_1}\}$

Table 5.1: Formal notation of Positive Exemplar Rule (Eqn. 5.1)

Fig. 5.6 is the graphical representation of the checking condition. Eqn. 5.1① checks that there exists an item, e_i , wrongly categorised by the student. Eqn. 5.1② ensures that every other item, except e_i , is also wrongly categorised. Hence, there is no other correct categorisation made. From Fig. 5.6, e_1 is the mistake and there is no other correct placement in Sm . To tutor the student about his mistake, Eqn. 5.1③ presents

the best example of the mistake, e_{11} , to the student to realise the property of G_{S_1} . It is subsequently compared with the properties of e_1 in Eqn. 5.1④ to allow the student to infer the similarity. Hence, implying that e_1 should be categorised under that property. This is called the '*Positive Exemplar Rule*'.

Rule 2: Negative Exemplar Rule

Figure 5.7: Example of Negative Exemplar Rule

Eqn. 5.2① and 5.2② are similar to the Eqn. 5.1① and 5.1② of the '*Positive Exemplar Rule*'. From the example given in Fig. 5.7, e_1 is the only mistake and there is no other correct placement. However, in this case, the best example of the category, where the mistake is found, is used in Eqn. 5.2⑥ to obtain the property G_{D_i} . As the input of the predicate $BestExample()$ is an item and not a category, the best example of a category is obtained by using an item that belongs to that category with $BestExample()$. Such an item is first searched within the playlist; if it is not found, that item will be selected from the domain model. This searching process is carried out by Eqn. 5.2③④⑤. Therefore, in the scenario shown in Fig. 5.7, e_{10} is selected from the playlist as e_k and the best example of e_{10} is e_{12} . Hence, e_{12} is the best example of the category G_{D_3} . e_{12} will be presented, in Eqn. 5.2⑥, to the student to realise the property of the category G_{D_3} . Subsequently, Eqn. 5.2⑦ contrasts the

obtained property against properties of e_1 to demonstrate the differences. Hence, implying that the item, e_1 , should not be classified under this group. This is called the 'Negative Exemplar Rule'.

	Semantic of notation	Formal notation	Actions based on example of Fig. 5.7
<i>If</i>	Misclassification exists	① $(\exists e_i \in G_{S_t} \text{ s.t. } e_i \notin G_{D_t})$	$e_i = e_1, t = 3$
	and	$\wedge$	
	No correct classification present	② $(\forall e_j \in G_{S_v}, e_j \notin G_{D_v}) \text{ where } i \neq j$	$e_j = \emptyset$
<i>Then</i>	Obtain an item that belongs to the category where the mistake is found	③ if $(\exists e_j \in \text{playlist s.t. } e_j \in G_{D_t})$ ④ then $(e_k = e_j)$ ⑤ else $(e_k \in Dm \text{ s.t. } e_k \in G_{D_t})$	$e_k = e_j = e_{10}$
	Obtain the properties of the best example of category where the mistake is found	⑥ $C = \text{ObtainProperties}(\text{BestExample}(e_k))$	$\text{BestExample}(e_{10}) = e_{12}$ $C = \{G_{D_3}\}$
	Contrast the derived properties against the properties of the mistake	⑦ $\text{Contrast}(C, \text{Prop}(e_i))$	$\text{Prop}(e_1) = \{G_{S_1}\}$

Table 5.2: Formal notation of Negative Exemplar Rule (Eqn. 5.2)

Rule 3: Linear Positive Exemplar Rule

Figure 5.8: Example of Linear Positive Exemplar Rule

	Semantic of notation	Formal notation	Actions based on example of Fig. 5.8
<i>If</i>	Misclassification exists	① $(\exists e_i \in G_{S_t} \text{ s.t. } e_i \in G_{D_v}) \text{ where } t \neq v$	$e_i = e_1, t = 2, v = 1$
	and	$\wedge$	
	All other groupings are correct (Implies only one mistake)	② $(\forall e_j \in G_{S_u}, e_j \in G_{D_u}) \text{ where } i \neq j$	$\{e_2, e_6, e_7\}$ are correctly grouped
	and	$\wedge$	
	At least one correct grouping made in the category that belongs to the mistake	③ $(\exists e_k \in G_{S_v} \text{ s.t. } e_k \in G_{D_v})$	$e_k = e_2 \text{ or } e_k = e_6 \text{ or } e_k = e_7$
<i>Then</i>	Extract all the correctly grouped items in the category that belongs to the mistake	④ $G_e \forall e_g \in G_e, e_g \in G_{S_v} \rightarrow e_g \in G_{D_v}$	$G_e = \{e_2, e_6, e_7\}$
	Generalise the extracted correctly grouped items	⑤ $C = \text{GeneraliseProperties}(G_e)$	$C = \{G_{D_1}\}$
	Compare the derived common properties against the properties of the mistake	⑥ $\text{Compare}(C, \text{Prop}(e_i))$	$\text{Prop}(e_1) = \{G_{S_1}\}$

Table 5.3: Formal notation of Linear Positive Exemplar Rule (Eqn. 5.3)

Eqn. 5.3① checks that there is at least one item, e_i , wrongly grouped. From the example in Fig. 5.8, e_1 is classified wrongly. Eqn. 5.3② states that the other items are correctly grouped; induces that e_i is the only error in Sm . Eqn. 5.3③ ensures that at least one of the correctly grouped items belonging to the same property as the mistake. Eqn. 5.3⑤ generalises the correctly grouped items, G_e , that belong to the same property as the mistake. The set of items, G_e , is extracted or determined with Eqn. 5.3④. In the case shown in Fig. 5.8, the items are $\{e_2, e_6, e_7\}$. Subsequently, the common property is compared with properties of e_1 in Eqn. 5.3⑥ to identify the similarity among them. With the knowledge of the similarity with the correct answers, the student knows that e_1 should be grouped together with the other items.

This teaching method is called the '*Linear Positive Exemplar Rule*'.

Rule 4: Linear Negative Exemplar Rule

Figure 5.9: Example of Linear Negative Exemplar Rule

The first two conditions, Eqn. 5.4① and 5.4②, have the same definition as the first two conditions, Eqn. 5.3① and 5.3②, of the '*Linear Positive Exemplar Rule*'. The difference is the third condition (Eqn. 5.4③) which ensures that there exists at least one of the correctly categorised items belonging to the property of the category where the mistake is placed. Hence, from Fig. 5.9, e_1 is the error and the correctly categorised items are $\{e_2, e_7, e_8\}$. To tutor the student, the correctly placed items, $\{e_2, e_7, e_8\}$, are extracted in Eqn. 5.4④ and generalised together in Eqn. 5.4⑤ to derive the common properties, which are subsequently contrasted against properties of e_1 in Eqn. 5.4⑥ to demonstrate the difference between them. When the student learns that the mistake, e_1 , is different from the correct placements, the student would be guided to realise that e_1 should not be grouped under that property. This is called the '*Linear Negative Exemplar Rule*'.

	Semantic of notation	Formal notation	Actions based on example of Fig. 5.9
<i>If</i>	Misclassification exists	① $(\exists e_i \in G_{S_t} \text{ s.t. } e_i \in G_{D_v}) \text{ where } t \neq v$	$e_i = e_1, t = 2, v = 1$
	and	$\wedge$	
	All other groupings are correct (Implies only one mistake)	② $(\forall e_j \in G_{S_u}, e_j \in G_{D_u}) \text{ where } i \neq j$	$\{e_2, e_7, e_8\}$ are correctly grouped
	and	$\wedge$	
	At least one correct grouping made in the category where the mistake is found	③ $(\exists e_k \in G_{S_t} \text{ s.t. } e_k \in G_{D_t})$	$e_k = e_2 \text{ or } e_k = e_7 \text{ or } e_k = e_8$
<i>Then</i>	Extract all the correctly grouped items in the category where the mistake is found	④ $G_e \forall e_g \in G_e, e_g \in G_{S_t} \rightarrow e_g \in G_{D_t}$	$G_e = \{e_2, e_7, e_8\}$
	Generalise the extracted correctly grouped items	⑤ $C = \text{GeneraliseProperties}(G_e)$	$C = \{G_{D_2}\}$
	Contrast the derived common properties against the properties of the mistake	⑥ $\text{Contrast}(C, \text{Prop}(e_i))$	$\text{Prop}(e_1) = \{G_{S_1}\}$

Table 5.4: Formal notation of Linear Negative Exemplar Rule (Eqn. 5.4)

Rule 5: Non-linear Positive Exemplar Rule

Figure 5.10: Example of Non-linear Positive Exemplar Rule

	Semantic of notation	Formal notation	Actions based on example of Fig. 5.10
<i>If</i>	Misclassification exists	① $(\exists e_i \in G_{S_t} \text{ s.t. } e_i \in G_{D_v})$ where $t \neq v$	$e_i = e_1, t = 2, v = 1$ (or $e_i = e_2, t = 3, v = 1$)
	and	$\wedge$	
	Another misclassification that possesses the same property exists	② $(\exists e_j \in G_{S_u} \text{ s.t. } e_j \in G_{D_v})$ where $i \neq j$ and $u \neq v$	$e_j = e_2, u = 3$ (or $e_j = e_1, u = 2$)
	and	$\wedge$	
	At least one correct grouping made in the category that belongs to the mistakes	③ $(\exists e_k \in G_{S_v} \text{ s.t. } e_k \in G_{D_v})$	$e_k = e_6$ or $e_k = e_7$
<i>Then</i>	Extract all the correctly grouped items in the category that belongs to the mistakes	④ $G_{e_1} \forall e_g \in G_{e_1}, (e_g \in G_{S_v} \rightarrow e_g \in G_{D_v})$	$G_{e_1} = \{e_6, e_7\}$
	Generalise the extracted correctly grouped items	⑤ $C_1 = \text{GeneraliseProperties}(G_{e_1})$	$C_1 = \{G_{D_1}\}$
	Extract all the incorrectly grouped items that possess the same property	⑥ $G_{e_2} \forall e_h \in G_{e_2}, (e_h \in G_{S_w} \rightarrow e_h \in G_{D_v}) \text{ where } w \neq v$	$G_{e_2} = \{e_1, e_2\}$
	Generalise the extracted incorrectly grouped items	⑦ $C_2 = \text{GeneraliseProperties}(G_{e_2})$	$C_2 = \{G_{S_1}\}$
	Compare the two sets of common properties to induce that the 2 sets are similar	⑧ $\text{Compare}(C_1, C_2)$	

Table 5.5: Formal notation of Non-linear Positive Exemplar Rule (Eqn. 5.5)

Eqn. 5.5① states that there exists an item e_i wrongly grouped and Eqn. 5.5② states that there exists another item e_j having the same property, G_{D_v} , as e_i is also wrongly grouped. From Fig. 5.10, the two items are e_1 and e_2 . Eqn. 5.5③ ensures that there exists some correctly categorised item belonging to the same property as the mistakes. To tutor the student, first Eqn. 5.5④ extracts all the correctly categorised

items, $\{e_6, e_7\}$ according to Fig. 5.10, and generalises together in Eqn.5.5⑤ to derive their common properties. Eqn. 5.5⑥ extracts all the mistakes, $\{e_1, e_2\}$, and generalises together in Eqn. 5.5⑦ to derive another set of common properties. Subsequently, the two sets of common properties are compared together in Eqn. 5.5⑧ to identify the similarity among them. Knowing the similarities, the student would induce a conclusion that the wrongly categorised items should be categorised together under the same property G_{D_1} . This is called the '*Non-linear Positive Exemplar Rule*'.

Rule 6: Non-linear Negative Exemplar Rule

Figure 5.11: Example of Non-linear Negative Exemplar Rule

The first 2 equations, Eqn. 5.6① and Eqn. 5.6②, have the same definition as Eqn. 5.5① and Eqn. 5.5② of '*Non-linear Positive Exemplar Rule*' to ensure more than one mistake exists. From Fig. 5.11, the mistakes are $\{e_1, e_6, e_7\}$. However, Eqn. 5.6③ ensures that there exists some correctly grouped item belonging to the property, G_{S_i} , where the first mistake, e_i , is placed. Similarly, to tutor the student, first the correctly categorised items, namely $\{e_4, e_{10}\}$ from Fig. 5.11, are extracted by Eqn. 5.6④ and generalised in Eqn. 5.6⑤ to derive a set of common properties. Subsequently, the wrongly categorised items are extracted by Eqn. 5.6⑥ and generalised together in Eqn. 5.6⑦ to identify another set of common properties. Lastly, the two sets of common properties are contrasted against each other in Eqn. 5.6⑧ to demonstrate

the difference between the two sets. Now, knowing that the two sets of items are of different common properties, the student can induce that these items should not be placed together under the same property but instead to regroup these items after studying the information on them. This is called the '*Non-linear Negative Exemplar Rule*'.

	Semantic of notation	Formal notation	Actions based on example of Fig. 5.11
<i>If</i>	Misclassification exists	① $(\exists e_i \in G_{S_t} \text{ s.t. } e_i \in G_{D_v})$ where $t \neq v$	$e_i = e_1, t = 3, v = 1$ (or $e_i = e_7, t = 3, v = 1$)
	and	$\wedge$	
	Another misclassification that possesses the same property exists	② $(\exists e_j \in G_{S_u} \text{ s.t. } e_j \in G_{D_v})$ where $i \neq j$ and $u \neq v$	$[e_j = e_6, u = 2$ or $e_j = e_7, u = 3]$ or $[e_j = e_6, u = 2$ or $e_j = e_1, u = 3]$
	and	$\wedge$	
	At least one correct grouping made in the category where the first mistake is found	③ $(\exists e_k \in G_{S_t} \text{ s.t. } e_k \in G_{D_t})$	$e_k = e_4$ or $e_k = e_{10}$
<i>Then</i>	Extract all the correctly grouped items in the category where the first mistake is found	④ $G_{e_1} \forall e_g \in G_{e_1}, (e_g \in G_{S_t} \rightarrow e_g \in G_{D_t})$	$G_{e_1} = \{e_4, e_{10}\}$
	Generalise the extracted correctly grouped items	⑤ $C_1 = \text{GeneraliseProperties}(G_{e_1})$	$C_1 = \{G_{D_3}\}$
	Extract all the incorrectly grouped items that possess the same property	⑥ $G_{e_2} \forall e_h \in G_{e_2}, (e_h \in G_{S_w} \rightarrow e_h \in G_{D_v})$ where $w \neq v$	$G_{e_2} = \{e_1, e_6, e_7\}$
	Generalise the extracted incorrectly grouped items	⑦ $C_2 = \text{GeneraliseProperties}(G_{e_2})$	$C_2 = \{G_{S_1}\}$
	Contrast the two sets of common properties to induce that a difference exist between the 2 sets	⑧ $\text{Contrast}(C_1, C_2)$	

Table 5.6: Formal notation of Non-linear Negative Exemplar Rule (Eqn. 5.6)

The Algorithm

Figure 5.12: The Algorithm for Intra-scheme Rules

Fig. 5.12 illustrates the algorithm of the intra-scheme rules. First the algorithm checks if the items were taught before. If they have been taught more than 3 times, implying that more than 3 rules were used to tutor the student on the same mistake, the system will instruct the student directly instead of using the rules and tutorials again. From the diagram, it can also be observed that the rules are organised into three steps. The first step involves Non-linear Positive and Negative Exemplar Rules; followed by the Linear Positive and Negative Exemplar Rules; lastly are the Positive and Negative Exemplar Rules. Focusing on the Positive rule types, it can be observed that Positive Exemplar Rule operates with only a set of *two* items, Linear Positive Exemplar Rule with a bigger set of items and Non-linear Positive Exemplar Rule have the largest set. Hence, the item set of the Positive Exemplar Rule is a proper subset item set of Linear Positive Exemplar Rule which in turn is a proper subset of Non-Linear Positive Exemplar Rule. Therefore, the checking sequence is from Non-linear Positive then Linear Positive and finally followed by Positive Exemplar Rules such that the sequence is from the largest to the smallest. If the sequence begins with smallest first, the algorithm would be biased towards Positive Exemplar Rule.

Within each step, the rules are organised to perform the ‘positive approach’ first followed by the ‘negative approach’ provided that the mistake has not been tutored before. The ‘positive approach’ group compares similar properties to hint at the correct answer, whereas the ‘negative approach’ group contrasts different properties to eliminate incorrect answers. Experiments performed on children have shown that infants as young as 3 month are able to recognise similar shapes of different forms, for example triangles with thick lines, thin lines, dotted lines, etc (Goswami, 1998). Hence, it is argued that the ability to observe similarity is an inborn basic skill for human. Therefore, the algorithm will first attempt to use comparison to derive similarity before contrast is used to demonstrate differences.

Furthermore, the ‘positive approach’ directly highlights the correct category; the student is guided from his mistake towards the correct answer. The ‘negative ap-

proach', on the other hand, uses the elimination strategy to *eliminate* the incorrect answers. The 'negative approach' will indicate that the erroneous item does not belong to the category. However, the student may still categorise the item incorrectly. This will cause multiple execution of the second group of rules depending on the new student model after the rearrangement of items by the student. The multiple executions will eliminate incorrect categories one at a time until only the correct category is left. This is an effective elimination strategy and is often employed in selection when there are multiple choices. Therefore, after an item is taught for the first time, it is guided by the algorithm to check for the 'negative approach' rules only.

When all the items are correctly categorised, the classification scheme is complete. Under this situation, no more rules will be executed, since the execution of the rules entails that there exists at least one incorrectly categorised item. When none of the intra-scheme rules are executed, the classification scheme is complete but it may or may not be significant. If the classification scheme is non-unique, then is it not significant and unsuitable to the current teaching goal. If the classification scheme is unique, then it is significant but it may still not be suitable for the current teaching goal set forth at the beginning of the game. Hence, the student model is subsequently passed to the inter-scheme rules to migrate the student's classification scheme to a *significant and suitable* classification scheme.

5.2.3 Inter-scheme Rules

Inter-scheme rules focus on guiding students to migrate from one classification scheme to another. The migration is accomplished by inducing the student to recognize the importance of the target classification scheme and leads on to new categories of a target scheme while removing the categories from an existing scheme. There are four rules in the set of inter-scheme rules; namely Positive Exemplar Introduction, Negative Exemplar Introduction, Regrouping and Elimination Rules. The Positive and Negative Exemplar Introduction Rules introduce a new category of the target

scheme to the students whilst the Elimination Rule removes old categories. The Re-grouping Rule focuses on moving elements from old categories to the newly introduced categories.

The Student Model (Sm) is passed over from the intra-scheme to inter-scheme rules when all items are categorized correctly. Whenever an element is moved or re-categorized due to execution of inter-scheme rule, the student model is again passed back to the intra-scheme rules to check on the completeness of the model as the inter-scheme rules do not have such a mechanism. Once the model has exited from the intra-scheme rules, it is again handed back to inter-scheme rules to continue the migration process. This cycle, as shown in Fig. 5.13, goes on until a significant and complete classification scheme is obtained. The formal definition of Sm with all items correctly categorized is also shown in Fig. 5.13. Hence, the termination of this loop occurs when $(\forall e_i \in playlist, (e_i \in G_{St}) \wedge (e_i \in G_{Dt})) \wedge (Group(Sm) = Group(Em))$ is true.

Figure 5.13: Cycle between Intra- and Inter-scheme Rules

The following sections describes the inter-scheme rules in detail. Similarly, examples are used to assist in the explanation of the rules. However, the Expert model (Em) is used instead of the Domain model (Dm). Referring to the example of the

‘Positive Exemplar Introduction Rule’, Fig. 5.14, the Em shown is the target classification form for $playlist = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$. The Sm in the figure reflects the actions of the student and it shows that the student has correctly grouped all items in $\{G_{S_1}, G_{S_2}, G_{S_3}\}$. By overlaying the Em with Sm , the significance of the categories of Sm can be determined.

Rule 7: Positive Exemplar Introduction Rule

Figure 5.14: Example of Positive Exemplar Introduction Rule

	Semantic of notation	Formal notation	Actions based on example of Fig. 5.14
<i>If</i>	The Em and Sm do not match exactly implying none of the important category is selected	① $(Group(Sm) \cap Group(Em) = \emptyset)$	$\{G_{S_1}, G_{S_2}, G_{S_3}\} \cap \{G_{E_4}, G_{E_5}\} = \emptyset$
<i>Then</i>	Extract all the items that belong to the one important category	② $G_e \forall e_g \in G_e, e_g \in Sm \rightarrow e_g \in G_{E_t}$	$G_e = \{e_1, e_2, e_3, e_4, e_5\},$ $G_{E_t} = G_{E_4}$
	Generalise the extracted items	③ $C = GeneraliseProperties(G_e)$	$C = \{G_{E_4}\}$
	Highlight the importance of the common property	④ $ShowImportance(C)$	

Table 5.7: Formal notation of Positive Exemplar Introduction Rule (Eqn. 5.7)

As shown in Fig. 5.14, the intersection of groups from Em and Sm is $NULL$ which satisfies the condition of Eqn. 5.7①. This means that the student selects none of the target categories from Em . On the other hand, if the student selects one or more categories from Em , the intersection will not be empty. To guide the student towards the target classification scheme, firstly the set of items, G_e , from a randomly selected group of Em , G_{E_t} , are extracted by Eqn. 5.7② and generalised together in Eqn. 5.7③ to induce its common property. Subsequently, Eqn. 5.7④ explicitly presents this common property as an important group and prompts the student to include it in their classification scheme.

Rule 8: Negative Exemplar Introduction Rule

Figure 5.15: Example of Negative Exemplar Introduction Rule

	Semantic of notation	Formal notation	Actions based on example of Fig. 5.15
<i>If</i>	The <i>Em</i> and <i>Sm</i> overlap partially implying that one or some of the important category is selected	① $(Group(Sm) \cap Group(Em) \neq \emptyset)$	$\{G_{S_1}, G_{S_2}, G_{S_3}, G_{S_5}\} \cap \{G_{E_4}, G_{E_5}\} = \{G_{S_5}\}$
	and	$\wedge$	
	The <i>Em</i> is not a proper subset of <i>Sm</i> implying that not all the important category is selected	② $(Em \not\subseteq Sm)$	$\{G_{E_4}, G_{E_5}\} \not\subseteq \{G_{S_1}, G_{S_2}, G_{S_3}, G_{S_5}\}$
<i>Then</i>	Extract all the items that belong to the important category that the student had selected	③ $G_{e_1} \forall e_g \in G_{e_1}, e_g \in G_{S_t}$ where $G_{S_t} \in Group(Sm) \cap Group(Em)$	$G_{e_1} = \{e_6, e_7, e_8, e_9, e_{10}\},$ $G_{S_t} = G_{S_5}$
	Generalise the extracted items	④ $C_1 = GeneraliseProperties(G_e)$	$C_1 = \{G_{E_5}\}$
	Extract all the items that belong to the another important category that the student had not selected	⑤ $G_{e_2} \forall e_h \in G_{e_2}, e_h \in S_m \wedge e_h \in G_{E_u}$ where $G_{E_u} \in (Group(Em) - Group(Sm))$	$G_{e_2} = \{e_1, e_2, e_3, e_4, e_5\},$ $\{G_{E_4}, G_{E_5}\} - \{G_{S_1}, G_{S_2}, G_{S_3}, G_{S_5}\} = \{G_{E_4}\},$ $G_{E_u} = G_{E_4}$
	Generalise the extracted items	⑥ $C_2 = GeneraliseProperties(G_{e_2})$	$C_2 = \{G_{E_4}\}$
	Compare the two sets of common properties in terms of higher-order property	⑦ $Compare(C_1, C_2)$	
	Highlight the importance of the unselected property	⑧ $ShowImportance(C_2)$	

Table 5.8: Formal notation of Positive Exemplar Introduction Rule (Eqn. 5.8)

Fig. 5.15 shows that the intersection of the groups of *Em* and *Sm* is not NULL, this is the condition indicated in Eqn. 5.8①. Eqn. 5.8② ensures that the *Em* is not a subset of *Sm*, implying that *Sm* does not contain all categories in *Em*. This means that the student only selected some but not all categories from *Em*. The

scenario shown in Fig. 5.15 satisfies these conditions. The objective of this rule is to highlight to the student the importance of the unselected category from Em that has a similar higher-order property as the selected category from Em in Sm . Referring to Fig. 5.15, the unselected group is G_{E4} and the selected group is G_{E5} . Firstly, Eqn. 5.8③ extracts items from a selected group of Em , G_{E4} , and generalises together in Eqn. 5.8④ to derive the common property. Subsequently, Eqn. 5.8⑤ extracts items of an unselected property of Em , G_{E5} , and generalises together in Eqn. 5.8⑥ to derive another common property. Lastly, the two common properties are compared together in terms of a higher-order property using Eqn. 5.8⑦ to demonstrate their similarity in terms of higher-order property. Eqn. 5.8⑧ indicates to the student that the two properties are of the same higher-order property and instructs him to include the unselected group in his classification scheme.

