A WEB-BASED LEARNING SUPPORT SYSTEM FOR INQUIRY-BASED LEARNING USING TREASURE HUNT

A Thesis
Submitted to the Faculty of Graduate Studies and Research
In Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Computer Science
University of Regina

By
Dong Won Kim
Regina, Saskatchewan
December 2009

© Copyright 2009: Dong Won Kim
Abstract

Web-based learning has become a very popular learning mechanism. However, Web-based learning may not always work effectively, and sometimes fails to meet its learning objectives. Students frequently express dissatisfaction with Web-based learning suggesting it is “hard to stay motivated.” Learning games offering challenges and entertainment may stimulate and motivate students to learn. Nevertheless, no empirical evidence proposes learning games will promote student learning without instructional activities.

Web-based learning support systems enable students, who are taking courses in which the systems are involved, to learn topics. Web-based learning support systems, combined with learning games, are able to efficiently promote learning by encouraging student participation. Inquiry-based learning is a student-centered educational method driven more by student questions than by an instructor’s lessons. Web-based learning support systems will hopefully take the idea of inquiry-based learning and combine the learning games in a way to maximize the effectiveness of learning.

In this thesis, we present a Web-based learning support system called OTHI - Online Treasure Hunt for Inquiry-Based Learning. The system is designed according to a treasure hunt model, which embodies the idea of inquiry-based learning using set theory. Its implementation exploits some open source softwares and off-the-shelf extensions, and uses not only Web technology but also online game technology by including a development of an online treasure hunt game as the learning game. We
also address the implementation and demonstration of the system with a data communications and network course, and discuss its usefulness for supporting students’ learning activities. This Web-based learning support system is able to provide students with an inquiry-based learning environment in which they actively partake in learning.
Acknowledgments

I wish to express my sincerest and deepest thanks to my supervisor, Dr. JingTao Yao. This thesis would not have been completed without his expertise, invaluable advice, encouragement, understanding and patience. I am also very grateful for his financial support.

In addition to my supervisor, I would like to express my appreciation to the rest of my supervisory committee: Dr. Lisa Fan and Dr. Wojciech Ziarko, for their valuable comments and suggestions regarding the revision of this thesis, as well as their time and effort.

I must acknowledge my friends and fellow students, especially Joseph Herbert for our scholarly debates, and exchanges of knowledge and skills during my graduate program. My thanks also goes out to the Faculty of Graduate Studies and Research, and the Department of Computer Science for their academic and financial support.
Post-Defense Acknowledgments

I would like to sincerely thank the external examiner, Dr. Rene Mayorga, for his invaluable comments and suggestions improving this thesis. I am also thankful to the defense chair, Dr. Dianliang Deng, for presiding the defense.
Dedication

I would like to give my sincerest thanks to my precious wife, Bookhee, and my wonderful daughters – Soyoung, Sunyoung and Seyoung – for their love, support, understanding, and sacrifices. I also dedicate this work and give special thanks to my parents and brothers for their understanding and encouragement during my studies.
Acronyms

IBL  Inquiry-Based Learning
LSS  Learning Support Subsystem
NPC  Non-Player Character
OTHI  Online Treasure Hunt for Inquiry-Based Learning
PAWL  Passive Asynchronous Web-based Learning
SOLC  Sun Open Learning Center
TFCL  Traditional Face-to-face Classroom Learning
TSS  Teaching Support Subsystem
WLSS  Web-based learning support systems
Contents

Abstract i

Acknowledgments iii

Post-Defense Acknowledgments iv

Dedication v

Acronyms vi

Table of Contents vii

List of Tables viii

List of Figures ix

Chapter 1 Introduction 1

Chapter 2 Background Information 5
  2.1 A Review of Web-Based Learning 5
    2.1.1 Web-Based Learning 6
    2.1.2 Student-Centric Learning 6
    2.1.3 Online Courses 7
    2.1.4 An Example of Web-based Learning: SOLC 9
Chapter 3  Definition of a Treasure Hunt Model  

3.1 Learning Order Relationship and Knowledge Spaces  
3.1.1 Learning Order Relationship  
3.1.2 Knowledge Spaces on Learning Order Relationship  

3.2 Treasure Hunts and Inquiry-Based Learning  
3.2.1 Treasure Hunts for Learning  
3.2.2 A Treasure Hunt Process for Inquiry-Based Learning  

3.3 A Treasure Hunt Model for Inquiry-Based Learning  
3.3.1 Structure of a Treasure Hunt Model  
3.3.2 Learning State of the Treasure Hunt Model  
3.3.3 Knowledge Construction of the Treasure Hunt Model  

3.4 Summary
Chapter 4  Implementation and Demonstration

4.1 Overall System Architecture ........................................... 47
4.2 Algorithms ................................................................. 49
   4.2.1 Learning Environment Building Algorithm ................. 50
   4.2.2 Treasure Hunting Algorithm ................................. 52
4.3 Development Environment and Available Technologies .......... 54
   4.3.1 Use of Open Source Softwares and Freely-available Servers .. 54
   4.3.2 Useful Tools and Other Technologies ....................... 56
4.4 Implementation of Essential Features ............................. 57
   4.4.1 Homepage of OTHI ............................................ 58
   4.4.2 Teaching Support Subsystem ............................... 59
   4.4.3 Learning Support Subsystem ................................ 61
   4.4.4 Treasure Hunt Game ......................................... 62
   4.4.5 Dependence of the Treasure Hunt Game on Learning ...... 66
4.5 A Demonstrative Example ............................................. 66
   4.5.1 Finding Learning Order Relationships ...................... 67
   4.5.2 Editing Topic Information .................................. 67
   4.5.3 Building an IBL Environment ............................... 69
   4.5.4 Experiencing IBL ............................................. 72
4.6 Usefulness of OTHI for Learning ................................. 77

Chapter 5  Conclusion

5.1 Concluding Remarks ................................................. 82
5.2 Summary of Contributions .......................................... 83
5.3 Limitations and Future Research .................................. 84

Appendix A  List of Symbols ................................. 86
Appendix B  Algorithms  90
Appendix C  XML Files  99
Appendix D  PHP Programs Added To Joomla  107
Appendix E  Java Programs of Treasure Hunt Game  111
Bibliography  122
List of Tables

2.1 Examples of U.S. universities offering online courses. .................. 8
2.2 Examples of IT companies providing online courses. .................. 8
2.3 Examples of companies providing online learning services. ........... 9
4.1 Open source softwares used to develop the system. ................. 54
4.2 Database and Web servers implemented for the system. ............ 56
4.3 Tools used in the design and development of the treasure hunt game. 56
4.4 Algorithms and other technologies applied to the treasure hunt game. 57
4.5 Combination of game components and learning objects. .............. 72
4.6 Comparative and contrastive evaluation of OTHI and other learning approaches. .................................................. 78
List of Figures

2.1 Presentation of Sun Open Learning Center.................. 10
2.2 Quiz game of Sun Open Learning Center.................. 11
2.3 Open question from the Sun Open Learning Center......... 12
2.4 WLSS, the intersection of Web-based systems and learning support systems................................. 14
2.5 An Example of WebQuest..................................... 23
3.1 An Example of Learning Order of Concepts.................. 28
3.2 Relative difficulty or relative difference in learning order of concepts in a textbook.......................... 28
3.3 Flowchart of a treasure hunt process....................... 35
3.4 Flowchart of a knowledge construction..................... 36
3.5 High-level view of the treasure hunt model................ 43
4.1 An Architecture of OTHI..................................... 48
4.2 Homepage of OTHI............................................ 58
4.3 User Menu of student........................................ 59
4.4 User Menu of instructor...................................... 59
4.5 Main menu of TSS............................................ 60
4.6 Login screen of the treasure hunt game.................... 63
4.7 Multiple users of the treasure hunt game................... 64
4.8 Edit Topic Orientation of TSS............................... 68
Chapter 1

Introduction

The Internet has emerged as a major, and perhaps eventually the major worldwide communication channel for information, knowledge and services [15]. The impact of the Internet can be seen in all aspects of people’s lives. Its impact on teaching and learning may be even greater. The Internet has led to advances in computer technologies, especially in World Wide Web technology. The advances in Web technology have shifted the paradigm of education from the traditional teacher-centered model to a collaborative student-centered model [22].

Web-based learning is growing in popularity and offers many benefits over traditional learning environments [21, 84]. The Web provides a distributed infrastructure for information processing, with global accessibility, and a user-friendly interface [82]. Web-based learning delivers various forms of learning content such as text, audio, graphics, animation and video, with reasonable prices through the Web. It increases opportunities for education [31].

However, students may passively take part in online classes which lack interaction with the learning material [37] and furnish inflexible instruction without consideration for each student’s background knowledge [21]. In addition, many online learning systems do not request that instructors simultaneously participate in their students’ learning process. This results in a lack of interaction between students and
instructors [85]. This passive, asynchronous Web-based learning makes it difficult for students to remain motivated and continue learning.

Digital games may be applied to reduce the negative feedback of Web-based learning when dealing with a lack of motivation for learning [33]. Digital games hold players’ attention and stimulate their internal motivation [30]. The aim of a rational player is to win the game or to be one of the top players in the game. Playing implies active participation. That is, the game must be player-centered, and at every opportunity, players need to make their own decisions to win the game. Students require this type of motivation in their learning. Jong et al. [33] investigated student perception of a game-based learning method with traditional Web-based learning. They found game-based learning was far more favored, more interesting, more explanatory, more stimulating, more challenging, and more capable of empowering students to learn with confidence and retain learned knowledge in their mind.

Learning support systems not only enhance student learning, but also motivate students to learn [63]. The recent development of the Web has increased the need to move traditional computerized support systems to the Web platform and enhance support systems into further user-friendly systems [82]. Web-based learning support systems (WLSS) are computerized learning support systems redesigned or modified to use Web technology to support student learning [79]. Students may not only eliminate study difficulties, but may also be encouraged to learn with the support of WLSS.

The goal of this research is to design and implement a Web-based learning support system which mitigates the limitations of Web-based learning and encourages active student participation in learning by using an online-learning game. As for the design of such a Web-based learning support system, it is necessary to discuss the integration of the online-learning game into the system in a way which maximizes the effectiveness of learning. The effectiveness could be improved if game-based learning was combined
with traditional teaching approaches [33]. Moreover, games combined with student-centered teaching approaches are more effective and attractive than games with basic drill-and-practice approaches [36].

Inquiry-Based Learning (IBL) is an educational approach driven more by student questions than by the instructor’s lessons, and involves active, student-centered learning [18, 22]. Lim [41] described Web-based learning environments, designed with IBL, as providing students with cognitive tools and helping them form a learning community in which instructors and students interact to solve complex problems. IBL is compatible with constructivist learning strategies [43]. In constructivism, knowledge is defined as the cognitive structure of a person and learning is the active process of constructing knowledge rather than the process of knowledge acquisition [39]. In IBL, students construct knowledge using an inquiry approach [74].

The thesis will present a Web-based learning support system called OTHI, which stands for Online Treasure Hunt for Inquiry-Based Learning. The system design is based on a treasure hunt model, which takes its inspiration from a treasure hunt as a method to apply IBL to the system and embodies the idea of IBL using set theory. The Web-based learning support system takes advantage of online game technology as well as Web technology in its implementation. OTHI employs an online treasure hunt game in which students conduct a treasure hunt on a certain topic and develop an answer to a given question, and includes a website to support instructors and students who utilize the online treasure hunt game for student learning. Some open source softwares and off-the-shelf extensions are used to implement the system. With this system, students are able to experience IBL and study the course topics in an enjoyable and interesting way.

This thesis is organized as follows: Chapter 2 describes background information regarding Web-based learning, WLSS, Web technologies and IBL, and presents the problem statement; Chapter 3 defines learning order relationship and its knowledge
spaces, and depicts a treasure hunt and a treasure hunt model; Chapter 4 details several algorithms for two major functions of OTHI, describes the implementation of a prototype of OTHI, demonstrates the prototype system with a data communications and network course, and discusses the usefulness of OTHI in student learning; and Chapter 5 gives the concluding remarks, the summary of contributions and directions for future research.
Chapter 2

Background Information

Chapter 2 offers some background information and technologies upon which this thesis is based. A detailed review of Web-based learning is provided, including student-centric learning, online courses and an example of Web-based learning. An overview of WLSS is also presented, with discussion of Web technologies such as Web search engines, Web services and online-learning games utilized to implement WLSS. In addition, the chapter provides an in-depth overview of IBL, including IBL processes and a Web-enabled IBL model. At the end of this chapter, a problem statement is presented with a brief discussion of a new approach.

2.1 A Review of Web-Based Learning

This section gives a detailed review of Web-based learning, including student-centric learning, online courses and an example of Web-based learning: Sun Open Learning Center (SOLC).
2.1.1 Web-Based Learning

The Web is a popular medium for storing, presenting, gathering, sharing, processing and using information [82]. The Web offers the following benefits for learning: 1) it provides a distributed infrastructure of educational content; 2) it can be used as a channel in which students collaborate and interact; 3) it delivers various forms of learning content and secure information in a timely manner; 4) it provides a student-friendly interface; 5) there are no time frames or geographic restrictions when attending online classes; and 6) it can remotely and instantly manage and retrieve knowledge.

Web-based learning is sometimes called online learning or e-Learning since it includes online course content [48]. The content may consist of various forms of multimedia like audio, graphics, animation and video. It not only enhances an individual’s problem-solving skills, but also motivates students to complete a task by the vividness of the presentation [85].

In addition to the student-friendly and eye-catching content, Web-based learning eliminates the barriers of time and geographical distance in education [31, 85]. Once the learning content is created, students can use the content at anytime, anywhere. Wherever students are located, they can take part in their online classes as long as the Internet is available [85]. It also allows students to take courses in a cost effective learning environment [21, 31]. Web-based learning gives many people a second chance to educate themselves [31].

2.1.2 Student-Centric Learning

Modern-day learning is shifting from teacher-centric to student-centric, which emphasizes relevance, personalization and learning flexibility [85]. Traditional education provides a teacher-centered form of education and group learning in a conventional
classroom [21]. A typical scenario of a traditional class involves a teacher instructing students, and students listening to the teacher and taking notes. This face-to-face learning has the advantage of being familiar and close, and is comfortable for both the teacher and students [85].

However, it is almost impossible for a teacher to satisfy all requirements of all students with his/her lecture allotment. Normally, a lecture is prepared for average students with an average background knowledge of a lecture topic. It is also very difficult to measure each student’s understanding of the class, even though the teacher takes into account student feedback (quizzes, exams and assignments) in the lecture. One barrier for traditional education is learning basically depends on limited time and a fixed place.

Alternatively, Web-based learning can provide a student-centered form of education, and self-directed, individual learning [21]. A student can choose the most suitable course for themself, based on his/her background knowledge. The student can replay learning materials until (s)he feels satisfied, or by-pass those materials with which (s)he is already familiar. If the student has questions in his/her online class, (s)he can easily receive assistance from groups of teachers and other students by using a communication tool supplied by the Web-based learning system which manages the online class. The student can determine his/her understanding of the class by using various types of tests at any time. Therefore, Web-based learning is considered as one means of reducing the disadvantages portrayed by traditional methods of learning.

2.1.3 Online Courses

Nowadays, many universities worldwide are offering thousands of online courses, including degree and certificate programs [37, 85]. Table 2.1 provides a list of U.S. universities offering online courses. As of 2009, Stanford University has 23 master of science degrees, completed entirely online. The University of California offers more
<table>
<thead>
<tr>
<th>Schools</th>
<th>Courses or programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>University of Phoenix (<a href="http://www.phoenix.edu">http://www.phoenix.edu</a>)</td>
<td>Arts and Sciences, Business and Management, Criminal Justice and Security, Education, Human Services, Nursing and Health Care, Psychology, Technology.</td>
</tr>
<tr>
<td>Stanford University (<a href="http://scpd.stanford.edu">http://scpd.stanford.edu</a>)</td>
<td>Aeronautics and Astronautics, Bioengineering, Biomedical Informatics, Civil and Environmental Engineering, Computer Science, Computational and Mathematical Engineering, etc.</td>
</tr>
<tr>
<td>University of California, Berkeley (<a href="http://explore.berkeley.edu">http://explore.berkeley.edu</a>)</td>
<td>Art and Design, Behavioral and Health Sciences, Basic Sciences, Business and Management, Computer Technology and IS, Education, Electronic Engineering, Humanities, etc.</td>
</tr>
<tr>
<td>Georgia Institute of Technology (<a href="http://www.dipe.gatech.edu/dl">http://www.dipe.gatech.edu/dl</a>)</td>
<td>Aerospace Engineering, Computational Science and Engineering, Electrical and Computer Engineering, Environmental Engineering, Industrial Engineering, Mechanical Engineering, etc.</td>
</tr>
<tr>
<td>University of Illinois Online, Urbana (<a href="http://www.online.uillinois.edu">http://www.online.uillinois.edu</a>)</td>
<td>Agricultural Business and Production, Biological Sciences and Life Sciences, Business Management and Administrative Services, Communications, etc.</td>
</tr>
<tr>
<td>University of Colorado Denver (<a href="http://www.cudenver.edu/Academics/CUno/line">http://www.cudenver.edu/Academics/CUno/line</a>)</td>
<td>Sociology, English Writing, Nursing, eLearning, School Library, Early Childhood Education, Business Administration, Geographic Information Systems, Public Administration, etc.</td>
</tr>
</tbody>
</table>

Table 2.1: Examples of U.S. universities offering online courses.

<table>
<thead>
<tr>
<th>Companies</th>
<th>Web-based learning solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco Systems (<a href="https://cisco.hosted.jivesoftware.com">https://cisco.hosted.jivesoftware.com</a>)</td>
<td>Cisco Learning Network</td>
</tr>
<tr>
<td>Dell (<a href="http://www.learndell.com/dls">http://www.learndell.com/dls</a>)</td>
<td>Dell Learning System</td>
</tr>
<tr>
<td>Hewlett Packard (<a href="https://www.rooms.hp.com">https://www.rooms.hp.com</a>)</td>
<td>HP Virtual Classroom</td>
</tr>
<tr>
<td>IBM (<a href="http://www.ibm.com/training/us/ilo">http://www.ibm.com/training/us/ilo</a>)</td>
<td>IBM Instructor-led Online</td>
</tr>
<tr>
<td>Microsoft (<a href="https://www.microsoftlearning.com">https://www.microsoftlearning.com</a>)</td>
<td>Microsoft E-Learning</td>
</tr>
<tr>
<td>SAP (<a href="http://www.sap.com/usa/services/education">http://www.sap.com/usa/services/education</a>)</td>
<td>SAP Education E-Learning, Virtual Live Classroom</td>
</tr>
<tr>
<td>Sun Microsystems (<a href="http://www.sun.com/training/elearning">http://www.sun.com/training/elearning</a>)</td>
<td>Web-based Training, Live Virtual Class</td>
</tr>
</tbody>
</table>

Table 2.2: Examples of IT companies providing online courses.

than 140 courses through Web-based learning. The University of Colorado, Denver offers 340 courses and 14 degrees available online.