Rule 9: Regrouping Rule

Figure 5.16: Example of Regrouping Rule

	Semantic of notation	Formal notation	Actions based on example of Fig. 5.16
<i>If</i>	The <i>Em</i> and <i>Sm</i> overlap partially implying that one or some of the important category is selected	① $(Group(Em) \cap Group(Sm) \neq \emptyset)$	$\{G_{S_1}, G_{S_2}, G_{S_3}, G_{S_5}\} \cap \{G_{E_4}, G_{E_5}\} = \{G_{S_5}\}$
	and	$\wedge$	
	There exists an item that is not grouped in an important category	② $(\exists e_i \in G_{S_t} \text{ s.t. } e_i \in G_{E_u})$ where $t \neq u$ and $G_{E_u} \in Group(Em) \cap Group(Sm)$	$e_i = e_6$ (or $e_i = e_{10}$), $t = 1$ (or $t = 3$), $u = 5$
<i>Then</i>	Extract all the items that are already grouped in that important category	③ $G_{e_1} \forall e_g \in G_{e_1}, (e_g \in G_{S_u} \rightarrow e_g \in G_{E_u})$	$G_{e_1} = \{e_7, e_8, e_9\},$ $u = 5$
	Generalise the extracted items	④ $C_1 = GeneraliseProperties(G_{e_1})$	$C_1 = \{G_{S_5}\}$
	Extract all the items that are not grouped in that important category	⑤ $G_{e_2} \forall e_h \in G_{e_2}, (e_h \in G_{S_v} \rightarrow e_h \in G_{E_u})$ where $v \neq u$ and $1 \leq v \leq n$ (n is the number of groups in <i>Sm</i>)	$G_{e_2} = \{e_6, e_{10}\}$
	Generalise the extracted items	⑥ $C_2 = GeneraliseProperties(G_{e_2})$	$C_2 = \{G_{S_5}\}$
	Compare the two sets of common properties in terms of higher-order property	⑦ $Compare(C_1, C_2)$	

Table 5.9: Formal notation of Regrouping Rule (Eqn. 5.9)

From Fig. 5.16, the intersection of groups from *Em* and *Sm* is non-empty which satisfies Eqn. 5.9①. Eqn. 5.9② states that there exists an item belonging to category G_{E_u} but is not grouped in that category. G_{E_u} is a category of *Em* selected by the student. This means that there exists a group in *Sm*, in a member of *Em*, that does not categorise all the items in that group. Referring to Fig. 5.16, the missing items are e_6 and e_{10} . The strategy of this rule is to let the student understand that the missing items should be placed under this group. First, Eqn. 5.9③ extracts the items already categorised in the property G_{E_u} and Eqn. 5.9④ generalises these items to

derive a common property. Subsequently, Eqn. 5.9⑥ generalises the missing items of the property extracted by Eqn. 5.9⑤ to derive another common property. Lastly, both common properties are compared together in Eqn. 5.9⑦ to show similar properties. Hence indicating that the items should be grouped together in property G_{Eu} .

Rule 10: Elimination Rule

Figure 5.17: Example of Elimination Rule

	Semantic of notation	Formal notation	Actions based on example Fig. 5.17
<i>If</i>	There exists a category that is not important and is empty	① $\exists G_{S_t} \in Group(Sm) \text{ s.t. } ((G_{S_t} = \emptyset) \wedge (G_{S_t} \notin Group(Em)))$	$G_{S_t} = G_{S_3}(Case1)$ or $G_{S_t} = G_{S_2}(Case2)$
<i>Then</i>	Show the redundancy of that category	② $ShowRedundancy(G_{S_t})$	

Table 5.10: Formal notation of Elimination Rule (Eqn. 5.10)

The condition of the rule, Eqn. 5.10①, will be true if there exists a group in Sm which does not have any item and the property does not belongs to Em . In both cases as shown in Fig. 5.17 there exists an empty group in Sm and this group is not

a member of Em . The difference between the two cases is that the Sm of Case 2 has all the property found in Em but Case 1 does not. The empty category is the result of the Regrouping Rule that moves items towards the important groups from Em leaving the unimportant groups empty. For both cases, Eqn. 5.10② highlights the redundancy of the empty group and instructs the student to remove it from his classification scheme.

The Algorithm

Figure 5.18: The Algorithm for Inter-scheme Rules

Fig. 5.18 illustrates the algorithm governing the execution of the set of inter-scheme rules that will migrate students from one classification scheme to another. The

algorithm begins with the removal of empty categories (Elimination Rule). This rule has to be repeated until all empty categories are removed, this is because an empty category will cause an incorrect decision or execution of the other inter-scheme rules and also causes confusion to the student. This is shown as a self-loop (1) in Fig. 5.18. After all the empty categories are removed, the student model is checked against the Positive Exemplar Introduction Rule. This rule will introduce the first category of the target classification scheme. If one or more categories of the target classification scheme already exist, this would imply that Positive Exemplar Introduction Rule is not applicable; the Regrouping Rule will be checked. The Regrouping Rule checks if these groups contain every item that it can categorise. If not, this rule will move the missing items into these groups. When the Regrouping Rule is not applicable, it means that the groups of the target scheme found in the student model have all the items that can be categorized. Finally, the Negative Exemplar Introduction Rule will introduce the remaining groups from the target classification scheme one at a time. Under the situation where the student model has been migrated to become similar to the expert model, the student model is now *significant and suitable*. Hence, the inter-scheme rules will be not applicable. Therefore, the non-execution of intra- and inter-scheme rule means the classification scheme in the student model is now significant and suitable. This is the end point of the game and instruction process.

5.3 Rule Analysis

The analysis that can be performed on a set of code include the termination conditions, execution sequence and complexities. The following section demonstrates the possibilities of this analysis on the EpiList I rules. These are mathematically represented in terms of sets operations.

5.3.1 Lemmas

Lemma 1 *The `GeneraliseProperties()` function guarantees the student identifies the common property.*

This function primarily requires the student to perform generalisation and obtain the common property of the items presented. However, there can be a situation where the student is not able to perform generalisation. In this situation, the student is put through an elimination process. When the student selects the incorrect property for the first and second time, the system would advise him to read the detailed text carefully. During incorrect selection of the property for the third time, the system would highlight the mistakes and disable that choice. Therefore, a student, who is unable to perform generalisation, would eventually identify the correct common property through this elimination process of disabling the incorrect choices.

Lemma 2 *The rules and algorithms ensure that the student carries out the actions of correcting their grouping errors and migrating the classification schemes.*

The rules require the student to make inferences in order to determine the actions to undertake after the execution of the rules. In the scenario where the student is unable to make the correct inferences, similar rules will be executed endlessly. The algorithm is able to detect the repetitive use of the similar rule and instruct the student directly. Regardless of which the student comprehends the inferences or learns by rote, EpiList I's rules and algorithms ensure that the student will carry out the correct actions.

5.3.2 Achieving the Target Classification Scheme

An important analysis of ITS is the proof of student achieving the target learning outcome. The target of EpiList I is the student obtains the target classification scheme. With Lemma 1, it is certain that the student is able to perform generalisations and complete the tutorials. Lemma 2 states that the student eventually carries

out the actions instructed by the tutorials. Hence, with Lemma 1 and 2, the intra- and inter-scheme rules are able to guide the student; however, the guidance has to be analysed to ensure that the student would certainly achieve the targeted learning outcome.

Algorithm 1 Overall algorithm of EpiList I

```

while  $Sm \neq Em$  do {Repeat until the desired scheme is acquired}
  {The following codes highlight and correct all mistakes.}
  for  $i = 1$  to number of mistakes do
    while  $i^{th}$  mistake is not corrected do {Repeat until the mistake is corrected}
      if  $i^{th}$  mistake is not tutored before then
        Run Positive Exemplar Rule or Linear Positive Exemplar Rule or Non-
        linear Positive exemplar Rule
      else {mistake is tutored before}
        Run Negative Exemplar Rule or Linear Negative Exemplar Rule or Non-
        linear Negative exemplar Rule
      end if
    end while
  end for
  {The following codes migrate the student's classification scheme.}
  for  $j = 1$  to number of unimportant category in  $Sm$  do
    if  $j^{th}$  category is empty then
      Run Elimination Rule {Remove empty and unimportant categories}
    end if
  end for
  for  $k = 1$  to number of important category in  $Sm$  do
    while  $k^{th}$  category does not group all its item do
      Run Regrouping Rule {Ensure the important category groups all items}
    end while
  end for
  if Number of important category in  $Sm = 0$  then
    Run Positive Introduction Rule {Introduce the first important category}
  else
    Run Negative Introduction Rule {Introduce the next important category}
  end if
end while

```

Algorithm 1 is the overall algorithm of EpiList I. The rules of EpiList I focus its attention on one mistake at a time. Even though other mistakes with the same property are gathered together to be taught in the Non-linear Positive or Negative

Exemplar Rules, the algorithm would ensure that the student has to correct the mistake that is currently in focus. The highlighting of a mistake is first compared with positive examples which will directly indicate the correct category for the mistake. If this strategy fails to guide the student to correct the mistake, the system will contrast the mistake against negative examples to eliminate the incorrect categories until only the correct category left for the student to group the item. One mistake at a time, this sequence is repeated i times for i number of mistakes.

The next portion of the algorithm is set to migrate the student's classification scheme to a significant and suitable classification scheme. The first step of the migration is to choose empty categories to be removed. After that, existing categories that belong to the target classification scheme (important categories) are checked to ensure that they have grouped all the items that possess their properties. Finally the algorithm introduces categories of the target classification scheme to the student one at a time. These 3 steps are repeated until the target classification scheme is achieved. The introduction of important categories and removal of unimportant categories surely migrates the student to the target classification scheme.

It can be concluded that the rules and algorithms of EpiList I achieve the desired learning outcome.

5.3.3 Worst-case Scenario

In the scenario of p number of items in the *playlist* and n number of categories the student selects. Assume that the expert model consists of m number of categories.

In the worst-case scenario for intra-scheme rules, all items are categorised incorrectly. To make matters worse, the mistakes are made one at a time; hence, the mistakes will be also tutored one at a time. For each mistake, the 'positive exemplar' Intra-Scheme Rules will first be fired. If the student does not induce the correct answer, the 'negative exemplar' Intra-scheme Rules will be fired $(n - 2)$ times to eliminate incorrect categories leaving only the correct category. In total, the intra-scheme

rules will be fired $n - 2 + 1 = n - 1$ times to correct one mistake. Therefore for p number of items, the Intra-scheme rules will be fired $p(n - 1)$ times.

For Inter-scheme Rules, the worst-case is when none of the categories from Em is selected. The Positive Exemplar Introductory Rule will be fired to introduce the first category followed by $m - 1$ times of Negative Exemplar Introductory Rule for the remaining categories of Em . At the same time, at most p times of Regrouping Rule will be fired to move the items to the categories of Em , and at most n times of Elimination Rules will be fired to remove the categories in Sm that are not in Em . Therefore, the inter-scheme rules will be fired $1 + (m - 1) + p + n = m + p + n$ times.

In total, $p(n - 1) + m + p + n = n(p + 1) + m$ times of firing both intra- and inter-scheme rules in the worst case.

5.3.4 Complexities

Complexities include time and space complexity. Time complexity measures the characteristics of the program in terms of the execution time; whereas the space complexity measures the memory usage characteristic during execution time.

Time Complexity

Complexities are measured in the worst-case scenario. From the worst-case scenario as described in the previous section, there are p number of items, n number of categories the student selects and m number of categories in the expert model. In the worst case, the total number times the rules is fired is $n(p + 1) + m$. Hence the execution complexity of EpiList I is $O(np)$.

Space Complexity

Space Complexity is the characteristic of the memory usage during program execution. Let b be the number of bytes of memory necessary for one item. Therefore the

memory required by the main interface is pb bytes. Since only one rule will be fired at one time, the memory required is independent between tutorials. Therefore, only the worst-case tutorial scenario is considered. For this scenario, all items are used in the tutorial. This could be the result of categorising all items in one category. The “Non-linear Negative Exemplar Rule” will be fired using all the correctly categorised items in that category to contrast against the incorrectly categorised items in the same category. Hence, the memory needed for worst-case tutorial is also pb bytes. In total, the maximum memory required is $2pb$ bytes; hence, the space complexity of EpiList I is $O(pb)$.

5.4 Summary

Due to the fact that explanations for these cognitive processes are diverse, a verbose definition of EpiList I would cause ambiguous understanding. This chapter demonstrated a formal approach of defining EpiList I in terms of simple sets and set operations to overcome this problem. The formal definition provided detailed, precise and correct explanations of EpiList I’s cognitive process and instructional sequences. This resulted in a correct and unambiguous understanding of EpiList I and its learning outcome. It also showed that this formal definition enables analysis to be performed on the program. Due to the generic characteristic of simple sets, EpiList I is domain and platform independent.

Chapter 6

System Representation of EpiList I

“Creativity involves breaking out of established patterns
in order to look at things in a different way.”

– Edward de Bono

6.1 Background

In the research field of ITS, design and analysis are the major issues addressed over the years. The ad-hoc development of ITSs in the early years resulted in tedious and complicated analysis of the systems. Over the years, researchers have successfully partitioned the issues by addressing the problem at individual components of ITS. The components that the researchers focus on include 1) Knowledge Representation, 2) Student Modelling and 3) Pedagogical Rules. However, the individual analysis of individual components does not address the issues related to the whole system. When the components are placed together to form the system, the issues that arise include the communication protocol and functional support among the components, and system functionalities and analysis.

This research proposes a methodology to describe an ITS to enable analysis and design of a system. This methodology uses system theories, specifically the control system, to describe an ITS system. With ITS described in terms of system represen-

tation, a sound and well defined background of system and control theories can be applied to analyse and design the ITS.

6.2 Generic System Representation of an ITS

6.2.1 Fundamental of Control Systems

Sophistication Levels of Control System

A control system is a combination of elements or sub-systems that attempt to maintain a quantity or a set of quantities, termed output, suitably related to another quantity or a sets of quantities, termed input (Gupta, 1983). In general, a control system can be represented by Fig. 6.1.

Figure 6.1: Block Diagram of Control System

Through decades of research and development of control systems, they can be divided into 4 levels of sophistication; namely:

1. Open loop (Simplest),
2. Closed loop,
3. Adaptive, and
4. Learning.

These system structures are depicted by Fig. 6.2.

Figure 6.2: Sophistication Levels of Control System

The simplest and most economical type of control system, Fig. 6.2(a), is the open loop control system. In an open loop system, the reference input is fed directly to the controller; based on this input the controller provides the corresponding output to the process. However, in the situation of noise factors changing the characteristic of the process, the open loop control would fail to produce a desirable output from the process. Hence, the closed loop control systems were designed.

An open-loop control style of teaching is like an educational documentary. Information is constantly presented to the viewer without determining the amount of information the viewer has comprehended. Most of the early educational software transfers the knowledge to the student in an open-loop control manner. Although the student can control the flow of information of such system, the system does not monitor the learning outcome of the student.

In the closed loop control system, Fig. 6.2(b), some information from the output of the process is fed back to the controller. The signal to the controller is now a function between the reference input and the feedback signal. A closed loop system

is sensitive to external noise and internal variations of the system; and thus better in control of unstable systems.

All ICAI and ITS are closed-loop control systems. They all have some form of feedback mechanisms to monitor the student and tutor accordingly. The feedback signal could be from the interactions between the student and system or from the questions posed by the system to the student. Nevertheless, the system no longer ‘blindly’ tutors the student as in the open-loop system, it ‘intelligently’ tutors the student by constantly monitoring them.

However, both open loop and closed loop possess the same disadvantage of requiring a human operator to control or fine-tune the systems. The adaptive system is designed to replace the human operator. Furthermore, when the characteristics of the process vary drastically, the controller has to be more sensitive and adaptive. An adaptive controller is able to adapt itself in response to the changes in the process.

An adaptive educational system not only has to ‘intelligently’ tutor the student; it has to be incorporated with several teaching strategies or pedagogical models, such that the system can adapt to different students by using different teaching strategies. However, a more complicated monitoring is necessary to determine the learning preferences of the student and adapt the system to the student using correct teaching strategies.

An adaptive controller adapts to the process based on the current state of the process. A learning controller is very similar to an adaptive controller; the difference being the learning controller learns and adapts itself based on past experiences. The learning controller predicts the possible future states of the process and pro-actively reacts by adapting itself to these predicted states.

A learning educational system has most resemblance to that of a human teacher. It not only adapts itself to the student, it can also ‘learn’ new pedagogical models to improve and enhance itself to better educate the student.

Classification of Control Systems

A control system can be categorized into one or more of the following types:

- Static or Dynamic Systems

Variables of a static system remain unchanged over time while those of the dynamic system change with time. Almost all educational software and ITS are dynamic systems as the student that the system is controlling changes over time. The student motivations, interest, perspective and concentration changes as time passes.

- Continuous or Discrete Systems

A continuous system continuously feeds back the process output and controls the process in real time. The discrete system samples the process output and generates the control signals at predetermined intervals. An ITS could be both continuous or discrete. A continuous ITS would monitor every move made by the student while a discrete ITS monitors at a fixed interval or when initiated by the student.

- Single-input-single-output (SISO) or Multiple-input-multiple-output (MIMO) Systems

The SISO system has only one input and output while the MIMO system has several inputs and outputs. An ITS system that tutors one student at a time is a SISO system as it receives information regarding one student and instructs only that student. On the other hand, a collaborative learning system is a MIMO system. Instructions are given to several students based on the several inputs from the students.

- Linear or Non-Linear Systems

A system that obeys the principle of superposition is a linear system otherwise it is a non-linear system. Almost all ITS are linear systems. During an interaction

between the student and a system, the student has a mind-set to learn from the system. Therefore, in this state, the student is considered to be rational although it is human nature to be irrational. An ITS provides one tutorial for a mistake and another tutorial for another mistake. When both mistakes are to be taught, both tutorials would be activated. Hence, the ITS obeys the principle of superposition.

- Multi-level and multi-layer Control

Multi-level control, usually for complex systems, separates the process horizontally into several dependent or independent tasks with a controller for each task. Multi-layer control decomposes the process into sub-problems with a higher layer controlling the lower layers. An ITS could be both a multi-level and multi-layer control system depending on the teaching requirements.

Components of Control Systems

This section provides detailed definitions of the components of a control system (Gupta, 1983; Ogata, 1997). These definitions are necessary for the mapping of ITS to the control paradigm. Fig. 6.3 shows the components for a closed loop system. Closed loop control is adopted because it best describes the operations of most ITSs. Mapping of ITS to the closed loop control framework is intuitive and easily comprehended.

Figure 6.3: A closed loop control system

- Reference input (r): This is the actual input signal to the control system. It is the desirable value of the process output.
- Error sensor: This is the sensing module that determines the error (ε) or the relative difference between the reference input and process output.
- Error (ε): This is the input signal to the controller that measures the error of the output. The controller uses this to generate its output to drive the plant to a desirable value. The mathematical expression is $\varepsilon = r - \hat{y}$.
- Controller (G_1): The module that generates the manipulating signal (m) to control the plant. The controller module is denoted as G_1 in Fig. 6.3.
- Manipulated variable (m): This is the control signal from the controller. Hence, $m = G_1\varepsilon$.
- Plant: A *plant* is a piece of equipment that performs a particular operation. The name “Plant” is used interchangeably with “Process”. A process is a natural, progressively continuing operation. The difference between “plant” and “process” is that the former refers to a physical object while the latter refers to an operation. The plant module is denoted as G_2 .
- Disturbance (d): A disturbance is a signal that tends to adversely affect the output of a system. If a disturbance is generated within the system, it is termed internal disturbance; while an external disturbance is generated outside the system and is considered as an desirable input to the system.
- Output (y): The quantity that must be maintained at a prescribed value. This is the output of the plant, where $y = G_2m$.
- Feedback Element (H): This module provides feedback of the output (usually a function of the output) to be compared against the reference input. The feedback element is denoted as H .

- Feedback Signal ($\hat{y}$): This is the signal from the feedback element that is directly used with the reference signal to compute the error signal (ε). Hence, $\hat{y} = Hy$.
- Putting all the formulas together, the output (using super-position theory):

With respect to reference input (r) *With respect to disturbance (d)*

$$y = G_2 m \qquad y = G_2 m + d \qquad (6.1)$$

$$= G_2 G_1 \varepsilon \qquad = G_2 G_2 \varepsilon + d \qquad (6.2)$$

$$= G_2 G_1 (r - \hat{y}) \qquad = G_2 G_1 (-\hat{y}) + d \qquad (6.3)$$

$$= G_2 G_1 (r - Hy) \qquad = G_2 G_1 (-Hy) + d \qquad (6.4)$$

$$= \frac{G_2 G_1 r}{1 + G_2 G_1 H} \qquad = \frac{d}{1 + G_2 G_1 H} \qquad (6.5)$$

Therefore:

$$y = \frac{G_2 G_1 r}{1 + G_2 G_1 H} + \frac{d}{1 + G_2 G_1 H} \qquad (6.6)$$

$$= \frac{G_2 G_1 r + d}{1 + G_2 G_1 H} \qquad (6.7)$$

6.2.2 Representing ITS in terms of Control Theories

Analysis of ITS

The architecture of an ITS is precisely and clearly presented in Section 3.3. However, Fig. 6.4 presented the block diagram of the architecture of an ITS.

Figure 6.4: Block Diagram of ITS

Analysing the architecture of ITS in terms of a control paradigm shows that the student is essentially the ‘*plant*’ that the control system is attempting to control. That would essentially make the ‘*Manipulated Variables*’ equivalent to the ‘*Instructional Sequences*’ that is controlling or guiding the student. Tracing backwards from the instructional sequences to the ‘*Pedagogical Model*’ and ‘*Student Error*’, they are functionally equivalent to the ‘*Controller*’ and ‘*Error Signal*’ respectively.

The ‘*Student Error*’ is generated from the ‘*Student Model*’; however, the ‘*Student Model*’ is not exactly equivalent to the ‘*Error Sensor*’. The ‘*Student Model*’ is more than a mere ‘*Error Sensor*’, it also includes the translation of the student’s response to a representation that is understandable by the system. Hence, the ‘*Student Model*’ is more suitable to be mapped as the ‘*Feedback Element*’ and it generates the ‘*Student Representation*’ or the ‘*Feedback Signal*’ that can be used by the system.

Moving back to the ‘*Error Sensor*’; although the sensor cannot be directly mapped to any components in ITS, it is an important component for the control system. Hence, in the control representation of ITS, the error sensor would use the ‘*Student Representation*’ (as the ‘*Feedback Signal*’) and ‘*Instructional Goals*’ (as the ‘*Reference Signal*’) to generate the ‘*Student Error*’ (as the ‘*Error Signal*’).

Since the control paradigm is to represent the system and its functionalities, it

does not explicitly define the interface nor the domain knowledge. The interface would be implicitly defined in the ‘Instructional Sequences’; whereas, the domain knowledge is implicitly incorporated into the whole control system readily available for each component to use.

Mapping ITS to Control paradigm

Table 6.1 shows the corresponding mapping of components of an ITS to a closed loop system (Wong, 1998). It is also a summary of the analysis of an ITS in terms of control paradigm.

Closed loop system	ITS
Reference input (r)	Instructional Goal
Error (ε)	Student Error
Controller (G_1)	Pedagogical model
Manipulated variable (m)	Instructional Sequence
Plant (G_2)	Student
Output (y)	Student's Response
Feedback Element (H)	Student Model
Feedback Signal ($\hat{y}$)	Student Representation

Table 6.1: Mapping of ITS to Control Paradigm

Figure 6.5: System view of ITS

Fig. 6.5 shows the general system view of ITS after it is mapped to a closed loop control paradigm. Any ITS that conforms to the architecture shown in Fig. 6.4 can be

mapped to this general system view. The student model captures student responses and a representation of the student is generated. The representation is compared against the domain model and the instructional goal to determine the current error that the student has made. Based on the error, different instructional sequences are delivered to the students. Subsequently, responses from the student are captured and fed back to the system. This completes the closed loop control instruction for the student.

The components of an control system can be grouped into two main category: namely, the signals and modules. The former refers to the signals that is passed around in the system and it consists of the output (y), feedback signal ($\hat{y}$), reference input (r), error (ε), manipulated variable (m) as well as the disturbance (d). The latter refers to the components that map two signals or generate one signal based on another signal and it consists of the plant (G_2), feedback element (H), and controller (G_1).

In system theories, the signals are usually in terms of values. The value could be an integer, float or binary. When mapped to an ITS, these signals come in a form of vectors to represent the information. Different ITS uses different forms of values or even vectors in the vectors. Sometimes, several different formats are used within an ITS. Nevertheless, the formats describe the student's responses, instructional goals, instructional sequences and student error that corresponds to the outputs, reference inputs, manipulated variables and error respectively. Hence, in general, the vectors are in the following formats:

- $\hat{y} = \{rep_1, rep_2, rep_3, \dots, rep_i, \dots\}$

rep_i represents the i^{th} attribute of student that reflects part of the current status of the student. With several attributes, the system can have a complete understanding of the student.