Meanwhile, companies have widely adopted Web-based learning solutions for their corporate training (See Table 2.2). The solutions allow individual employees to have learning-on-demand opportunities while reducing training time and costs [85]. The companies offer their Web-based learning solutions not only to employees, but also customers, as a training service. As listed in Table 2.3, many other companies such as Ninth House, Cengage Learning, Tutorials.com, SkillSoft and Kaplan Financial
### Table 2.3: Examples of companies providing online learning services.

<table>
<thead>
<tr>
<th>Companies</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cengage Learning</td>
<td><a href="http://cengage.co.uk">http://cengage.co.uk</a></td>
</tr>
<tr>
<td>Kaplan Financial Education</td>
<td><a href="http://www.kfeducation.com">http://www.kfeducation.com</a></td>
</tr>
<tr>
<td>Ninth House</td>
<td><a href="http://ninthhouse.com">http://ninthhouse.com</a></td>
</tr>
<tr>
<td>SkillSoft</td>
<td><a href="http://skillssoft.com">http://skillssoft.com</a></td>
</tr>
<tr>
<td>Tutorials.com</td>
<td><a href="http://tutorials.com">http://tutorials.com</a></td>
</tr>
</tbody>
</table>

Education, provide Web-based learning services.

#### 2.1.4 An Example of Web-based Learning: SOLC

Sun Open Learning Center (SOLC) ([http://www.sun.com/training/solc](http://www.sun.com/training/solc)) represents a typical form of Web-based learning providing online courses about OpenSolaris and Sun Solaris. It is free to anyone who has a Sun account. The two courses available for Sun Solaris are: Introduction to Solaris and Intermediate Solaris. Managing ZFS Pools and OpenSolaris Labs Sandbox courses are available for OpenSolaris. Introduction to Solaris course topics include: Using Components of the Desktop System, Manipulating and Managing Files and Directories, Searching and Process Manipulation, Working with the Shell, and Archiving Files and Remote Transfer. Intermediate Solaris course topics include: Booting SPARC and x86 Based Systems, Controlling System Processes, Create and Manage Users, Installing Solaris 10, etc. Students may choose a course with no prerequisites.

A SOLC course may consist of up to five components: a presentation, a quiz game, an open question, supporting documents and lab exercises. Figure 2.1 depicts a presentation regarding a topic belonging to the Introduction to Solaris course. The presentation mainly contains PowerPoint slides, multi-level navigation and playback controls. A PowerPoint slide explains a *sub-topic* with text information, images and voice narration. The multi-level navigation has four levels: Outline, Thumbnails, Notes and Search. The names of the PowerPoint slides are displayed in a hierarchical order in the Outline. By simply left clicking on a slide name, students are taken to
the PowerPoint slide without interruption. The *Thumbnails* represent small versions of the PowerPoint slides. The *Notes* show a description of a voice narration. The *Search* lists the names of PowerPoint slides containing a specific word. They can move to the next slide or the previous slide and control the voice narration by using the playback controls.

The quiz game provides students with four quizzes to do with a topic they have learned. As depicted in Figure 2.2, it displays available time and scores in the upper left-hand corner and in the upper right-hand corner, respectively. Students need to answer the four quizzes within 45 seconds by choosing the right answer from the four answers. Voice narration is also provided with each quiz.

As for the open question, students are requested to explain what they know about
a given question as they click the tab “Do You Know?” in Figure 2.2. Figure 2.3 shows an example of an open question. When students submit an answer to a question, the system shows its Expert Answer with no marks given for the answer. When they click the tab “Info” in Figure 2.3, it presents a list of hypertext links pointing to Web documents which relate to the topic.

Students are able to get access a virtual lab environment with lab exercises and asynchronous support from an instructor upon buying a Sun on-demand lab. It enables students to master key skills by completing real world lab exercises at a virtual lab. On the other hand, SOLC offers OpenSolaris Learning Cloud Service which allows students to freely try the features of OpenSolaris for one hour. The lab desktop of the OpenSolaris Learning Cloud Service gives students access to a virtual desktop where they can complete lab exercises for the OpenSolaris operating system.

The SOLC has some drawbacks in terms of learning. Many of the course topics have several sub-topics. A one-time presentation of the course covers all sub-topics related to a main topic and requires students to spend a lengthy amount of time.
and attention on the presentation, sitting in one spot without *interactivity*. Students cannot be expected to comprehend all the sub-topics the presentation contains in one sitting.

Students may enjoy the quiz game and see the Expert Answer to the open question before completing the presentation, since all four quizzes and the one open question do not interfere with them easily catching the main ideas of the topic. However, students cannot expect such learning activities from other courses which consist of only the presentation. Although the courses offer the quiz game and question, four quizzes and one open question are not enough to review all the sub-topics.

Students will not be aware of other students’ learning performance and there is no way of measuring their progress in a course. Furthermore, there are no features which allow students, who are taking the same courses, to contact each other. Hence, it is difficult to share ideas and issues regarding the learning topics with other students.
2.2 An Overview of Web-based Learning Support Systems

Web-based learning support systems (WLSS) assist, support, and aid students with learning activities. WLSS can be utilized to mitigate problems students have in conventional classrooms, even with online courses. Furthermore, Web-based learning may also receive support from WLSS to solve unforeseen problems. In this section, an overview of WLSS is presented in order to further investigate any possible solutions WLSS may provide.

2.2.1 Web-based Learning Support Systems

WLSS support instructors and students to achieve better teaching and learning outcomes [21]. Web technology refers to all technologies which implement, maintain, and use the Web. Advances in the Web technology remove any barriers of time and place for supporting various human activities and generate further momentum for the design and implementation of computerized support systems [82].

Web-based support systems [80] take advantage of Web technology to provide further user-friendly support environments [82]. The most popular and successful computerized support systems are decision support systems. If people develop decision support systems using Web technology, those systems become Web-based decision support systems [57]. Likewise, if people take into account learning, computerized learning support systems combined with Web technology can be intuitively called Web-based learning support systems [79]. As represented in Figure 2.4, any Web-based system which contributes to students’ learning can be viewed as a WLSS.

Fan and Yao [21] described the main features of WLSS for the implementation of student-centered learning environments. The main features include: encouraging students to communicate with each other; delivering adapted learning content based on students’ background knowledge; providing an interactive interface; and evaluating
students’ learning process. They also presented three views in terms of the design of WLSS, namely teacher, student and administrator views. Teachers consider WLSS to be convenient tools to create and modify learning content, while students regard WLSS as learning support tools containing a variety of learning content. Administrators might use WLSS to maintain the systems.

Pinkwart et al. [55] expressed four support criteria of WLSS: learning process support, community support, task support, and organizational support. They applied the criteria to an implementation of a WLSS called iPAL (Internet Portal to Augment Learning) system, which extends a freely available content management system for learning via necessary off-the-shelf extensions and several self-developed modules.

Hirai and Hazeyama [29] developed a WLSS, called Concerto, and applied it to an actual university course. The system focuses on question-posing by students, and supports not only question-posing and assessment of questions posed among peers, but also discussions about the questions. The experimental result of the system showed the exchange of questions and answers between students contributed to an improvement in understanding as to what they had learned.

2.2.2 Architecture of Web-based Learning Support Systems

Generally, WLSS architecture forms a thin-client and server structure in which enterprise-wide applications, installed on a server, are available to multiple users using
client softwares. The client software is responsible for transmitting input and output messages between the remote server and the user, while the server performs requested computing services as a database server, an application server, a file server, or a Web server. The server also sends back a response to the client software to present a processing result, indicating a confirmation of completion of the requested operation, or giving a notice of a failure of an operation.

WLSS have a web browser with plug-in modules to work as the client software. Web programs are placed and executed in the Web server, and communicate with a database management system or other programs. The web browser provides an interface between the Web programs and the user. A processing result of the Web programs is finally presented through the web browser to the user; the user gives feedback on the result to the Web programs using the web browser.

WLSS architecture also presents a three-layered architecture proposed by Yao and Yao [82]. The three-layered architecture includes an interface and presentation layer, a data layer and a management layer. The interface and presentation layer is located on the top of the architecture. This layer presents and permits interaction with information in a variety of visual and audio formats, and supports communication and expression through a web browser.

The data layer is placed on the bottom of the architecture, and represents knowledge bases, as well as databases. This layer is responsible for storing and manipulating large quantities of information. The management layer of the architecture is located between the interface and presentation and data layers, and serves as the middleware of the three-tier client and server architecture. The management layer performs complex computations with data taken from the data layer in order to process user requests received from the interface and presentation layer.
2.3 Web Technologies Supporting WLSS

This section describes several Web technologies used to build WLSS, such as Web search engines, Web Services and online-learning games. Those technologies may empower WLSS and extend the availability of WLSS. This research takes advantage of those technologies in order to achieve its goal.

2.3.1 Web Search Engines

There are many Internet-enabled applications and services. One of the primary tasks of the Internet is for information retrieval and information seeking [25]. Hundreds of millions of Web pages are available on the Internet and connected by hypertext links. People find a particular nugget of information required to fulfill some need they have in the use of the Internet [7]. Without search engines, a vast amount of information on the Internet would be quite interesting but perhaps not very useful [25, 53].

Web search engines have three major functions: collecting new information, collating and extracting useful information, and presenting it in a publicly available Web interface [46]. Information is collected by intelligent agents such as spiders crawling over Web pages via connecting links. All documents, found by the intelligent agents, are converted and saved in databases in forms a computer can understand and process. Each collected document is then passed through an information extraction system to extract document representatives and bibliographic information to form indexes [46]. The indexes will be grouped based on the similarity of the documents. The search engines entertain user queries and provide a list of relevant documents, in a user-friendly format.

Citation indexing, implemented firstly in 1963 by Garfield [47], is today one of the standard indexing techniques. The citation index indicates how many academic
papers were cited from other papers. It reveals relationships between papers, and represents the relative importance of each paper. That is, the more citation numbers a paper has, the more importance is attributed to the paper. Google believes every hyperlink embedded in a Web page is a citation to the other Web page [7]. The PageRank system, used by Google, shows how many Web pages and sites link to a page. It reflects the relative importance of a site. The Google system performs a type of full text search which looks for keywords within the title, abstract, and full text of an article. PageRank bares a very important role in the system to prioritize the results of Web keyword searches [5]. The PageRank algorithm enables Google to dominate other Web search engines in terms of usage [79].

The Web is considered to be a large and searchable virtual library and an information resource for scientists, researchers and students [81]. Web search engines make it possible for students to access the virtual library and provide students with information relevant to their topics, in order to improve student learning performance.

### 2.3.2 Web Services

Web services are considered an emerging web technology. They are self-contained, self-describing, modular applications operating over the Internet. In the development of Web services, the same technical standards are applied to self-description, publication, location, communication, invocation, and data exchange capabilities. It increases the interoperability and reusability of Web-based applications and, as well, greatly reduces the time and effort spent on implementing the applications. The field of online learning may benefit from the advantage of Web services [34], especially Web services which can be combined together to implement a WLSS.

Google Calendar is a Web service provided by Google and allows applications to read and update calendar events using Google Calendar Data API. A client application is able to use the Google Calendar Data API to create new events, edit or...
delete existing events, and query events matching particular criteria [24]. The Google Calendar is a useful coordination tool for students to make appointments with other colleagues and manage their schedules to work together [52].

MetaWeblog API is an application programming interface which enables the implementation of blog-based web services by allowing applications to publish and update the content and attributes of weblog posts [40]. A *weblog* (usually shortened to *blog*) offers an excellent new channel for discussion, communication and collaboration not only in research [78], but also in learning. Furthermore, *blogs* enable students to develop a deep understanding of, and take more responsibility for, their own knowledge [70].

### 2.3.3 Online-Learning Games

Modern digital games provide young people with learning opportunities every second and motivate students to learn [58]. Recently, game-based learning has been adopted for adult education, as well as children’s learning [16]. It is reported online-learning games are more effective than standalone learning applications, encouraging collaborative learning [9, 34].

Online games are highly graphical 2- or 3-dimensional video games played over the Internet [67]. Players, through their self-created digital characters or avatars, are allowed to interact not only with Non-Player Characters, which are computer-controlled characters, but with other players’ avatars. *Online-learning games* are derived from an attempt to not only bring the concept of the online games to interactive learning tools, but also provide students with learning experience through the online games.

Dziabenko *et al.* [16] implemented an online-learning game called “UNIGAME: Social Skills and Knowledge Training.” The game is accessible through its website, which furnishes the following features: training and help, community area, user registration, game introduction, virtual conference etc. They believed the game would
be used as a complement to lectures or as a standalone learning process, enabling students to better understand the theory as they practically apply it.

Michaelson et al. [51] developed an online-learning game for finance education of undergraduates in the FINESSE project. The aim of the game is to provide students with a Web-based learning environment to manage a portfolio of securities with real-time share price data. Shang et al. [62] developed an online-learning game called FARMTASIA, a farming simulation game, using the learning framework VI-SOLE. This framework shows three stages to facilitate students’ learning as follows: scaffolding learning, student game-based learning, and debriefing and reflection.

Global Goonzu [23] is a Massively Multiplayer Online Role-Playing Game simulating real world society. A player can experience this world as a warrior, merchant or politician. Its free market economic system enables players to experience real-time trading stock, real estate and other goods. Furthermore, players can take part in the shareholder election, to elect or to become a town chief. Although the game has been developed for fun, it is very useful in the areas of economics and political education.

Childress and Braswell [9] described current uses of multiplayer online games in learning. They derived a process of using multiplayer online games for cooperative learning activities from their experiences in Second Life [61], a Massively Multiplayer Online Role-Playing Game. They also emphasized multiplayer online games can provide students with many meaningful and enriching learning experiences, and the virtual environments of the games enable instructors to easily and efficiently design highly-social cooperative learning activities.

2.4 An Overview of Inquiry-Based Learning

This section presents an overview of inquiry-based learning (IBL) including IBL processes and a Web-enabled IBL model, WebQuest. IBL requires active student
participation through an inquiry activity. The IBL process represents an inquiry activity containing the following four phases: presenting phase, retrieving phase, developing phase and evaluating phase.

2.4.1 Inquiry-Based Learning

Inquiry-Based Learning is a student-centered instructional approach in which students seek truth, information, or knowledge by asking questions [8, 22]. IBL enables students to conduct self-directed investigations of problems and issues presented to them [41]. Russell [59] described inquiry as an investigation of a certain problem. Fleissner et al. [22] defined inquiry “as a seeking for truth, information, or knowledge; that is, seeking information by questioning.” Chan [8] described students constructing their perspectives of natural and human-designed worlds through inquiry activities.

The educational philosophy of IBL is founded on the ideals and principles of constructivism [43]. In constructivism, knowledge is defined as the cognitive structure of a student [39]. The cognitive structure is changed, reconstructed and reorganized by the student’s experience [54]. The constructivist believes students are the center of education, and learning is the product of self-organization and reorganization of the cognitive structure of the students [43, 77].

In other words, learning is an active, constructive process rather than process of knowledge acquisition, and teaching is the support of students’ constructive processing of understanding rather than the delivery of information to the students [39]. In addition, learning outcomes do not depend on what an instructor presents, but rather upon what information is encountered and how students process it, based on preconceived notions and existing personal knowledge [77]. Inquiry, in constructivist teaching, offers an opportunity to better understand facts and formulas, and encourages student participation in learning [41, 39].
2.4.2 Processes of Inquiry-Based Learning

Edelson [17] presented a three-step model of IBL: a motivational step, an acquire step and a refine step. Each step has activity strategies. The motivational step involves creating a demand for knowledge and eliciting curiosity. The acquire step allows students to discover new knowledge and receive knowledge from others. Finally, in the refine step, students perform reorganizing, applying and examining their knowledge. Lim [41] defined the process of inquiry as a model of an inquiry cycle which takes account of the circular, recursive and continuous notion of inquiry. The inquiry process involves: 1) presenting an overarching question/problem or a case scenario; 2) designing students’ own learning plan and problem solving strategies; 3) carrying out an investigation; 4) developing meaningful ideas and constructing new knowledge; and 5) applying their conclusion to a new situation and preparing new questions for the next cycle of inquiry.

The following four phases regarding the process of IBL are commonly described in the literature [17, 18, 41, 74].

1. **Presenting phase**: In the presenting phase, the instructor presents information and background in order not only to stimulate student interest in or curiosity about a topic, but also to help students develop a plan for their inquiry.

2. **Retrieving phase**: Students, in the retrieving phase, explore and query data and information for their inquiry, and develop skills and strategies to not only select relevant information, but also adjust and modify inquiries.

3. **Developing phase**: In the developing phase, students perform developing ideas and construct new knowledge. This phase requires students to evaluate and interpret the data and information collected in the retrieving phase.

4. **Evaluating phase**: In the evaluating phase, the developed ideas and student
inquiry process are evaluated by the instructor and other students, and constructive feedback is given enabling improvement in the future.

Throughout the entire process of IBL, students may revise their ideas or decide to go forward with the intended direction by looking back at the question, process and direction of the inquiry. Students also communicate with others to fully understand given information, and clarify or solidify their findings.

### 2.4.3 A Web-enabled IBL Model: WebQuest

WebQuest [72] is a WLSS for IBL and is designed according to the guidelines proposed by Dodge [14]. Well-designed WebQuests promote learning practices by integrating the idea of IBL with Internet resources, open-ended questions and authentic tasks stimulating students’ motivation [8]. WebQuest is also a flexible model for Web-enabled IBL [22]. Instructors are able to very simply create their own WebQuests if they can create a document with hyperlinks [72]. Figure 2.5 illustrates an example of a WebQuest on human learning.

A WebQuest should contain at least the following six components [22]:

1. **Introduction**: Provides some background information allowing students to be ready to investigate a quest.

2. **Task**: Specifies an interesting and doable task, duty, or assignment.

3. **Resource**: Contains a collection of information sources - links and references - necessary to complete the task.

4. **Process**: Describes the steps students should go through in accomplishing the task.

5. **Evaluation**: Shows an assessment rubric informing how student performance will be evaluated.
6. **Conclusion**: Brings closure to the quest, reminds students of what they have learned, and encourages them to apply the experience to other domains.

An optional component, *credits*, shown in Figure 2.5, denotes information about permissions to use and modify the WebQuest. Another optional component, *teacher page*, guides other teachers who want to implement the WebQuest.

MacGregor and Lou [45] pointed out there is very little in the way of empirical research on the effects of WebQuests on student learning, even if many educators have voted in favour of the WebQuest model and numerous WebQuests have been created due to the flexibility and easy creation of WebQuest. They found students asked teachers for assistance more frequently than teachers expected during the WebQuest implementation in actual classrooms and the teachers felt they should design their WebQuests more carefully to provide more supportive activities and materials.
2.5 Problem Statement

Recently, Web-based learning has emerged as a promising solution to lifelong learning and is fast becoming the most popular alternative to traditional face-to-face education [2, 85]. However, Web-based learning is not always effective and sometimes fails to meet its learning objectives [75].

Some online learning systems only present text-based learning materials, which may have students feeling bored and less engaged during online learning [85]. Even the most current multimedia-based online learning systems have unstructured learning materials, which are delivered without consideration of each student’s personal background and needs [21, 75, 85]. Students may passively take part in learning activities in a Web-based learning environment where one size fits all instructional design solutions will suffer because of its passive nature [75].

Moreover, in many online learning systems, instructors are not required to simultaneously participate in their students’ learning process and students normally study a subject in isolation [26, 85]. This results in a lack of interaction between a student and his/her instructor [26, 34, 85]. Passive, asynchronous Web-based learning makes it difficult for students to become motivated towards learning [33, 37, 85]. The goal of this research is to design and implement a WLSS to stimulate students’ motivation and encourage students to take part in their learning.

2.5.1 New Approach

Learning games encourage students and motivate toward study [58]. However, there is little empirical evidence a learning game is able to promote learning without interaction, led by instructors and instructional activities [13, 35]. Therefore, learning games might be better used as a means to support educational courses and to enlarge the instructional effect of the courses. In this sense, WLSS may well be conceptually
a better framework than online learning systems in adopting the learning games. The learning games combined in WLSS may support student learning under the supervision of instructors, while online learning systems basically focus on delivering online courses.

Nevertheless, this approach faces another problem: “how to integrate learning games into WLSS in a way that maximizes learning effectiveness.” Jong et al. [33] suggested traditional teaching approaches be combined with game-based learning in terms of the improvement of the learning effectiveness. Ke [35] reported learning games, carefully aligned with sound teaching approaches, promote good learning outcomes. Kebritchi and Hirumi [36] described games with student-centered approaches are more effective and attractive than games with basic drill-and-practice approaches.

IBL is a student-centered teaching method and is more effective than traditional teaching for achieving a variety of student learning outcomes [66]. Web-based learning environments, designed with IBL, are able to furnish students with cognitive tools and help them form a learning community in which instructors and students interact to solve complex problems [41].

According to the previous researches [33, 35, 36, 41, 66], we choose IBL as a teaching approach whereby WLSS and learning games can be combined and solve the problem. We propose and apply a treasure hunt model, which represents the idea of IBL, to the design of OTHI, a WLSS for IBL. We hope this system provides students with an appropriate IBL environment in which they can experience IBL and actively take part in learning. In the following chapter, we present the treasure hunt model.
Definition of a Treasure Hunt Model

Chapter 3 addresses a treasure hunt for learning and defines a treasure hunt model in which the idea of IBL is incorporated. First of all, to support the definition of the treasure hunt model, learning order relationship and its knowledge spaces are presented first in this chapter.

3.1 Learning Order Relationship and Knowledge Spaces

This section defines a learning order relationship and its knowledge spaces, including a learning order relation. The definitions will be utilized in further discussion as to what learning objects the students have learned and what objects might be next in Section 3.3.2, as well as presenting how to construct knowledge in Section 3.3.3. A portion of this section have been published in a conference of SPIE [79].

3.1.1 Learning Order Relationship

Wong and Yao [73] defined a user preference in order to introduce an adaptive linear model for an information retrieval system. They presented a strict preference relation and an indifference relation, which is an equivalence relation regarding the preference relation called a weak order. Learning order relationship represents a
user preference of a student who tries to choose and learn a more suitable learning object between two related learning objects in accordance with his/her background knowledge.

Generally, the posterior concepts of a course textbook require students to know the prior concepts of the textbook for the sake of understanding. Although some concepts are not mandatory in order to learn other concepts in a document, those concepts are required to comprehend in order to understand the document itself, which is represented by a broader concept. As long as concepts are discussed in the same document, they cannot be logically independent from others, and they can be compared with each other in order to measure the level of difficulty. In a collection of documents, if the content of any two documents are semantically related to each other, the relative difficulty of understanding or learning between the two documents is judged by students, based on their background knowledge.

*Objects* to be dealt with in the course of this thesis might be significant words, concepts, topics, educational materials, information units, documents, functions, commands, etc [32]. *Learning order of objects* provides the correct order in which a student is supposed to learn the other objects which remain after learning one object.

**Definition 3.1** We say that the relationship between two objects and satisfying the learning order of objects is called learning order relationship.

In other words, the learning order relationship between two related objects can be clearly expressed as follows: *Learning one object is a prerequisite for learning another object.* It reflects the student’s relative preference for the objects, based on his/her background knowledge. Figure 3.1 shows an example of the learning order of concepts in a textbook. The student may prefer “Concept 2 of chapter 3” to “Concept 3 of chapter 3” as their next learning concept.

The learning order relationship between two related objects can also determine
which object is more difficult. Students naturally feel the posterior concepts in a course textbook are more difficult than the prior concepts, as the posterior concepts require students to have knowledge of the prior concepts first. In addition, students tend to make estimates and *educated guesses* [27] regarding a new concept in a class, and they feel relaxed if the concept is very close to their guess. As shown in Figure 3.2, the more a concept differs from the student’s knowledge base, the more difficult the concept is to guess and understand. Thus, for a course, the last concept in the learning order of concepts becomes the most different and difficult concept. In the same way, the more an object is different from what a student has known, the more difficult the object becomes to guess and understand.

A concept is said to be a sub-concept or a super-concept of another concept [83]. A group of related objects can become a super-object by overlapping the objects. On the contrary, an object can be disjointed into its sub-objects.
**Definition 3.2** We say that a learning order relationship exists between any two objects in a new set which is formed by overlapping or disjointing objects in a set of related objects.

In a course design, a teacher considers learning objects students will learn in class. The teacher may overlap or disjoint objects of the course in order to build a set of objects to teach, based on how detail each topic will be presented to students with his/her lecture allotment. Likewise, in the system, an instructor may overlap or disjoint objects of a course, since the system needs to limit the maximum number and minimum number of objects the instructor is able to define for the course due to limited system resources and effective service delivery. Even in this case, the learning order relationship is still available to represent the user preference between the objects belonging to the new set, as stated in Definition 3.2.

### 3.1.2 Knowledge Spaces on Learning Order Relationship

The knowledge of a student can be profiled by assessing the student’s competence. From this standpoint, in a knowledge space theory, a topic can be defined by a finite set of problems, questions, or *items* the student must learn to solve [19]. Some problems may be solvable by the student only if some other problems have been previously mastered by the student [20]. Let $Q$ be a set of *items* for a particular topic. If $K$ is a subset of $Q$, $K$ is called a *knowledge state* and the set of all possible knowledge states is called the *knowledge structure* on $Q$ [60].

Let $\mathcal{K}$ be a knowledge structure on $Q$ and $q \in Q$. Then $\mathcal{K}(q)$ is the collection of all knowledge states containing the item $q$ and denoted by:

$$\mathcal{K}(q) = \{ K \in \mathcal{K} | q \in K \}. \quad (3.1)$$
The intersections of all such knowledge states in $\mathcal{K}(q)$ are described by:

$$\bigcap \mathcal{K}(q) = \bigcap_{K \in \mathcal{K}(q)} K.$$  \hspace{1cm} (3.2)

For a knowledge structure $\mathcal{K}$ on $Q$, the relation $R \subseteq Q \times Q$:

$$q_1 R q_2 \iff q_1 \in \bigcap \mathcal{K}(q_2), \forall q_1, q_2 \in Q,$$  \hspace{1cm} (3.3)

is called a surmise relation on $Q$ [4] and interpreted as meaning every knowledge state $K$ containing $q_2$ should also contain $q_1$ [44]. The surmise relation is a quasi-order, which is reflexive and transitive. For $q_1, q_2 \in Q$, another interpretation of $q_1 R q_2$ is a student can surmise a correct response to problem $q_1$ from a correct response to problem $q_2$. It can also be said mastering item $q_1$ is a prerequisite for mastering item $q_2$ [1]. Thus, the surmise relation $R$ on $Q$ describes the prerequisite relationship between two items in the set $Q$ [1, 4].

Let $T$ be a finite nonempty set of all objects in a collection of related documents. Then, a binary relation on $T$, with respect to Definition 3.1, is called a learning order relation $\preceq$. The learning order relation $\preceq$ on $T$ can be formally defined by: for $t_1, t_2 \in T$,

$$t_1 \preceq t_2 \iff \text{anyone who is able to learn } t_2 \text{ should also be able to learn } t_1.$$  \hspace{1cm} (3.4)

t_1$ is called a prior object of $t_2$, and $t_2$ is called a posterior object of $t_1$. The prior object $t_1$ of $t_2$ can be regarded as a prerequisite for mastering $t_2$. The learning order relation $\preceq$ on $T$ also describes the prerequisite relationship between two objects in the set $T$.

A subset of $T$ can be the knowledge state, and a knowledge structure on $T$ can be restricted by the learning order relationships between objects in $T$. Thus, for a
learning order relation \( \preceq \) on \( T \), a knowledge structure, associated with the relation, is defined by:

\[
\mathcal{K}_R = \{ K | (\forall t, t' \in T, t' \preceq t, t \in K) \Rightarrow t' \in K \}. \tag{3.5}
\]

It means a knowledge state \( K \in \mathcal{K}_R \) contains all the prerequisites of an object \( t \) if \( t \) is an element of \( K \), and the prerequisites are linearly ordered along with one single dimension, taking the learning order relation as deterministic. Hence, the object \( t \) can only have one set of prerequisites. The knowledge structure \( \mathcal{K}_R \) contains the empty set \( \emptyset \) and the entire set \( T \), and is closed under set union and intersection.

**Definition 3.3** We say that a learning order relation \( \preceq \) of the knowledge structure \( \mathcal{K}_R \) on \( T \) is a quasi-order relation on \( T \) satisfying the following axioms:

1. (1) if for \( \forall t \in T, t \preceq t \) holds,
2. (2) if for \( \forall (t, t', t'') \in T^3, (t \leq t') \) and \( (t' \leq t'') \), then \( (t \leq t'') \).

As previously described, in the knowledge structure \( \mathcal{K}_R \), an object can have only one set of prerequisites, which is able to provide one learning path to the student. However, in practice, it is possible for an object to have more than one set of prerequisites. A mapping \( \sigma : T \to 2^{2^T} \) is called a surmise function \([60]\) if it satisfies the following conditions: 1) \( P \in \sigma(t) \Rightarrow t \in P \); 2) \( P \in \sigma(t), t' \in P \Rightarrow (\exists P' \in \sigma(t'), P' \subseteq P) \); and 3) \( P \in \sigma(t) \Rightarrow (\forall P' \in \sigma(t), P' \not\subseteq P) \). The interpretation of a surmise function \( \sigma(t) = \{P_1, \ldots, P_n\} \) is that every student who is able to learn an object \( t \) is also able to learn all objects from at least one element \( P_i \) of \( \sigma(t) \). The elements in \( \sigma(t) \) are presented as sets of prerequisites for an object \( t \).

The pair \( (T, \sigma) \) represents a surmise system \([76]\). For the surmise system, a knowledge structure \( \mathcal{K}_S \) on \( T \), in which an object \( t \) can have sets of its prerequisites \( \sigma(t) \),
is described as:

\[ K_S = \{K | (\forall t \in T, t \in K) \Rightarrow (\exists P \in \sigma(t), P \subseteq K)\}. \quad (3.6) \]

The knowledge structure \( K_S \) associated with the surmise system \((T, \sigma)\) is called a knowledge space on \( T \), which is closed under union. The knowledge space can represent a collection of all possible learning paths, for all objects in \( T \), with respect to the learning order relationship. We hereafter use \( K_S \) to denote the knowledge space unless explicitly stated otherwise.

Let \( T^K_O \) be the outer fringe [1] of \( K \in K_S \). \( T^K_O \subset T \) is a set of all objects \( t \) such that adding \( t \) to \( K \) forms another knowledge state \( K' \in K_S \). Let \( T^K_I \subset T \) be the inner fringe [1] of \( K \). \( T^K_I \) is a set of all objects \( t \) such that removing \( t \) from \( K \) forms another knowledge state \( K'' \in K_S \). Thus, objects in the outer fringe \( T^K_O \) are the next objects to be suggested to the student whose current knowledge state is \( K \).

**Example 3.1** Suppose a learning order relation \( \preceq \) on \( T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\} \) is specified by:

\[
\begin{align*}
 t_1 \preceq t_2, & \quad t_1 \preceq t_3, \quad t_1 \preceq t_4, \quad t_2 \preceq t_5, \quad t_3 \preceq t_5, \quad t_4 \preceq t_6, \quad t_5 \preceq t_7, \quad t_6 \preceq t_7.
\end{align*}
\]

The learning order relation \( \preceq \) on \( T \) implies the following relationships:

\[
\begin{align*}
 t_1 \preceq t_1, & \quad t_2 \preceq t_2, \quad t_3 \preceq t_3, \quad t_4 \preceq t_4, \quad t_5 \preceq t_5, \quad t_6 \preceq t_6, \quad t_7 \preceq t_7, \\
 t_1 \preceq t_2, & \quad t_1 \preceq t_3, \quad t_1 \preceq t_4, \quad t_1 \preceq t_5, \quad t_1 \preceq t_6, \quad t_1 \preceq t_7, \quad t_2 \preceq t_5, \\
 t_2 \preceq t_7, & \quad t_3 \preceq t_5, \quad t_3 \preceq t_7, \quad t_5 \preceq t_7, \quad t_4 \preceq t_6, \quad t_4 \preceq t_7, \quad t_6 \preceq t_7.
\end{align*}
\]

From Equation (3.5), a knowledge structure \( K_R \) on \( T \) is obtained:

\[ K_R = \{\emptyset, \{t_1\}, \{t_1, t_2\}, \{t_1, t_3\}, \{t_1, t_4\}, \{t_1, t_4, t_6\}, \{t_1, t_2, t_3, t_5\}, T\}. \]
It shows every object has only one set of prerequisites, providing one learning path. For example, to learn $t_6$ a student should learn $t_1$ and $t_4$, and there are no other paths available to reach $t_6$. More learning paths to the object $t_6$ can be found with a surmise function $\sigma(t_6) = \{\{t_1, t_4, t_6\}, \{t_1, t_2, t_4, t_6\}, \{t_1, t_3, t_4, t_6\}, \{t_1, t_2, t_3, t_4, t_6\}, \{t_1, t_2, t_3, t_5, t_6\}\}$. We can form a knowledge space $\mathcal{K}_S$ on $T$ as follows:

$$\mathcal{K}_S = \{ \emptyset, \{t_1\}, \{t_1, t_2\}, \{t_1, t_3\}, \{t_1, t_4\}, \{t_1, t_2, t_3\}, \{t_1, t_2, t_4\}, \{t_1, t_3, t_4\}, \{t_1, t_4, t_6\}, \{t_1, t_2, t_3, t_4\}, \{t_1, t_2, t_3, t_5\}, \{t_1, t_2, t_4, t_6\}, \{t_1, t_3, t_4, t_6\}, \{t_1, t_2, t_3, t_4, t_5\}, \{t_1, t_2, t_3, t_4, t_6\}, \{t_1, t_2, t_3, t_5, t_6\}, T \}. $$

For a knowledge state $K = \{t_1, t_2, t_3\}$ in $\mathcal{K}_S$, the outer fringe of $K$ and the inner fringe of $K$ are the sets $T^O_K = \{t_4, t_5\}$ and $T^I_K = \{t_2, t_3\}$, respectively. Therefore, a student may choose one of two objects, $t_4$ and $t_5$, as the next object to learn if the student has learned objects $t_1$, $t_2$ and $t_3$.

3.2 Treasure Hunts and Inquiry-Based Learning

In this section, an overview of current treasure hunts for learning and a treasure hunt process for IBL are presented in order to disclose the fundamental idea of our approach to the problem.

3.2.1 Treasure Hunts for Learning

Treasure hunt was originally an outdoor activity and a game played by children and occasionally by adults. To play treasure hunt, an adult prepares a list of hidden objects for children to find. Each team of children receives a duplicate list of the hidden objects. The winner is the first team to find all the items on the list.

Mechling [49] laid out a treasure hunt game played by a Boy Scout troop. The game requires game organizers to spend a few days preparing the hunt as follows: 1)
deciding game boundaries, the location for hiding the treasure and the location of the stations along the route from the camp to the treasure site; 2) creating a chain of clues that lead the game teams from one station to another; and 3) allocating the tasks and necessary equipment to the stations. He also described the important elements of a modern treasure hunt: rhymed clues, riddles, clues written in code, performances at stations of the hunt, and foodstuffs as treasures. During the treasure hunt, the scouts are requested to follow all the rules and use their physical skills and strategies to overcome given challenges.

Dr. Clue (http://www.drclue.com), a corporate team-building company specializing in business-focused treasure hunts, presented a reinvented treasure hunt model in which groups work together to solve a series of tricky riddles and clues, leading to mystery locations where answers appear. Blas et al. [3] introduced an online treasure hunt game in the SEE project providing students with a virtual learning environment. Students are included in online meetings and discuss previously studied themes under the active supervision of a guide, in a virtual museum. A treasure hunt in the SEE project helps students find a solution to cultural riddles provided by the museum, and enables them to review their learning in exciting ways. Hamelin [27] described an IBL experiment, in the form of a treasure hunt on the Web. It has been shown a treasure hunt can be a very effective tool in developing searching abilities, using the Internet [27].

### 3.2.2 A Treasure Hunt Process for Inquiry-Based Learning

A treasure hunt can be well matched to IBL. Treasure is considered to be information, truth, or knowledge, and hunt implies inquiry, which is a systematic investigation. Treasure hunt is an inquiry activity in which one systematically seeks knowledge with questions.

Figure 3.3 shows the process of the treasure hunt which we exploit in order to
design a WLSS for IBL. The process has the following four phases:

1. **Presenting Phase**: In an orientation, a guide provides students with a question regarding a topic which they will investigate via a treasure hunt, as well as the outlines of the topic. The background information is available from a resource and helps students easily understand the objectives of the topic and the question.

2. **Retrieving Phase**: Once students receive their orientation from the presenting phase, they are required to explore every necessary station using the given clues and meet guides who offer a related sub-topic description called help. Students need to understand the sub-topic, using its resource, so as to construct
a correct answer to the question. While exploring stations, students may meet and overcome several obstacles, and find treasures.

3. Developing Phase: When students obtain help at a station, they may develop ideas for the answer to the question through the “Construct knowledge” process, shown in Figure 3.3. A more detailed flowchart is depicted in Figure 3.4. Students analyze their collected information, and write an idea, from their analysis, into their logbook. If they require more information to understand the given information, they may search a resource. By reviewing the ideas in the logbook, the students determine where to find more helps with the question, and thereby construct a correct answer to the question.

4. Evaluating Phase: Students are required to take a test when they find a treasure at a station. In the evaluation, students must answer the question they received in the presenting phase, referring to ideas in their logbook. The answer is compared with the sample answer (prepared by the instructor) to determine whether they pass the test. If the students pass, they receive points and a
reward which is useful in overcoming obstacles they will meet in future rounds, and they are also able to know available topics for the next round of treasure hunt. The winner is the student or team with the greatest number of points.

Each phase has its corresponding phase in the IBL process described in Section 2.4.2. The IBL process is also observed in this process. Therefore, it can be said treasure hunt does include the idea of IBL.

3.3 A Treasure Hunt Model for Inquiry-Based Learning

Before designing a WLSS, featuring treasure hunt, it is necessary to clearly define a treasure hunt by creating a treasure hunt model. In so doing, we come to understand what components and functions are needed, and how the functions might work for the system conducting the treasure hunt for IBL. The treasure hunt model is thoroughly discussed in this section.

3.3.1 Structure of a Treasure Hunt Model

The treasure hunt model formalizes the treasure hunt, discussed in Section 3.2.2, using set theory in order to efficiently integrate the learning strategies of IBL into WLSS. It also provides an efficient way to combine the Web and online game technologies in the designing of WLSS for IBL.