- $r = \{goal_1, goal_2, goal_3, \dots, goal_j, \dots\}$

$goal_j$ represents one, j^{th} , of the goals of the system that is to be accomplished by the student. With all the individual goals defined, the vector r represents the learning outcome or instructional goals of the system.

- $\varepsilon = \{error_1, error_2, error_3, \dots, error_k, \dots\}$
 $error_k$ indicates an error, k^{th} , made by the student. Hence, ε represents all the errors made by the student. The errors are computed using the student's responses and instructional goals.
- $m = \{instruction_1, instruction_2, instruction_3, \dots, instruction_l, \dots\}$
 $instruction_1$ represents the l^{th} instructional sequence of the system. Therefore, m is the set of instruction sequences that are used by the system. One or more instruction sequences can be used at a time according to the errors committed by the student.

The modules in a control system; namely: the plant, controller, and feedback element, map the output of the module with its input with a transfer function. The transfer function mathematically defines the relationship between the inputs and outputs of the modules. When mapped to an ITS, the modules are the student, pedagogical model, and student model and these modules, except the student, use rules to map the inputs to the outputs. The rules in the student model, termed as *StudentModelRules*, generates the student representation based on the student's response. Whereas the rules of pedagogical model, termed as *ControllerRules*, gather the error signals and determine the instruction sequences used to tutor the student.

Although the student cannot be modelled with functions or rules, a predicate is used instead to express the student's reactions or responses to the instructions. This predicate, *StudentReactions*, simply indicates the student's responses to the instructions presented to them. Hence, the generic system representation of the rules are as follows:

- $\hat{y} = StudentModelRule(y)$

- $m = \text{ControllerRule}(\varepsilon)$
- $y = \text{StudentReaction}(m)$

The *Error Sensor* checks for errors in the student representation. In a closed loop control system, the error is defined as $\varepsilon = r - \hat{y}$. Similarly, the error sensor of an ITS operates on the Instructional goal and student representation to derive the student error. Hence, the error sensor of an ITS can be represented as $\varepsilon = \text{operation}(r, \hat{y})$. For different ITS, different types of operation are used to evaluate the student's error. The most common or popular operation is the “*overlay*” method (Goldstein, 1982). The overlay method performs a simple set intersection operation on the instructional goal (r) and student representation ($\hat{y}$) and subsequently derives the student error (ε) from the set intersection.

Combining the rules and signals above would gives the following:

$$\begin{aligned}
 y_i &= \text{StudentReaction}(m_i) \\
 &= \text{StudentReaction}(\text{ControllerRule}(\varepsilon_i)) \\
 &= \text{StudentReaction}(\text{ControllerRule}(\text{Overlay}(r, \hat{y}_i))) \\
 &= \text{StudentReaction}(\text{ControllerRule}(\text{Overlay}(r, \text{StudentModelRule}(y_{i-1}))))
 \end{aligned}$$

y_i is the student's reaction towards a set of instructional sequence, m_i , at an instant i . This set of instructions is based on the error of that instance, ε_i , which is subsequently computed by overlaying the instructional goal, r , with the student representation, $\hat{y}_i$. This student representation is generated from the student's reaction of the instance before the current instance; namely: y_{i-1} . The combined resultant formulae describes the changes of the student's reaction from a previous instance, $i - 1$, to a current instance, i . It defines the characteristics or instructional capabilities of the whole system.

“Disturbance” is always present in a student while using an ITS. The internal disturbance could be the lack of attention or concentration, and motivation (Kellogg, 1995; Solso, 1993). External disturbance includes interruption and an unfriendly environment. However, these disturbances are psychological aspect of the students. The effect of these disturbances is that student does not effectively learn from the instructional sequences. The control loop should be able to detect this problem and continuously tutor the student until he completes the lesson. Therefore, the disturbances to the student are not explicitly presented in the representation.

6.3 System View of EpiList I

6.3.1 Classification of EpiList I

According to ITS definitions, EpiList I can be classified as a problem-solving monitor and an ITS shell. A problem-solving monitor is a system that monitors, analyses, detects and corrects the “bug” or error of the student in a problem-solving process. EpiList I requires the student to create a significant and suitable classification scheme to correctly categorize the set of items provided. Throughout the classification process, EpiList I has two stages of monitoring. In the first stage, EpiList I monitors the classification error committed by the student; whereas during the second stage, EpiList I monitors the student classification scheme and guides him to discover the desirable and correct classification scheme.

As mentioned in the previous chapter, EpiList I is a domain and platform independent system. Hence, it satisfies the properties of an ITS shell. EpiList I at current stage of development, also qualifies an ITS shell where the Human-Computer Interfaces, strategies, instruction sequence and pedagogical model are static within the system.

When mapped to a control paradigm, EpiList I is a dynamic, discrete, MIMO,

linear, multi-level and single-layer system

- Dynamic System

The variables of EpiList I are the factors or attributes of the students. The attributes include perception, knowledge and intelligence. After using EpiList I for some time, the misconceptions of a student in the domain would be corrected. The student gains further knowledge, and an increase in intelligence by understanding the functions and rules of the system. Hence, the perception, knowledge and intelligence of the student would improve over time; however, the rate of improvement is dependent on the individual.

- Discrete System

The student activates the tutorials to begin the tutoring process. Subsequently, the student model captures the classification made by the student, analyses it and highlights the errors. Therefore, EpiList I does not continuously monitor the student and is considered a discrete system.

- MIMO System

A MIMO system has several inputs and outputs. The input of EpiList I is the instructional goal or the objective of the system. Since the objective requires the students to derive a significant and suitable classification scheme to categorize all items correctly and implicitly acquire the cognitive skills, EpiList I is a multiple (multi-objective) input system. The outputs of EpiList I are the attributes of the students which are the domain and cognitive knowledge. Hence, EpiList I is a MIMO system with the instruction goals for input and the student's attributes for output.

- Linear System

As mentioned above, a student is rational during an interaction with an ITS. Each tutorial (or an output sequence of the controller) of EpiList I is focused to

tutor one form of mistakes. When more than one form of mistake is present, the corresponding tutorials for each form will be presented. This follows the rule of superposition; hence, EpiList I is a linear system. As a linear system, analysis of a linear system, based on system theories, can be applied to EpiList I. The analysis includes identification of steady state and convergence analysis.

- Multi-level and Single-layer System

The rules and tutorials of EpiList I are grouped into intra- and inter-scheme rules. The intra-scheme rules govern the grouping process while the inter-scheme rules govern the classification scheme performed by the student. The inter-scheme rules are activated only after the set of intra-scheme rules; therefore, the inter-scheme rules are pegged at a level higher than the intra-scheme rules. Hence, EpiList I is a 2-level system with intra-scheme rules at the first level and the inter-scheme rules at the second level. At each level, there is only one set of rules or one closed-loop control to guide the student; hence, EpiList I is a single-layer system.

6.3.2 Mapping of EpiList I to Control Paradigm

To map EpiList I to a control paradigm, each component of the generic representation of ITS has to be defined individually and corresponding to parts of EpiList I. The followings are the definitions of the components:

1. The Plant, G_2

The plant is the student. As mentioned above, EpiList I is a 2-level control system; the first level controls the grouping while the second level controls the classification scheme. Therefore, the student can be viewed as performing two tasks; the first task is to correctly group all items and the second task is to derive the significant and suitable classification scheme. The two tasks are controlled by the intra-scheme and inter-scheme respectively. The tasks are denoted as S_1

and S_2 , as used in Fig. 6.6.

During the initial phase, EpiList I allows the student to create his or her own classification scheme. This is visualised with a switching mechanism. The switch is initially set to student's selection of classification scheme. The switch is triggered by the error signal of the intra-scheme. When the student completes the categorization on his or her classification scheme, the error signal would reflect no error which subsequently triggers the switch to point to the closed loop 2 and activates the inter-scheme rules.

2. Reference Input, r

Reference input is the instructional goal of the ITS. The instruction goal in EpiList I is to ensure that the student derives the significant and suitable classification scheme.

The objective of the intra-scheme rules of control loop 1 is to derive a complete classification. The student has to correctly group all the items. The reference input at this level is the domain model to ensure that all the categorizations are correct ($r_1 = Dm$, see Fig. 6.6). Therefore, the reference input r_1 is a vector of sets. Each set in the vector represents a category or property. The elements in a set represent the items that possess such a property and can be grouped in that category. Assuming that the domain model consists of o number of properties and $x_1, x_2, \dots, x_o$ represents the number of element in 1^{st} , 2^{nd} , $\dots$, o^{th} categories. The reference input, r_1 , of such domain is a vector of o number of sets and each set consists of x number of elements.

$$r_1 = \{\{e_{11}, e_{12}, e_{13}, \dots, e_{1x_1}\}, \{e_{21}, e_{22}, e_{23}, \dots, e_{2x_2}\}, \dots, \{e_{o1}, e_{o2}, e_{o3}, \dots, e_{ox_o}\}\}$$

The objective of the inter-scheme rules of control loop 2 is to induce in the student the choices of the proper selection through the tutorial interaction of a significant and suitable classification scheme. The expert model defines the significant and suitable classification scheme. It is the reference input at the second level control ($r_2 = Em$, see Fig. 6.6); therefore, r_2 is a vector of groups.

Assuming the expert model consists of q number of categories that make up the significant and suitable classification scheme. The reference input, r_2 , of such an expert model is a vector of q number of elements of which each element represents a category.

$$r_2 = \{group_1, group_2, \dots, group_q\}$$

3. Feedback Element and Signal, H and $\hat{y}$

From the previous section, the feedback element and signal is the student model and student representation respectively. The student model of EpiList I is not explicitly presented and described; however, the Student model (Sm), or the student representation, is clearly presented in Section 5.2.1 of Chapter 5. The student model should generate Sm according to the list-making interface when the student initiates the tutorials.

The student model or the feedback elements are denoted as H_1 and H_2 . The feedback signal, $\hat{y}_1$, or student representation of H_1 is a vector with similar syntax to r_1 ; while $\hat{y}_2$, the feedback signal for H_2 , is similar to r_2 . The vector of $\hat{y}_2$ could be extracted from $\hat{y}_1$, since $\hat{y}_2 = Group(\hat{y}_1)$. Therefore, $\hat{y}_1 = H_1(y)$ and $\hat{y}_2 = H_2(y)$.

4. Error Sensor and Signal, ε

The error sensor is the component that checks for errors. The operation of error sensing of EpiList I uses the *overlay* method. The overlay method takes the intersection of two models to derive the relationship between the two models. The sensing component of the first level overlays the Student model (Sm) with the Domain model (Dm) to determine the grouping errors; while the second level overlays Sm with the Expert model (Em) to determine the classification scheme errors of the students.

The overlaying performed in the first level extracts the mistakes made during grouping of items. Hence, the error signal of the control loop 1, ε_1 , is essentially

a vector of 2 sets; one set is the correctly grouped items while the other set is the incorrectly grouped items. In a scenario where the student correctly groups $item_1, item_2, item_3$ and incorrectly groups $item_7, item_8, item_9$. The error signal would be as follows:

$$\varepsilon_1 = \{\{item_1, item_2, item_3\}, \{item_7, item_8, item_9\}\}$$

The error sensor of the second level performs the intersection of the Sm and the Em to determine the correctness of the classification scheme. The error signal of this control loop is also a vector of 2 sets, however, the sets are vectors of categories which represent important categories that are selected by the student. Therefore, in the scenario where the important categories are $Group_1, Group_2, Group_3$ and the student selects $Group_2$ in his classification scheme; the error signal, ε_2 , would be as follows:

$$\varepsilon_2 = \{\{Group_2\}, \{Group_1, Group_3\}\}$$

5. Controller, G_1

When mapped onto the control paradigm, the pedagogical model in an ITS is the controller while the instructional sequence is the manipulated variable. The pedagogical model of EpiList I contains the rules of EpiList I. The actions of the rules are the instructional sequences. The controller takes in the error sensor's signal and decides the appropriate instructional sequences for the student. Based on the conditions or situations indicated by the error signals, the controller decides the sequence or rule to activate to tutor the student. This decision is made by the algorithm of the intra- and inter-scheme rules.

The controller or pedagogical models for the intra-scheme rules and inter-scheme rules are denoted as P_1 and P_2 respectively. Subsequently, the instructional sequences are represented as m_1 and m_2 respectively. Therefore, $m_1 = P_1(\varepsilon_1)$ and $m_2 = P_2(\varepsilon_2)$.

6.3.3 System Representation of EpiList

Figure 6.6: System View of EpiList

Fig. 6.6 shows the operations of EpiList I as a control structure. The figure shows the two tasks required of the student and the two closed loop control systems dedicated to each of the tasks. In addition, the switch that selects the mode of classification scheme control is also depicted. Initially, the switch is set at A to allow student to select his classification scheme. With this scheme the first level control loop would ensure that the student correctly groups all items. When that happens, the error signal, ε_1 , would indicate that no error is detected, which is the trigger condition for the switch to activate the second control loop.

When the switch is turned to B, the classification scheme of the student is now controlled by the second level control system. Subsequently, the second level control system would guide the student to migrate to the targeted classification scheme, r_2 , while the first level control system continues to maintain correct categorization at all

times.

6.4 System Analysis and Limitations of EpiList I

6.4.1 Convergence of the System Steady State

In control theory, the convergence of a system refers to the reduction of the error signal till no error is detected. Such reduction is brought about by controlling the plant to produce an output and feedback signal that is similar to the reference input; that is to say, the controller has successfully controlled the system to achieve its desired output. Proving that a system would converge is an important step to prove the correctness and stability of a system.

In terms of ITS, the convergence of an ITS system representation refers to the student attaining the desired learning outcome as indicated by the reference input (or instructional goals). By showing that the system will converge to the instructional goals; it proves that the pedagogical design or instructional sequences are correct and capable of tutoring the student to acquire the targeted knowledge.

In the following sections, the convergence of EpiList I is clearly presented.

First level control loop

The first level control loop of EpiList I focuses on tutoring incorrectly grouped items. The steady-state of control loop 1 would be the correct grouping of all items. The transfer function for the first level control loop is as follows:

$$\begin{aligned}
 y &= S_1(m_1) \\
 &= S_1(P_1(\varepsilon_1)) \\
 &= S_1(P_1(\text{overlay}(r_1, \hat{y}_1))) \\
 &= S_1(P_1(\text{overlay}(r_1, H_1(y))))
 \end{aligned}$$

' y ' would reach steady-state value when $\hat{y}_1$ converged to r_1 . The distance or difference between $\hat{y}_1$ and r_1 denotes the mistakes made by the students in their groupings. Hence, the convergence of the control loop 1 to the student's selected scheme is dependent on the proper rectification of the categorization mistakes. The intra-scheme rules, P_1 , would highlight to the students the mistakes made by them and subsequently attempt to guide them towards the desired classification.

The intra-scheme rules use generalization and comparisons to highlight the mistakes. The comparisons made between the mistakes and positive or negative examples indicate the correct or incorrect class respectively of the mistakes. The rules would first attempt to use similar examples to highlight the correct class of the mistakes to the student. If the student fails to correct the mistakes, the rules will proceed to use different examples to guide the student in an *elimination* process. The elimination process is the most fundamental method of pruning incorrect options one at a time, leaving the final option as the solution. Therefore, for each mistake involving n classes, the student would encounter at most $n - 1$ times of intra-scheme rules that use negative examples to eliminate the incorrect classes.

Lemma 1 and 2 from Section 5.3.1 ensure the successful execution and elimination process of the rules. Hence, the instructions of the intra-scheme rules will eventually guide the student to rectify his mistakes. Thus, the student could attain the instructional goals of the first level control loop of EpiList I. This shows that $\hat{y}_1$ will converge to r_1 .

Second level control loop

The second level control loop of EpiList I focuses on the acquisition of a significant and suitable classification scheme for an instance of the system. The steady-state of this loop is the student acquiring the targeted classification scheme. The transfer

function for the second level control loop is simplified as follows:

$$y = S_2(P_2(\text{overlay}(r_2, H_2(y))))$$

The distance or difference between $\hat{y}_2$ to r_2 is the degree of dissimilarity between the targeted and the student's classification scheme. To show that $\hat{y}_2$ converges to r_2 , the distance has to be decreased by introducing the target classification scheme to the student. The inter-scheme rules, P_2 , guide the student to migrate from their classification scheme to the target classification scheme to reduce the dissimilarity between the two schemes.

The inter-scheme rules perform the migration process by introducing the classes of the targeted scheme to the student with the *Introductory* rules; at the same time removing the undesirable classes from the student's selected classification scheme with the *Elimination* rule. Another important tutorial for the migration is guiding the student to group items into categories of the target classification scheme. This is the *Regrouping* rule.

In a scenario where there are m number of items and x number of categories in the target classification scheme of this instance of scenario. The worst case is that the student selects none of the categories from the target scheme in his own classification scheme. The inter-scheme rules are required to execute the introductory rules x times to introduce the categories of the target classification scheme and the elimination rules y times to remove the categories of his classification scheme. At the same time, the regrouping rule is executed m times to move the items into the target scheme.

Lemma 2 is also applicable to inter-scheme rules to ensure that the student carries out the migration of schemes as instructed in the tutorials. Thus, the inter-scheme rules could guide the student to migrate to the target classification scheme; hence, $\hat{y}_2$ would converge to r_2 .

Open cognitive control loop

The tutorials of EpiList I employ the processes of generalizations and comparisons to highlight the student's error. Hence, through the use of EpiList I, it can be seen that the student implicitly acquires the generalization and comparison skills.

“Implicit learning has been characterized as an subconscious process by which knowledge is acquired without deliberate attempt to learn. In contrast, explicit learning, occurs when there is active engagement in conscious strategies to discover the rules underlying a task (O'Brien-Malone & Maybery, 1998).” Thus, instructions that require the student to perform implicit or explicit learning are denoted as implicit or explicit instructions respectively.

Figure 6.7: Open loop cognitive control system

With reference to the implicit learning of cognitive skills, the instructions are directly presented to the student without any feedback. Thus, implicit cognitive teaching is an open loop control system, as shown in Fig. 6.7, where there is no assurance by the system that the student has acquired the desired cognitive skill competency.

6.4.2 Limitations of EpiList I

Limitations of Open loop Cognitive Instructions

It is widely known in classical control theory that an open loop control system possesses severe limitations. In a dynamic system, an open-loop control system will not adequately perform the desired task (Ogata, 1997). Since EpiList I is a dynamic

system, one cannot assume that the student is able to acquire the necessary generalization and comparison skills implicitly. The lack of proper generalization and comparison skills hinders the student's ability to correct categorization errors.

Another limitation of open loop control system is structural instability (Gupta, 1983). That is, slight changes to the process would cause an open-loop control system to have a large error. For cognitive control in EpiList I, if the students misinterpreted the process of generalization or comparison, then the application of cognitive control might lead to poorer performance of the students.

Limitation of Implicit Cognitive Instruction

Implicit learning is time consuming. Long hours are needed for the students to implicitly learn the cognitive skills in EpiList I. Explicit learning, which requires less time, is desirable.

Implicit learning is also highly sensitive to individual differences in intelligence (Kirsner et al., 1998). Due to this fact, the effectiveness of implicit instruction in EpiList I varies greatly from one student to another. Hence, using explicit instructions can enhance the overall average efficiency of EpiList I.

6.5 Summary

This chapter first demonstrates the representation sequence or process to formally represent an ITS within a control paradigm. This enables the analysis of the ITS systems using control theories. This control representation methodology is used to map the architecture of EpiList I as a control representation. The control representation of EpiList I, as shown in this chapter, is a two level closed loop control over the categorisations and classification scheme of the student, as well as a third level open loop control over the cognitive skills.

This chapter further examines open loop control. EpiList I develops the general-

isation and comparison skills in an open loop manner. In another words, EpiList I develops these skills of the students without monitoring the student's improvement on these skills. This is a limitation of EpiList I. It monotonously involves the students in the same sequences of generalisations and comparisons without ensuring that the students acquire these skills because the primary educational objectives of EpiList I is the grouping of items and classification schemes. Closing the open loop control of the cognitive skills in EpiList I will overcome the limitation. By monitoring the students' skills and feeding the information back to the system, EpiList I can determine the students' deficiency in these skills and tutor them accordingly; ensuring that they can properly learn and acquire the skills. This proposed system with the closed loop cognitive skills instruction is EpiList II, which will be discussed in greater detail in the following chapters.

Part III

EpiList II and A-EpiList

Chapter 7

EpiList II

“The books that help you most are those which make you think that most.”

– Pablo Neruda

7.1 Background

The control analysis of EpiList I highlights the open loop instructions for cognitive skills. This also raises the issues as well as limitations of EpiList I. A solution is to form closed loop instructions for the cognitive skills in EpiList I. EpiList II is an extension of EpiList I that comprises a higher level to form a closed loop encapsulating EpiList I.

The cognitive skills covered in EpiList I are classification, generalization and comparison skills. These are monitored by the third level control of EpiList II. The third level control not only monitors the skills, it also includes the teaching and development of these cognitive skills. Hence, meta-cognitive teaching theories or models would be incorporated at this level of control to facilitate such instructions. The design of EpiList II presented in this chapter would focus on the acquisition of generalization and comparison skills.

In the research field of ITS, design and analysis of systems are major issues addressed over the years (Vasandani & Govindaraj, 1990; Ng et al., 1995). The ad-hoc

development of ITSs in the early years resulted in tedious and complicated analysis of the systems. The previous chapter shows that the analysis methodology employed by EpiList I is based on the representations and theories of sound system theory. This methodology is also used to design EpiList II and to demonstrate its capabilities in assisting the design of ITS.

7.2 Formal Refinement of Cognitive Skills

A cognitive skill consists of several definitions as well as difficulty levels. This section attempts to formally define the cognitive skills of EpiList I to comprehend concisely the skills involved. This section also presents the different levels of the cognitive skills in terms of their difficulties as well as the formal method to determine them.

7.2.1 Generalisation Skills

The definition of generalization from the Webster Dictionary is “*the act of bringing individuals under a genus or class*”. This definition is the objective of performing the act of generalization. The process of generalization involves firstly the determination of the classes that each item is to be grouped under, and subsequently to determine the appropriate classes that are able to categorise all the available items. The formal definition of generalization is

$$\begin{aligned} GeneralizeProperties(\{e_1, e_2, \dots, e_n\}) &= prop(e_1) \cap prop(e_2) \cap \dots \cap prop(e_n) \\ &= \{G_{D_\alpha}, G_{D_\beta}, \dots, G_{D_\theta}\} \end{aligned}$$

where *GeneralizeProperties(a set of items)* returns a set of properties or groups that is obtained by generalizing the set of items, $\{e_1, e_2, \dots, e_n\}$. Refer to Section 5.2.1 for the definition of *prop(e_i)*.

In cognitive load theory, “the term cognitive load refers to the total amount of mental activity imposed on the working memory at an instance of time. The major factor that contributes to cognitive load is the number of elements that needs to be attended to.” The human working memory has a threshold for the amount of activity or elements. When the cognitive load is above this threshold, any learning process will be difficult. Although, “the measure used for cognitive load does not equate mathematically to the task difficulty” (Cooper, 1998), it provides a good and clear evidence on the difficulty of the task performed.

Based on such a measure of cognitive load, there are 3 levels of difficulty for performing generalization; namely, *simple*, *moderate* and *difficult generalizations*. This categorisation of generalisation skills is also found in Section 4.4.2 which also includes the definitions and explanations of the refined rules. The following sections will formally define the different levels of the skill along the approach based on the formal representation method proposed in Chapter 5.

Simple Generalisation Skills

In simple generalization, the items to be generalized are from the same group. Therefore the items are very similar to each other and almost all properties of an item are common throughout the set of items. The cognitive load of simple generalization is relatively small where it involves only properties of an item. Hence, a generalization is defined as a simple generalization if and only if the number of generalized classes is more or less the same as the average number of classes of the items.

Formal definition :

$$\begin{aligned}
 & [|GeneraliseProperty(\{e_1, e_2, \dots, e_n\})| \approx 1/n \sum_{i=1}^n |prop(e_i)|] \\
 & \leftrightarrow [GeneraliseProperty(\{e_1, e_2, \dots, e_n\}) \text{ is a simple generalisation.}]
 \end{aligned}$$

Difficult Generalisation Skills

This generalization involves items from several different groups. Due to its diversity, only at most one common property exists. Therefore, generalization of these items to identify the single common property is very difficult. The cognitive load is also the highest with the greatest number of properties involved in this kind of generalization. Hence, a generalization is a difficult generalization if and only if the number of generalized classes is one.