The treasure hunt model consists of seven components: agents, treasures, information, events, actions, stations, and logbooks. Its structure is defined as follows:

\[ S = (AGT, TRE, INF, EVT, ACT, STA, LOG). \]  \hspace{1cm} (3.7)

- \( AGT \) is a finite nonempty set of agents representing students or guides in the system, and expressed as \( AGT = SD \cup GI \), where \( SD \) is the set of all students
and $GI$ is the set of all guides. Guides help students find treasures by providing useful information about the treasures.

- $TRE$ is a finite nonempty set of treasures. A treasure is the conclusion of a topic and a reward for learning the topic. The reward is useful in helping students overcome obstacles to find other treasures.

- $INF$ is a finite nonempty set of information about the treasures. This is defined as $INF = \{TP, RS, QS, CL\}$, where $TP$ is a finite nonempty set of learning topics, $RS$ is a finite nonempty set of resources, $QS$ is a finite nonempty set of questions, and $CL$ is a family of nonempty sets of clues. A learning topic is a finite nonempty set of learning objects necessary to master a topic. The resource is a collection of information sources related to a learning object. The question is what students must investigate while learning a topic. The clue is a rhymed hint about the next learning object to be found and learned.

- $EVT$ is a finite nonempty set of events which invite student participation. There are four different types of events: orientation events, help events, obstacle events and evaluation events.

- $ACT$ is a finite nonempty set of actions students can perform in order to find the treasure. Some major actions are as follows: finding the information, analyzing the information, retrieving the resource, documenting the findings and determining the direction.

- $STA$ is a finite nonempty set of stations where the event occurs and students initiate the actions. In accordance with the events, the stations can be grouped into the following types: orientation stations, help stations, obstacle stations and evaluation stations.
- **LOG** is a finite nonempty set of *logbooks* containing all records of the students’ important achievements during the treasure hunt. Students record their findings in the logbooks, the contents of which assist them in constructing new knowledge.

Let \( n \) be the number of topics for a course and let \( m \) be the number of sub-topics of each topic. \( TP \) of INF in Equation (3.7) can be expressed as \( TP = \{ L_1, L_2, \ldots, L_n \} \), where \( L_i \) (for \( 1 \leq i \leq n \)) is the learning topic. Learning topic \( L \) is described as \( L = \{ k_0, k_1, k_2, \ldots, k_m, k_{m+1} \} \), where \( k_0 \) is the *introduction* of the topic, \( k_i \) (for \( 1 \leq i \leq m \)) is a *sub-topic* of the topic, and \( k_{m+1} \) is the *conclusion* of the topic. In the treasure hunt model, a set \( T \) of all learning objects, which students are required to understand in order to complete a *course* through the treasure hunt, is defined as:

\[
T = \bigcup_{L \in TP} L. \tag{3.8}
\]

Let \( IT \subset T \) be a set of all the *introductions* of topics for a course, let \( SU \subset T \) be a set of all the *sub-topics*, and let \( CO \subset T \) be a set of all the *conclusions* of topics. Thus, the set \( T \) of all the objects of the course is also defined as:

\[
T = IT \cup SU \cup CO. \tag{3.9}
\]

The learning objects are allocated to stations and guides using a learning object allocation function \( f_{OA} \) given by:

\[
f_{OA} : T \rightarrow STA \times GI. \tag{3.10}
\]

If \( TN \) is a finite nonempty set of names of topics for a course, an orientation function
$f_{OT}$ representing the orientation event of $EVT$ in Equation (3.7) is described as:

$$f_{OT} : STA \times GI \times TN \to IT \times QS \times CL.$$ \hspace{1cm} (3.11)

The orientation function $f_{OT}$ allows a guide to provide an introduction, a question and a clue for a given topic name at a station. The introduction is called an orientation object. A help function $f_{HL}$ for the help event of $EVT$ is expressed as:

$$f_{HL} : STA \times GI \times QS \to SU \times CL.$$ \hspace{1cm} (3.12)

The definition of the help function $f_{HL}$ describes a guide furnishing a sub-topic description and a clue for a given question at a station. The sub-topic description is called a help object.

Let $f_{SM}(q, t, t')$ be a similarity function to measure the similarity of a student answer $t'$ to a sample answer $t$, provided by the instructor, for a given question $q \in QS$. If $S_q$ is a set of significant terms appearing in the question $q$, $S_t$ is a set of significant terms appearing in the sample answer $t$, and $S_{t'}$ is a set of significant terms appearing in the student answer $t'$, then the similarity can be calculated by:

$$f_{SM}(q, t, t') = \frac{|S_{t'} \cap (S_q \cup S_t)|}{|S_q \cup S_t|}.$$ \hspace{1cm} (3.13)

It means the similarity is measured by the ratio between the number of significant terms, included in both the student answer and the sample answer or question, and the number of significant terms, included in the sample answer or question.

If $RW$ is a set of rewards the student receives after passing a test at an evaluation event, the set of treasures $TRE$ in Equation (3.7) can be described by $TRE = CO \times RW$, where $CO$ is a set of all the topic conclusions as previously described.
Therefore, an evaluation function $f_{EV}$ for the evaluation event of $EVT$ is defined as:

$$f_{EV} : STA \times QS \times R(f_{SM}) \rightarrow TRE,$$  \hspace{1cm} (3.14)

where $R(f_{SM})$ is the range of $f_{SM}$. The evaluation function $f_{EV}$ expresses the fact a student is able to obtain a reward and a conclusion of the topic as a treasure, at a particular station, based on the similarity of their answer to a sample answer for a question. The conclusion is called an *evaluation object*.

The obstacle event gives students enjoyment, and hinders their ability to find the treasures. If $OB$ is the finite nonempty set of obstacles, an obstacle function $f_{OB}$ for the obstacle event of $EVT$ is described as:

$$f_{OB} : STA \times OB \times 2^{RW} \rightarrow \mathbb{N},$$  \hspace{1cm} (3.15)

which represents how well students overcome an obstacle at a station using rewards, which they obtained. Students acquire some points while struggling with the obstacle at a station, using the rewards they received.

### 3.3.2 Learning State of the Treasure Hunt Model

The system needs to know students’ current learning states in order to arrange the next objects the students are to learn, based on each student’s learning state. It is very clear the treasure hunt model should define the learning states and explain how to derive the next objects.

In Section 3.1.2, we discussed a way to discover the next objects from the current knowledge state of the student, using the learning order relationship. Now, the concept of the knowledge space, defined by Equation (3.6), allows us to discuss the learning state, which shows a student’s learning progress. Furthermore, we are able to
present how the current knowledge state of a student is changed by student’s learning experience.

If $LS$ is a learning state structure, $LS$ can be defined as a tuple:

$$LS = (SD, t_0, \lambda, LE, f_{LE}),$$

where $SD$ is a finite set of students, $t_0$ is a course introduction, $\lambda$ is the sequence of stations, $LE$ is the set of learning experiences a student has in order to complete a course, and $f_{LE}$ is an experience interpretation function.

Let $K_S$ be a knowledge space on $T$, defined in Equation (3.8), and $\lambda$ be the sequence of stations a student visits in the treasure hunt. Learning, in the treasure hunt model, takes place at a station $s_\lambda \in STA$ as a student takes a series of actions $A_\lambda$ based on the student’s current knowledge state $K_\lambda \in K_S$, for a given object $t_\lambda \in T_{K_\lambda}^O$, the outer fringe of $K_\lambda$. The series of actions $A_\lambda$ is expressed as $A_\lambda = (a_1, \cdots, a_l)$, where $l$ is the number of actions which the student takes at the station, and for $1 \leq i \leq l$, $a_i \in ACT$ from Equation (3.7).

The set $LE$ of learning experiences can be expressed as:

$$LE = \bigcup_\lambda (s_\lambda \times K_\lambda \times t_\lambda \times A_\lambda).$$

If there is a function $f(\lambda) : \mathbb{N} \to LE$, the experience interpretation function $f_{LE}$ is defined as:

$$f_{LE} : t_0 \times \prod_\lambda f(\lambda) \to K_n,$$

where $t_0$ is the course introduction and $K_n \in K_S$ is the new knowledge state of the student. $K_n$ will be used as the current knowledge state in the next station, replacing the old knowledge state of the student. As described in Section 3.1.2, for the new knowledge state $K_n$, the next available objects for the student can be denoted by
According to the learning state structure, the past learning states of a student are retrieved by changing the sequence $\lambda$. $LE$ in the structure may be a data storage, wherein the learning experiences of a student are kept. The system writes the learning experiences into the data storage. The past learning experiences can be the learning history of the student. The learning history is also retrieved from $LE$ by changing the sequence $\lambda$.

### 3.3.3 Knowledge Construction of the Treasure Hunt Model

It is important that the treasure hunt model defines the process of the knowledge construction, depicted in Figure 3.4. Figure 3.5 shows a high-level view of the treasure hunt model, including the knowledge construction. Learning objects in $T$ make up a knowledge space $K_S$ by applying learning order relationships between the objects. The student chooses a learning object $t_\lambda$ among the available objects derived from the knowledge space $K_S$, based on his/her current knowledge state $K_\lambda$.

In fact, for Equation (3.17), all the essential actions for learning are involved in the knowledge construction process. The first information, given to the student at the
station $s_\lambda \in STA$, is the object $t_\lambda$ the student chose for this round of treasure hunt. The student is able to construct knowledge by writing an idea, from the analysis of given information, into the logbook, and attain a new knowledge state $K_n \in K_S$. The student may retrieve the resource related to the object $t_\lambda$ to obtain more information and clearly understand it. The current knowledge state $K_\lambda$ is replaced with the new knowledge state $K_n$ for the next round.

In the system, the blog works as the logbook. If $IS$ is an idea structure describing the idea posted to the blog, $IS$ can be defined as a tuple:

$$IS = (QS, \lambda, SBJ, CMT, REL), \quad (3.19)$$

where $QS$ is a finite nonempty set of questions from $INF$ of Equation (3.7), $\lambda$ is the sequence of stations, $SBJ$ is a set of subjects, $CMT$ is a set of comments, and $REL$ is a set of relationships. The subject is the title of the comment, and the relationship represents the relationship between the question and the comment. A writing idea function $f_{WI}$, posting an idea to the blog, is defined as:

$$f_{WI} : QS \times K_S \times T \times RS \rightarrow IS, \quad (3.20)$$

where $T$ is the set of all the objects as defined in Equation (3.8), $K_S$ is the knowledge space on $T$, and $RS$ is a finite set of resources from $INF$ of Equation (3.7). The function $f_{WI}$ is a mapping function specifying an idea developed from the given learning object and related resource, based on the current knowledge state of the student in order to find a correct answer to a question.

The idea structure $IS$ presents fields which may compose a blog entry form. The function $f_{WI}$ also describes the information available from the system, allowing the student to fill out the fields on the entry form.
3.4 Summary

This chapter presents several different types of treasure hunts for learning, including a Boy Scout troop’s treasure hunt, a team-building treasure hunt, a treasure hunt in an online-learning game, and a treasure hunt using the Web. The chapter also specifies the process of a treasure hunt which implies the idea of IBL, and formalizes the treasure hunt into a treasure hunt model using set theory.

The treasure hunt model, which is a new Web-enabled IBL model, is made up of the following components: agents, treasures, information, events, actions, stations, and logbooks. Agents are live objects in the system; treasures represent topics students need to master; information is the content of the learning objects of the topics, as well as questions and clues necessary to obtain the treasures; events take place to provide students with the information and obstacles, and evaluate their treasure hunt; actions are the actions students can take according to the events; stations are the places where the events occur; and logbooks have the acquired information and ideas developed by students during their journeys to the treasures. The treasure hunt model furnishes the following functions: a learning object allocation function, an orientation function, a help function, a similarity function, an evaluation function, an obstacle function, an experience interpretation function, and a writing idea function.

To facilitate the definition of the treasure hunt model, we proposed a learning order relationship and its knowledge spaces at the beginning of this chapter. The learning order relationship represents a student’s preference to two related learning objects, and enables the student to specify the learning order of objects in a correct manner, based on his/her preconceived notions and existing personal knowledge. We also define a learning order relation, which is a binary relation on the objects with respect to the learning order relationship.

The knowledge space theory is involved to further investigate the properties of the
learning order relationship in this chapter. The knowledge structure and knowledge
space of the learning order relationship are defined. Through these definitions, the
chapter presents a surmise relation, a quasi-order relation, a surmise function and a
surmise system regarding the learning order relationship. Furthermore, the chapter
derives sets of prerequisites of an object from the knowledge space, and describes the
outer fringe and inner fringe of the current knowledge state of the student.

Finally, the chapter addresses the learning state, learning experience and knowl-
dge construction process of the treasure hunt model. The learning state shows the
student’s learning progress including the learning experiences of the student. The
knowledge construction process is expressed with both the idea structure which rep-
resents the idea posted to the blog, and the writing idea function which describes
posting an idea to the blog. The following chapter presents the implementation and
demonstration of a WLSS, complying with the treasure hunt model.
Chapter 4

Implementation and Demonstration

Chapter 4 details several algorithms for two major functions of OTHI, describes the implementation of its prototype, presents the demonstration of the prototype system with a topic from a data communications and network course, and discusses the usefulness of OTHI. OTHI is a WLSS for IBL and is designed on the basis of the treasure hunt model. The system employs an online treasure hunt game to allow students to experience IBL while playing the game.

4.1 Overall System Architecture

The architecture of OTHI is depicted in Figure 4.1. It basically follows the thin-client and server structure and three-layer architecture presented in Section 2.2.2. OTHI has the following major subsystems: (1) a treasure hunt game, (2) a teaching support subsystem, (3) a learning support subsystem, and (4) a membership management subsystem. The teaching support subsystem implements the teacher’s view of WLSS, while the learning support subsystem implements the student’s view of WLSS as described in Section 2.2.1.

The membership management subsystem manages various members such as the
present, past and affiliate members, and supports three different types of user accounts: students, instructors and administrators. The members’ profiles are managed on the user knowledge base. One of its most important features is the identification and authentication of each user. The user is able to access available services based on the authentication.

The teaching support subsystem enables instructors to design their courses as IBL. Instructors define course outlines, topics, questions, clues and related information using this subsystem. The system stores the information into the course knowledge base and generates IBL environments into the treasure hunt game. From this generation, treasure boxes containing questions, conclusions and rewards are placed at certain stations. The generation also allows non-player characters to have their own words.

Figure 4.1: An Architecture of OTHI.
to guide students as the guides of the treasure hunt model.

The learning support subsystem provides students with resources related to each topic as well as information about the treasure hunt. Students can search for more information about a topic using a Web search engine, if necessary. In addition, they are able to post their findings onto the blog, the data of which are stored in the user knowledge base. The system also automatically posts their achievements during the treasure hunt to the blog. The blog helps students develop their ideas and compose an answer to a question by tracing their findings.

Treasure hunt game provides a playground for the treasure hunt. All required information for the treasure hunt game is located in the game database. Students become motivated, and experience both the treasure hunt and IBL by playing the game. The learning support subsystem provides students with a virtual place, such as their home, where students complete homework and prepare for their next classes, while the treasure hunt game provides a different virtual place, such as a classroom where students learn from a teacher.

4.2 Algorithms

As the system is implemented based on the treasure hunt model, the system should provide two major functions as follows: (1) building a learning environment and (2) treasure hunting. Using the functions, an instructor is able to build a learning environment in which students are able to experience IBL through a treasure hunt. This section describes the two functions with algorithms: a learning environment building algorithm and a treasure hunting algorithm. Appendix B contains the algorithms.
4.2.1 Learning Environment Building Algorithm

The learning environment building algorithm presented in Algorithm 5 is an extension of both Algorithm 2 and Algorithm 4. Algorithm 2 and Algorithm 4 depict how a knowledge space $K_S$ and outer fringes of knowledge states in $K_S$, described in Section 3.1.2, are constructed for a course. A set $T$ of all objects of the course is already defined by the instructor, as described in Algorithm 7, and used as a data set of Algorithm 2. In Algorithm 2, line 1 initializes the knowledge space $K_S$, a knowledge state $K$, a set of next possible learning objects $T_N$ and an learning object $\vec{t}$. $K_S$ contains the empty set $\emptyset$ at line 2. Line 3 creates a new knowledge state and assigns it to $K$ by invoking the function Create_new_knowledge_state() with two parameters: a knowledge state $K$ and an learning object $\vec{t}$. Line 5 allows the instructor to specify the next possible learning objects for students who have mastered $\vec{t}$, and adds the objects to $T_N$. Line 6 gets an outer fringe $T^O_K$ of $K$ from $T_N$. Line 7 invokes the recursive procedure Building_knowledge_space_and_outerfringe() to get the rest of the knowledge states of $K_S$ and their outer fringes.

Algorithm 4 shows how the procedure Building_knowledge_space_and_outerfringe() works to complete building the knowledge space $K_S$ and get the outer fringes. The procedure has three parameters: a knowledge space $K_S$, a knowledge state $K$ and its outer fringe $T^O_K$. Lines 2-8 repeat until all objects in $T^O_K$ are processed. Line 2 invokes the function Create_new_knowledge_state() to create a new knowledge state. Line 3 adds the new knowledge state $K$ to $K_S$ if $K$ is not an element of $K_S$. Lines 5-7 are identical to lines 5-7 in Algorithm 2, and performed if $K$ does not have all the elements of $T$.

Algorithm 5 is the learning environment building algorithm, which is quite similar to Algorithm 2, except that it has one more data set to use and invokes the procedure Building_learning_environment() with an additional parameter, question. A set $STA$ of all stations the system provides is added as the data of the algorithm. It means
STA is already defined by the administrator of the system and each station in STA has some information about guides and treasure boxes which are placed at the station, as shown in Appendix C.1. Line 7 initializes question. Line 8 invokes the procedure Building_learning_environment() to actually generate components necessary to depict a treasure hunt for IBL in the treasure hunting algorithm.

The procedure Building_learning_environment() called in Algorithm 5 is shown in Algorithm 6. The procedure is also similar to Algorithm 4, except that it separately deals with each learning object according to the object’s type: an orientation object, a help object and an evaluation object, and invokes itself instead of the procedure Building_knowledge_space_and_outerfringe(). The procedure also includes the learning object allocation function defined in Equation (3.10).

In the procedure Building_learning_environment(), lines 2-33 repeat until all objects in $T^K_O$ are processed. Lines 3-8 are computed if the object $\vec{t}$ is an orientation object. Lines 3-5 find a station $\vec{s}$ which should have at least one guide available to keep some information. Lines 6-8 enable a guide to keep orientation information. Once the guide has the information, the guide is no longer available to keep other information. Lines 11-15 are processed if the object $\vec{t}$ is a sub-topic object. Lines 11-13 are identical to lines 3-5. Lines 14-15 get a guide from the station and give the guide the sub-topic information. Lines 18-25 are performed if the object $\vec{t}$ is an evaluation object. Lines 18-20 find a station $\vec{s}$ which should have at least one treasure box available to keep a treasure. Line 21 gets an available treasure box $\vec{b}$ from the station. Lines 22-25 put evaluation information into the treasure box $\vec{b}$.

According to Equation (3.8), for a learning topic $L$, the first object to learn is about the introduction of $L$ and the last object to learn is about the conclusion of $L$. To master the topic $L$, the student who had the orientation ends up taking the evaluation.

**Definition 4.1** We say that there must be an evaluation object, corresponding to an
orientation object, as a posterior object of the orientation object in the learning order of objects.

Line 32 recursively calls the procedure itself, and the parameter \textit{question} will be used for an evaluation object, which must be a posterior object of the orientation object \( \vec{t} \) in accordance with Definition 4.1.

4.2.2 Treasure Hunting Algorithm

The learning environment building algorithm describes a major function of the teaching support subsystem, while the treasure hunting algorithm describes a major function of the treasure hunt game. Algorithm 8 is the treasure hunting algorithm, whose input data contain outer fringes of all knowledge states \( \in \mathcal{K}_S \) and a set \( STA \) of all stations in the system. Guides and treasure boxes, placed at the stations, already have the information and rewards to be delivered to students, according to the learning environment building algorithm.

Line 1 initializes variables. \( n \) counts how many topics are mastered; \( clue \) contains a clue; \( \vec{t} \) is an object; \( \vec{s} \) is a station; \( points \) are experience points; \( RW \) represents a reward inventory the student possesses; and \( K \) is a knowledge state. Line 2 introduces the course to the student. Line 3 adds the course introduction object to the current knowledge state \( K \), and gets an outer fringe \( T_{\mathcal{O}K} \) of \( K \). Lines 5-33 repeat until the student learns all topics of the course. Line 5 allows the student to choose one object \( \vec{t} \in T_{\mathcal{O}K} \), which contains all the objects available to learn for the next round of treasure hunt. Line 6 invokes the function \textit{Explore\_stations\_and\_overcome\_obstacles()} to find an appropriate station for the object \( \vec{t} \). As presented in Algorithm 10, the function shows the student overcomes obstacles and gets some experience points while exploring stations.

Lines 8-13 are performed if the object \( \vec{t} \) is an orientation object. Otherwise, if the
object \( \vec{t} \) is a help object, lines 16-22 are processed. If the object \( \vec{t} \) is an evaluation object, lines 25-32 are executed. As defined in Equation (3.16), Equation (3.17) and Equation (3.18), line 8 calls the function Orientation() to give the student an orientation about a topic, line 16 invokes the function Help() to provide the sub-topic information, and line 26 invokes the function Evaluation() for the student to take a test if a treasure is found in a treasure box at the station \( \vec{s} \). Lines 10-12 are processed if the orientation was provided. Line 10 adds the object \( \vec{t} \) to the current knowledge state \( K \), and get an outer fringe \( T^O_K \) of \( K \). Line 11 displays a list of possible objects to learn and enables the student to choose an object \( \vec{t} \) for the next round. Line 12 calls the function Get\_clue\_for\_next\_station() to have a clue, leading to the station where the next object \( \vec{t} \) will be treated.

Lines 18-21 are processed if the sub-topic information was provided. Line 18 invokes the function Knowledge\_construction() to construct an idea using the sub-topic information and get a new knowledge state \( K \) as described in Algorithm 15. Line 19 gets an outer fringe \( T^O_K \) of \( K \). Line 20 is identical to lines 11, and line 21 is quite similar to line 12. Lines 28-30 are performed if the reward is available. Line 28 adds the reward to the reward inventory \( RW \). Line 29 is identical to line 10. Line 30 initializes variables and increases \( n \) by 1.

According to Equation (3.8), for a learning topic \( L \), the last learning object is the evaluation object of \( L \). It means a new topic will be introduced to the student at the next time, and the next object must be the orientation object of the new topic.

**Definition 4.2** We say that every element of the outer fringe of an evaluation object should be an orientation object in the learning order of objects.

Line 30 shows the evaluation event does not provide any clue and the orientation information \( \vec{o} \) about the current topic is no longer necessary, since a new topic will be treated for the next round according to Definition 4.2.
Table 4.1: Open source softwares used to develop the system.

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blojsom</td>
<td>3.2</td>
<td>Java-based multi-blog system</td>
<td><a href="http://wiki.blojsom.com">http://wiki.blojsom.com</a></td>
</tr>
<tr>
<td>Joomla</td>
<td>1.5.12</td>
<td>Content management system</td>
<td><a href="http://www.joomla.org">http://www.joomla.org</a></td>
</tr>
<tr>
<td>Project Darkstar</td>
<td>0.9.10</td>
<td>Online game server</td>
<td><a href="http://www.projectdarkstar.com">http://www.projectdarkstar.com</a></td>
</tr>
<tr>
<td>Slick</td>
<td>0.7.2</td>
<td>Java library for 2D game</td>
<td><a href="http://slick.cokeandcode.com">http://slick.cokeandcode.com</a></td>
</tr>
<tr>
<td>Stendhal</td>
<td>0.69</td>
<td>Multiplayer online adventures game</td>
<td><a href="http://arianne.sourceforge.net/">http://arianne.sourceforge.net/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="http://arianne.sourceforge.net/games/game_stendhal">http://arianne.sourceforge.net/games/game_stendhal</a></td>
</tr>
</tbody>
</table>

4.3 Development Environment and Available Technologies

Developing a website including an online game usually requires many system resources, financial resources and manpower. Although this is a prototype system, it is too heavy of a workload for only one software developer if (s)he develops the whole system on their own. Therefore, it is imperative to exploit open source softwares and servers available for free. A Java-based development environment has a greater chance of finding applicable open source softwares and freely-available systems than C# and DotNet-based development environments. Hence, Java is the primary language of choice. In this section, all the software packages and core technologies, used in the development, are presented in detail.

4.3.1 Use of Open Source Softwares and Freely-available Servers

Table 4.1 shows open source softwares involved in the development. Joomla is a content management system developed with PHP, which is a general-purpose scripting language for Web development, and uses MySQL database as its data storage. Joomla provides almost all the components the prototype system needs for its essential Web features: the homepage, the learning support subsystem, the membership management subsystem, forums, and interfaces to other subsystems and functions.

In addition, Joomla has many free extensions, to which many developers around the world contribute. For example, GCalendar, developed by Allon Moritz, using Google Calendar Data API, described in Section 2.3.2, offers a component which
shows Google Calendars inside a Joomla-based web page. Kunena Forum (http://www.kunena.com) is one of the popular extensions in the forum category of Joomla. GCALendar and Kunena Forum are applied to the implementation of the system without modifications.

**Blojsom** is a Java-based multi-blog system combined with MySQL database. The logbooks presented in Equation (3.7) are implemented using this blog system. **Blojsom** has its own identification and authentication mechanism. Hence, to integrate this blog system into the learning support subsystem, based on Joomla, it is inevitable some modules and database tables, in both Joomla and Blojsom, will be added and modified.

**Project Darkstar** is an open source, online game server. It is responsible for communicating between game server and clients. It also maintains the last status of game objects the game server manages, and forms the infrastructure of the treasure hunt game. **Slick** is a Java-based 2-dimensional game development library through which one is able to easily access OpenGL, which is an API for creating 2D and 3D graphics. The Slick library manages the images and sounds of the treasure hunt game. The client portion of the game is implemented based on the Slick library.

**Stendhal** is an open source, multiplayer online adventure game. All the graphic images and sounds, implemented in the treasure hunt game, are obtained from Stendhal. The treasure hunt game world is also made up of 2-dimensional game maps and related XML codes taken from Stendhal. Thus, it is possible to remove a large amount of graphic related work and reduce a large amount of development time.

Table 4.2 presents the database and Web servers, installed to form the development environment. **Apache HTTP Server** is a freely-available HTTP (Web) server. Joomla-based web pages of OTHI run on this server. **Apache Tomcat** is a Java servlet container, which implements the Java Servlet and JavaServer Pages technologies. **Apache Tomcat** is responsible for the execution of Blojsom and the teaching support
subsystem of OTHI. MySQL is a database server, and manages the user knowledge base. Berkeley DB is an embedded database with bindings in Java and manages the game data base.

### 4.3.2 Useful Tools and Other Technologies

In the design and development of the system, the tools, listed in Table 4.3, are very helpful. StarUML is an open source UML tool running on Win32 platform. It supports UML 2.0 and has four modeling approaches: 4+1 View Model, Default Approach, Rational Approach and UML Component. The treasure hunt game is designed with the Default Approach of this modeling tool. NetBeans is a modular, standards-based, integrated development environment (IDE) for Java developer. Its profile feature shows the CPU and memory usage of each Java class and method, and supports finding bottlenecks in the treasure hunt game. BMFont generates fonts for letters appearing in the game. Paint.net supports drawing and editing graphic images. Tiled is an editor for the 2-dimensional map and helps organize the 2D game map for the game.
Table 4.4 lists a few algorithms and other technologies used in developing the game. A* Pathfinding Algorithm and Dijkstra’s Algorithm are pathfinding algorithms. The game exploits the algorithms to give the most suitable paths to avatars which are digital characters moving forward in the game. Porter stemmer is an algorithm for suffix stripping, which is a crucial technology in a Web search engine. The game uses the algorithm to extract significant words from the question, sample answer and student answer in order to measure the similarity, as denoted in Equation (3.13). Java Web Start provides a method of deploying the game to the user. XML files work as the course knowledge base. Appendix C presents some important xml files of this system.

### 4.4 Implementation of Essential Features

OTHI has the following major components: the homepage, teaching support subsystem, learning support subsystem, treasure hunt game and membership management subsystem. There is no need to describe the membership management subsystem further. In this research, all the functions the subsystem provides are implemented by Joomla Administration. Some functions of the subsystem are treated in Section 4.4.1. One may refer to the Joomla website, specified in Table 4.1, if (s)he wishes more information. In this section, the implementation of the remaining components is presented.
4.4.1 Homepage of OTHI

As shown in Figure 4.2, the homepage consists of a logo, a top menu, Search, Main Menu, Key Concepts, Links, Latest News, Popular, and User Login. The logo signals the student to solve a puzzle and answer a question. Home, News, Forums and FAQ make up the top menu. The Search in the upper right-hand corner of the page performs a local search. The Main Menu provides Home, OTHI Overview, More about OTHI, Treasure Hunt Game, Local Search, and News Feeds. The Key Concepts furnish the system’s background concepts such as: inquiry-based learning, Web-based learning support systems, treasure hunt and the treasure hunt model. The Links offer links to a few useful websites to obtain more information and a link to the administrator web page as well. The Latest News lists links to the latest news about
the system and the Popular shows links to the popular articles the website provides. The User Login has two input fields, Username and Password. When a user logs into the system, a User Menu appears above the Main Menu. A welcome message is displayed in the middle of the homepage.

Available content and features are restricted by the user group to which the user belongs. Figure 4.3 and Figure 4.4 show the instructor’s User Menu containing more features than the student’s User Menu. There are four Front-end User Groups: Registered, Author, Editor and Publisher. The content and features of OTHI are categorized by access levels: Public, Registered and Special. Only the Author, Editor, or Publisher group can access the special-level content and features. Instructors have the Publisher privilege of being allowed to access the teaching support subsystem through a menu Organize IBL, whose access level is the Special, while students are assigned to the Registered.

4.4.2 Teaching Support Subsystem

The teaching support subsystem (TSS) allows instructors to lay out their courses and build their own IBL environments into the treasure hunt game. Figure 4.5 shows
the main features of TSS as follows.

1. **Edit Course Introduction.** The instructor inputs the course title, course objectives and course references on the Edit Course Introduction page, as described in Algorithm 1.

2. **Edit Topic Orientation.** On the Edit Topic Orientation page, the instructor edits the topic title, introduction and question which is offered to a student by an NPC at an orientation station.

3. **Edit Topic Evaluation.** On the Edit Topic Evaluation page, the instructor enters a topic conclusion and a sample answer to the question, which the instructor presented on the Edit Topic Orientation page. In addition to the conclusion and sample answer, a clue, which directs the student to the evaluation station of the topic, is also specified on the Edit Topic Evaluation page.
4. **Add Sub-Topic.** The instructor defines a sub-topic title, a description of the sub-topic and a clue on the Add Sub-Topic page. The description is the core information regarding the sub-topic. The instructor may consider a two or four-line rhymed clue which is ambiguous enough to hide its meaning at first glance, but stimulate students to discover the meaning. The clue will direct the student to the help station of the sub-topic.

5. **Build Learning Environment.** Build Learning Environment enables the instructor to define connections between the information specified on the Edit Topic Orientation, Edit Topic Evaluation and Add Sub-Topic pages, and game objects in the treasure hunt game. The instructor also defines the learning order relationship between concepts by using the field “Next stations” on the Build Learning Environment page.

Those definitions are converted into records in XML files that are inputs to the game. Appendix C presents the XML files. Algorithm 7 depicts Edit Topic Orientation, Edit Topic Evaluation and Add Sub-Topic features. Build Learning Environment feature is described in Algorithm 5 and Algorithm 6, as detailed in Section 4.2.1.

### 4.4.3 Learning Support Subsystem

The learning support subsystem supports both the treasure hunt and IBL of each student. The subsystem also provides topic resources for students and aides them in managing the complex, extended activities of the treasure hunt. The following is the available features of the subsystem.

1. **Calendar.** As described in Section 2.3.2, the calendar enables the student to make an appointment with others to collaborate in the treasure hunt. Once a student creates a Google Calendar and the system administrator enrolls its
Calendar ID through the Joomla Administration, other people can share the calendar.

2. **Browse Blog.** It opens a blog window and allows the student to read articles posted by the system, as well as by the student.

3. **Post Blog.** The student posts new articles and revises old articles through this feature. As previously described, the blog works as the logbook of the treasure hunt model.

4. **Google Search.** Google Search supports students in finding the resources for the treasure hunt. The Links in the lower left-hand corner of Figure 4.2 include a link to the Google Search.

To attach Browse Blog and Post Blog features, implemented with Blojsom package, to the homepage based on Joomla, a Joomla component is developed using a programming language PHP. Appendix D contains the program sources.

### 4.4.4 Treasure Hunt Game

Treasure hunt game is a 2-dimensional multiplayer online learning game developed with Java and its client program is deployed using Java Web Start. Its JNLP (Java Network Launching Protocol) definition is presented in Appendix C.9. When the game server is started, it loads all configurations about game objects into memory, including the course definitions, from the XML files, presented in Appendix C, and initializes the game objects, as denoted in lines 77-85 of Appendix E.2. Appendix E contains: the server main program, client main program, authentication server program and database definition program. The database definition program provides database and table information necessary to process the authentication of players,
using the authentication server program, and manage the blog data. The client main program specifies the server ip address and tcp port number to connect.

As denoted in Section 4.2.2, Algorithm 8 shows the game’s major functions. Players can access the treasure hunt game through the authentication process, as shown in Figure 4.6. Basically, a player tries to find a hidden treasure, denoted in a quest with given clues, exploring the game world and overcoming challenges. If the player acquires enough experience points, (s)he can level up, become stronger and faster, and is allowed to equip herself/himself with higher level weapons and armors. Completing the quests, overcoming challenges, and finding treasures give the player rewards and experience points. Some equipment, tools, and aids are required in order to find the treasures and defeat the monster which is considered to be one of the obstacles in the treasure hunt model. The player can buy those valuable items with golds, acquired as rewards.

The hidden treasure may be an appropriate answer to the question presented in a quest. The quest is a request the player explores and finds the answer by visiting
stations and constructing ideas. The Non-Player Character (NPC) works as the guide and personifies one of the sub-topics of the topic. The names and places of all NPCs in the game world may consist of words relevant to the sub-topics. If the player meets an NPC and the help event occurs, the NPC tells the player the description of a sub-topic and a clue for the next station. It offers an opportunity to further investigate the given question and learn the topic in-depth. The treasure hunt game posts the information, provided by the NPC, to the blog, as mentioned in Section 3.3.3. Each found treasure causes an evaluation event, in which the player should take a test and give the answer developed by the treasure hunt. The test score demonstrates the student’s understanding of the topic. A lack of score results is a failed quest.

Figure 4.7 is a screenshot of the treasure hunt game, showing multiple users logged on. There is an orange bar called Control Panel on the bottom of the screenshot.
It has nine buttons. HELLO, HELP and BYE buttons are required to conduct a conversation with an NPC. The HELLO button attracts the NPC’s attention, makes the NPC stop moving, and starts a conversation with the NPC. The HELP button asks the NPC for the information, addressed in Equation (3.7). The BYE button ends the conversation and allows the NPC to go on its way.

The conversations between the player and the NPC are left in a message log. The Message Log window, on the left side of Figure 4.7 allows the conversations in the message log to be displayed on the game screen. The CLEAR button on the Control Panel empties the message log. A small button on the right side of the CLEAR button disables and enables the Message Log window. A list of the next learnable objects is displayed when the player clicks the NEXT button. The TREASURES button displays the information regarding the treasures found by the player, and the QUESTS button displays the information about the quests the player has received from the NPCs. The STOP button halts the playing of the game.

A left click of the mouse points to a spot where the player’s avatar moves toward. If the player’s avatar is close enough to attack a monster, a check symbol is displayed when the mouse hovers over the monster. A right click of the mouse attacks the monster if the check symbol is on it. If the player’s avatar is near a treasure box, it is opened by left clicking on the treasure box and the player takes note if it has a treasure. Every avatar has one name appearing on the bottom of it. When the mouse is over the avatar, its game level is also displayed.

A status panel is always displayed in the upper right-hand corner of the game screen, as shown in Figure 4.7. The status panel depicts the status of the player in the game with the following information: Level, Gold, HP and EXP. The Level shows the game level of the player and the Gold indicates how many golds the player has. HP stands for Health Points, and the player cannot move for a while if the HP is zero due to an attack from a monster. The EXP represents the experience points a player
has obtained. If the EXP bar is full, the player levels up, the EXP is set to zero, and the maximum of the HP is increased.

### 4.4.5 Dependence of the Treasure Hunt Game on Learning

Students are eager to receive a quest, prepare for an answer, find a treasure, and answer the open question, since they receive a better reward from completing a quest than from hunting monsters. Suppose a level-one player tries to level up by hunting rats. The player must hunt 100 rats and spends a great deal of time, taking care of the health points. Hunting a rat requires the player to hit the rat ten times at sacrificing of his/her health points, just offers the player one experience point and drops a cheese or some golds if the player is lucky.

On the contrary, as represented in Figure 4.12, a quest reward may include 1,000 golds and 500 experience points, which is sufficient for a one-level player to level up. If the player receives remarkable marks for the question, the reward includes one of the following: a tool, an equipment, an armor or a weapon. After completing the quest, the player becomes a level-two player who can hunt the rat by hitting it only once. The rats can no longer prohibit the player from finding other treasures. The player may challenge higher level of treasures. Students quickly come to recognize learning a topic is the best way to upgrade their avatars and to win respect from other students.

### 4.5 A Demonstrative Example

This section depicts how each feature of the system works for learning in a data communications and network course. Suppose an instructor prepares a lecture regarding the data link layer (DLL) of the course using the TSS; the topic must be the data link layer.
4.5.1 Finding Learning Order Relationships

According to the instructor’s lecture notes on the data link layer, the instructor may have the following learning objects: DLL introduction, DLL duties, error detection, flow control, parity checking, etc. In addition, the learning order relationships between the objects, are obtained using the learning order relation \( \preceq \), given by Equation (3.4), on the set of the objects as follows:

- DLL Introduction \( \preceq \) DLL Duties, DLL Duties \( \preceq \) Error Detection,
- DLL Duties \( \preceq \) Flow Control, Error Detection \( \preceq \) Parity Checking,
- Flow Control \( \preceq \) Parity Checking.