Formal Definition :

$$[|GeneraliseProperty(\{e_1, e_2, \dots, e_n\})| = 1]$$

$$\leftrightarrow [GeneraliseProperty(\{e_1, e_2, \dots, e_n\}) \text{ is a difficult generalisation.}]$$

Moderate Generalisation Skills

In this form of generalization, items are derived from only two different or opposite groups. Although the items are quite different, there are still some similarities among them. Therefore, some common properties are still present for this set of items. Hence, the difficulty in locating these common properties would be more difficult than simple generalization but less difficult than difficult generalization. The cognitive load of this level of generalization is in between simple and difficult generalizations. Hence, a generalization is considered a moderate generalization if and only if the number of generalized classes is between simple and difficult generalization.

Formal Definition :

$$[1 < |GeneraliseProperty(\{e_1, e_2, \dots, e_n\})| < 1/n \sum_{i=1}^n |prop(e_i)|]$$

$$\leftrightarrow [GeneraliseProperty(\{e_1, e_2, \dots, e_n\}) \text{ is a moderate generalisation.}]$$

7.2.2 Comparison Skills

The objective of comparison is to identify the similarities and differences between two or more items. Hence, the identification of similarities is named as comparing sub-skill while the identification of differences is named as the contrasting sub-skill.

Comparing sub-skill

The comparing sub-task is the identification of similar properties among two items or two sets of items. The properties identified are actually the common properties of the items. Therefore the comparing sub-task is essentially the generalization process. The following is the formalisation of the comparing sub-skill.

Compare of 2 items:

$$\begin{aligned} \text{Comparing}(e_1, e_2) &= \text{prop}(e_1) \cap \text{prop}(e_2) \\ &= \{G_{D_\alpha}, G_{D_\beta}, \dots, G_{D_\theta}\} \end{aligned}$$

Compare of 2 sets of items:

$$\begin{aligned} &\text{Comparing}(\{e_1^1, e_2^1, \dots, e_n^1\}, \{e_1^2, e_2^2, \dots, e_n^2\}) \\ &= \text{GeneraliseProperty}(\{e_1^1, e_2^1, \dots, e_n^1\}) \cap \text{GeneraliseProperty}(\{e_1^2, e_2^2, \dots, e_n^2\}) \\ &= \{G_{D_a}, G_{D_b}, \dots, G_{D_c}\} \cap \{G_{D_x}, G_{D_y}, \dots, G_{D_z}\} \\ &= \{G_{D_\alpha}, G_{D_\beta}, \dots, G_{D_\theta}\} \end{aligned}$$

The cognitive load of comparison is the same as the cognitive load of generalization provided in the previous section. Likewise, the cognitive load of comparing is also inversely proportional to the number of common properties. The following is the

formalised definition of the difficulty levels of comparing 2 items or 2 sets of items.

Formal definition :

$$Comparing(A, B) \text{ is } \begin{cases} \text{simple,} & \text{if } |Comparing(A, B)| \approx \frac{1}{n} \sum_{i=1}^n |prop(e_i)| \\ \text{moderate,} & \text{if } 1 < |Comparing(A, B)| < \frac{1}{n} \sum_{i=1}^n |prop(e_i)| \\ \text{difficult,} & \text{if } |Comparing(A, B)| = 1 \end{cases}$$

where e_i are the elements in the sets A and B .

Contrasting sub-skill

The contrasting sub-task is the identification of the different properties between a set of items. The identification process begins with the determination of the properties of each item, and subsequently determines the properties that are not common properties of the set of given items. The following is the formalisation of the contrasting sub-skill.

Contrast of 2 items:

$$\begin{aligned} Contrasting(e_1, e_2) &= [prop(e_1) \cap prop(e_2)]^{-1} \\ &= prop(e_1)^{-1} \cup prop(e_2)^{-1} \\ &= \{G_{D_\gamma}, G_{D_\delta}, \dots, G_{D_\eta}\} \end{aligned}$$

Contrast of 2 sets of items:

$$\begin{aligned} Contrasting(\{e_1^1, e_2^1, \dots, e_n^1\}, \{e_1^2, e_2^2, \dots, e_n^2\}) \\ &= [GeneraliseProperty(\{e_1^1, e_2^1, \dots, e_n^1\}) \cap GeneraliseProperty(\{e_1^2, e_2^2, \dots, e_n^2\})]^{-1} \\ &= [\{G_{D_a}, G_{D_b}, \dots, G_{D_c}\}]^{-1} \cup [\{G_{D_x}, G_{D_y}, \dots, G_{D_z}\}]^{-1} \\ &= \{G_{D_\gamma}, G_{D_\delta}, \dots, G_{D_\eta}\} \end{aligned}$$

When two items were very similar, most of the properties are common to both items. The set of unique properties formed from the first step would contain more common properties and few non-common properties. Therefore, the search for non-common properties causes higher cognitive load. Hence, the cognitive load of contrasting sub-task is proportional to the number of common properties. The following is the formal definition of the difficulty levels of the contrasting sub-skill.

Formal definition :

$$\text{Contrasting}(A, B) \text{ is } \begin{cases} \text{simple,} & \text{if } |\text{Contrast}(A, B)| = 1 \\ \text{moderate,} & \text{if } 1 < |\text{Contrast}(A, B)| < \frac{1}{n} \sum_{i=1}^n |\text{prop}(e_i)| \\ \text{difficult,} & \text{if } |\text{Contrast}(A, B)| \approx \frac{1}{n} \sum_{i=1}^n |\text{prop}(e_i)| \end{cases}$$

where e_i are the elements in the sets A and B .

Types of comparison skills

Comparison occurs most often during the execution of the intra-scheme rules of EpiList I. The positive exemplar rules make use of the comparing sub-skill whereas the negative exemplar rules make use of the contrasting sub-skill.

The formal definition of the comparison skills presented in the previous section involves either two items or two sets of items. However, these are not the only type of comparison. A third form of comparison involves one item against a set of items. Therefore, in total, there are 3 types of comparison skills in conjunction with the 3 levels of difficulty.

The first type is the comparison of one item against another item. The properties of the items are used directly without any execution of generalisation. This type of comparison will be termed as the *one-to-one* comparison. The second type involves one item against one set of items. For this type of generalisation, the set of items is generalised to derive its common properties which subsequently are used to compare

against the properties of the item. This type of comparison is termed as *one-to-group* comparison. Beside this definition of deriving the common properties among the item and the set of items, an alternative way of describing the one-to-group comparison is to determine if the item can be assimilated into the set of items. This approach allows one to determine if the item *possesses* the common properties of the set of items. The last type of comparison compares two set of items; hence, it is termed as *group-to-group* comparison. Group-to-group comparison requires two generalisations, one for each set of items.

These three types of comparisons; namely, one-to-one, one-to-group, and group-to-group comparison, consist of different difficulty levels. Each comparison type can be further refined to be simple, moderate or difficult. Hence, the combination of types and difficulty levels are shown in Table 7.1.

Types of comparison	Difficulty levels		
	Simple	Moderate	Difficult
one-to-one	Simple one-to-one comparison	Moderate one-to-one comparison	Difficult one-to-one comparison
one-to-group	Simple one-to-group comparison	Moderate one-to-group comparison	Difficult one-to-group comparison
group-to-group	Simple group-to-group comparison	Moderate group-to-group comparison	Difficult group-to-group comparison

Table 7.1: Combinations of types and difficulty level of comparison skill

The horizontal axis of Table 7.1 represents the difficulty levels in ascending order of difficulties from left to right. The vertical axis represents the types of comparison in the order of one-to-one, one-to-group, and group-to-group. Each intersection point within the table is a combination of a difficulty level and a comparison type, yielding

a possible tutorial interaction in EpiList II for explicit instruction of comparison skill. For example, the top-left intersection produces the combination of “*Simple one-to-one comparison*”. Hence, the 9 combinations produced are clearly presented in Table 7.1 outlining 9 possible tutorial interactions.

7.3 System Design of EpiList II

Figure 7.1: System view of EpiList II

Fig. 7.1 depicts the system design of EpiList II. The limitations of EpiList I are overcome using a closed-loop cognitive instruction by wrapping EpiList II over EpiList I. Hence, as shown in the figure, EpiList II consists of a third level of closed-loop instructions of the cognitive skills to the student. This third level of closed-loop control would explicitly monitor, instruct and develop the cognitive skills in the student. The following sections present the detailed design of the third closed-loop instruction.

7.3.1 Student Modeler, H_3 , and Student Representation, $\hat{y}_3$

The fundamental cognitive skills that EpiList I employs are the generalization and comparison skills. From Section 7.2, both generalisation and comparison skills can be further refined into 3 difficulty levels and types. The cognitive closed-loop is required to monitor the student's competency in generalisation and comparison skills.

As mentioned in the previous section, the generalisation skill can be refined into simple, moderate and difficult generalisations. The student modeler has to monitor all the three levels of difficulties in performing the generalisation skills. In order to perform moderate generalisation, the student must first be able to perform the task of simple generalisation. Similarly, difficult generalisation requires the ability to perform moderate generalisation. Hence, there is a hierarchy of importance and pre-requisites for the skills. The student representation of the skills should not only reflect the competency of the skills but also the importance of the hierarchy of the skills.

The monitoring of comparison skills should also be refined according to the difficulty levels. Furthermore, the set of comparison skills also consists of 3 different types for each difficulty level. This hierarchy of importance in terms of difficulties is critical to the acquisition and tutoring of the cognitive skills. On the other hand, the types of comparison skills are not interdependent. The ability to perform one type of comparison skills at a simple level does not necessarily demonstrate the capability to perform the same type of comparison at a moderate or difficult level. On the other

hand, the ability to perform one type of comparison at a simple level does show that the student is capable of performing another type of comparison at the same difficulty level. Therefore, the student modeler only monitors the different difficulty levels but not the different types of comparisons.

Also mentioned above, comparison skills can be separated into comparing and contrasting sub-skills. The comparing sub-skill is the identification of similarities and the contrasting sub-skill is to identify differences. In Chapter 4, it has been argued that the preference to identify similarities is inborn in humans. Therefore, the student modeler should place the comparing sub-skill at a higher hierarchical level than the contrasting sub-skill. Both comparing and contrasting sub-skills, as shown in the formalisation of these skills, operate on top of generalisation; hence, generalisation has the highest priority among comparison skills.

Other than the generic primitive cognitive skills; namely: generalisation and comparison skills, EpiList I also implicitly imparts reasoning skills to the student. Hence, EpiList II would explicitly monitor the student's reasoning skills within a closed-loop system and explicitly instruct the student on the acquisition of such reasoning skills. In terms of hierarchical importance, all the primitive cognitive skills are crucial in the reasoning process. Therefore, the primitive cognitive skills should be placed ahead of the reasoning skills. The reasoning skills attempted in EpiList I requires the students to reason for themselves about their own mistakes, and there are two forms of reasoning employed; namely: positive and negative reasoning. The positive reasoning comprises of those rules that use positive exemplars while the negative reasoning consists of those rules that use negative exemplars.

As a whole, the student modeler, H_3 (see Fig. 7.1), has to monitor *eleven* skills that are employed in EpiList I. The eleven skills are 1) 'simple generalisation', 2) 'moderate generalisation', 3) 'difficult generalisation', 4) 'simple comparing', 5) 'moderate comparing', 6) 'difficult comparing', 7) 'simple contrasting', 8) 'moderate contrasting', 9) 'difficult contrasting', 10) 'positive reasoning', and 11) 'negative reasoning'.

Furthermore, the sequence priority is given due consideration in term of cognitive importance among the eleven skills. The student modeler has to constantly monitor the actions undertaken in EpiList I to determine the correct and incorrect performance of the skills.

These skills need to be explicitly represented in order to be understood by the system. WUMPUS (Goldstein, 1982) represents its skills set in terms of "used-bit". Whenever the system detects the student using a particular skill, it would set the "used-bit" for that skill. WUMPUS highlights the limitations of this representation. That is:

- When a skill is not used, it does not implies the incompetence in that skill.
- A single instance of correct usage of a cognitive skill does not allow one to infer the competence in that cognitive skill. This could be due to true competence or merely luck.
- A single instance of incorrect usage of a cognitive skill does not allow one to infer the incompetence in that skill. This could be due to true incompetence or a careless mistake.
- An equal number of correct and incorrect usage of a cognitive skill results in uncertainty over the competency in the skills.

To overcome the limitations of WUMPUS skills representation method, instead of a single "bit" to keep track of the usage of a particular cognitive skill, a "counter" is employed. The counter traces the number of correct and incorrect usage of the cognitive skill. A correct use increases the counter while an incorrect use decrements it. This trace counter, to be used in EpiList II, is termed as the 'UseCounter'. The UseCounter is essentially the student representational vector, $\hat{y}_3$. The UseCounter is a vector of eleven integer values with each value representing one skill.

Figure 7.2: An example of UseCounter

Fig. 7.2 shows an example of the UseCounter. Whenever a skill is performed correctly, the corresponding value of the UseCounter is incremented. The value would be decremented if the skill is performed incorrectly. For example, when the student correctly performs positive reasoning, the 10th integer value, which represents such a skill, will be incremented by one. Therefore, if the third value, which is corresponding to difficult generalisation, is 3; it indicates that the student performed difficult generalizations 3 times more than incorrect performance on that skill. On the other hand, if the value is negative as the ‘Moderate Contrasting’ in Fig. 7.2, this indicates that the student incorrectly performed moderate contrasting 2 times more than correct performance on that skill. Hence, a high negative value in UseCounter reflects an incompetence of the student in using that corresponding skill. The following is the formal definition of the UseCounter.

Formal definition :

$$\text{If } UseCounter[i] \left\{ \begin{array}{l} > 0, \text{ The student has correctly performed cognitive skill} \\ & i \text{ } UseCounter[i] \text{ times more than incorrectly used.} \\ = 0, \text{ The student either has not performed cognitive skill} \\ & i \text{ or has equal number of correctly and incorrectly} \\ & \text{attempts in that skill.} \\ < 0, \text{ The student has incorrectly performed cognitive skill} \\ & i \text{ } UseCounter[i] \text{ times more than correctly used.} \end{array} \right.$$

Using this UseCounter, the issues of “use-bit” in WUMPUS are overcome. As the UseCounter keeps track of the student’s performance, it can precisely determine the competency and incompetency of the student in cognitive skills. However, the UseCounter is unable to distinguish the difference between equal and zero usage of cognitive skills. Therefore, other method is necessary to correctly identify the cognitive competency level of the student. This UseCounter is passed to the error-sensor to be used to compute the error signal, ε_3 .

7.3.2 Target Cognitive Skills, r_3

The target cognitive skills, denoted as r_3 , are the desired values for the UseCounter vector. Therefore, it is also a vector with the same number of values and this vector is termed as the ‘WantedValue’. Fig. 7.3 shows an example of the WantedValue.

Figure 7.3: An example of WantedValue

The first element of the vector wanted-value, as shown in Fig. 7.3, represents simple generalization and has a value of 7. This means that the student had to correctly perform the simple generalization at least 7 times more than the incorrect performance of the skill.

From the previous section, the cognitive skills have different levels of importance. The WantedValue vector can also be used to provide different levels of difficulty for the curriculum. A skill with higher importance requires more practice; hence, the corresponding value in the WantedValue vector is allocated with a higher value. A higher WantedValue value ensures sufficient practice for the student and enforces a greater confidence that the student is competent in the corresponding cognitive skill.

Therefore, a higher WantedValue reflects the greater importance of a cognitive skill. However, a consistent increase of all the skills in WantedValue has different consequences. This increment requires the student to show a higher competency in the skills; hence, increasing the difficulties of the system. In this manner, the difficulties of the curriculum in the system can be controlled. A WantedValue with low overall values indicates a simple curriculum while a WantedValue with high overall values reflects a difficult curriculum. A variation of highs and lows in the WantedValue also provides a variety in the curriculum that would focus on different cognitive skills if necessary. Referring to the example shown in Fig. 7.3, generalisation skills have a

higher overall value than the comparison skills. As mentioned before, generalisation skills are more important than comparison skills; hence, the values corresponding to the generalisation skills in the WantedValue vector have a higher value. This demonstrates the flexibilities of the curriculum as controlled through setting of the WantedValue.

7.3.3 Error sensing and Error signal, ε_3

The error sensor at the third level closed-loop instruction in EpiList II uses the WantedValue and UseCounter to compute the error signal, ε_3 . Hence, the error signal is a vector representing the competency of the student in performing the corresponding cognitive skills. This vector of error signals is termed as the 'ErrorValue'. The ErrorValue is computed by simply subtracting the UseCounter from the WantedValue.

Formal definition :

$$ErrorValue[i] = WantedValue[i] - UseCounter[i] \text{ where } 1 \leq i \leq |Skills| \quad (7.1)$$

Using the WantedValue in Fig. 7.3 and UseCounter in Fig. 7.2 to compute the ErrorValue with Eqn. 7.1. The result is shown in Fig. 7.4.

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline 7 & 6 & 5 & 6 & 5 & 4 & 5 & 4 & 3 & 3 & 3 \\ \hline \end{array} & \text{WantedValue} \\
 - & \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline 1 & 1 & 3 & 1 & 2 & 5 & 0 & -2 & -1 & 2 & 3 \\ \hline \end{array} & \text{UseCounter} \\
 \hline
 \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline 6 & 5 & 2 & 5 & 3 & -1 & 5 & 6 & 4 & 1 & 0 \\ \hline \end{array} & \text{ErrorValue}
 \end{array}$$

Figure 7.4: An Example of ErrorValue

Figure 7.5: Value range of ErrorValue

Fig. 7.5 shows the possible value ranges of the ErrorValue. The following is the interpretation of the ranges.

- A: ($\text{ErrorValue} \leq 0$) Competence in skill - This value range is derived when the value of the UseCounter is equal or greater than the value of the WantedValue. This reflects that the student is capable of performing that corresponding skill.
- B: ($1 \leq \text{ErrorValue} \leq \text{WantedValue} - 1$) Insufficiency in skill - This value range is derived from a positive value of UseCounter but less than the WantedValue. This reflects that the student has insufficient practice to achieve the desirable level as stated in the WantedValue.
- C: ($\text{ErrorValue} = \text{WantedValue}$) Uncertain of skill - This particular value is attained when the UseCounter is zero. This value has inconclusive results as it indicates two conditions. The first condition is that the student has not performed a skill while the second condition is that the student has performed the skill correctly as many times as he has done incorrectly. Hence, more execution of scenarios is necessary to determine the comprehension of skills.
- D: ($\text{ErrorValue} > \text{WantedValue}$) Incompetence in skill - The last range of value is derived when the UseCounter is negative. This reflects that the student is incompetent in performing that skill.

The following gives the formal definition from the interpretations of the ErrorValue.

Formal definition :

$$If \ ErrorValue[i] \begin{cases} \leq 0, & Competence \ of \ skill \ i \\ = 1 \dots (WantedValue[i] - 1), & Insufficient \ of \ skill \ i \\ = WantedValue[i], & Uncertain \ of \ skill \ i \\ > WantedValue[i], & Incompetence \ of \ skill \ i \end{cases}$$

7.3.4 Cognitive Rule, P_3

The vector error-value, ε_3 , is an input to the cognitive rule, P_3 , to generate the respective instructional sequences for the student. The instructional sequences are divided into 2 forms. The first form is a set of tutorials that explicitly tutor the student on the skills that they have demonstrated an incompetence in. The other form is the set of scenarios that explicitly assesses the student's competence in the skills that have not be adequately assessed or when the system is uncertain of the student's comprehension in the skills. As clearly shown in Fig. 7.1, the tutorials explicitly teach the students cognitive skills in the third level; whereas, the scenarios are fed down to the lower level in EpiList I to generate the necessary conditions to test the students.

The tutorials are activated whenever incompetence is detected from the Error-Value. However, incompetency in more than one skill is detected from the Error-Value; therefore, ways of determining which skills should be tutored must be devised. It has been mentioned above that the values of the WantedValue and the position of the skills within the UseCounter represents the level of importance of the skills. Hence, this importance can be used to determine the order of the tutorials when incompetence in more than one skill is detected.

A high ErrorValue signifies two situations. The first situation is, as mentioned

above, a negative value in UseCounter that would elevate the value of the ErrorValue when applied to the formulae in Eqn. 7.1. The elevation of value in the ErrorValue infers that the student is incompetent at performing the corresponding skill indicated by the value. Second situation arises when there is a high WantedValue value. The high value in the ErrorValue in this situation does not necessarily infer an incompetency; but it infers an insufficient execution of the corresponding skill. Focus on the Simple Generalisation skill (first value) in Fig. 7.4 as an example, although the value of the UseCounter is a positive value, but due to the high WantedValue value, the result in the ErrorValue would be high as well. Therefore, once a skill is selected by the Cognitive Rules, the skill has to be checked against the WantedValue to determine the correct situation of the high ErrorValue and to carry out the corresponding actions.

The following is the formal definition of the cognitive rules and it uses Fig. 7.4 as an example.

Formal Definition:

	Semantic of notation	Formal notation	Actions based on example
<i>If</i>	Identify a skill with the highest ErrorValue value	① $\exists i \text{ s.t.}$ $\{(ErrorValue[i] = \max_j ErrorValue[j])\}$	$i = 0,$ $ErrorValue[0] = 6$
	and	$\wedge$	
	Ensure that other ErrorValue with the similar highest value has a lower WantedValue or at a further position in the vector	② $\forall x \in K, (WantedValue[x] < WantedValue[i]) \vee \{(WantedValue[x] = WantedValue[i]) \rightarrow (x < i)\}$ Where $K = \{k_1, k_2, \dots, k_y\} \forall k \in K, (ErrorValue[i] = ErrorValue[k])$	$K = \{7\},$ $ErrorValue[7] = 6,$ $WantedValue[7] < WantedValue[0]$
<i>Then</i>	Check against WantedValue to determine to tutor or provide scenario	③ <i>IF</i> $(ErrorValue[i] > WantedValue[i])$	$ErrorValue[0] < WantedValue[0]$ (FALSE)
	Provide Tutorial (Details in Section 7.4.2)	④ <i>THEN</i> $Tutorial(i)$	
	Provide Scenario (Details in Section 7.4.3)	⑤ <i>ELSE</i> $Scenario(i)$	Scenario for skill 0 (Simple Generalisation) will be provided

Table 7.2: Formal notation of the Cognitive Rules (Eqn. 7.2)

Eqn. 7.2① extracts the skill with the highest ErrorValue. However, it is necessary to ensure that there is no other skill with the same ErrorValue. If such a skill is detected, Eqn. 7.2② extracts these skills and ensure they possess a lower WantedValue or are at a lower position in the vectors. Hence, the selected skill has the highest ErrorValue with higher WantedValue or position in the vectors. The selected skill is compared with its WantedValue in Eqn. 7.2③ to determine if the student exhibits incompetence in this skill. If it is, the tutorial would be activated by Eqn. 7.2④. Otherwise, the scenario would be provided as described/specified by Eqn. 7.2⑤. In this section, details of the tutorials and scenarios are not presented. They will be covered in the following section.

7.4 Architecture of EpiList II

This section describes in detail the various functional computations in the architecture of EpiList II. Section 7.4.1 covers the pedagogical model supporting EpiList II. Detailed discussion on the tutorial mechanisms is highlighted in Section 7.4.2 and the assessment of the student's skills competency through scenario interaction is presented in Section 7.4.3.

7.4.1 Pedagogical model of EpiList II

The pedagogical model for the explicit cognitive teaching in EpiList II is based on the cognitive apprenticeship model proposed by Prof Collins (Collins et al., 1991). The instructional sequence of cognitive apprenticeship is as follows:

1. Demonstrate (or make visible) the cognitive tasks,
2. Demonstrate the application of the task under different context, and
3. Assess the student

According to the model, explicit instructions for the cognitive skills should first guide the students to perform the cognitive task step by step. This would also demonstrate to the students the sequence involved in such a cognitive skill. To further reinforce the student's understanding, the sequence is again demonstrated to the student under a different context. Subsequently, the instruction would generate a testing scenario to assess the student's understanding of the cognitive skill taught.

EpiList II embeds this pedagogical model in its instructions. However, due to limitations in EpiList I, demonstrating the skill under a different context is not possible. The limitation is that EpiList I consists of only one epistemic form which is List Making. A different context refers to another situation or scenario whereby the skill is used; hence creating another set of database within EpiList I would only involve the student in performing the skill in different content instead of different context.

Therefore, instructional sequences in EpiList II's tutorials would first guide the student to perform the skills step by step and to be followed by an assessment to ensure that the student has correctly acquired the skill. If the student fails the assessment again, EpiList II repeats the instructional sequence based on the assessment and a new set of assessment questions will be formulated to assess the student. To avoid an endless loop, the repetition of the instructional sequence is limited to 3. When that happens, the correct answers will be given to the student. This last attempt is instruction by rote.

7.4.2 Tutorials of EpiList II

The tutorials of EpiList II explicitly instruct the student to perform the skills step by step in order to allow the student to understand the processes involved in the skills. The skills that EpiList II attempts to teach include generalisation skill, comparing and contrasting sub-skill, and the reasoning skills.