Since the object “DLL Duties” has two outer fringes – Error Detection and Flow Control – from the learning order relationships, the instructor can determine two possible learning paths: 1) DLL introduction \( \rightarrow \) DLL Duties \( \rightarrow \) Error Detection \( \rightarrow \) Flow Control \( \rightarrow \) Parity Checking; and 2) DLL introduction \( \rightarrow \) DLL Duties \( \rightarrow \) Flow Control \( \rightarrow \) Error Detection \( \rightarrow \) Parity Checking. The instructor may allow students to choose one of the two paths by defining the two objects as the outer fringes of “DLL Duties.” Otherwise, the instructor fixes the learning path by choosing one. In this case, the first learning path is chosen.

4.5.2 Editing Topic Information

Before defining a topic using the Edit Topic pages shown in Figure 4.8 and Figure 4.9, the instructor must select a course, which is already defined on the Edit Course Introduction of TSS. Similarly, to define a sub-topic with the Add Sub-Topic of TSS, the instructor is also required to select both a course and a topic, to which the sub-topic belongs. For the learning object “DLL introduction,” the instructor enters the topic title with “Data Link Layer,” and also its introduction and question
Edit Topic Orientation

Course: CS335 Data Communications and network

**Topic Title**

Data Link Layer

**Topic Introduction**

Data link layer supervises the flow of information between adjacent network nodes. It uses error detection or correction techniques to ensure that a transmission contains no errors.

**Topic Question**

Describe what the data link layer works for networking.

Figure 4.8: Edit Topic Orientation of TSS.

Edit Topic Evaluation

Course: CS335 Data Communications and Network

**Topic: Data Link Layer**

**Topic Question**

Describe what the data link layer works for networking.

**Sample Answer**

The data link layer is layer two of the seven-layer OSI model as well as of the five-layer TCP/IP reference model. It receives services from the physical layer and provides services to the network layer. The data link layer is responsible for carrying a packet from one node to another and has only a local responsibility. Its main functions are error control and flow control.

**Conclusion**

The task of the data link layer is to convert noisy lines into communication channels free of transmission errors for use by the network layer.

**Treasure Clue**

The treasure of the data link layer is in a storage of the town hall.

Figure 4.9: Edit Topic Evaluation of TSS.
on the Edit Topic Orientation page as depicted in Figure 4.8. Appendix C.3 shows
the output of this work. On the Edit Topic Evaluation page, the instructor enters a
sample answer to the question presented on the Edit Topic Orientation page, a con-
clusion of the topic and a clue as to its evaluation station. Its results are displayed
in Appendix C.4.

Other learning objects, except for “DLL introduction,” become the sub-topics
of the data link layer. The instructor defines the sub-topics on the Add Sub-Topic
pages. Figure 4.10 shows the definition of a sub-topic “DLL Duties.” The instructor
enters its title and description, and also a clue as to its help station. Other sub-
topic definitions can be performed in a similar manner. The results are shown in
Appendix C.5.

4.5.3 Building an IBL Environment

Once the instructor finishes making up a topic with the Edit Topic Orientation,
Edit Topic Evaluation and Add Sub-Topic pages, the instructor must build a learning
Building Learning Environment

Course
CS335: Data Communications and Network

Game Map
int_network_townhall

Guide
Mayor

Topic
Data Link Layer
Nick Name
Town Hall
Nick Name

Station
Orientation
DLL Introduction

Next Stations

Introduction by guide
I want to introduce the data link layer to you. The data link layer supervises the flow of information between adjacent network nodes. It uses error detection or correction techniques to ensure that a transmission contains no errors.

Question by guide
You need to describe what the data link layer works for networking in order to complete this quest.

Figure 4.11: Build Learning Environment for orientation

Building Learning Environment

Course
CS335: Data Communications and Network

Game Map
int_townHall_storage

Treasure Box
Box 13

Topic
Data Link Layer
Nick Name
Town Hall Storage

Station
Evaluation

Conclusion
The task of the data link layer is to convert noisy lines into communication channels free of transmission errors for use by the network layer.

Reward
Gold: 1000
Experience Points: 500

Tools
Pick

Armors
Leather scale armor

Weapons

Figure 4.12: Build Learning Environment for evaluation.
environment into the treasure hunt game for the topic, using the Build Learning Environment of TSS. As shown in Figure 4.11, for the topic “Data Link Layer,” the instructor assigns a game map “Town Hall” as the orientation station and NPC “Mayor” as the guide, in accordance with the learning object allocation function, defined in Equation (3.10). The instructor also modifies the introduction of the topic to simulate a conversation. For the evaluation station, the instructor chooses a treasure box, “Box 13,” in “Town Hall Storage,” and sets 1,000 gold, 500 game experience points and so on, as a reward for the quest, as depicted in Figure 4.12.

Meanwhile, according to the building definition of “DLL Duties,” shown in Figure 4.13, NPC “Boy” in “Town Hall” will deliver the description, shown in Figure 4.10, to the students. Sub-topic “Error Detection,” specified in the field “Next Stations,” is an outer fringe of the sub-topic. The combination of game objects and corresponding learning objects is described in Table 4.5. Table 4.5 lists the related events, stations, game objects and learning objects. Appendix C.6, Appendix C.7 and Appendix C.8 are the results of this work.
Table 4.5: Combination of game components and learning objects.

<table>
<thead>
<tr>
<th>Events</th>
<th>Stations</th>
<th>Game Objects</th>
<th>Learning Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orientation</td>
<td>Town Hall</td>
<td>NPC Mayor</td>
<td>DLL Introduction</td>
</tr>
<tr>
<td>Help</td>
<td>Town Hall</td>
<td>NPC Boy</td>
<td>DLL Duties</td>
</tr>
<tr>
<td>Help</td>
<td>Network City</td>
<td>NPC Alice</td>
<td>Error Detection</td>
</tr>
<tr>
<td>Obstacle</td>
<td>Network Library Entrance</td>
<td>Rats</td>
<td></td>
</tr>
<tr>
<td>Help</td>
<td>Network Library</td>
<td>NPC Monogenes</td>
<td>Flow Control</td>
</tr>
<tr>
<td>Obstacle</td>
<td>Town Hall Storage Entrance</td>
<td>Rats</td>
<td></td>
</tr>
<tr>
<td>Evaluation</td>
<td>Town Hall Storage</td>
<td>Box 13</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>

4.5.4 Experiencing IBL

Figure 4.14 shows a journey to the treasure of the data link layer. In the game world, the name of the learning object each NPC will deal with is displayed as the name of the NPC. From the orientation function, defined in Equation (3.11), as depicted in Figure 4.15, a student receives the quest about the data link layer from a NPC “DLL Introduction,” including the background information and question. The student posts what (s)he has understood, with the orientation information, to the blog as the writing idea function, described in Equation (3.20). To write an idea about it, the student may search for more information with key words “data link layer” or “introduction data link layer” using the *Google Search*.

After posting the idea, the learning object “DLL introduction,” indicated in Table 4.5, is added to the knowledge state of the student according to the experience interpretation function expressed in Equation (3.18). For the next available sub-topics, there is only one sub-topic available, “DLL Duties,” in accordance with the building definition for the orientation shown in Figure 4.11. Hence, NPC “DLL Introduction” tells the student the clue: “A boy is working in a town hall.” The student finds the boy called “DLL Duties” in a town hall.

When the student meets NPC “DLL Duties,” as described by Equation (3.12), (s)he may learn the basic duties of the data link layer from the NPC. To gain further
Figure 4.14: A Treasure Map.

Figure 4.15: Orientation for Data Link Layer in the treasure hunt game.
understanding, the student searches for additional information, relevant to the sub-topic “DLL Duties,” using the search engine. Then, the idea regarding the sub-topic is posted to the blog, and the sub-topic becomes a member of the knowledge state of the student. For the next sub-topic “Error Detection”, NPC “DLL Duties” gives the student the clue: “A woman is waiting for you in a city.” The student moves to find the woman called “Error Detection” in a city.

The woman can be found in “Network City.” The woman talks about the error detection of the data link layer and shows a clue for NPC “Flow Control” as the next object to learn. The student meets rats when (s)he tries to enter the “Network Library” on the way to find NPC “Flow Control,” as shown in Figure 4.16. The student is required to defeat the rats in order to enter this place and check if the NPC is there. According to the obstacle function, defined in Equation (3.15), some experience points are obtained as the student expels the rats using his/her weapons.

After overcoming all the challenges, the student finds a treasure box in “Town Hall
Figure 4.17: Ask for an answer in the treasure hunt game.

Figure 4.18: Complete a quest in the treasure hunt game.
Figure 4.19: Blog posts about sub-topic Flow Control.

Storage. When the student opens it, as shown in Figure 4.17, (s)he is asked to give an answer to the question to complete this quest in accordance with the evaluation function described in Equation (3.14). For the treasure, as defined in Figure 4.12, the student may gain 1,000 gold and 500 game experience points if (s)he passes. Figure 4.18 represents the student achieving 79 points for the question. Enough experience points enable the student to level up. The gold the student received will be spent on arms to help defeat the obstacles (s)he will face in the next round of the treasure hunt. The game level of the student is one of the measures indicating how well (s)he has performed learning.

As previously described, during the journey to the treasure hunt, the student is required to write his/her idea about each sub-topic in the blog. Each post to the blog
forms the idea structure, defined in Equation (3.19). Figure 4.19 shows the blog posts about the sub-topic “Flow Control,” written by the system, as well as by the student. “Flow Control” is the subject and the “sub-topic” represents the relationship of the idea structure. The posts also contain comments below the subject portions. The question is shown on top of the blog. This mechanism can help the student organize ideas about the question and give the appropriate answer when the student finds the treasure and takes the test.

4.6 Usefulness of OTHI for Learning

The usefulness of fully implemented OTHI can be shown by means of contrastive evaluation of OTHI supported learning against other learning approaches: traditional face-to-face classroom learning (TFCL), and passive, asynchronous Web-based learning (PAWL). The evaluation was performed by investigating the comparative advantages and disadvantages of each learning approach in previous studies related to Web-based learning, inquiry-based learning, or learning games.

The following five attributes were developed to deal with the pros and cons of the learning approaches.

1. **Availability**: Learning availability, including the accessibility and reuseability of learning services.

2. **Flexibility**: Flexibility and adaptability of learning content and process, according to students’ individual needs.

3. **Interactivity**: Real-time interactivity between students and instructors or learning content, and among students.

4. **Motivation**: Learning motivation and engagement, encouraging students to actively participate in learning.
5. **Student-centricity**: Student-centricity in the process of teaching and learning.

Table 4.6 presents the results found in an investigation of the evaluation. Each learning approach was evaluated and classified (low, medium, high) based on its comparative advantages and disadvantages as reported in the literature for the attributes and compared with the other learning approaches. The first column lists the evaluation attributes, and the values of the attributes for each learning approach are found in the second, third, fourth and fifth columns. The last column parades the literature from which the results are served for each attribute.

The availability of TFCL is relatively low when compared with other learning approaches, while PAWL shows a high availability of learning services. The OTHI supported learning approach may include either PAWL or TFCL. If OTHI supports students in PAWL, its availability is the same as the availability of PAWL. Otherwise, if OTHI supports students in TFCL, the overall availability of the learning services cannot attain the availability of the pure web-based learning due to the low availability of TFCL, in spite of the support of OTHI to improve the availability.

Previous studies reported TFCL provides students with higher flexibility and interactivity than PAWL. In a traditional classroom, if a student does not understand a particular section of a learning material, the student can raise questions and receive an immediate presentation about the material from the teacher who uses an example, a story, or just more detail. The teacher can adapt content and pace to the rate at

Table 4.6: Comparative and contrastive evaluation of OTHI and other learning approaches.

<table>
<thead>
<tr>
<th>Evaluation attributes</th>
<th>Traditional learning (TFCL)</th>
<th>Web-based learning (PAWL)</th>
<th>TFCL with OTHI</th>
<th>PAWL with OTHI</th>
<th>Related research papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>[2, 10, 12, 26, 37, 75, 84, 86]</td>
</tr>
<tr>
<td>Flexibility</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>[6, 10, 12, 21, 33, 50, 75, 85, 86]</td>
</tr>
<tr>
<td>Interactivity</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>[10, 11, 28, 34, 42, 56, 62, 65, 68, 85]</td>
</tr>
<tr>
<td>Motivation</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>[3, 13, 17, 30, 31, 33, 35, 38, 58, 62, 66, 69, 71]</td>
</tr>
<tr>
<td>Student-centricity</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>[18, 21, 22, 26, 31, 37, 39, 68, 74, 86]</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>8</td>
<td>14</td>
<td>13</td>
<td>(Low=1, Medium=2, High=3)</td>
</tr>
</tbody>
</table>

78
which students understand the material. In PAWL, neither the instructor nor the
delivery system can adapt the course presentation to different students, and online
students perceive less interactivity compared to students in the traditional classroom.

On the other hand, the features of OTHI (such as the learning state, forum, Web
search engine and online treasure hunt game) are able to enhance the flexibility and
interactivity of PAWL. A student who uses OTHI in learning is able to find and
learn appropriate topics based on his/her learning state, and does not need to waste
time learning irrelevant or already known material. In addition, the student cannot
only find appropriate answers to questions about the topics at the forum or on the
Web using the search engine, but is also able to receive instant feedback regarding
the questions from other students who have already investigated the questions in the
treasure hunt game. The guide, an animated character [65], in the treasure hunt
game provides a more interactive learning experience to the student. Nevertheless,
PAWL with OTHI cannot attain the flexibility and interactivity of TFCL, unless the
animated character can compete with the human instructor.

As for the motivational aspect of the learning approaches, TFCL is more motiva-
tional than PAWL. Zhang et al. [86] specified that one of the advantages of TFCL
is “motivating students,” as compared to PAWL. Furthermore, previous studies on
Web-based learning reported PAWL makes it difficult for students to become moti-
vated for learning.

OTHI supported learning provides students with higher motivation for learning
than TFCL. Learning games can enhance students' learning motivation when applied
in classrooms or other learning scenarios. Particularly, online-learning games can
arouse the motivation of the students by their enjoyment of playing the game and
learning a topic at the same time. The online treasure hunt game of OTHI can be
a valuable learning motivator and promote learning motivation. Furthermore, the
inquiry activities that the game furnishes can create a motivation to learn. The guide
works as an animated pedagogical character [64, 13] and also helps to increase the student’s motivation and engagement.

TFCL exhibits relatively lower student-centricity, representing a teacher-centered learning, when compared to other learning approaches. The high availability of PAWL increases the student-centricity of the learning approach, while the low availability of TFCL decreases its student-centricity, despite its high flexibility and interactivity, since the flexibility and interactivity are mostly controlled by the teacher, not by the student. Even though PAWL shows high availability compared to OTHI supported learning, its student-centricity is lower than OTHI supported learning due to its low flexibility, interactivity and motivation, on which student-centricity is also dependent. In addition, OTHI supported learning contains inquiry-based learning in which students are the center of learning and the role of the instructor is to assist the students in constructing their own knowledge.

The last row of Table 4.6 indicates the total results of the evaluation, which may represent the learning effectiveness of the learning approaches. TFCL with OTHI shows the highest effectiveness, while PAWL shows the lowest effectiveness. In addition, the results reveal TFCL is still a better learning approach than PAWL, and the two OTHI supported learning approaches are more effective than the others.

However, the total results may be changed if each attribute is given some weight to reflect its relative importance under a certain learning situation. The evaluation results of Table 4.6 can be gained only if every attribute has the same weight which is 1. Some courses may give a relatively high weight to the availability and low weight to other attributes so as to provide students with better learning environments for the courses. In real learning situations, the evaluation results may vary due to the consideration of additional evaluation attributes, as well as various attribute weights. For example, collaborativity may be added as one attribute to measure the effectiveness of the learning approaches for certain courses.
Although the results shown in Table 4.6 do not provide any empirical evidence regarding the effectiveness of OTHI, at least it can be said the results present the usefulness of OTHI in improving the learning effectiveness of other learning approaches.
Chapter 5

Conclusion

Chapter 5 concludes this research, summarizes its contributions, and presents possible future research directions.

5.1 Concluding Remarks

To address inquiry-based learning, we have developed a prototype of a WLSS called OTHI. We also have presented a treasure hunt model as a method of integrating the idea of IBL into the system. According to the treasure hunt model, the inquiry activity is considered the treasure hunt, and students learn a topic by developing an answer to a question regarding the topic through the treasure hunt.

The goal of this research was to design and implement a Web-based learning support system for IBL, which encourages students to actively take part in their learning by employing an online-learning game. Meanwhile, regarding the design of the system, this research was also required to find a solution to the integration of learning games into WLSS in a way that maximizes learning effectiveness.

Through this research, not only was a solution found, but also a Web-based learning support system was designed and its prototype was implemented with a solution for further research. As described in Chapter 2, the problem of Web-based learning
was reviewed, and its background and available technologies were studied with several approaches to the problem. As presented in Chapter 3, the system’s framework, the treasure hunt model, was defined, and its necessary components and functions for the system were discussed. The prototype of the system was implemented and demonstrated, and its usefulness was discussed, as addressed in Chapter 4.

5.2 Summary of Contributions

The following is the summary of contributions.


   The learning order relationship was conceptually and formally defined, including a learning order relation and term relation graph. The definition was extended to delineate the knowledge spaces of the learning order relationship by exploiting the knowledge space theory. The learning order relationship was applied to the modelling of an inquiry-based learning process.

2. Identified IBL forms a treasure hunt.

   The process of IBL was observed in a treasure hunt in which treasure is considered to be information, truth, or knowledge, and hunt implies inquiry. The process of the treasure hunt was reviewed in-depth. Phases, in the process of the treasure hunt, could, one by one, correspond to phases in the process of IBL. We found treasure hunt can represent IBL.

3. Proposed a new Web-enabled IBL model with which an implementation of a WLSS complies.

   The new Web-enabled IBL model, the treasure hunt model, was created by
formalizing the treasure hunt, which represents IBL. The components and functions of the model were devised. Furthermore, the student’s learning state and knowledge construction process of the treasure hunt model were defined with the learning order relationship. The treasure hunt model was applied to the implementation of the prototype of a WLSS for IBL called OTHI.

4. **Employed online game technology to appropriately implement the idea of IBL into a WLSS.**
   
   The idea of IBL was appropriately implemented into OTHI by employing online game technology. An online treasure hunt game was developed based on the treasure hunt model, which contains the idea of IBL. OTHI, with the treasure hunt game, is capable of providing students with an IBL environment in which students are able to experience both the treasure hunt and IBL.

5. **Developed a prototype system with the treasure hunt model.**
   
   The prototype of OTHI was developed with several algorithms based on the treasure hunt model and showed how the system works for IBL. The evaluation of OTHI was performed by investigating the comparative advantages and disadvantages of each learning approach in previous studies.