Tutorial for Generalisation skill

The tutorial instruction that teaches generalisation skills would firstly guide the students to identify the properties possessed by each item. Subsequently, the instruction would display the items to be generalised together with their properties, and instruct the student to identify the properties that are common among all the items on display. Subsequently, explaining the task he has undertaken as a generalisation process. This tutorial is illustrated using Example 3, where the student is unable to generalise 'Tiger', 'Crow' and 'Lizard'. The list of categories to be used in the example is {Cold-blooded, Warm-blooded, Breathe through lungs, Lay eggs}.

Example 3

Semantic	Dialogue
Obtain properties of first item.	System: (Displays Tiger and a list of categories.) Which of the following categories does this animal belong to?
Student makes a mistake.	Student: (Selects “Breathe through lungs” and “Cold-blooded”).
Tutors on the mistake.	System: (Display details of Tiger) Does Tiger belong to the category “Cold-blooded”. Student: (Selects “No”) System: Correct. Now go back and make the correction.
Student corrects his/her mistake.	Student: (De-selects “Cold-blooded” and selects “Warm-blooded”).
Obtain properties of second item.	System: (Displays Crow and the same list of categories.) Which of the following categories does this animal belong to? Student: (Selects “Breathe through lungs”, “Warm-blooded” and “Lay eggs”).
Obtain properties of third item.	System: (Displays Lizard and the same list of categories.) Which of the following categories does this animal belong to? Student: (Selects “Breathe through lungs”, “Cold-blooded”).
Generalise the three items.	System: (Displays Tiger, Crow and Lizard as well as their categories that the student had selected previously.) Which of the following categories do all these animals belong to? Student: (Selects “Breathe through lungs”).
Explain the skill.	System: Well done. This is the process of generalisation. Generalisation is to obtain the categories that all the displayed animals belong to.
Assess the student.	System: Now I am going to test your generalisation skill. (Display opportunistic assessment to assess the student) Student: (Answer the assessment correctly.)
	System: Well done. Now go back to the tutorial.

The following is the formal definition of the tutorial for generalisation skill. In this formal definition, a list termed as *skill* is first introduced. *Skill*[*i*] describes the skill at position *i* of the UseCounter.

	Semantic of notation	Formal notation	Actions based on example
<i>If</i>	Skill is a Simple Generalisation	① ($skill[i] = SimpleGeneralisation$)	
	or	$\vee$	
	Skill is a Moderate Generalisation	② ($skill[i] = ModerateGeneralisation$)	
	or	$\vee$	
	Skill is a Difficult Generalisation	③ ($skill[i] = DifficultGeneralisation$)	Example 3 is a difficult generalisation.
<i>Then</i>	Obtain the number of items involves in the generalisation	④ let $n = items\ to\ be\ Generalised $	$n = 3,$ $e_1 = Tiger,$ $e_2 = Crow,$ $e_3 = Lizard$
	Identify the properties of each item	⑤ $p_x = ObtainProperties(e_x),$ where $1 \leq x \leq n$	$p_1 = \{Warm-blooded, Breathe through lungs\},$ $p_2 = \{Warm-blooded, Breathe through lungs, Lay eggs\},$ $p_3 = \{Cold-blooded, Breathe through lungs, Lay eggs\}$
	Generalise the items	⑥ $P = GeneraliseProperties(e_1, e_2, \dots, e_n)$	$P = \{Breathe through lungs\}$

Table 7.3: Formal notation of Generalisation Tutorial (Eqn. 7.3)

When the selected skill, i , to be taught is either Simple, Moderate or Difficult Generalisations, the same instructional sequence is activated. Eqn. 7.3④ determines the number of items involved in the process of generalisation. Subsequently, the properties of each item is to be derived by the student as in Eqn. 7.3⑤. Finally, Eqn. 7.3⑥ presents the items as well as their properties for the student to determine the similarities. This will essentially guide the student to perform generalisation.

Tutorial for Comparison Skills

Comparison skills consist of comparing and contrasting sub-skills and each sub-skill in turn consists of 3 different types: one-to-one, one-to-group and group-to-group, and 3 different difficulty levels: simple, moderate and difficult. Although the UseCounter does not distinguish between the types of comparison skills exhibited by the student, it is important to identify the types of comparison skills to be used during the tutorials. This is because different types of comparison skills would require slightly different approaches to tutor the students.

Tutorial for one-to-one Comparing Sub-skill

Since the one-to-one comparison skill is the simplest, its instruction is also the simplest and most direct. Firstly, the instruction would guide the student to identify the properties found in two items. Subsequently, it would instruct the student to infer if the two items belong to the same category based on the properties that have been identified. Example 4 is an example of such a tutorial and this example tutors the student to compare a 'Tiger' and 'Snake'. The list of categories is the same as Example 3.

Example 4

Semantic	Dialogue
Obtain the properties of first item.	System: (Displays Tiger and a list of categories.) Which of the following categories does this animal belong to? Student: (Selects “Breathe through lungs” and “Warm-blooded”.)
Obtain the properties of second item.	System: (Displays Snake and the same list of categories.) Which of the following categories does this animal belong to? Student: (Selects “Breathe through lungs” and “Cold-blooded”.)
Compare the two items.	System: (Displays Tiger and Snake as well as the categories they belong to.) Do the animals belong to the category “Breathe through lungs” Student: (Selects “Yes”.)
Explain the skill.	System: Well done. This is the process of the one-to-one comparing. One-to-one comparing determines if 2 items belong to the a similar category.
Assess the student.	System: Now I am going to test your skill in one-to-one comparison. (Display opportunistic assessment to assess the student) Student: (Answer the assessment correctly.)
	System: Well done. Now go back to the tutorial.

The following is the formal notation of the one-to-one comparing sub-skill. The predicate ‘ $Type(i)$ ’, that is used in the notation, returns the type of comparison skill corresponding to the i^{th} position in the UseCounter. Hence, the possible return values of the predicate include ‘one-to-one’, ‘one-to-group’ and ‘group-to-group’.

All the comparison processes in EpiList I require the student to perform comparison among different items with respect to a particular property. In another words, the comparison processes in EpiList I determine if two items or two sets of items are similar in terms of a specific property. From Example 4, ‘Tiger’ and ‘Snake’ are compared to identify similarity with respect to the property “Breathe through lungs”. On the other hand, if the property is “Warm-blooded”, the two animals would be different. Since only this form of comparison is used in EpiList I, EpiList II is designed to tutor only this form of comparison.

Therefore, the tutorials of EpiList II that tutor comparison skills have to guide the student to identify the similarity or difference with respect to a specific property. This specific property will be termed as ‘interested’ property from here on. The ‘interested’ property is determined through EpiList I. EpiList II monitors the tutorials of EpiList I to determine the student’s deficiency in skills as well as to extract the items, property list and the ‘interested’ property to be used in its tutorials. The ‘interested’ property is denoted as ‘ P ’ in the subsequent formal notations of comparison skills.

	Semantic of notation	Formal notation	Actions based on example
<i>If</i>	Skill is a Simple or Moderate or Difficult Comparing	① $[(skill[i] = SimpleComparing) \vee (skill[i] = ModerateComparing) \vee (skill[i] = DifficultComparing)]$	Example 4 is a difficult comparing sub-skill
	and	$\wedge$	
	The comparison is a one-to-one comparison	② $(Type(i) = One - to - One)$	compare {Tiger} against {Snake}
<i>Then</i>	Identify the properties of first item	③ $p_1 = ObtainProperties(e_1)$	$e_1 = Tiger,$ $p_1 = \{Warm-blooded, Breathe through lungs\}$
	Identify the properties of second item	④ $p_2 = ObtainProperties(e_2)$	$e_2 = Snake,$ $p_2 = \{Cold-blooded, Breathe through lungs, Lay eggs\}$
	Compare the two items	⑤ $((P \in p_1) \wedge (P \in p_2)) = TRUE$	$P = \{Breathe through lungs\}$ exists in p_1 and p_2 .

Table 7.4: Formal notation of One-to-one Comparing Tutorial (Eqn. 7.4)

Eqn. 7.4① checks that the skill to be taught is a comparing sub-skill and Eqn. 7.4② ensures that it is a one-to-one comparison. Firstly the student is guided to derive the properties of the 2 items by Eqn. 7.4③ and 7.4④. Subsequently, Eqn. 7.4⑤ guides the student to determine which 2 items possess the ‘interested’ property in order to

induce that the 2 items are similar with respect to the ‘interested’ property.

Tutorial for one-to-group Comparing Sub-skill

The linear comparing sub-skill consists of comparing an item against a set of items. The instruction first guides the student to identify the properties of an item. Subsequently, it instructs the student to identify the common properties of a set of items. Finally, the student compares the item against the common properties of the set of item to determine if they belong to the same category. Example 5 is an example of this tutorial and it tutors the student to compare a ‘Tiger’ against a set consisting of ‘Snake’ and ‘Lizard’. The list of categories to be used in this example is {Cold-blooded, Warm-blooded, Breathe through lungs, Lay eggs} and the ‘interested’ property, P , is “Breathe through lungs”.

Example 5

Semantic	Dialogue
Obtain properties of an item.	System: (Displays Tiger and a list of categories.) Which of the following categories does this animal belong to? Student: (Selects “Breathe through lungs” and “Warm-blooded”.)
Obtain the properties of a set of items.	System: (Displays Snake and Lizard as well as their properties. Display the same list of categories.) Which of the following categories are common to these animals? Student: (Selects “Breathe through lungs” and “Cold-blooded”.)
Compare the item against the set of items.	System: (Displays Tiger and its categories. Display Snake and Lizard and their common properties.) Do these animals belong to the category “Breathe through lungs” Student: (Selects “No”.)
Student answered wrongly.	System: Wrong. Observe the properties of the animals carefully and determine whether they all belong to “Breathe through lungs”. Student: (selects “Yes”)
Explain the skill.	System: Well done. This is the process of the one-to-group comparing. This form of comparing determines if one item is similar to a group of items with respect to a specific category.
Assess the student.	System: Now I am going to test your one-to-group comparison. (Display opportunistic assessment to assess the student) Student: (Answer the assessment correctly.)
	System: Well done. Now go back to the tutorial.

Formal notation:

	Semantic of notation	Formal notation	Actions based on example
<i>If</i>	Skill is a Simple or Moderate or Difficult Comparing	① $[(skill[i] = SimpleComparing) \vee (skill[i] = ModerateComparing) \vee (skill[i] = DifficultComparing)]$	Example 5 is a difficult comparing sub-skill
	and	$\wedge$	
	The comparison is a one-to-group comparison	② $(Type(i) = One - to - Group)$	compare {Tiger} against {Snake, Lizard}
<i>Then</i>	Identify the properties of an item	③ $p_1 = ObtainProperties(e_1^1)$	$e_1^1 = Tiger,$ $p_1 = \{Warm-blooded, Breathe through lungs\}$
	Obtain the number of items in the set	④ let $n =$ number of items in the set	$n = 2$
	Identify the properties of a set of item	⑤ $p_2 = GeneraliseProperties(e_1^2, e_2^2, \dots, e_n^2)$	$e_1^2 = Snake,$ $e_2^2 = Lizard,$ $p_2 = \{Cold-blooded, Breathe through lungs, Lay eggs\}$
	Compare the item against the set of items	⑥ $((P \in p_1) \wedge (P \in p_2)) = TRUE$	$P = \{Breathe through lungs\}$ exists in p_1 and p_2 .

Table 7.5: Formal notation of One-to-group Comparing Tutorial (Eqn. 7.5)

When the one-to-group Comparing Sub-skill is to be taught, Eqn. 7.5③ first guides the student to determine the properties of the first item. Subsequently, the student generalises the set of n items in Eqn. 7.5⑤. Finally, Eqn. 7.5⑥ guides the student to identify and induce that they are similar with respect to a specific property.

Tutorial for Group-to-group Comparing Sub-skill

The group-to-group comparing sub-skill involves the comparison of one set of items against another set of items. The instruction first guides the student to perform two

generalisations; one for each set. Subsequently, it instructs the student to compare the two sets of common properties to determine if they belong to the same category. Example 6 shows a group-to-group comparison tutorial and the student is to compare ‘Tiger’ and ‘Crow’ against ‘Snake’ and ‘Lizard’. The list of properties and ‘interested’ property is similar to Example 5.

Example 6

Semantic	Dialogue
Generalise to derive common properties of first set of items.	System: (Displays Tiger and Crow as well as their properties. Display a list of categories.) Which of the following categories are common to these animals? Student: (Selects “Breathe through lungs” and “Warm-blooded”.)
Generalise to derive common properties of second set of items.	System: (Displays Snake and Lizard as well as their properties. Display the same list of categories.) Which of the following categories are common to these animals? Student: (Selects “Breathe through lungs” and “Cold-blooded”.)
Compare the common properties of the two sets of items.	System: (Displays Tiger, Crow and their common properties. Display Snake, Lizard and their common properties.) Do these animals belong to the category “Breathe through lungs” Student: (Selects “Yes”.)
Explain the skill.	System: Well done. This is the process of the group-to-group comparing. This form of comparison determines if two sets of items are similar with respect to a specific property.
Assess the student.	System: Now I am going to test your ability to perform group-to-group comparison. (Display opportunistic assessment to assess the student) Student: (Answer the assessment <i>incorrectly</i> .)
Student fails the assessment.	System: (Reactivate the group-to-group comparing sub-skills tutorial.)
Re-tutor the student.	Student: (Go through the tutorial and correctly answer the re-assessment.)
	System: Well done. Now go back to the tutorial.

Formal notation:

	Semantic of notation	Formal notation	Actions based on example
<i>If</i>	Skill is a Simple or Moderate or Difficult Comparing	① $[(skill[i] = SimpleComparing) \vee (skill[i] = ModerateComparing) \vee (skill[i] = DifficultComparing)]$	Example 6 is a difficult comparing sub-skill
	and	$\wedge$	
	The comparison is a group-to-group comparison	② $(Type(i) = Group - to - Group)$	compare {Tiger, Crow} against {Snake, Lizard}
<i>Then</i>	Obtain the number of items in the first set	③ let m = number of items in the first set	$m = 2$
	Identify the properties of the first set of items	④ $p_1 = GeneraliseProperties(e_1^1, e_2^1, \dots, e_m^1)$	$e_1^1 = Tiger,$ $e_2^1 = Crow,$ $p_1 = \{Warm-blooded, Breathe through lungs\}$
	Obtain the number of items in the second set	⑤ let n = number of items in the second set	$n = 2$
	Identify the properties of the second set of items	⑥ $p_2 = GeneraliseProperties(e_1^2, e_2^2, \dots, e_n^2)$	$e_1^2 = Snake,$ $e_2^2 = Lizard,$ $p_2 = \{Cold-blooded, Breathe through lungs, Lay eggs\}$
	Compare the item against the set of items	⑦ $((P \in p_1) \wedge (P \in p_2)) = TRUE$	$P = \{Breathe through lungs\}$ exists in p_1 and p_2 .

Table 7.6: Formal notation of Group-to-group Comparing Tutorial (Eqn. 7.6)

Eqn. 7.6① and ② ensure that the skill to be taught is the Group-to-Group Comparing Sub-skill. Firstly, Eqn. 7.6④ and ⑥ guide the student to perform generalisation on the two sets of items. Subsequently, Eqn. 7.6⑦ guides the student to induce that the two sets of common properties possess the similar ‘interested’ property; therefore, the two sets of items are similar with respect to that property.

Tutorial for Contrasting Sub-skills

Similar to comparing sub-skills, the contrasting sub-skill tutorials have to identify the 3 forms of comparison; namely: one-to-one, one-to-group and group-to-group. The process of contrasting is almost identical to that involved in the comparing sub-skills. The only difference is the last step in the process. In the comparing sub-skill, the last step is to derive that the items are similar in common properties with respect to a specific property; however, in the last step of a contrasting sub-skill one has to determine that the common properties of items does not possess the ‘interested’ property; hence, implying that they are different with respect to the ‘interested’ property.

Therefore, the tutorial sequence of the contrasting sub-skill is exactly the same as the comparing sub-skill, with the final step identifying a difference instead of a similarity with respect to a specific property. Hence, the last step in the tutorials guides the student to determine that the common properties do not include the ‘interested’ property. Example 7 shows the tutorial of group-to-group contrasting sub-skills. In order to highlight the differences between the tutorials of contrasting and comparing sub-skills, differences are underlined. The condition and environment are identical to Example 6 where ‘Tiger’ and ‘Crow’ is compared against ‘Snake’ and ‘Lizard’, and the list of categories is {Cold-blooded, Warm-blooded, Breathe through lungs, Lay eggs}. However, the ‘interested’ property is “Cold-blooded”.

Example 7

Semantic	Dialogue
Generalise to derive the common properties of the first set of items.	System: (Displays Tiger and Crow as well as their properties. Display a list of categories.) Which of the following categories are common to these animals? Student: (Selects “Breathe through lungs” and “Warm-blooded”.)
Generalise to derive the common properties of the first set of items.	System: (Displays Snake and Lizard as well as their properties. Display the same list of categories.) Which of the following categories are common to these animals? Student: (Selects “Breathe through lungs” and “Cold-blooded”.)
Compare the common properties of the two sets of items. Student answered <i>incorrectly</i> . Student selects the correct answer.	System: (Displays Tiger, Crow and their common properties. Display Snake, Lizard and their common properties.) Do these animals belong to the category “ <u>Cold-blooded</u> ” Student: (Selects “Yes”.) System: Wrong. Observe the properties of the animals carefully and determine whether they all belong to “ <u>Cold-blooded</u> ”. Student: (Select “ <u>No</u> ”.)
Explain the skill.	System: Well done. This is the process of the group-to-group contrasting. This form of contrasting determines if one group of items are different to another group of items with respect to a specific category.
Assess the student.	System: Now I am going to test your group-to group contrasting. (Display opportunistic assessment to assess the student) Student: (Answer the assessment correctly.)
	System: Well done. Now go back to the tutorial.

The formal notations of the contrasting sub-skill is almost identical to those of comparing. The only difference in the tutorial is in the last instruction, Eqn. 7.6⑦ $((P \in p_1) \wedge (P \in p_2))$, where the student is guided to determine if the common properties possess the ‘interested’ property. For the comparing sub-skills, the two sets of common properties possess the ‘interested’ property; hence, the result of Eqn. 7.6⑦ would be ‘TRUE’. In the situation for contrasting sub-skills, the result would be ‘FALSE’ as both sets of common properties do not possess the ‘interested’ property.

Formal notation:

	Semantic of notation	Formal notation	Actions based on example
<i>If</i>	Skill is a Simple or Moderate or Difficult Comparing	① $[(skill[i] = SimpleContrasting) \vee (skill[i] = ModerateContrasting) \vee (skill[i] = DifficultContrasting)]$	Example 7 is a moderate contrasting sub-skill
	and	$\wedge$	
	The comparison is a group-to-group comparison	② $(Type(i) = Group - to - Group)$	contrast {Tiger, Crow} against {Snake, Lizard}
<i>Then</i>	Obtain the number of items in the first set	③ let m = number of items in the first set	$m = 2$
	Identify the properties of the first set of items	④ $p_1 = GeneraliseProperties(e_1^1, e_2^1, \dots, e_m^1)$	$e_1^1 = Tiger,$ $e_2^1 = Crow,$ $p_1 = \{Warm-blooded, Breathe through lungs\}$
	Obtain the number of items in the second set	⑤ let n = number of items in the second set	$n = 2$
	Identify the properties of the second set of items	⑥ $p_2 = GeneraliseProperties(e_1^2, e_2^2, \dots, e_n^2)$	$e_1^2 = Snake,$ $e_2^2 = Lizard,$ $p_2 = \{Cold-blooded, Breathe through lungs, Lay eggs\}$
	Compare the item against the set of items	⑦ $((P \in p_1) \wedge (P \in p_2)) = FALSE$	$P = \{Cold-blooded\}$ exists in p_2 but not p_1 .

Table 7.7: Formal notation of Group-to-group Contrasting Tutorial (Eqn. 7.7)

Similar to the formal notations of comparing sub-skills, Eqn. 7.7① ensures that the skill to be taught is a contrasting sub-skill while Eqn. 7.7② checks that the comparison is a group-to-group comparison. The tutorial also first guides the student to perform two generalisations, Eqn. 7.7④ and ⑥, to determine the common properties of the two sets of items, similar to the tutorials of comparing sub-skills. The difference is the last step; Eqn. 7.7⑦ guides the student to identify that the common properties does not possess the ‘interested’ property and hence induce that the items are different

with respect to the ‘interested’ property.

Eqn.7.7 is the formal notation of the tutorial for group-to-group contrasting sub-skill. As for one-to-one and one-to-group contrasting sub-skills, their formal notation is almost identical to Eqn. 7.4 and Eqn. 7.5 respectively with the only difference or changes as presented above. Hence, the formal notations of the tutorials for one-to-one and one-to-group contrasting sub-skills are not presented here.

Tutorial for Reasoning Skills

Reasoning skills in EpiList I consist of positive and negative reasoning skills. The instructional sequence of these skills explicitly explains and describes the reasoning processes to the student. Hence, the first step in the instruction is to obtain a set of common properties of the correctly grouped items; followed by a set of common properties of the incorrectly grouped items. Subsequently, the two sets of items as well as their common properties are displayed and the tutorial explicitly explains the reasoning process of EpiList I of grouping the incorrectly categorised items together with or separate from the correctly categorised items.

Example 8 is an example of the tutorial for the negative reasoning skill. This example uses the example of the intra-scheme rules in Section 4.3.1. If the student is unable to reason from this rule to correct the mistake, the tutorial for negative reasoning skill will be activated.

Similar to the tutorial of comparison skills, the tutorials of reasoning skills have an ‘interested’ property. If it is a positive reasoning skill, the ‘interested’ property is the correct category of the incorrectly grouped items. Hence, the ‘interested’ property of positive reasoning skill can be found in the common properties of both correctly and incorrectly grouped items.

Example 8

Semantic	Dialogue
Generalise to derive the common properties of the correctly grouped items.	System: (Displays Whale and Dolphin as well as their properties. Display a list of categories.) Which of the following categories are common to these animals? Student: (Selects “Live in water” and “Warm-blooded”.)
Generalise to derive the common properties of the incorrectly grouped items.	System: (Displays Tiger and Bear as well as their properties. Display the same list of categories.) Which of the following categories are common to these animals? Student: (Selects “Live on Land” and “Warm-blooded”.)
Compare the two sets of common properties.	System: (Displays Whale, Dolphin and their common properties on one side. Display Tiger, Bear and their common properties on the other side.) Do the animals on the left similar to the animals on the right in terms of “Live in water”? Student: (Select “No”.)
Explicitly explains the reasoning process.	System: Correct. The animals on both side are different with respect to “Live in water”. however, they are grouped together in the “Live in water” category. Therefore, the animal on the right should not be grouped together with the animals on the left. Now, go back and move the animals on the right out from the category “Live in water”.

On the other hand, the ‘interested’ property of a negative reasoning skills is the category where the incorrectly grouped items are found. Hence, the ‘interested’ property of negative reasoning skills can be found in the common properties of the correctly grouped items but not in the common properties of incorrectly grouped items.

In the formal notation of the tutorial of reasoning skills, a new predicate is used to define the explicit explanation of the reasoning skills. The predicate, *ExplicitExplain(ReasoningSkill)*, explicitly explains the reasoning skill to the student.

Formal notation:

	Semantic of notation	Formal notation	Actions based on example
<i>If</i>	Skill to be taught is a Reasoning Skill	① ($skill[i] = ReasoningSkill$)	Example 8 is a negative reasoning skill
<i>Then</i>	Obtain the number of correctly grouped items	② let m = number of correctly grouped items	$m = 2$
	If there is only one correctly grouped item	$if(m = 1)then$	
	Identify the properties of the correctly grouped items	③ $p_1 = ObtainProperties(e_1^1)$	
		$else$	
	Generalise to obtain the common properties of the correctly grouped items	④ $p_1 = GeneraliseProperties(e_1^1, e_2^1, \dots, e_m^1)$	$e_1^1 = Whale,$ $e_2^1 = Dolphine,$ $p_1 = \{Live\ in\ water,\ Warm-blooded\}$
	Obtain the number of incorrectly grouped items	⑤ let n = number of correctly grouped items	$n = 2$
	If there is only one incorrectly grouped item	$if(n = 1)then$	
	Identify the properties of the incorrectly grouped items	⑥ $p_2 = ObtainProperties(e_1^2)$	
		$else$	
	Generalise to obtain the common properties of the incorrectly grouped items	⑦ $p_2 = GeneraliseProperties(e_1^2, e_2^2, \dots, e_n^2)$	$e_1^2 = Tiger,$ $e_2^2 = Bear,$ $p_1 = \{Live\ on\ land,\ Warm-blooded\}$
	Determine the type of reasoning	$if(P \in p_2)then$	$P = "Live\ in\ water" \notin p_2$
	Explicitly explain positive reasoning skill	⑧ $ExplicitlyExplain(PositiveReasoning)$	
		$else$	
	Explicitly explain positive reasoning skill	⑨ $ExplicitlyExplain(NegativeReasoning)$	

Table 7.8: Formal notation of Reasoning Tutorial (Eqn. 7.8)

When reasoning skills are detected by EpiList II to be taught, the tutorial for reasoning skills will be activated. Firstly, the common properties of the correctly grouped items are derived. If there is only one correctly grouped item, $m = 1$, the tutorial guides the student to identify the properties of the item with Eqn. 7.8③. If there is more than one incorrectly grouped item, $m > 1$, Eqn. 7.8④ guides the student to generalise the items to derive the common properties. Similarly, Eqn. 7.8⑥ and Eqn. 7.8⑦ guide the student to either obtain the properties or generalise to derive the common properties of the incorrectly grouped items depending on the number of the incorrectly grouped items. Subsequently, these two sets of common properties would be used in Eqn. 7.8⑧ and Eqn. 7.8⑨ to explicitly explain the reasoning processes.

7.4.3 Assessments(Scenarios) of EpiList II

The assessments of EpiList II can be undertaken in two forms. The first is the assessment during a tutorial; this ensures that the student understands and acquires cognitive skills. The second form of assessment in EpiList II come in the form of scenarios to determine skill competency of the student. The former assessment is ‘*Opportunistic*’ while the latter is ‘*Non-opportunistic*’.

Opportunistic Assessment

As mentioned above, the opportunistic assessment is performed immediately after the tutorials of the cognitive skills in EpiList II. The tutorials in EpiList II are dependent on the tutorials in EpiList I. The tutorial of EpiList I is a series of instructions that guides the student in the steps to complete the cognitive skills. Hence, each page of the tutorial interaction with the student in EpiList I can potentially be an exhibition of inaccurate steps that can be detected by EpiList II as incompetency in the skill. In another words, the tutorials of EpiList II that teach cognitive skills are based on only one page of the tutorial interactions in EpiList I. Therefore, the

opportunistic assessments that follow the tutorials of EpiList II, should also provide a similar one page condition as that used in EpiList I to precisely and correctly assess the cognitive skills.

The signal from EpiList II to the lower level to generate the opportunistic assessment for the student should generate a single page scenario from the tutorials of EpiList I. The signal that generates the single page scenario is therefore named as the '*SingleScenario*'. Fig. 7.6 is the flow chart for the generation of single scenarios.

Figure 7.6: The algorithm of SingleScenario

The algorithm first randomly extracts twenty items in preparation for the creation of a scenario. Subsequently, from these twenty items, the algorithm attempts to extract the necessary items that can be arranged to assess the targeted cognitive skill. If this attempt fails, the process is repeated until the extraction to assess the cognitive skill is successful. After the extraction is successful, the extracted items are

compiled into a script that is passed down to the lower control loop, EpiList I, to generate the opportunistic assessment.

Non-opportunistic Assessment

The non-opportunistic assessment is undertaken to assess the student whenever uncertainty in the cognitive skills is detected. The non-opportunistic assessment is presented or activated without any indications or conditions from EpiList I; hence, activating the non-opportunistic assessment is not by opportunity but is well planned and designed. As it is not activated in relation to EpiList I, SingleScenarios are not applicable. Therefore, a different approach and scenario signals to generate the non-opportunistic assessment is required.

The approach adapted here is to generate a scenario in EpiList I that will activate a tutorial completely and probe for the student's competency in correctly executing the cognitive steps during the tutorial. Hence, the signal to generate the necessary scenario is termed as *FullScenario*. One important step of the signal generation is the determination of which tutorial is to be activated. Table 7.9 shows the corresponding cognitive skills that a tutorial or rule would require the student to be proficient in. Hence, based on the type of cognitive skills to be tested, the correct or desired tutorial can be determined.

From Table 7.9, it can be seen that several tutorials require the proficiency of more than one cognitive skill. Therefore, to assess the student on a particular cognitive skill, several tutorials can be used as FullScenario for the non-opportunistic assessment. Hence, the particular tutorial that is to be used to assess the student has to be determined. The system ensures that the selection of the tutorials is fair and just. It is accomplished simply by a random selection of un-activated tutorials.

	Generalisation skill	Comparing sub-skill	Contrasting sub-skill	Positive reasoning skill	Negative reasoning skill
Positive Exemplar Rule		✓		✓	
Negative Exemplar Rule			✓		✓
Linear Positive Exemplar Rule	✓	✓		✓	
Linear Negative Exemplar Rule	✓		✓		✓
Non-linear Positive Exemplar Rule	✓	✓		✓	
Non-linear Negative Exemplar Rule	✓		✓		✓
Positive Introduction Rule	✓ (Only Simple)				
Negative Introduction Rule	✓ (Only Simple)				
Regrouping Rule	✓	✓			

Table 7.9: Relations between tutorials and cognitive skills

Figure 7.7: The algorithm of FullScenario

Fig. 7.7 is the algorithm to generate a *FullScenarios* for non-opportunistic assessments. Similar to the *SingleScenarios* algorithm, twenty items are repeatedly extracted from the database until a smaller set of items can be extracted to teach the targeted skill. After the items are extracted and determined, the type of tutorial of EpiList I is determined. Based on the selected tutorial, the system groups the items into the categories with mistakes to generate the necessary scenario in EpiList I that will precisely trigger the desired tutorial.

7.5 Evaluation of EpiList II

The evaluation of EpiList II was undertaken at MGS. The evaluation of EpiList I in MGS a year ago highlighted its cognitive development capabilities and potential as well as limitations and inefficiencies. The solution is a closed-loop cognitive instruction, resulting in the development of EpiList II. Therefore, the objective of EpiList II's evaluation is to demonstrate its improved efficiency in cognitive skill development. The evaluation of EpiList II was based on four classes of primary 4 students. This makes up a total of 140 ten years old children.

Similar to the evaluation of EpiList I, The evaluation consists of 3 phases. The first phase is the pre-evaluation test, which is a written test. The second stage is the hands-on session where the students go through the system twice. Lastly, the third phase is another written test, the post-evaluation test. However, in this evaluation of EpiList II, there is a slight change to the questionnaires and style to produce a controlled evaluation.

7.5.1 Structure of controlled evaluation

The controlled evaluation draws from the lessons derived from the evaluation of EpiList I. The first change was made to the questionnaires while the second change was made to the number of students having hands-on for EpiList II.

The questionnaires were separated into pre- and post-evaluation tests. It has been argued that the improvement shown in EpiList I is due to the lower difficulty level in the post-evaluation. Therefore, the evaluation of EpiList II attempts to rebut this argument. The method is to switch the questions between the pre- and post-evaluation tests. The questions that test for the cognitive skills in both tests are swapped. If improvement is still registered in this swapped evaluation tests, it shows that the improvement is not due to the difference in the difficulty levels.

The second change was made in response to the argument that the improvement

in the cognitive skills was not due to EpiList I but to the normal curriculum of the school. Therefore, for the evaluation this time round, not all the students will be involved in the hands-on session to use EpiList II. The students are divided into groups, two classes will go through all three phases of evaluation while the other two classes will only go through the first and last phases which are the answering of pre- and post-evaluation. In this way, the cause of cognitive skill improvement of the student can be differentiated between EpiList and school curriculum.

7.5.2 Result and analysis of EpiList II evaluation

The evaluation of EpiList II, as mentioned above, consists of 2 changes; first is the switching of question and the second is the controlled group of students that would not be exposed to EpiList II. The results will be presented in 2 sections with each section covering the result of each change.

Result of the controlled group evaluation

This section presents the results of the controlled group evaluation. This group of students answer the pre- and post-evaluation tests without being exposed to EpiList II. Fig. 7.8 shows the results of the evaluation tests on this group of students.

Figure 7.8: Result of the Controlled Group Evaluation

From the results shown in Fig. 7.8, the difference between the pre- and post-evaluation test is as small as 1% except for the difficult generalisation skill. From these results, it can be deduced that although the school's curriculum contributes to the improvement of cognitive skills proficiency, the contribution is relatively small and negligible.

Using these results to compare with the results of evaluation in EpiList I as depicted in Fig. 4.3, it can be seen that this year's primary 4 students, who evaluated EpiList II, are better than last year's primary 4 students who evaluated EpiList I in term of competency in cognitive skills employed in EpiList I. However, after the usage of EpiList I, the proficiency of last year's students exceeds this year's students. This shows that the improvement registered in EpiList I was due to the system and not other factors.

Result of the EpiList II controlled evaluation

This section focuses on the evaluation of EpiList II. The students in this evaluation follow the three phases of evaluation to evaluate EpiList II. The questions of this evaluation are swapped between the pre- and post-evaluation tests.

Figure 7.9: Result of the EpiList II Controlled Evaluation

The first and second bars of the chart in Fig. 7.9 are the results of the pre- and post-evaluation tests of EpiList I. These results are used for the comparison with the results of the pre- and post-evaluation test of EpiList II, which is represented by the third and fourth bars of Fig. 7.9. Focusing on the results of the EpiList II, improvement in competency is demonstrated across the different cognitive skills. The improvement is, on average, 15% for generalisation skills and 9% for comparison skills. This categorically demonstrates the fact that the questions of both pre- and post-evaluation tests are unbiased and are able register improvement in competency of skills.

When the results of EpiList II's evaluation is compared with the evaluation results of EpiList I, it can be observed that the improvement in EpiList II is smaller than

that of EpiList I. This could be due to the fact mentioned earlier that the cohort that evaluated EpiList II is better in terms of cognitive skills competency. As most of the students were already competent in cognitive skills, therefore, to increase the competency level further requires more effort and time.

However, a significant greater improvement in cognitive competency can be observed from the reasoning skills. As high as 51% of the students, as shown in Fig. 7.9, have acquired or understood the positive and negative reasoning skills as compared to the average of 8% in EpiList I. This great difference demonstrates the effectiveness of explicit teaching employed in EpiList II.

7.6 Summary

EpiList II is an enhancement of EpiList I. This chapter presents the design of EpiList II based on the formal definition and system analysis methodology proposed in the previous chapters. EpiList II is a third level closed-loop control system that wraps around EpiList I. EpiList II monitors the execution of cognitive skills in EpiList I and explicitly does cognitive skills tutoring whenever a deficiency in the skills is detected.

This chapter refines the cognitive skills into 3 difficulty levels for the incremental development of the closed-loop instruction. It also identifies the type of comparison involved in EpiList I; namely: one-to-one, one-to-group, and group-to-group comparison. Subsequently, the design of EpiList II components in terms of system theories is presented and at the same time these components are formally described for easy and precise understanding. The explicit or closed-loop instruction that transfer cognitive skills to the student is based on the theory of “Cognitive Apprenticeship” proposed by Prof Collins.

The chapter goes on to describe the notations and formal definitions of both opportunistic and non-opportunistic assessments that will be used to explicitly assess the student for his acquaintance and competence of cognitive skills. Finally, the

evaluation and analysis of EpiList II is presented in the last section of the chapter. The evaluation and analysis shows the fair and unbiased nature of the pre- and post-evaluation test questions. It also highlights the effectiveness of EpiList II in the transferring of positive and negative reasoning skills.

Chapter 8

A-EpiList

“To improve is to change;
to be perfect is to change often.”

– Winston Churchill

8.1 Background

EpiList II employs a cognitive loop over the underlying EpiList I structure. The improvement from an open-loop to a closed-loop system is demonstrated in EpiList II. This chapter demonstrates further enhancements made to EpiList using sound system theories.

In system theories, a closed-loop control is a linear response to the output of the process. Under the situation when the environment changes and affects the process, the linear response of the closed-loop control becomes ineffective in controlling the process. Hence, an adaptive control system is required in this situation. A similar situation also occurs in EpiList II. The ‘process’ in EpiList II is the student who is undergoing a guided instruction on classification. Different students have different capabilities and preferences in cognitive and reasoning skills. Therefore, the controller or the instructions employed by EpiList II should be adapted to the differences among students. The adaption capabilities of RAPSODY (Okamoto, 2000) highlight the ad-

vantages of adapting to different users/students to maximise their learning outcome.

The adaptation of EpiList II produces a flexible instruction and it is termed Adaptive EpiList or in short, A-EpiList. As mentioned in the previous chapter, the objective of the system design of EpiList is to address the issues of ad-hoc development of ITS. Incorporation of such a design into EpiList demonstrates the design capabilities and implications of adaptive system theories on ITS design.

Figure 8.1: Developments of EpiList

A-EpiList is an enhancement that encapsulates EpiList II which in turn encapsulates EpiList I. These layered encapsulations are depicted in Fig. 8.1. Hence, A-EpiList is considered as an incremental enhancement of EpiList II. It is not a fresh system developed from scratch but a incremental design over an existing system. The following sections show how sound system theories can support such incremental design.

8.2 Basics of Adaptive Control System

Figure 8.2: Block diagram of an adaptive system

Fig. 8.2 depicts an adaptive control system. The ‘Performance computation mechanism’ monitors the output of the plant and generates the performance parameters. A performance parameter is a measure of the dynamic characteristic of the closed-loop system in one aspect. The set of performance parameters describe the current situation or condition of the system in terms of quality or efficiency. From the values, the adaptive system can determine if it is performing within a desired level as indicated by the ‘Figures of Merit’ specified through a set of reference values for each of the indices. Based on the findings from the performance indices, the ‘Adaptive computer’ determines or computes the necessary changes to be made to the controller of the system.

Therefore, from the definition above, the adaptive control system performs three main functions (Aggarawal, 1981):

1. Identification,
2. Decision, and
3. Modification

“Identification refers to the measurement of the dynamics characteristics of the process to be controlled. (Aggarawal, 1981, pp. 728)” It is quite similar to the feedback element where it has to monitor the plant and feedback the information regarding the plant in a format understandable by the system. Identification probes for the dynamic characteristics and parameters of the plant. Such characteristics are defined by the performance parameters which are computed by the performance computation mechanism. The performance parameters reflect the current quality of the closed-loop control system.

‘Decision’ refers to the determination of the action or adaptation to undergo with respect to the performance parameters. From values of the performance parameters, the adaptive system has to determine the necessary adaption to perform on the controller to improve the performance.

Finally, ‘modification’ is the actual action of adapting the controller. This definition can be viewed in two parts. The first part is the modification procedure is decided from the decision process. The second part refers to the adaptable characteristic of the controller to be able to change accordingly. This also brings out the fact that controller of the closed-loop control has to be redesigned and rebuilt to be adaptable.

8.3 Design of A-EpiList

This section presents the step-by-step design of A-EpiList in terms of the 3 main functions discussed above. It demonstrates how EpiList II is incorporated with the 3 functions to arrive to A-EpiList.

Figure 8.3: System view of A-EpiList

Fig. 8.3 presents the system view of A-EpiList. As shown in the diagram, the adaptive system employs the changes in *UseCounter*, $\Delta UseCounter$, for the adaptive rules to tune the intra-scheme rules to suit the individual student. A careful examination of the intra- and inter-scheme rules of EpiList I shows that currently the adaptation is only suitable to tune the intra-scheme rules. Two characteristics

of the inter-scheme rules render them unadaptable. The first is the fact that all 4 inter-scheme rules employ only generalisation and comparing sub-skills. Hence, there is no user specific preference identifiable in the inter-scheme rules to adapt to. Secondly, the algorithm of the inter-scheme rules is considered to be rigid or unadjustable. The algorithm of inter-scheme rules cannot be changed since the sequence of introducing the category, regrouping items and removing a category has to be carried out sequentially; changing the sequence would result in unsuccessful migration of the classification scheme. Therefore, the adaptation rules are only applicable to the intra-scheme rules.

8.3.1 Identification of A-EpiList

Identification involves monitoring the student to derive a performance parameter or quality of the system based on responses by the student. Identification also includes the representation of its finding for the use by the system.

Therefore, the first step in the identification process is to define the performance parameters of EpiList II to be monitored. Since, the focus of instructions in EpiList is on the primitive generic cognitive skills: generalisation and comparison skills, as well as reasoning skills, hence, the monitoring of EpiList II is undertaken in 2 parts; with the former monitoring cognitive skills while the latter monitoring reasoning skills.

To simplify the implementation, the ‘Student modeler’ of the closed-loop cognitive instruction also monitors the same set of cognitive and reasoning skills. The correct and incorrect application of skills are reflected in the *UseCounter*. Therefore, the adaptive system makes effective use of this *UseCounter* in its identification to derive the performance measurement of EpiList II.

Based on the *UseCounter*, the adaptive system has to identify the performance of the student in terms of cognitive and reasoning skills. The *UseCounter* consists of eleven values; each representing one type of skill. By monitoring the changes in each value, the adaptive system can directly determine the necessary information from the

UseCounter. Therefore, the ‘Performance computation mechanism’ in A-EpiList is based on the changes of the *UseCounter*, $\Delta UseCounter$.

8.3.2 Decision of A-EpiList

The decisions of A-EpiList are undertaken by the adaptation rules of A-EpiList. The adaptive rules have to decide on the modifications to carry out based on performance parameters, $\Delta UseCounter$, derived from the performance computation mechanism. The performance parameters indicate the correct and incorrect execution of generalisation skill, and comparing and contrasting sub-skills. If the student is able to rectify his mistakes, the reasoning skills of the *UseCounter* would increase and vice versa. Therefore, the adaptation rules examine such changes to decide on the correct actions to undertake.

Performance Indices	Skills	Actions
Increment of <i>UseCounter</i> value	Comparing sub-skill	Promote ‘positive’ rules
	Contrasting sub-skill	Promote ‘negative’ rules
	Reasoning skill	Promote the activated rule
Decrement of <i>UseCounter</i> value	Comparing sub-skill	Demote ‘positive’ rules
	Contrasting sub-skill	Demote ‘negative’ rules
	Reasoning skill	Demote the activated rule

Table 8.1: Adaptation Rules

Table 8.1 shows the 6 adaptation rules used in A-EpiList. The input condition of the rules is the $\Delta UseCounter$, the directional changes of *UseCounter* showing a possible proficiency or deficiency in a skill or rule nature. However, the generalisation

skill is omitted since generalisation is used in *all* the rules of EpiList II. Therefore, the student's ability or inability in performing generalisation skills does not have any significance in any particular rule. The adaptive rules only check for the comparing and contrasting sub-skills, and the reasoning skills.

The comparing sub-skills are coupled together with the 'positive' rules of the intra-scheme rules. As the 'positive' rules employ positive exemplar and comparing sub-skills in the tutorials, the ability to perform this sub-skill is critical to the 'positive' rules. Whenever an increment is detected in *UseCounter*, the adaptive rules would increase the importance of the 'positive' intra-scheme rules by "promoting" the rules. This approach is, likewise, applicable to the 'negative' intra-scheme rules and contrasting sub-skills. When there is a decrement in the *UseCounter*, the adaptive rules would "demote" the use of 'negative' intra-scheme rules.

Reasoning skills on the other hand are not coupled specifically to any intra-scheme rules. Although the positive reasoning skill is related to 'positive' intra-scheme rules, promoting the positive rules based on the positive reasoning skill is inconclusive and inaccurate. To accurately adapt the intra-scheme rules, the precise rule that triggers the change in the *UseCounter* value has to be identified and adapted accordingly. Hence, as shown in the table, whenever there is a change in $\Delta UseCounter$ that represents reasoning skills, the rule that caused the change will be promoted or demoted respectively to address the preference of the student in undertaking such a reasoning approach.

8.3.3 Modification of A-EpiList

A-EpiList essentially takes in the promoting and demoting signals from the performance computation mechanism to increase or decrease the preference for the intra-scheme rules.

The first design procedure is to revamp the intra-scheme rules to become adaptable and changeable. The algorithm of intra-scheme rules guides the system to perform

the conditional checks in a predetermined and hard-coded sequence. In order to be adaptable, the algorithm or checking sequence has to be dynamically represented in the system. This is accomplished with an array of 6 values and the positions in the array decide the sequence of the conditional checks. In another words, the rule indicated by the first value in the array is the first rule to be checked, followed by the rule indicated by the second value and so forth. This array is named as the *Sequence* array. Table 8.2 shows the representational value for each of the intra-scheme rules. Odd representational values represent positive rules while even representational values represent negative rules. Furthermore, the greater the value, the greater the amount of cognitive loading is required to perform such a reasoning process.

Intra-scheme Rules	Representation Value
Direct Positive Exemplar Rule	1
Direct Negative Exemplar Rule	2
Linear Positive Exemplar Rule	3
Linear Negative Exemplar Rule	4
Non-linear Positive Exemplar Rule	5
Non-linear Negative Exemplar Rule	6

Table 8.2: Representation of Intra-scheme Rules

According to the algorithm of intra-scheme rules in Section 5.2.2, the initialisation of the *Sequence* array is given as: $Sequence = \{5, 6, 3, 4, 1, 2\}$. A ‘promote’ signal moves the value representing the desired rule up one position in the *sequence* array. Similarly, the ‘demote’ signal moves the value down the array. In this manner, the conditional checking sequence can be changed accordingly to adapt the algorithm to individual students.

Hence, the promotion of the positive rules would move value 1, 3 and 5 one position

up in the *Sequence* array and demotion would bring the values down in position. The same thing also occurs to values 2, 4 and 6 which represent the negative rules in the promotion and demotion sequences. However, the promotion or demotion of this set of 3 rules must maintain their sequence among themselves.

In a scenario where the student correctly executes a comparing sub-skill and the adaptive rules would promote the positive rules; namely: value 1, 3 and 5. Example 9 shows the changes before and after the promotion. If there is more than one rule to be promoted, the promotion of the rules will follow the order from highest to lowest position.

As shown in the example, the first rule of the sequence before the promotion is rule 6 which is ‘Non-linear Negative Exemplar Rule’. After the promotion, the first rule becomes rule 5 which is ‘Non-linear Positive Exemplar Rule’. Hence, the preference of this sequence switches from ‘negative’ approach to the ‘positive’ approach. This demonstrates the modification abilities of the intra-scheme rules.

Example 9

Before	: <i>Sequence</i> = {6, 5, 3, 4, 2, 1}
Promote Rule 5	: <i>Sequence</i> = {5, 6, 3, 4, 2, 1}
Promote Rule 3	: <i>Sequence</i> = {5, 3, 6, 4, 2, 1}
Promote Rule 1	: <i>Sequence</i> = {5, 3, 6, 4, 1, 2}
After	: <i>Sequence</i> = {5, 3, 6, 4, 1, 2}

8.4 Evaluation

This section presents the evaluation of A-EpiList. Instead of demonstrating the cognitive skills development abilities in EpiList I and EpiList II, the evaluation objective of A-EpiList is to demonstrate the tuning ability and effectiveness of A-EpiList.

8.4.1 Evaluation Procedure

A-EpiList is evaluated by a small group of five students. These students include two nine-year-old (primary 3), two ten-year-old (primary 4), and a eleven-year-old (primary 5) student. These five students were selected from average grade classes which essentially form a sample of the cohort. The two primary 3 students are labelled as ‘Student3a’ and ‘Student3b’. The two primary 4 students are labelled as ‘Student4a’ and ‘Student4b’. Lastly, the primary 5 student is labelled as ‘Student5a’.

As mentioned above, the cognitive skills competency of the student is of no significance to this evaluation, therefore the student is not required to complete any written test that derives skill competency. The student only has to use A-EpiList for 30 minutes. The student will be presented with the initialised order of the sequence array. A-EpiList will tuned the array accordingly during the execution of the program. In order to capture the tuning capabilities of A-EpiList, the system is programmed to capture and log the sequence of intra-scheme rules throughout the execution of the program. By examining the logs of the students, the adaptive or tuning capability of A-EpiList can be determined as well as the cognitive preferences of the student.

8.4.2 Evaluation results and analysis

Student	Final value of <i>Sequency</i> array
Student3a	{2, 6, 4, 5, 3, 1}
Student3b	{4, 2, 6, 1, 5, 3}
Student4a	{1, 2, 3, 5, 6, 4}
Student4b	{2, 6, 4, 1, 3, 5}
Student5	{3, 5, 1, 2, 6, 4}

Table 8.3: Evaluation results of A-EpiList

Table 8.3 shows the final value of the *Sequence* array of each student. These final values were extracted from the logs of the students. From the analysis of the results of Table 8.3, the primary 3 students had a cognitive preference for ‘negative’ rules. The top 3 rules of ‘Primary3a’ and ‘Primary3b’ are all ‘negative’ intra-scheme rules. This shows that the students have more difficulty comprehending the reasoning processes of the ‘positive’ rules than the elimination process in the ‘negative’ rules. A-EpiList tuned itself to suit the preference of the student by placing the ‘negative’ rules above the ‘positive’ rules.

Focusing on the results of ‘Student4a’, there is no significant preference as shown in the primary 3 students. However, A-EpiList tuned the order of the intra-scheme rules to place the direct rules with the top two priorities. This shows that this student has a preference for lower cognitive loads as the direct rules require the least cognitive load. From this result, A-EpiList does not only adjust to the student based on his abilities in performing comparison but also on his allowance for cognitive load.