### 5.3 Limitations and Future Research

The usefulness of OTHI was presented by the contrastive evaluation of OTHI supported learning against other learning approaches. However, there is no empirical evidence for the effectiveness of the system. Hence, an empirical study on OTHI will become a main area of future research. It requests to upgrade the prototype system into a production system and apply the system to a college course, in order to access how the system, with the model, actually works for learning. Through the empirical study, the strength of this approach is better understood.
The lack of development time narrowed the scope of features to be implemented for the prototype system. However, the following components and features should be upgraded or implemented for the production system. Some web pages of the prototype system are void of information. The empty pages need to be filled with useful information or links. As for the learning support subsystem (LSS) of OTHI, the following features are being considered for further implementation: 1) providing information about courses, topics, questions, sub-topics and clues; 2) offering a pretest page where students are able to see possible marks for their answer before giving the answer in the treasure hunt game; and 3) showing students’ learning progress with their current knowledge states, as defined in Equation (3.16).

Another major improvement for the LSS will take place when replacing the role of the Google Search in the LSS with a new learning-oriented Web search engine. The indexing method of the new Web search engine is called learning order indexing and considers the learning order relationships between related learning objects in order to reflect user preference. This requires more research in developing a theoretical framework for the indexing method with the learning order relationship.

On the other hand, the main upgrade for the TSS of OTHI will allow the instructor to define and build obstacle stations into the treasure hunt game so monsters can provide players with simple surprise quizzes rather than attacking them in the game. As for the treasure hunt game, game objects will be capable of dealing with multiple topics, and appropriate articles, students post in the blog, will also be taken into account to award some experience points and golds in the game so as to further encourage students to post their ideas.

Finally, I wish to bring to light possible research areas where this preliminary study could be used to solve problems.
Appendix A

List of Symbols

\((T, \sigma)\) surmise system
\(\preceq\) learning order relation
\(A_\lambda\) series of actions
\(ACT\) set of actions
\(AGT\) set of agents
\(CL\) set of clues
\(CMT\) set of comments
\(CO\) set of conclusions
\(EVT\) set of events
\(f_{EV}\) evaluation function
\(f_{HL}\) help function
\(f_{LE}\) experience interpretation function
$f_{OA}$ learning object allocation function

$f_{OB}$ obstacle function

$g_{OT}$ orientation function

$f_{SM}$ similarity function

$f_{WI}$ writing idea function

$GI$ set of guides

$INF$ set of information

$IS$ idea structure

$IT$ set of introductions

$K, K_\lambda$ knowledge state

$K_n$ new knowledge state

$K$ knowledge structure

$K_R$ knowledge structure of learning order relationship

$K_S$ knowledge space of learning order relationship

$L$ learning topic

$LE$ set of learning experiences

$LOG$ set of logbooks

$LS$ learning state structure

$OB$ set of obstacles
q question

QS set of questions

R surmise relation

\( R(f_{SM}) \) range of \( f_{SM} \)

REL set of relationship

RS set of resources

RW set of rewards

S structure of treasure hunt model

\( s_\lambda \) station

SBJ set of the subjects

SD set of students

STA set of stations

SU set of sub-topics

T set of learning objects

\( t_0 \) course introduction

\( t_\lambda \) learning object and outer fringe of \( K_\lambda \)

\( t_1, t_2 \) learning objects or terms

\( T^I_K \) inner fringe of \( K \)

\( T^O_K \) outer fringe of \( K \)
$TN$ set of names of topics

$TP$ set of learning topics

$TRE$ set of treasures

$\lambda$ sequence of stations

$\sigma$ surmise function
Appendix B

Algorithms

/* Course Introduction Defining Algorithm */

Input: A course identification number courseid

Output: A course introduction object $\vec{t}_0$

1. Create and initialize a learning object $\vec{t}_0$.
2. $\vec{t}_0$.course_id := courseid;
3. Input a title $cname$ of the course; $\vec{t}_0$.course_title := $cname$;
4. Input the course objectives $obj$; $\vec{t}_0$.course_objectives := $obj$;
5. Input the course references $refs$; $\vec{t}_0$.course_references := $refs$;

Algorithm 1: Course Introduction Defining Algorithm

/* Knowledge Space and Outer Fringe Building Algorithm */

Data: A set $T$ of all learning objects of a course

Input: The course introduction object $\vec{t}_0$

Output: A knowledge space $K_S$ on $T$ and an outer fringe $T_K^O$ for every knowledge state $K \in K_S$

1. $K_S, K, T_N := \emptyset; \vec{t} := \vec{t}_0$;
2. Add $K$ to $K_S$; /* A knowledge space includes the empty set $\emptyset$. */
3. $K := \text{Create_new_knowledge_state}(K, \vec{t})$;
4. Add $K$ to $K_S$;
5. Get next possible learning objects $T_N$ of learning object $\vec{t}$ from $T$;
6. $T_K^O := T_N$;
7. Call Building_knowledge_space_and_outerfringe($K_S, K, T_K^O$);

Algorithm 2: Knowledge Space and Outer Fringe Building Algorithm
/ Function Create new knowledge state(K,$\vec{t}$)
/* Input: A knowledge state K and a learning object $\vec{t}$ */
/* Output: A new knowledge state $K'$ */
1 $K' := \emptyset$;
2 add all elements of $K$ to $K'$;
3 add $\vec{t}$ to $K'$;
4 return $K'$;

**Algorithm 3:** Function Create new knowledge state($K,\vec{t}$)

/* Building knowledge space and outer fringe($K_S, K, T_O K$) */
/* $K_S$ is a knowledge space, $K$ is a knowledge state. */
/* $T_O K$ is an outer fringe of $K$. */
foreach learning object $\vec{t} \in T_O K$ do
  1 $K := $ Create new knowledge state($K,\vec{t}$);
  2 if $K \notin K_S$ then add $K$ to $K_S$;
  3 if $T \notin K$ then get next possible learning objects $T_N$ of learning object $\vec{t}$ from $T$;
  4 $T_O K := T_N$;
  5 call Building knowledge space and outer fringe($K_S, K, T_O K$);
end

**Algorithm 4:** Procedure Building knowledge space and outer fringe($K_S, K, T_O K$)

/* Learning Environment Building Algorithm */
Data: A set $T$ of all objects and a set STA of all stations
Input: The course introduction object $\vec{t}_0$
Output: A knowledge space $K_S$ on $T$, an outer fringe $T_O K$ for every knowledge state $K \in K_S$, and guides and treasure boxes having information and rewards to be delivered to students
1 $K_S, K, T_N := \emptyset; \vec{t} := \vec{t}_0$;
2 add $K$ to $K_S$; /* A knowledge space includes the empty set $\emptyset$. */
3 $K := $ Create new knowledge state($K,\vec{t}$);
4 add $K$ to $K_S$;
5 get next possible learning objects $T_N$ of learning object $\vec{t}$ from $T$;
6 $T_O K := T_N$;
7 question := null;
8 call Building learning environment($K_S, K, T_O K, question$);

**Algorithm 5:** Learning Environment Building Algorithm
/* Building learning environment($K_S$, $K$, $T^O_K$, question) */
/* Refer to Equation (3.10). */
/* $K_S$ is a knowledge space, $K$ is a knowledge state, $T^O_K$ */
/* is an outer fringe of $K$, and question contains a question. */

foreach learning object $\vec{t} \in T^O_K$ do
  if $\vec{t}$ is an orientation object then
    repeat
      get a station $\vec{s} \in STA$;
      until $\vec{s}$ has a guide available to keep an information;
      get an available guide $\vec{g} \in \vec{s}.guides$;
      $question := \vec{t}.question$;
      $\vec{g}.introduction := \vec{t}.introduction$; $\vec{g}.question := question$;
    end
  else if $\vec{t}$ is a help object then
    repeat
      get a station $\vec{s} \in STA$;
      until $\vec{s}$ has a guide available to keep an information;
      get an available guide $\vec{g} \in \vec{s}.guides$;
      $\vec{g}.subtopic_description := \vec{t}.subtopic_description$;
    end
  else if $\vec{t}$ is an evaluation object then
    repeat
      get a station $\vec{s} \in STA$;
      until $\vec{s}$ has a treasure box available to keep a treasure;
      get an available treasure box $\vec{b} \in \vec{s}.treasure_boxes$;
      $\vec{b}.question := question$;
      $\vec{b}.sample_answer := \vec{t}.sample_answer$;
      $\vec{b}.conclusion := \vec{t}.conclusion$;
      get a reward $rw \in RW$; $\vec{b}.reward := rw$;
    end
  $K := \text{Create\_new\_knowledge\_state}(K, \vec{t})$;
  if $K \notin K_S$ then add $K$ to $K_S$;
  if $T \nsubseteq K$ then
    get next possible learning objects $T_N$ of learning object $\vec{t}$ from $T$;
    $T^O_K := T_N$;
    call Building\_learning\_environment($K_S$, $K$, $T^O_K$, question);
  end
end

Algorithm 6: Procedure Building\_learning\_environment($K_S$, $K$, $T^O_K$, question)
/* Learning Object Defining Algorithm */
/* Refer to Equation (3.8) and Equation (3.9). */

**Input:** A course name, course

**Output:** A set $T$ of all learning objects of a course

1. $T := \emptyset$
2. find a set $TP$ of all topics the course deals with;
3. **foreach** learning topic $L \in TP$ **do**
4. /* Definition of an orientation object of $L*/
5. create and initialize an object $\vec{t}$;
6. input a topic title $tname$ of $L$; $\vec{t}.name := tname$;
7. input an introduction $intro$ of $L$; $\vec{t}.introduction := intro$;
8. input a question $quest$ of $L$; $\vec{t}.question := quest$;
9. $\vec{t}.type := \text{“an orientaion object”}$;
10. add $\vec{t}$ to $T$;
11. /* Definition of an evaluation object of $L*/
12. create and initialize an object $\vec{t}$;
13. $\vec{t}.name := tname$;
14. input a conclusion $con$ of $L$; $\vec{t}.conclusion := con$;
15. display $quest$;
16. input a sample answer $ans$ for $quest$; $\vec{t}.sample\_answer := ans$;
17. input a clue $clue$, leading to this evaluation; $\vec{t}.clue := clue$;
18. $\vec{t}.type := \text{“an evaluation object”}$;
19. add $\vec{t}$ to $T$;
20. /* Definitions of help objects of $L*/
21. find all sub-topics of $L$;
22. $i := 0$; $m :=$ the number of all sub-topics of $L$;
23. **repeat**
24. create and initialize an object $\vec{t}$;
25. $\vec{t}.topic\_name := tname$;
26. select a sub-topic $\text{subtopic} \in L$;
27. input a sub-topic title $sname$; $\vec{t}.name := sname$;
28. input a description $desc$; $\vec{t}.subtopic\_description := desc$;
29. input a clue $clue$, leading to this sub-topic; $\vec{t}.clue := clue$;
30. $\vec{t}.type := \text{“a help object”}$;
31. add $\vec{t}$ to $T$; remove $\text{subtopic}$ from $L$; $i := i + 1$;
32. **until** $i \geq m$;
33. **end**

**Algorithm 7:** Learning Object Defining Algorithm
/* Refer to Equation (3.16), Equation (3.17), Equation (3.18) */

Data: Outer fringes of all knowledge states $K_S$ and a set $STA$ of all stations

Input: A course introduction object $\vec{t}_0$

Output: A knowledge state $K$ containing all learning objects $\in T$

1. $n := 0$; $\vec{t}, \vec{t}_0, \vec{s} := \text{null}$; $\text{points} := 0$; $RW, K := \text{the empty set}$;
2. display $\vec{t}_0$\_course\_title, $\vec{t}_0$\_course\_objectives and $\vec{t}_0$\_course\_references;
3. add $\vec{t}_0$ to $K$; $T^O_K := \text{Outerfringe}(K)$;
4. while $n < \text{number of topics of the course}$ do
5.   if $\vec{t} = \text{null}$ then select an object $\vec{t} \in T^O_K$;
6.     $\vec{s} := \text{Explore\_stations\_and\_overcome\_obstacles}(\vec{t}, \text{clue}, RW, \text{points})$;
7.     if $\vec{t}$ is an orientation object then
8.       $\vec{o} := \text{Orientation}(\vec{s}, \vec{t})$;
9.       if $\vec{o} \neq \text{null}$ then
10.          add $\vec{t}$ to $K$; $T^O_K := \text{Outerfringe}(K)$;
11.          display a list of all objects $\in T^O_K$; select an object $\vec{t}$ in the list;
12.          clue := Get\_clue\_for\_next\_station($\vec{o}$\_guide, $\vec{t}$);
13.       end
14.     end
15.     else if $\vec{t}$ is a help object then
16.        $\vec{h} := \text{Help}(\vec{s}, \vec{t})$;
17.        if $\vec{h} \neq \text{null}$ then
18.           $K := \text{Knowledge\_construction}(K, \vec{o}$\_question, $\vec{h}, \vec{t})$;
19.           $T^O_K := \text{Outerfringe}(K)$;
20.           display a list of all objects $\in T^O_K$; select an object $\vec{t}$ in the list;
21.           clue := Get\_clue\_for\_next\_station($\vec{h}$\_guide, $\vec{t}$);
22.        end
23.     end
24.     else if $\vec{t}$ is an evaluation object then
25.        if a treasure is found in a box $\vec{b} \in \vec{s}$\_treasure\_boxes then
26.           reward := Evaluation($\vec{b}$);
27.           if $\text{reward} \neq \text{null}$ then
28.              add reward to $RW$
29.              add $\vec{t}$ to $K$; $T^O_K := \text{Outerfringe}(K)$;
30.              $\vec{t}, \vec{o}, \text{clue} := \text{null}$; $n := n + 1$;
31.           end
32.        end
33.     end
34. end

Algorithm 8: Treasure Hunting Algorithm
/* Function Outerfringe(K) */

Data: Outer fringes of all knowledge states ∈ Kₜ
Input: A knowledge state K
Output: An outer fringe Tₖₒ of K

1 get an outer fringe Tₖₒ of K from data;
2 return Tₖₒ;

Algorithm 9: Function Outerfringe(K)

/* Function Explore_stations_and_overcome_obstacles(⃗t, clue, RW, points) */
/* Refer to Equation (3.15). */

Input: A learning object ⃗t, a clue clue, a set of obtained rewards RW and experience points points
Result: Accumulated points and a station ⃗s ∈ STA.

1 repeat
2 moves to a station ⃗s ∈ STA using clue;
3 if ⃗s is an obstacle station then
4   obstacle := ⃗s.obstacle;
5   get experience points exp by defeating obstacle with available tools and equipments ∈ RW;
6   points := points + exp;
7 end
8 until ⃗s is the orientation, help or evaluation station of ⃗t;
9 return ⃗s;

Algorithm 10: Function Explore_stations_and_overcome_obstacles(⃗t, clue, RW, points)

/* Function Get_clue_for_next_station(⃗g, ⃗t) */

Input: A guide ⃗g and an object ⃗t
Output: A clue clue
1 ⃗g.clue := ⃗t.clue;
2 clue := ⃗g.clue;
3 display ⃗g.clue;
4 return clue;

Algorithm 11: Function Get_clue_for_next_station(⃗g, ⃗t)
/* Function Orientation(\vec{s},\vec{t}) */
/* Refer to Equation (3.11). */

**Input:** A station \vec{s} and an object \vec{t}

**Output:** An orientation information \vec{o}

1. \vec{o} := null;
2. get an attention from a guide \vec{g} ∈ \vec{s}.guides by saying hello;
3. ask \vec{g} for a help about \vec{t};
4. if \vec{g} has the information about \vec{t} then
   5. display \vec{g}.introduction;
   6. if student wants to learn this topic then
      7. \vec{o}.introduction := \vec{g}.introduction;
      8. \vec{o}.question := \vec{g}.question;
      9. \vec{o}.guide := \vec{g};
      10. display \vec{g}.question;
      11. post the introduction and question to blog;
   else
      12. display “Bye”;
   end
5. else
6. display “No information”;
7. end
8. return \vec{o};

**Algorithm 12:** Function Orientation(\vec{s},\vec{t})
/* Function Evaluation($\vec{b}$)
 /* Refer to Equation (3.14).

**Input:** A treasure box $\vec{b}$
**Output:** A reward $rw$ for the answer

1. $rw := \text{null}$;
2. display $\vec{b}$.question;
3. input a student answer $a'$;
4. $a := \vec{b}$.sample_answer;
5. $marks := \text{Similarity}(\vec{b}.\text{question}, a, a') * 100$; /* Refer to Equation (3.13) */;
6. if $marks > 50$ then
7.   $rw := \vec{b}$.reward;
8.   display $\vec{b}$.conclusion;
9.   post the conclusion to blog;
10. end
11. return $rw$;

Algorithm 13: Function Evaluation($\vec{b}$)

/* Function Help($\vec{s}, \vec{t}$)
 /* Refer to Equation (3.12).

**Input:** A station $\vec{s}$ and an object $\vec{t}$
**Output:** A help information $\vec{h}$

1. $\vec{h} := \text{null}$;
2. get an attention from a guide $\vec{g} \in \vec{s}.\text{guides}$ by saying hello;
3. ask $\vec{g}$ for a help about $\vec{t}$;
4. if $\vec{g}$ has the information about $\vec{t}$ then
5.   $\vec{h}.\text{subtopic_description} := \vec{g}.\text{subtopic_description}; \vec{h}.\text{guide} := \vec{g}$;
6.   display $\vec{g}.\text{subtopic_description}$;
7.   post the sub-topic description to blog;
8. else
9.   display “No information”;
10. end
11. return $\vec{h}$;

Algorithm 14: Function Help($\vec{s}, \vec{t}$)
/* Function Knowledge_construction(K, question, ℎ, ℎ) */
/* Refer to Equation (3.20). */

Data: Resources RS
Input: A knowledge state K, a question question, help information ℎ, and a learning object ℎ\nOutput: A new knowledge state K

1. K := ∅;
2. add all elements of K to K;
3. initialize an idea structure is /* Refer to Equation (3.19). */;
4. is.sequence := the number of records in blog + 1;
5. is.question := question;
6. display question;
7. subtopic_desc := ℎ. subtopic_description;
8. display subtopic_desc;
9. analyze subtopic_desc with regard to question;
10. input a subject sbj using ℎ. name;
11. is.subject := sbj;
12. input a comment cmt using subtopic_desc;
13. is.comment := cmt;
14. is.relationship := “Sub Topic”;
15. while need more information about cmt do
16. retrieve information inf about cmt from RS;
17. analyze inf with regard to question;
18. update the comment cmt using inf;
19. is.comment := cmt;
20. end
21. post is to blog;
22. add ℎ to K;
23. return K;

Algorithm 15: Function Knowledge_construction(K, question, ℎ, ℎ)
1. Should also define speed to make it talk. It is used to calculate the talking speed.