Similar to the primary 3 students, ‘Student4b’ presented himself to be inclined towards ‘negative’ rules as well. Further analysis of the top 3 rules, which are all ‘negative’ rules, show that this student is inclined towards lower cognitive load as well. Lastly, the primary 5 student, ‘Student5’, produced a sequence that indicated he is more proficient in performing comparing sub-skills and efficient in handling higher cognitive loads.

Overall, this evaluation demonstrated the adaptivity and tuning capabilities of A-EpiList. With the lower skill competency and cognitive load capabilities of the lower primary students, A-EpiList adapted itself to teach the students with elimination rules using the least cognitive load. On the other hand, with the higher skill competency and cognitive load capability of higher primary students, A-EpiList presented to them rules that require higher cognitive load and higher reasoning process than the elimination process.

8.5 Summary

In this chapter, further enhancement is made to EpiList II based on system theories. Adaptation is employed in EpiList II to generate A-EpiList that has the flexibility to adapt and suit the preference of different students to maximise the learning outcome.

The enhancements made to EpiList II give rise to A-EpiList. The design of an adaptive system consists of 3 parts: identification, decision and modification. The detailed design of each part of A-EpiList is discussed in this chapter. This section also presents the evaluation of A-EpiList demonstrating the tuning capabilities of the rules.

Chapter 9

Conclusion

“There must be a beginning of any great matter,
but the continuing unto the end until it be
thoroughly finished yields the true glory.”

– Francis Drake

This research was sparked off by the importance of teaching cognitive skills to the student. It motivated the development of an ITS that also teaches and develops cognitive skills in the student. This research examined in depth the issues involved in the development of an ITS; focusing on its effects on ITS that deal with cognitive skill development. Investigations into educational research have identified that the most basic and critical cognitive skills include classification, generalisation and comparison skills. These are the skills focused by this research.

The Epistemic Forms and Games; namely the ‘List-making’ and ‘Compare-and-Contrast’ games, proposed by Prof Collins was selected as the basis of the ITS prototype to be developed. It was selected because they were specifically designed for the development of cognitive skills and were systematically described by Prof Collins which made them ideal for the use in this research. On top of that, the teaching model of inductive and deductive models defined by Eggen and Kauchak (1988) was incorporated as the teaching model in the system. The notion behind the selection is that these models use both generalisation and comparison skills to accomplish their

teaching goals.

Based on these educational theories and models, a prototype of the ITS that attempted to explicitly develop cognitive skills was crafted. This prototype was termed 'EpiList'. EpiList I was the first generation of EpiList. It explicitly develops the content knowledge of the student by involving the student in a list-making process. During the tutorial, the student was tutored on his misclassification or misconception using the inductive and deductive models. Hence, EpiList I implicitly develops generalisation and comparison skills through the tutorials.

A field-evaluation was conducted for EpiList I. A total of 300 students took part in the evaluation of EpiList I. Pre- and post-evaluation tests capture the cognitive skill competency level of the student before and after the use of EpiList I. The evaluation results empirically showed the capabilities of EpiList I in enhancing the efficiency of the student in performing cognitive skills. The evaluation results showed an overall 20% improvement in the competency level of the students. It demonstrated the capabilities of an ITS to develop generic cognitive skills in a one-to-one interaction. Nevertheless, this low value highlighted the inefficiency and limitations of EpiList I.

Besides the development of EpiList I, the research also proposed 2 methodologies to assist in the design, definition, development and analysis of ITS. The first methodology was the formal definition of ITS in terms of simple sets and set operations while the second methodology was the system view and analysis of ITS. These two methods were demonstrated by applying them to analyse and interpret EpiList I.

The formal definition using simple sets and set operations was successfully applied to the description of EpiList I. This allowed the system to be easily, precisely and unambiguously understood by everyone. It was a methodology that was neither non-verbose nor overtly technical such that both educators and engineers could readily comprehend the inner structure of the system. The system view and analysis enabled EpiList I to be enhanced using the paradigm of system theories. The limitations and problems of EpiList I were also highlighted through system theories.

EpiList II is version II of EpiList, an enhanced EpiList I. It explicitly developed the student's cognitive skills through a process of closed-loop instructions. The design and development of EpiList II was based on system theories and attempted to overcome the limitations inherent in EpiList I. At the same time, the design and architecture of EpiList II was also formally defined in terms of the simple sets and set operations. This demonstrated the effective application of the formal definition and system representation methodologies.

The evaluation of EpiList II similarly highlighted the capabilities of EpiList in the explicit development of cognitive skills. From the evaluations, it is learnt that the cohort whom evaluated EpiList II is more competence in cognitive skills than the cohort whom evaluated EpiList I. Nevertheless, this evaluation showed a significant increment in the acquisition of the reasoning skills. The evaluation of EpiList II was undertaken in a controlled environment. The objective of having a controlled environment was to ensure that the results of the evaluation was correct and uninfluenced by external factors.

In additional, further enhancements were applied to EpiList II based on the adaptive system of system theories. The enhanced system was called A-EpiList. A-EpiList was an adaptive system that employed information from the cognitive closed-loop instructions in EpiList II to adapt and adjust the content of the closed-loop instructions in EpiList I. Similarly, the adaptive system was formally defined using simple sets and set operations. The design and development of A-EpiList further demonstrated the ease of application and advantage of the system representation and formal definition.

9.1 Contribution and Significance

The main focus of this work is to develop an ITS that teaches cognitive skills. The research efforts in this direction successfully incorporated instructions that teach specific cognitive skills. Such skills include mathematical and scientific analytical

skills, electronic circuit troubleshooting skills and programming skills. This work goes a step further by developing a generic ITS that teaches generic cognitive skills. EpiList successfully incorporates pedagogical models and instructions into computational algorithms to facilitate cognitive skills development. Although the cognitive skills developed by EpiList; namely: classification, generalisation and comparison skills, are primitive cognitive skills, it successfully demonstrates the capabilities of an ITS to develop generic cognitive skills.

Over the years, educators have complained that the difficulties in comprehending ITS and its teaching strategies would make the evaluation and analysis of ITS tedious. At the same time, the translation or implementation of teaching strategies into practical computational models is also tedious and complicated. This work proposes a formal definition methodology that would bridge the gap between educators and engineers. This formal definition uses simple sets and set operations to represent the computational algorithms or instructional sequences in an ITS. The unambiguous nature of sets and set operations makes the formally defined ITS easily and unambiguously understood. This bridge goes both ways, educators can use this formal definition methodology to describe pedagogical theories and models to be readily available for implementation in computational algorithm in ITS.

The development of several ITS is carried out in an ad-hoc approach. There is generally no systematic design or development of ITS. Due to this ad-hoc development, analysis or improvement of the ITS is difficult or even impossible. This work proposes a design and analysis methodology that allows systematic design and development of an ITS. This methodology is based on sound system theories. The modules of an ITS are mapped into the system theory paradigm to derive a system view of ITS. An ITS designed and developed using sound system theories can be analysed and enhanced.

Employing the design and analysis methodology that uses system theory in EpiList, each version of EpiList can be analysed and enhanced to generate layered versions

of EpiList. The system enhancement on EpiList I gives rise to EpiList II, similarly, it gives rise to A-EpiList from EpiList II. EpiList I is a closed-loop content instructions. EpiList II is an enhancement of EpiList I to include a closed-loop cognitive instructions. The next enhancement on EpiList II adds a layer of adaptive control that essentially forms A-EpiList. All 3 layers of EpiList are designed using system theories. Furthermore, they are formally defined with simple sets and set operations for clearer and simpler comprehension of the systems.

The field evaluations on the 3 versions of EpiList categorically demonstrates the correctness and effectiveness of EpiList I, EpiList II and A-EpiList. On top of that, the evaluation includes 1-to-1 interviews and queries on selected students to articulate their thinking processes. This validates that the instructions can guide the student in performing cognitive skills. From this evaluation, this work also proposes an evaluation process that can evaluate ITS which employ instructions that teach generic cognitive skills.

9.2 Limitations and Future Work

Currently, the generic cognitive skills taught by this work are classification, generalisation and comparison skills. Such skills are the most fundamental of generic cognitive skills. Although these skills are important for subsequent higher level cognitive skills, it is essential for ITS to incorporate instructions that teach higher level cognitive skills. The next cognitive skill that can be taught and developed is causal (cause-and-effect) analytical skills. To accomplish such development, pedagogical theories and models that teach such skills have to be analysed and translated into computational algorithms for the implementation of ITS. Formal definition methodology can be employed at this stage to describe the pedagogical models. This would simplify the comprehensiveness of the models for implementation.

The simple sets and set operations presented in this work only utilises basic oper-

ations of sets. As the complexities and difficulties of the cognitive skills to be taught increase, more complex set operations are necessary to formally define the skills. Apart from simple sets, more complicated set representation, for example relations in sets, can be employed to formally define the causal relationship. On top of that, higher-order logics can be used to describe more complex instructions.

Based on the system theories, further enhancements can be made to EpiList. According to system theories, there are four class of systems: open-loop, closed-loop, adaptive and learning. EpiList incorporated the first three classes of control systems and the implementation of each class brings forth an enhancement in EpiList. The fourth class of system, the learning system, could be implemented to further enhance the instruction's teaching capabilities. However, such implementation requires tremendous efforts and time. A simpler enhancement is to develop a multi-layered system that consists of more than one closed-loop controlling the same aspect of the plant. In terms of ITS, a multi-layered ITS is a system that incorporates more than one pedagogical model to tutor the student on one mistake. This implementation is more visible. However supervisory rules are necessary between the layers of control.

References

- Addison, K. L., & Yakimowski, M. E. (2003). *An evaluation of the direct instruction program*. http://www.bcps.k12.md.us/Student_Performance/Program_Evaluation/direct_instruction.asp. (Last visited: 24th Dec 2004)
- Adkins, J. (1997). *Metacognition: Designing for transfer*. Unpublished master's thesis, University of Saskatchewan.
- Aggarawal, K. K. (1981). *Control systems analysis and designs*. Khanna Publishers.
- Ang, H. M. (1999). *EpiComp: An ITS Shell for Multiple Concepts Compare and Contrast Game* (Final Year Report). Nanyang Technological University, Singapore.
- Barr, A., & Feigenbaum, E. A. (1981). *The handbook of artificial intelligence*. Heuristics Tech Press.
- Belkada, S., Cristea, A. I., & Okamoto, T. (2001). Discovernet: Adaptive simulation-based learning environment and support system for designing neural networks. *Journal of Educational Technology Research*.
- Blakey, E., & Spence, S. (1990). *Developing metacognition*. ERIC Clearinghouse on Information Resources, Syracuse NY. (ERIC Identifier: ED327218)
- Bloom, B. S. (1956). *Taxonomy of Educational Objectives: Book 1 Cognitive Domain* (B. S. Bloom, Ed.). Longman, London.
- Boulton-Lewis, G. M., & Halford, G. S. (1991). Processing capacity and school learning. In G. Evans (Ed.), *Learning and teaching cognitive skills* (p. 27-50). Australian Council for Educational Research Ltd.

- Brady, L. (1985). *Model and methods of teaching*. Prentice-hall of Australia PTE LTD.
- Brown, J. S., Burton, R. R., & Kler, J. D. (1982). Pedagogical, natural language and knowledge engineering techniques in sophie i, ii and iii. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (p. 227-282). Academic Press.
- Bruner, J. (1966). *Toward a theory of instruction*. Cambridge: Harvard University Press.
- Burns, H., & Parlett, J. W. (1991). Intelligent tutoring systems: Evolutions in design. In H. Burns, J. W. Parlett, & C. L. Redfield (Eds.), *The evolution of intelligent tutoring systems: Dimensions of design*. Lawrence Erlbaum Associates Publishers.
- Burton, R. R., & Kler, J. de. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (p. 79-98). Academic Press.
- Cheng, P. C. H. (1999). Unlocking conceptual learning in mathematics and science with effective representation systems. *Computer and Education*, 33, 109-130.
- Clancey, W. J. (1982). Tutoring rules for guiding a case method dialogue. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (p. 201-223). Academic Press.
- Clarke, J. H., & Biddle, A. W. (1993). *Teaching critical thinking: reports from across the curriculum*. Prentice-Hall, Inc.
- Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator, The Journal of the American Federation of Teachers*.
- Collins, A., & Ferguson, W. (1993). Epistemic forms and epistemic games: Structures and strategies to guide inquiry. *Education Psychologist*, 28(1), 25-42.
- Collins, A., & Stevens, A. L. (1983). A cognitive theory of inquiry teaching. In C. M. Reigeluth (Ed.), *Instructional design theories and models: An overview*

- (p. 247-278). Hillsdale, NJ: Erlbaum.
- Cooper, G. (1998). *Research into cognitive load theory and instructional design at UNSW*. http://education.arts.unsw.edu.au/CLT_NET_Aug_97.HTML. (Last visited: 9th Oct 2004)
- Craig, G. P., & Reed, B. J. (2004). *Counterpoint: Cognitivism*. http://www.educ.drake.edu/doc/pedagogy/student_papers/craig_reed/def-cog.html. (Last accessed: 08 September 2004)
- DeMar, G. (1988). *Behaviorism*. http://www.forerunner.com/forerunner/X0497_DeMar_-_Behaviorism.html. (Last accessed: 08 September 2004)
- Drescher, G. L. (1987). Object-oriented logo. In *Artificial intelligence and education* (Vol. 1, p. 153-166). Ablex Publishing Corporation.
- Eggen, P. D., & Kauchak, D. P. (1988). *Strategies for teachers: teaching content and thinking skills*. Prentice-Hall.
- Elliot, S. N., Kratochwill, T. R., Cook, J. L., & Travers, J. F. (2000). *Educational psychology: Effective teaching, effective learning*. McGraw-Hill.
- Ennis, R. H. (1987). A taxonomy of critical thinking dispositions and abilities. In J. B. Baron & R. J. Sternberg (Eds.), *Teaching thinking skills: Theory and practice* (p. 9-26). W.H. Freeman and Company.
- Faria, L., Vale, Z., Ramos, C., & Marques, A. (2000). An its for control centre operators training: Issues concerning knowledge representation and training scenarios generation. *IEEE Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*.
- Goh, G. M. (2003). *Evaluation results of epilist* (Technical Report No. ISL-TR-01/03). School of Computer Engineering, Nanyang Technological University, Singapore.
- Goldstein, I. P. (1982). The genetic graph: a representation for the evolution of procedural knowledge. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (p. 51-77). Academic Press.

- Good, T. L., & Levin, J. R. (2001). Educational psychology yesterday, today and tomorrow: Debate and direction in an evolving field. *Educational Psychologist*, 36(2), 69-72.
- Goswami, U. (1998). *Cognition in children*. Psychology Press.
- Grigoriadou, M., Papanikolaou, K., Kornilakis, H., & Magoulas, G. (2001). Inspire: an intelligent system for personalized instruction in a remote environment. *Proceedings of Third Workshop on Adaptive Hypertext and Hypermedia*, 13-24.
- Gupta, S. D. (1983). *Control system theory*. Khanna Publishers.
- Hergenhahn, B. R., & Olson, M. H. (1997). *An introduction to theories of learning*. Prentice-Hall, Inc.
- Kellogg, R. T. (1995). *Cognitive psychology*. SAGE Publication.
- Kinshuk, Tretiakov, A., Hong, H., & Patel, A. (2001). Human teacher in intelligent tutoring system: a forgotten entity. In *IEEE International Conference on Advanced Learning Technologies*.
- Kirsner, K., Speelman, C., Maybery, M., O'brien-Malone, A., Anderson, M., & MacLeod, C. (1998). *Implicit and explicit mental processes* (K. Kirsner, C. Speelman, M. Maybery, A. O'brien-Malone, M. Anderson, & C. MacLeod, Eds.). Lawrence Erlbaum Associates, Publishers.
- Kolar, S., & D'Ambrosio, L. (2002). *Vygotsky resources*. <http://www.kolar.org/vygotsky/>. (Last accessed: 08 September 2004)
- Lefkowitz, L. S., Farrell, R., & Yoo, I. S. (1993, May). A knowledge-based intelligent tutoring system for telephony operations support systems. *IEEE International Conference on Communications*, 3, 1880 - 1884.
- Leitch, R., Shen, Q., Coghill, G., & Chantler, M. (1999). Choosing the right model. *IEE Proceedings of Control Theory Applications*, 146(5), 435-449.
- Maciejewski, A. A., & Kang, Y. S. (1994). A student model of katakana reading proficiency for japanese language intelligent tutoring system. *IEEE Trans. Syst., Man, Cybern.*, 24(9).

- Mayer, R. E. (2001). What good is educational psychology? the case of cognition and instruction. *Educational Psychologist*, 36(2), 83-88.
- McFarland, T. D., & Parker, R. (1990). *Expert system in education and training*. New Jersey: Educational Technology Publications, Inc.
- Mitchell, C. M., Chappell, A. R., Gray, W. M., Quinn, A. B., & Thurman, D. A. (2000). OFMspert II: intelligent tutoring for complex control. *IEEE Trans. Syst., Man, Cybern.*, 2, 1152-1157.
- Ng, F., Butler, G., & Kay, J. (1995, May). An intelligent tutoring system for the dijkstra-gries methodology. *IEEE Trans. Software Eng.*, 21(5), 415-428.
- Nixon, M. (1971). *Children's classification skills*. Australian Council for Educational Research.
- Nkambou, R. (1997). Using fuzzy logic in its-course generation. *Ninth IEEE International Conference on Tools with Artificial Intelligence*, 190 - 193.
- Nour, M., Abed, E., & Hegazi, N. (1995). *A proposed student model algorithm for an intelligent tutoring system* (Internal Report). SICE, Sapporo.
- Nussbaum, M., Rosas, R., Peirano, I., & Cardenas, F. (2001). Development of intelligent tutoring systems using knowledge structures. *Computer and Education*, 36, 15-32.
- Nwana, H. S. (1991). FITS: a fraction intelligent tutoring system. In *Proceedings ninth national conference on artificial intelligence (aaai-91)* (Vol. 1, p. 49-54).
- O'Brien-Malone, A., & Maybery, M. (1998). Implicit learning. In K. Kirsner, C. Speelman, M. Maybery, A. O'brien-Malone, M. Anderson, & C. MacLeod (Eds.), *Implicit and explicit mental processes* (p. 37-55). Lawrence Erlbaum Associates, Publishers.
- Ogata, K. (1997). *Modern control engineering*. Prentice-Hall, Inc.
- Okamoto, T. (2000). RAPSODY: distance ecological model for self- and collaborative-learning. *International Workshop on Advanced Learning Technologies*, 249-250.
- Papert, S. (1987). Microworlds: Transforming education. In R. W. Lawler & M. Yaz-

- dani (Eds.), *Artificial intelligence and education* (Vol. One, p. 79-94). Ablex Publishing Corporation, USA.
- Peachey, D. R., & McCalla, G. I. (1986). Using planning techniques in intelligent tutoring systems. *Int. Journal of M-M Studies*, 24.
- Reeves, T. (1997). *Evaluating what really matters in computer-based education*. <http://www.educationau.edu.au/archives/cp/REFS/reeves.htm>. (Last accessed: 05 July 2004)
- Saskatchewan Education. (1991). *Instructional approaches: A framework for professional practice*. <http://www.sasked.gov.sk.ca/docs/policy/approach/index.html>. Regina, SK: Saskatchewan Education. (Last visited: 24th Dec 2004)
- Saskatchewan Learning. (2004). *Indirect instruction*. <http://www.sasked.gov.sk.ca/docs/wellness/indirect.html>. Regina, SK: Saskatchewan Learning. (Last visited: 24th Dec 2004)
- Sleeman, D. (1987). PIXIE: A shell for developing intelligent tutoring systems. In *Artificial intelligence and education vol 1* (p. 239-265). Ablex Publishing Corporation.
- Sleeman, D. H., & Brown, J. S. (1982). *Intelligent tutoring systems*. Academic Press, London.
- Smith, L. (2000). *A brief biography of jean piaget*. <http://www.piaget.org/aboutPiaget.html>. (Last accessed: 08 September 2004)
- Smith, P. L., & Ragan, T. J. (1999). *Instructional design*. Prentice-Hall, Inc.
- Smith, R. L., Graves, H., Blaine, L. H., & Marinov, V. G. (1975). Computer-assisted axiomatic mathematics: information rigor. In *Computer in education part 1: IFIP*. Amsterdam: North Holland.
- Solso, R. L. (1993). *Cognitive psychology*. Allyn & Bacon.
- Stathacopoulou, R., Magoulas, G. D., & Grigoriadou, M. (1999, July). Neural network-based fuzzy modeling of the student in intelligent tutoring systems. *IJCNN: International Joint Conference on Neural Networks*, 5, 3517 - 3521.

- Stevens, A., Collins, A., & Goldin, S. E. (1982). Misconceptions in students' understanding. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (p. 13-24). Academic Press.
- The PSI Cafe. (2001). *Jerome bruner*. <http://www.psy.pdx.edu/PsiCafe/KeyTheorists/Bruner.htm>. (Last accessed: 08 September 2004)
- Vanlehn, K. (1996). Cognitive skill acquisition. *Annual Review of Psychology*, 47, 513-39.
- Vasandani, V., & Govindaraj, T. (1990). Knowledge representation and human-computer interaction in an intelligent tutor for diagnostic problem solving. *IEEE International Conference on Systems, Man and Cybernetics*, 665-667.
- Vasandani, V., & Govindaraj, T. (1991, Oct). Intelligent diagnostic problem solving tutor: an experimental evaluation. *IEEE International Conference on Systems, Man, and Cybernetics*, 3, 1739-1744.
- Vasandani, V., & Govindaraj, T. (1995, July). Knowledge organization in intelligent tutoring systems for diagnostic problem solving in complex dynamic domains. *IEEE Trans. Syst., Man, Cybern.*, 25(7), 1076-1096.
- Vasandani, V., Govindaraj, T., & Mitchell, C. M. (1989). An intelligent tutor for diagnostic problem solving in complex dynamic systems. *IEEE International Conference on Systems, Man and Cybernetics*, 2, 772-777.
- Way, E. C. (1991). *Knowledge representation and metaphor*. Kluwer Academic Publishers.
- Wenger, E. (1987). *Artificial intelligence and tutoring system*. Morgan Kaufmann Publishers, Inc, California.
- Williams, M. D. (2000). *Integrating technology into teaching and learning: Concepts and applications*. Prentice Hall.
- Wilson, B. G., & Cole, P. (1996). Cognitive teaching models. In D. H. Jonassen (Ed.), *Handbook of research in instructional technology*. New York: Scholastic Press.

- Wong, L. H. (1998). *A structured system's view of an intelligent tutoring system based on inquiry teaching approach*. Unpublished doctoral dissertation, School of Applied Science, Nanyang Technological University, Singapore.
- Wong, L. H., Quek, C., & Looi, C. K. (1998a). Tap-2: A framework for an inquiry dialogue based tutoring system. *International Journal of Artificial Intelligence in Education*, 9, 88-110.
- Wong, L. H., Quek, C., & Looi, C. K. (1998b). TAP: a software architecture for an inquiry dialogue-based tutoring system. *IEEE Trans. Syst., Man, Cybern. A*, 28(3), 315-325.
- Woolfolk, A. E. (1995). *Educational psychology*. Allyn and Bacon.
- Wynne, C. D. L. (1998). A natural history of explicit learning and memory. In K. Kirsner, C. Speelman, M. Maybery, A. O'brien-Malone, M. Anderson, & C. MacLeod (Eds.), *Implicit and explicit mental processes* (p. 255-269). Lawrence Erlbaum Associates, Publishers.
- Yoshikawa, A., Shintani, M., & Ohba, Y. (1992, Aug). Intelligent tutoring system for electric circuit exercising. *IEEE Trans. Educ.*, 35(3), 222-225.

Appendix A

EpiList I's Evaluation Questions

G.M. Goh, C. Quek

Teacher's Profile

Name:

Age:

Highest qualification:
(With specialization if available)

Number of years teaching:

Current subjects teaching:

Level of students teaching:

G.M. Goh, C. Quek

Teacher Questionnaires

Section A – Highlighting of mistakes

1. Do the tutorials highlight the mistakes clearly?
Yes No
2. Is the highlighting of mistakes presented systematically?
Yes No
3. Do you think the students could induce their mistakes from the tutorials?
Yes No
4. Could EpiList rectify the students' misconception regarding the animal kingdom?
Yes No
5. Are the instructions of the tutorials clear and precise?
Yes No
6. Do the instructions of the tutorials sufficiently guide the students to identify their mistakes?
Yes No
7. For each tutorial, do you feel that different students would have different level of understanding?
Yes No

Section B – Migration of classification scheme

1. Is the introduction of the important classes carried out clearly? (Important classes are classes to be taught to the student in each session of EpiList)
Yes No
2. Is the instruction to move animals into important classes clear and understandable?
Yes No
3. Is the migration from one classification scheme to another classification scheme carried out smoothly?
Yes No

G.M. Goh, C. Quek

4. At each step of the tutorials, do you know specifically which skill is being used or you are involved in?

YesNo
5. Do the instructions of the tutorials clearly explain the usage of animals as examples?

YesNo
6. Do the explicit explanations of the skills used in the tutorials improve the students' effectiveness in acquiring the skills?

YesNo
7. Would you prefer to be able to design the instructional sequence and explanation of EpiList?

YesNo

Section E – Assessment

1. Are the questions of the assessments clear and understandable?
Yes No
2. Are the scenarios of the assessments easy to understand?
Yes No
3. Do the “Show items belonging to the same class” and “Show tutorial” features assist in answering the questions?
Yes No
4. Do the assessments specifically test on the students’ understanding of the tutorials?
Yes No
5. Could the students’ level of understanding of the tutorials be determined by using the assessments?
Yes No

Section F – General

- Are the detailed texts clear and easy to understand?
Yes No
- Do the detailed texts provide sufficient information to learn about the animals?
Yes No

G.M. Goh, C. Quek

3. Are the detailed texts presented at the tutorials helpful to the students while answering the MCQs?

Yes

No

4. Is the interface of EpiList friendly and easy to use?

Yes

No

5. Do you think that EpiList serve as a good teaching tool to teach the skills of classification, generalization and comparison?

Yes

No

6. Would you use EpiList to teach these skills?

Yes

No

7. What other comment do you have?

G.M. Goh, C. Quek

Pre-Evaluation Test

Name: _____ Class: _____

- Q1. Which of the following properties is the common property of Tiger, Lion, Dog and Horse?
- ☐ They can fly.
 - ☐ They are cold blooded.
 - ☐ They live on land.
 - ☐ They have scales on their skin.
- Q2. Using the four animals of Question 1, which of the following are NOT their common properties?
- ☐ They are carnivorous.
 - ☐ They are mammals.
 - ☐ They have lungs.
 - ☐ They are warm blooded.
- Q3. Which of the following properties is the common property of Whale, Bat, Dog and Crow?
- ☐ They live on land.
 - ☐ They are warm blooded.
 - ☐ They can swim.
 - ☐ They all lay eggs.
- Q4. Which of the following animal is the odd-one-out?
- ☐ Tiger
 - ☐ Frog
 - ☐ Tortoise
 - ☐ Angelfish
- Q5. Which of the following sentences best describe the animal that is odd-one-out from Question 4?
- ☐ It lives in water.
 - ☐ It lays eggs.
 - ☐ It has four legs.
 - ☐ It is warm blooded.
- Q6. The animals Angelfish, Whale, Dog and Frog can be classified into two groups with 2 items in each group. Which of the following are the two groups?
- ☐ "Give birth to life young" and "Give birth through laying eggs"
 - ☐ "Live in water" and "Live on land"
 - ☐ "Breathe through lungs" and "Breathe through gills"
 - ☐ "Carnivorous" and "Herbivorous"

G.M. Goh, C. Quek

This diagram will be used for Questions 7 to 10.

Group 1	Group 2	Group 3	Group 4
Dog	Frog	Crow	Angelfish
Lion	Lizard	Bat	Whale
Tiger	Tortoise	Horse	Shark
Cat	Snake	Dolphin	Turtle

Q7. Which 2 Groups have the same common property?

- ☐ Group 1 and 2.
- ☐ Group 1 and 3.
- ☐ Group 1 and 4.
- ☐ Group 2 and 3.
- ☐ Group 2 and 4.
- ☐ Group 3 and 4.

Q8. What is the common property of the 2 groups in Question 7?

- ☐ They live on land.
- ☐ They are cold blooded.
- ☐ They give birth through laying eggs.
- ☐ None of the above

Q9. What is the common property of Group 4?

- ☐ They are cold blooded.
- ☐ They live in water.
- ☐ They give birth to live young.
- ☐ They breathe through gills.

Q10. Could the common property of Group 4 be the common property of the other groups?

- ☐ Yes.
- ☐ No.

Q11. List down 2 cold-blooded animals that live on land.

Q12. All reptile and _____ are cold blooded whereas all _____ and birds are warm blooded.

Q13. Cat, snake, dolphin and turtle are examples of animal that

_____.

Q14. Carnivorous, _____ and _____ belong to the same higher-order property of "Food".

Q15. The common property of Snake, Crow and Angelfish is

_____.

G.M. Goh, C. Quek

For Question 16 to 20, answer if the sentences are True or False.

Q16. All mammals live on land.

☐ True ☐ False

Q17. Lizards and Snakes are reptiles; therefore they are cold blooded.

☐ True ☐ False

Q18. Dogs, Lions, Cats, Whale, and Horses give birth to life young.

☐ True ☐ False

Q19. Whale and Shark belong to the same family because they both live in water.

☐ True ☐ False

Q20. All reptiles are cold blooded. Snake is a reptile therefore it is cold blooded.

☐ True ☐ False

G.M. Goh, C. Quek

Post-Evaluation Test(Primary 4)

Name: _____ Class: _____

- Q1. Which of the following animal is the odd-one-out?
- ☐ Tiger
 - ☐ Deer
 - ☐ Bat
 - ☐ Kangaroo
- Q2. Which of the following sentences best describe the animal that is odd-one-out from Question 1?
- ☐ It is carnivorous.
 - ☐ It is warm blooded.
 - ☐ It can fly.
 - ☐ It lives on land.
- Q3. Excluding the odd-one-out animal from Question 1, which of the following properties is the common property of the remaining 3 animals?
- ☐ They are herbivorous.
 - ☐ They are carnivorous.
 - ☐ They are omnivorous.
 - ☐ None of the above.
- Q4. Which of the following pairs of animals have the same common property as Catfish and Whale?
- ☐ Frog and Lizard.
 - ☐ Crocodile and Angelfish.
 - ☐ Cobra and Viper.
 - ☐ Crow and Swam.
- Q5. Which of the following animals CAN be grouped together with the animals in Question 5.
- ☐ Tiger.
 - ☐ Deer.
 - ☐ Ostrich.
 - ☐ Turtle.
- Q6. Which of the following sets of properties CANNOT group all animals in the animal kingdom?
- ☐ “Live in air” and “Live in water”.
 - ☐ “Cold blooded” and “Warm blooded”.
 - ☐ “Carnivorous”, “Herbivorous” and “Omnivorous”.
 - ☐ “Breathe through lungs” and “Breathe through gills”.
- Q7. Which of the following is not characteristic of the different types of classes?
- ☐ They have the same higher-order property.
 - ☐ They can group all the animals.
 - ☐ Each animal can only be grouped in one important class
 - ☐ None of the above.

G.M. Goh, C. Quek

This diagram will be used for Questions 8 to 13.

Group 1	Group 2	Group 3	Group 4
Bat	Cobra	Crow	Frog
Tiger	Lizard	Duck	Dolphin
Deer	Catfish	Emu	Shark
Kangaroo	Batfish	Ostrich	Slowworm

- Q8. Which of the followings is not the common property of Groups 1 and 3?
- ☐ They are warm blooded.
 - ☐ They breathe through lungs.
 - ☐ They live on land.
 - ☐ They give birth to live young.
- Q9. Which of the following families best describe Group 3?
- ☐ Mammals.
 - ☐ Reptiles.
 - ☐ Birds.
 - ☐ Fishes.
- Q10. Which of the following properties is the common property of the Group 2?
- ☐ They live in water.
 - ☐ They have scales on their skins.
 - ☐ They are warm blooded.
 - ☐ They breathe through lungs.
- Q11. Which of the following properties can also be is the common property of the Group 2?
- ☐ They are cold blooded.
 - ☐ They live on land.
 - ☐ They have feathers on their skins.
 - ☐ They breathe through gills.
- Q12. If “Angelfish” is incorrectly grouped in Group 4, which group is the correct group for this animal?
- ☐ Group 1.
 - ☐ Group 2.
 - ☐ Group 3.
 - ☐ Group 4.
- Q13. If one of your friends has made the mistake mentioned in Question 12, what would you do to help your friend to identify the mistake?

G.M. Goh, C. Quek

Q14. List down 2 warm blooded animals that live in water.

Q15. _____ and _____ belong to the type of class “Body temperature”.

Use the following helping words to answer questions 16 and 17:

correctly incorrectly categorizing
compare contrasting generalizing

Q16. When you arrange a set of animals into classes, you are _____ the properties of the animals.

Q17. The tutorials of EpiList would first generalize the _____

grouped animals and _____ the common property with the mistakes.

For Question 18 to 21, answer if the sentences are True or False.

Q18. “Birth to life young” and “Lay eggs” belongs to the same type of classes.

☐ True ☐ False

Q19. If there is no correctly grouped animal, the best example will be used.

☐ True ☐ False

Q20. “Cold blooded” and “Give birth through laying eggs” belong to the same type of classes.

☐ True ☐ False

Q21. Mistakes with same property will be highlighted one at a time.

☐ True ☐ False

Q22. If you have incorrectly grouped Tiger in the class “Live in water” but correctly grouped Deer in the class “Live on land”. Describe how EpiList would highlight this mistake.

Q23. Does EpiList improve your generalization and comparison skills?

☐ No ☐ Slightly
☐ Yes ☐ Greatly

G.M. Goh, C. Quek

Q24. Does the program, EpiList, allow you to exercise your thinking process?

☐ Yes ☐ No

In your own words, explain your thinking process.

Q25. What is your opinion regarding EpiList?

Q26. Do you agree that EpiList is a useful tool to help you to understand the reasoning processes during a classification exercise?

☐ No ☐ Slightly
☐ Yes ☐ Greatly

Please explain your reason(s).

Q27. Would you like to see or use similar reasoning tools in the teaching of science based subjects?

☐ No ☐ No sure
☐ Yes ☐ Some of the topics only

Q28. Which topic do you think EpiList is useful in?

G.M. Goh, C. Quek

Post-Evaluation Test (Primary 5)

Name: _____ Class: _____

- Q1. Which of the following animal is the odd-one-out?
- ☐ Tiger
 - ☐ Deer
 - ☐ Bat
 - ☐ Kangaroo
- Q2. Which of the following sentences best describe the animal that is odd-one-out from Question 1?
- ☐ It is carnivorous.
 - ☐ It is warm blooded.
 - ☐ It can fly.
 - ☐ It lives on land.
- Q3. Excluding the odd-one-out animal from Question 1, which of the following properties is the common property of the remaining 3 animals?
- ☐ They are herbivorous.
 - ☐ They are carnivorous.
 - ☐ They are omnivorous.
 - ☐ None of the above.
- Q4. Which of the following pairs of animals have the same common property as Catfish and Whale?
- ☐ Frog and Lizard.
 - ☐ Crocodile and Angelfish.
 - ☐ Cobra and Viper.
 - ☐ Crow and Swam.
- Q5. Which of the following animals CAN be grouped together with the animals in Question 5.
- ☐ Tiger.
 - ☐ Deer.
 - ☐ Ostrich.
 - ☐ Turtle.
- Q6. Which of the following sets of properties CANNOT group all animals in the animal kingdom?
- ☐ "Live in air" and "Live in water".
 - ☐ "Cold blooded" and "Warm blooded".
 - ☐ "Carnivorous", "Herbivorous" and "Omnivorous".
 - ☐ "Breathe through lungs" and "Breathe through gills".
- Q7. Which of the following is not characteristic of the important classes?
- ☐ They have the same higher-order property.
 - ☐ They can group all the animals.
 - ☐ Each animal can only be grouped in one important class
 - ☐ None of the above.

G.M. Goh, C. Quek

This diagram will be used for Questions 8 to 13.

Group 1	Group 2	Group 3	Group 4
Bat	Cobra	Crow	Frog
Tiger	Lizard	Duck	Dolphin
Deer	Catfish	Emu	Shark
Kangaroo	Batfish	Ostrich	Slowworm

- Q8. Which of the followings is not the common property of Groups 1 and 3?
- ☐ They are warm blooded.
 - ☐ They breathe through lungs.
 - ☐ They live on land.
 - ☐ They give birth to live young.
- Q9. Which of the following families best describe Group 3?
- ☐ Mammals.
 - ☐ Reptiles.
 - ☐ Birds.
 - ☐ Fishes.
- Q10. Which of the following properties is the common property of the Group 2?
- ☐ They live in water.
 - ☐ They have scales on their skins.
 - ☐ They are warm blooded.
 - ☐ They breathe through lungs.
- Q11. Based on the common property of Question 10, which of the following properties has the same higher-order property?
- ☐ They are cold blooded.
 - ☐ They live on land.
 - ☐ They have feathers on their skins.
 - ☐ They breathe through gills.
- Q12. If “Angelfish” is incorrectly grouped in Group 4, which group is the correct group for this animal?
- ☐ Group 1.
 - ☐ Group 2.
 - ☐ Group 3.
 - ☐ Group 4.
- Q13. If one of your friends has made the mistake mentioned in Question 12, what would you do to help your friend to identify the mistake?

G.M. Goh, C. Quek

Q14. List down 2 warm blooded animals that live in water.

Q15. _____ and _____ belong to the same higher-order property of “Body temperature”.

Use the following helping words to answer questions 16 and 17:

correctly incorrectly categorizing
compare contrasting generalizing

Q16. When you are finding a common property among the animals, you are

_____ the animals.

Q17. The tutorials of EpiList would first generalize the _____

grouped animals and _____ the common property with the mistakes.

For Question 18 to 21, answer if the sentences are True or False.

Q18. “Birth to life young” and “Lay eggs” belongs to the same higher-order class.

☐ True ☐ False

Q19. It is preferred to group animals in the important classes.

☐ True ☐ False

Q20. If an important class is empty, it should be removed.

☐ True ☐ False

Q21. The important classes have similar higher-order property.

☐ True ☐ False

Q22. If you have incorrectly grouped Tiger in the class “Live in water” but correctly grouped Deer in the class “Live on land”. Describe how EpiList would highlight this mistake.

Q23. Does EpiList improve your generalization and comparison skills?

☐ No ☐ Slightly
☐ Yes ☐ Greatly

G.M. Goh, C. Quek

Q24. Does the program, EpiList, allow you to exercise your thinking process?

☐ Yes ☐ No

In your own words, explain your thinking process.

Q25. What is your opinion regarding EpiList?

Q26. Do you agree that EpiList is a useful tool to help you to understand the reasoning processes during a classification exercise?

☐ No ☐ Slightly
☐ Yes ☐ Greatly

Please explain your reason(s).

Q27. Would you like to see or use similar reasoning tools in the teaching of science based subjects?

☐ No ☐ No sure
☐ Yes ☐ Some of the topics only

Q28. Which topic do you think EpiList is useful in?

EpiList II's Evaluation Questions

G.M. Goh, C. Quek

Controlled MGS Pre-Evaluation Test

Name: _____

Class: _____

Section A – Multiple Choice Questions

1. Which of the following categories does a fish belong to?
☐ Cold blooded ☐ Live on land
☐ Omnivorous ☐ Breathe through lungs
2. A lion and a dolphin both belong to which of the following categories?
☐ Cold blooded ☐ Live on land
☐ Omnivorous ☐ Breathe through lungs
3. Do “Dolphin”, “Tiger”, “Whale” and “Crow” belong to the category “Warm Blooded”?
☐ Yes ☐ No
4. Which of the following sentences is correct?
☐ Both crocodile and whale are cold-blooded.
☐ Both crocodile and turtle live on land.
☐ Both bear and turtle are warm blooded.
☐ Both turtle and whale are live in water.
5. Which one of the animals below is the odd-one-out?
☐ Lion ☐ Shark
☐ Snake ☐ Frog
6. Which of the categories below does the odd-one-out animal of Question 5 belong to?
☐ Warm blooded ☐ Live in water
☐ Lay eggs ☐ Breathe through gills
7. Which of the following categories does not belong to both shark and dolphin?
☐ Cold blooded ☐ Live on land
☐ Carnivorous ☐ Breathe through lungs
8. What are the differences between “Cobra” and “Crocodile”?
☐ Cobra is cold-blooded animal but crocodile is not.
☐ Crocodile is a meat-eater but cobra is not.
☐ Cobra lays eggs but crocodile does not.
☐ Crocodile can live in water but cobra cannot.
☐ There is no difference between them.

G.M. Goh, C. Quek

9. Which of the categories below describe the similarity between bear and whale?
() Cold blooded () Live on land
() Skin covered with hair () Breathe through lungs
10. Which of the following categories do “Dolphin”, “Angelfish”, “Whale” and “Turtle” belong to?
() Give birth to life young.
() Cold Blooded.
() Live in water.
() Breathe through lungs.
() None of the above.

Section B – Fill in the blanks

11. _____ and _____ are examples of warm blooded animals that live in water.
12. Birds are _____ blooded, live on _____, breathe through _____ and lay _____.
13. Give **two** differences between “Whale” and “Shark”.

_____.
14. _____, _____ and _____ are categories that describe the food an animal eats.
15. Generalisation is to _____

_____.
16. Comparison is to _____

_____.

G.M. Goh, C. Quek

Controlled MGS Post-Evaluation Test

Name: _____

Class: _____

Section A – EpiList I

Instruction: This section focuses on EpiList I. Please click on “EpiList I (Upper Primary)” before answering this section.

Part 1 – Error in Classifying

Whenever you made mistakes in the grouping of the animals, tutorials would appear to highlight the mistakes to you.

1. Do you understand the questions asked by the tutorials?

()Yes ()No

2. Do you know your own mistakes after the tutorials?

()Yes ()No

3. Are the tutorials difficult to understand?

()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

4. Are you able to correct your mistakes after the tutorials?

()Yes ()No

5. How many times do you have to go through the same tutorial to understand your mistakes and correct them?

()1 times ()2 times ()3 times
()4 times ()5 times

G.M. Goh, C. Quek

6. If a student had grouped “Tiger” under “Live in water” and grouped “Cat” under “Live on land”. The tutorial highlighted that Tiger and Cat are similar under the category “Live on land”. Which of the following should the student do?

- ☐ Group “Cat” under “Live in water”.
- ☐ Group “Tiger” under “Live on land”.
- ☐ Group “Cat” under “Live in water” and “Tiger” under “Live on land”.
- ☐ None of the above.

Can you explain your answer?

7. Another student had grouped “Dolphin”, “Dog” and “Cobra” under “Live in water”. The tutorial had highlighted that Dog and Cobra are different from Dolphin under the category “Live in water”. Which of the following should the student do?

- ☐ Move “Dolphin” out of “Live in water”.
- ☐ Move “Dolphin” and “Cobra” out of “Live in water”.
- ☐ Move “Dog” and “Cobra” out of “Live in water”.
- ☐ Move “Dolphin” and “Dog” out of “Live in water”.

Can you explain your answer?

G.M. Goh, C. Quek

Part 2 – Obtaining the important categories

After you had correctly classify all the items, EpiList I would introduce the important categories to you.

8. You are able to understand the important categories introduced to you.

()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

9. Can you correctly add the important categories immediately after the first introduction of important categories?

()Yes ()No

If the answer is “No”, are you able to do it after a few tries of the introduction?

()Yes ()No

10. There is a tutorial to instruct you to move animals into important categories. Is this tutorial clear and understandable?

()Yes ()No

11. Do you obtain a set of important categories in EpiList I?

()Yes ()No

If your answer is “Yes”, list down below what are the categories that you had obtained.

12. Do you know what make a category important?

()Yes ()No

If the answer is “Yes”, write down the things that you think would make a category important.

G.M. Goh, C. Quek

13. Are the categories “Live in water”, “Live on land” and “Live in air” important categories?

()Yes ()No

14. Are the categories “Cold Blooded” and “Warm Blooded” important categories?

()Yes ()No

Part 3 – Generalisation and Comparison Skills

The tutorials of EpiList I use generalisation and comparison to highlight the mistake and introduce the important categories.

15. Generalise frog, dolphin, fish and shark will give you which of the following categories?

()Warm blooded ()Live in water
()Carnivorous ()Breathe through gills

16. Which of the following sentences is incorrect?

()Crocodile and Cobra are similar in the place they live in.
()Crocodile and Cobra are similar in the types of food they eat.
()Crocodile and Cobra are similar in their body-temperature.
()Crocodile and Cobra are similar in the way they give birth to their young.

17. Compare between snake and lion, they are different in term of which of the following categories?

()Cold blooded ()Live on land
()Carnivorous ()Breathe through lungs

18. The common category that birds, lizards, and fish belong to is _____.

G.M. Goh, C. Quek

19. List some categories that “Dolphin”, “Tiger”, “Whale” and “Crow” belong to.

Part 4 – General

20. Are the detailed texts clear and easy to understand?

()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

21. Do the detailed texts provide enough information for you to learn more about the animals?

()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

22. Is EpiList I friendly and easy to use?

()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

23. How much time did you use to complete EpiList I?

() less than 30 min()30 minutes ()1 hour
()2 hours ()5 hours ()10 hours

24. Would you use EpiList I frequently to know more about animals?

()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

25. What other comment do you have for EpiList I?

G.M. Goh, C. Quek

Section B – EpiList II

Instruction: This section focus on EpiList II. Please click on “EpiList II” before answering this section.

Part 1 – Teaching Generalisation and Comparison Skills

When you continuously answer incorrectly within the tutorials, it shows that you weak in generalisation or comparison skills. EpiList II would explicitly demonstrate or teach these skills to you.

26. Do you understand the tutorials that teach generalisation and comparison?

()Yes ()No

27. What is “Generalisation”?

28. What is “Comparison”?

Part 2 – Teaching the Reasoning Processes

The tutorials use generalisation and comparison to highlight the mistakes. After a series of generalisation and comparison through the tutorials, you should be able to identify your own mistakes and correct them accordingly. If EpiList II detects that the correction is not made, it would again but explicitly highlight the mistakes to you.

29. Does EpiList II correctly detect your inability to correct your mistakes?

()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

G.M. Goh, C. Quek

30. Is the second highlight of the mistakes clearer and more understandable?

- ()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

31. Are you to identify and correct your mistakes after the second highlight?

- ()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

32. If a student had correctly grouped “Tiger” and “Dog” in the category “Warm Blooded” but incorrectly grouped “Turtle” in the same category. Label the sequence (1, 2, 3 and 4) in the brackets below the steps taken by the tutorial that would highlight the “Turtle” as the incorrectly grouped animal.

- ()Obtain the category that “Tiger” and Dog” are grouped under.
()Induce that “Turtle” is incorrectly grouped and should be moved to other category.
()Determine if “Turtle” also belongs to the same category.
()Compare “Turtle”, Tiger” and “Dog” together to identify their differences and should not be grouped together with “Tiger” and “Dog”.

Part 3 – Testing for Skills

After EpiList II had explicitly taught you the skills, it will test you accordingly.

33. Are you able to understand the questions in the test?

- ()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

34. Are the questions of the tests clear and understandable?

- ()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

G.M. Goh, C. Quek

35. Do the tests help you to understand the generalisation or comparison skills better?

- ()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

After you had correctly grouped all the animals into important categories, you receive a series of test questions.

36. Are the questions clear and understandable?

- ()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

37. If the answering method confusing and complicated?

- ()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

38. Do these test questions improve your skills?

- ()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

Part 4 –General

39. You learnt more about generalisation and comparison from EpiList II.

- ()Strongly Agree ()Agree ()Neutral
()Disagree ()Strongly Disagree

40. How much time did you use to complete EpiList II?

- () less than 30 min()30 minutes ()1 hour
()2 hours ()5 hours ()10 hours

G.M. Goh, C. Quek

41. Would you use EpiList II frequently to know more about generalisation and comparison skills?

()Strongly Agree ()Agree ()Neutral

()Disagree ()Strongly Disagree

Please explain you answer.

42. Have you ever use an educational software that also teaches generalisation and comparison skills?

()Yes ()No

If yes, what is the name of that software?

How do you rate it against EpiList II?

()Better than EpiList II

()As good as EpiList II

()Worse than EpiList II

43. What other comment do you have for EpiList II?

List of Publications

Journal (In press)

G.M. Goh, C. Quek, *EpiList: An Intelligent Tutoring System Shell for Implicit Development of Generic Cognitive Skills that Support Bottom-up Knowledge Construction*, to appear in IEEE Transaction on Systems, Man and Cybernetics, Part A.

Journal under preparation

G.M. Goh, C. Quek, *EpiList II: Explicit Instructions for Development of Generic Cognitive Skills*, submitting in March.

G.M. Goh, C. Quek, *A-EpiList: Adaptive Instructional Sequence for Development of Generic Cognitive Skills*, under preparation.

G.M. Goh, C. Quek, *Formal Definition of An Intelligent Tutoring Systems*, under preparation.

Conference

G.M. Goh, C. Quek, (2004) *EpiList: An ITS Shell to Develop Generic Cognitive Skills for Bottom-up Knowledge Construction*, International Conference on Computers in Education 2004, ICCE04, Australia.

Goh, G.M., Quek, C. (2005) *EpiList II: Explicit Instructions for Development of Generic Cognitive Skills*, International Conference on Computers in Education 2005, ICCE05, Singapore.