XML Files

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE places SYSTEM "places.dtd">
<places>
<!-- Should also define speed to make it talk. It is used to calculate the talking speed -->
<place name="0_network_city" level="0" x="500064" y="500000" file="Level 0/network/city.tmx">
  <npc x="26" y="21" name="Alice" imgRef="woman_000_npc" speed="0.1" direction="WEST">
    <path sx="26" sy="21" dx="26" dy="25"/>
    <path sx="26" sy="25" dx="26" dy="21"/>
  </npc>
  <portal x="42" y="37" ref="tavern_entrance">
    <destination place="int_network_tavern_0" ref="entrance"/>
  </portal>
  <portal x="59" y="37" ref="house_exit_1">
    <implementation class="ca.uregina.thg.server.entities.portal.OneWayPortalDestination"/>
  </portal>
  <portal x="53" y="37" ref="temple_entrance">
    <destination place="int_network_temple" ref="entrance_center_right"/>
  </portal>
  <!-- Deleted some definitions to be published with the thesis. -->
  <place name="int_network_library" file="interiors/network/library.tmx">
    <npc x="28" y="17" name="Monogenes" imgRef="oldmannpc" speed="0.1" direction="WEST">
      <monster name="rat" x="28" y="20" direction="SOUTH"/>
```

99
<monster name="rat" x="20" y="25" direction="NORTH"/>
<food name="cheese" x="28" y="28"/>
<food name="cheese" x="18" y="20"/>
<portal x="8" y="30" ref="entrance_left">
<destination place="0_network_city" ref="library_entrance_left"/>
</portal>
<portal x="21" y="30" ref="entrance_right">
<destination place="0_network_city" ref="library_entrance_right"/>
</portal>
</place>

Deleted some definitions to be published with the thesis.

<place name="int_network_townhall" file="interiors/network/townhall.tmx">
<npc name="Boy" x="12" y="18" class="ca.uregina.thg.server.entities.npcs.network.townhall.BoyNPC"/>
<npc name="Mayor" x="13" y="2" class="ca.uregina.thg.server.entities.npcs.network.townhall.MayorNPC"/>
<treasure name="Box 9" x="24" y="38"/>
<treasure name="Box 10" x="22" y="38"/>
<treasure name="Box 11" x="20" y="38"/>
<treasure name="Box 12" x="18" y="38"/>
<portal x="14" y="46" ref="entrance_left">
<destination place="0_network_city" ref="townhall_entrance_left"/>
</portal>
<portal x="15" y="46" ref="entrance">
<destination place="0_network_city" ref="townhall_entrance"/>
</portal>
<portal x="16" y="46" ref="entrance_right">
<destination place="0_network_city" ref="townhall_entrance_right"/>
</portal>
<portal x="30" y="5" ref="storage_entrance">
<destination place="int_townhall_storage" ref="townhall_entrance"/>
</portal>
</place>

<place name="int_townhall_storage" file="interiors/network/townhall_storage.tmx">
<monster name="rat" x="6" y="3" direction="SOUTH"/>
<monster name="rat" x="13" y="5" direction="NORTH"/>
<monster name="rat" x="13" y="9" direction="EAST"/>
<monster name="rat" x="15" y="7" direction="WEST"/>
<food name="cheese" x="4" y="7"/>
Listing C.1: Place XML

```xml
<food name="cheese" x="6" y="10"/>
<food name="cheese" x="15" y="12"/>
<treasure name="Box 13" x="2" y="2"/>
<treasure name="Box 14" x="4" y="2"/>
<treasure name="Box 15" x="6" y="2"/>
<treasure name="Box 16" x="8" y="2"/>
<treasure name="Box 17" x="10" y="2"/>
<treasure name="Box 18" x="12" y="2"/>
<portal x="1" y="26" ref="townhall_entrance">
  <destination place="int_network_townhall" ref="storage_entrance"/>
</portal>
</place>
<!--
Deleted some definitions to be published with the thesis. -->
</place>
</places>
```

Listing C.2: Course Introduction XML

```xml
<introduction>
  <id>CS335</id>
  <title>Data Communications and Network</title>
  <objective>Course Objective</objective>
  <references>Course References</references>
</introduction>
```

Listing C.3: Orientation XML

```xml
<topic title="Data Link Layer">
  <introduction>Data link layer supervises the flow of information between adjacent network nodes. It uses error detection or correction techniques to ensure that a transmission contains no errors.</introduction>
  <question>Describe what the data link layer works for networking.</question>
</topic>
```
The data link layer is layer two of the seven-layer OSI model as well as of the five-layer TCP/IP reference model. It receives services from physical layer and provides services to network layer. The data link layer is responsible for carrying a packet from one node to another and has only a local responsibility. Its main functions are error control and flow control.

The task of the data link layer is to convert noisy lines into communication channels free of transmission errors for use by the network layer.

The treasure of the data link layer is in a storage of the town hall.

Listing C.4: Evaluation XML

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<sub-topics>
  <sub-topic main-title="Data Link Layer" sub-title="DLL Duties">
    <description>The duties of the data link layer include packetizing and framing, addressing error control, and flow control.</description>
    <clue>A boy is working in a town hall.</clue>
  </sub-topic>
  <sub-topic main-title="Data Link Layer" sub-title="Error detection">
    <description>The data link layer performs the error check using the Frame Check Sequence (FCS) in the trailer and discards the frame if an error is detected. The corrupted message frame may need to be retransmitted under control of an upper layer.</description>
    <clue>A woman is waiting for you in a city.</clue>
  </sub-topic>
  <sub-topic main-title="Data Link Layer" sub-title="Flow control">
    <description>The data link layer defines the data values used in the flow control signaling between two transmitting hosts. There are two types of flow control implemented in data communications such as software flow control and hardware flow control.</description>
    <clue>An old man is sitting in a library.</clue>
  </sub-topic>
</sub-topics>
```

Listing C.5: Sub-topics XML

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<stations>
  <station name="Orientation" place="int_network_townhall" place_nick_name="Town Hall" topic="Data Link Layer">
    <!-- Station details here -->
  </station>
</stations>
```
<guide>Mayor</guide><next_quests>DLL Duties</next_quests><station><station name="DLL Duties" place="int_network_townhall" place_nick_name="Town Hall" topic="Data Link Layer"><guide>Boy</guide><next_quests>Error detection</next_quests><clue>A boy is working in a town hall.</clue><station><station name="Error detection" place="0_network_city" place_nick_name="Network City" topic="Data Link Layer"><guide>Alice</guide><next_quests>Flow control</next_quests><clue>A woman is waiting for you in a city.</clue><station><station name="Flow control" place="int_network_library" place_nick_name="Network Library" topic="Data Link Layer"><guide>Monogenes</guide><next_quests>Evaluation</next_quests><clue>An old man is sitting in a library.</clue><station><station name="Evaluation" place="int_townhall_storage" place_nick_name="Town Hall Storage" topic="Data Link Layer"><treasure_box>Box 13</treasure_box><clue>The treasure of the data link layer is in a storage of the town hall.</clue></station></station></station></station></station><station><station name="Evaluation" place="int_townhall_storage" place_nick_name="Town Hall Storage" topic="Data Link Layer"><treasure_box>Box 13</treasure_box><clue>The treasure of the data link layer is in a storage of the town hall.</clue></station></station></stations>

Listing C.6: Stations XML

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<guides>
<guide name="Mayor" nick_name="DLL Introduction" place="int_network_townhall" topic="Data Link Layer">
<talk>I want to introduce you the data link layer. The data link layer supervises the flow of information between adjacent network nodes. It uses error detection or correction techniques to ensure that a transmission contains no errors.</talk>
<question>You need to describe what the data link layer works for networking in order to complete this quest.</question>
<clue/>
</guide>
<guide name="Boy" nick_name="DLL Duties" place="int_network_townhall" topic="Data Link Layer">
<talk>The duties of the data link lay include packetizing and framing, addressing error control, and flow control.</talk>
<question/>
```
The data link layer performs the error check using the Frame Check Sequence (FCS) in the trailer and discards the frame if an error is detected. The corrupted message frame may need to be retransmitted under control of an upper layer.

The data link layer defines the data values used in the flow control signaling between two transmitting hosts. There are two types of flow control implemented in data communications such as software flow control and hardware flow control.

The data link layer is layer two of the seven-layer OSI model as well as of the five-layer TCP/IP reference model. It receives services from physical layer and provides services to network layer. The data link layer is responsible for carrying a packet from one node to another and has only a local responsibility. Its main functions are error control and flow control.

The task of the data link layer is to convert noisy lines into communication channels free of transmission errors for use by the network layer.
Listing C.8: Treasures XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="http://localhost/thg" href="thg.jnlp">
  <information>
    <title>Treasure Hunt Game</title>
    <vendor>OTHI, Computer Science, University of Regina</vendor>
    <description>Treasure Hunt Game</description>
    <description kind="short">Treasure Hunt Game is a multiplayer online learning game.</description>
    <homepage href="http://localhost"/>
    <icon href="default.jpg"/>
    <icon kind="splash" href="othi_long_logo_72x72.png"/>
    <offline-allowed/>
  </information>
  <!-- <update check="always" policy="prompt-update"/> -->
  <update check="always" policy="always"/>
  <security>
    <all-permissions/>
  </security>
  <resources>
    <j2se version="1.6+"/>
    <jar href="thgclient.jar" main="true" download="eager"/>
    <jar href="sgs_client.jar" download="eager"/>
    <jar href="sgs_shared.jar" download="eager"/>
    <jar href="mina_core.jar" download="eager"/>
    <jar href="jogg.jar" download="eager"/>
    <jar href="jorbis.jar" download="eager"/>
    <jar href="slf4j_api.jar" download="eager"/>
    <jar href="slf4j_jdk14.jar" download="eager"/>
    <jar href="lwjgl.jar" download="eager"/>
    <jar href="jinput.jar" download="eager"/>
    <jar href="slick.jar" download="eager"/>
  </resources>
  <resources os="Windows">
    <j2se version="1.6+"/>
    <nativelib href="natives_win32.jar" download="eager"/>
  </resources>
  <resources os="Linux">
    <j2se version="1.6+"/>
    <nativelib href="natives_linux.jar" download="eager"/>
  </resources>
  <resources os="Mac OS X">
    <j2se version="1.6+"/>
    <nativelib href="natives_mac.jar" download="eager"/>
  </resources>
</jnlp>
```
Listing C.9: JNLP Definition for Treasure Hunt Game
Appendix D

PHP Programs Added To Joomla

Listing D.1: Default.php

```php
<?php // no direct access
defined('_JEXEC') or die('Restricted access');

define("BROWSE_MODE", "0");
define("EDIT_MODE", "1");
?

<?php
if ( $this -> blog_mode == BROWSE_MODE ) {
header("Location: ". $this -> blogURL);
}
?

<html>
<body onLoad="document.forms['othi'].submit()">
<form action="<?php echo $this -> blogURL; ?>?flavor=admin&plugins=edit-blog-entries&action=new-blog-entry" method="post" name="othi">
<input type="hidden" name="username" value="<?php echo $this -> username ?>" />
<input type="hidden" name="password" value="<?php echo $this -> password ?>" />
</form>
</body>
</html>
```

Listing D.1: Default.php
class OthiController extends JController {
    /**
     * Method to display the view
     * @access public
     */
    function display () {
        parent::display();
    }
}

<?php
    // no direct access
    defined ( ' _JEXEC ' ) or die ( ' Restricted access ' );

    jimport ( ' joomla.application.component.view' );

    class OthiViewOthi extends JView {
        function display ($tpl = null) {
            $model =& $this->getModel ();
            $blogURL = $model->getBlogURL ();
            $this->assignRef ( ' blogURL ', $blogURL );

            $params = & JComponentHelper::getParams (' com_othi ' );
            $blog_mode = $params->get (' blog_mode ' );
            $this->assignRef ( ' blog_mode ', $blog_mode );

            $user =& JFactory::getUser ();
            $username = $user->username;
        }
    }
}
Listing D.3: View.html.php

```php
<?php
// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

// Require the base controller
require_once ( JPATH_COMPONENT . DS . 'controller.php' );

// Require specific controller if requested
if( $controller = JRequest::getWord( 'controller' ) ) {
    $path = JPATH_COMPONENT . DS . 'controllers' . DS . $controller . '.php';
    if ( file_exists( $path ) ) {
        require_once $path;
    } else {
        $controller = '';
    }
}

// Create the controller
$classname = 'OthiController'. $controller ;
$controller = new $classname();

// Perform the Request task
$controller->execute( JRequest::getVar( 'task' ) );

// Redirect if set by the controller
$controller->redirect();
```

Listing D.4: Othi.php

```php
<?php
/**
 * Hello Model for Hello World Component
 *
 * @package   Joomla.Tutorials
 * @subpackage Components
 */
```
// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
import( 'joomla.application.component.model' );

/**
* Hello Model
*
* @package Joomla.Tutorials
* @subpackage Components
*/
class OthiModelOthi extends JModel {

function getBlogURL() {
    $user =& JFactory::getUser();
    $username = $user->username;
    $db =& JFactory::getDBO();
    $query = "SELECT c.property_value, d.name FROM jos_othi_users AS a, jos_othi_courses AS b, blog_properties AS c, blog_category AS d"
        " WHERE a.username = "'".$username."'" and a.blog_category_id = d.category_id and a.course_id = b.course_id and "
        " b.blog_id = c.blog_id and c.property_name = 'blog-url' ";
    $db->setQuery( $query );
    $row = $db->loadRow();
    return $row['0'].$row['1'];
}
}
Appendix E

Java Programs of Treasure Hunt Game

```java
package ca.uregina.thg.client;
import ca.uregina.thg.common.GameDefault;
import org.newdawn.slick.AppGameContainer;
import org.newdawn.slick.GameContainer;
import org.newdawn.slick.SlickException;
import org.newdawn.slick.state.StateBasedGame;

/**
 * Client main program
 * @author Dong Won Kim
 */
public class TreasureHuntGame extends StateBasedGame {
    public static String GAMEHOST = "localhost";
    public static String GAMEPORT = "13134";

    public TreasureHuntGame() {
        super("OTHI - Online Treasure Hunt for Inquiry-based learning!");
    }

    public TreasureHuntGame(String arg0) {
        super(arg0);
    }

    @Override
    public void initStatesList(GameContainer container) {
        addState(new THGkeeper());
        addState(new THGground());
    }

    public static void main(String[] args) {
        if (args.length > 0) {
            GAMEHOST = args[0];
            // Rest of the code
        }
    }
}
```
if (args.length > 1) {
    GAMEPORT = args[1];
}
}

try {
    AppGameContainer container = new AppGameContainer(new TreasureHuntGame());
    container.setDisplayMode(GameDefault.SCREEN_WIDTH, GameDefault.SCREEN_HEIGHT, false);
    container.start();
} catch (SlickException e) {
    e.printStackTrace();
}

Listing E.1: Client main program

package ca.uregina.thg.server;

import java.io.Serializable;
import java.io.IOException;
import java.util.Properties;
import java.util Enumeration;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.sun.sgs.app.AppContext;
import com.sun.sgs.app.AppListener;
import com.sun.sgs.app.ClientSession;
import com.sun.sgs.app.ClientSessionListener;
import com.sun.sgs.app.Channel;
import com.sun.sgs.app.ChannelManager;
import com.sun.sgs.app.Delivery;
import com.sun.sgs.app.ManagedReference;
import com.sun.sgs.app DataManager;
import com.sun.sgs.app.NameNotBoundException;

import ca.uregina.thg.server.config.*;

import ca.uregina.thg.common.Commands;
import ca.uregina.thg.common.GameDefault;

/**
 * Loading properties of game objects and verifying player
 * @author Dong Won Kim
 */
public class THGServer implements AppListener, Serializable {

private static final Logger logger = Logger.getLogger(
    THGServer.class.getName());

private static final long serialVersionUID = 1L;

private static final String ENTRY_PLACE = "int_network_townhall";
private static final float ENTRY_X = 15;
private static final float ENTRY_Y = 19;

/** The name of the first channel: {@value # CHANNEL_1_NAME} */
public static final String CHANNEL_1_NAME = "Foo";
/** The name of the second channel: {@value # CHANNEL_2_NAME} */
public static final String CHANNEL_2_NAME = "Bar";
/**
 * The first {@link Channel}. *(The second channel is looked up by name only.)
 */
private ManagedReference<Channel> channel1 = null;
private int maxPlayerId = Integer.MIN_VALUE;
private ManagedReference thoWorldRef = null;

@override
public void initialize(Properties props) {
    ChannelManager channelManager = AppContext.
    getChannelManager();
    // Create and keep a reference to the first channel.
    Channel c1 = channelManager.createChannel(CHANNEL_1_NAME,
        null, Delivery.RELIABLE);
    channel1 = AppContext.getDataManager().createReference(c1);

    // We don’t keep the second channel object around, to demonstrate
    // looking it up by name when needed.
    channelManager.createChannel(CHANNEL_2_NAME, null,
        Delivery.RELIABLE);

    THGWorld thoWorld = new THGWorld();
    DataManager dm = AppContext.getDataManager();
thoWorldRef = dm.createReference(thoWorld);
dm.setBinding(GameDefault.THOWORLD, thoWorld);

String appRoot = props.getProperty("com.sun.sgs.app.root ");

ConfigDefault.setConfigDir(appRoot + "/config/");

try {
    LevelDefinition.loadConfiguredLevel();
    MonsterDefinition.loadConfiguredMonsters();
    FoodDefinition.loadConfiguredFoods();
    StationDefinition.loadConfiguredStations();
    PlaceDefinition.loadConfiguredPlaces();
}

} catch (IOException e) {
    logger.log(Level.SEVERE, null, e);
}

if (maxPlayerId < 0) maxPlayerId = 0;

thoWorld.worldStart();

private void printProperties(Properties arg0) {
    String app_root = arg0.getProperty("com.sun.sgs.app.root ");
    System.out.println("com.sun.sgs.app.root : " + app_root)
    Enumeration e = arg0.propertyNames();
    while (e.hasMoreElements()) {
        Object o = e.nextElement();
        System.out.println("property : " + o.toString());
    }
}

@Override
public ClientSessionListener loggedIn(ClientSession arg0) {
    String name = arg0.getName();
    Player player;

    DataManager dm = AppContext.getDataManager();
try {
    player = (Player) dm.getBinding(GameDefault.USERPREFIX + name);
    if (player.isLoggedIn()) return null;
} catch (NameNotBoundException ex) {
    player = new Player(name);
    dm.setBinding(GameDefault.USERPREFIX + name, player);
    ++maxPlayerId;
}

player.setLoggedIn(true);
PlayerListener pl = new PlayerListener(player.getName(), arg0);

dm.markForUpdate(player);
player.setPlayerListener(pl);
setupPlayer(player);

return pl;
}

private void setupPlayer(Player player) {
    if (player.getId() == Integer.MIN_VALUE) player.setId(maxPlayerId);

    if (player.getGameBoard() == null) {
        DataManager dm = AppContext.getDataManager();
        GameBoard gameBoard = (GameBoard) dm.getBinding(GameDefault.GAMEBOARD + ENTRY_PLACE);
        player.setGameBoard(gameBoard);
    }

    if (player.getPlace() == null) {
        player.locatePlace(ENTRY_PLACE, ENTRY_X, ENTRY_Y);
    }

    sendPlayerInitialInfo(player);
    logger.info("Complete setting up player " + player.getId() + " session name : " + player getPlayerListener().getSession().getName());
}

protected void sendPlayerInitialInfo(Player player) {
    DataManager dm = AppContext.getDataManager();
    Competence pCompetence = (Competence) dm.getBinding(GameDefault.PLAYER_COMPETENCE + player.getName());
```java
ArmedArmor pArmor = (ArmedArmor) dm.getBinding(GameDefault.PLAYER_ARMOR + player.getName());

GameLevel gameLevel = pCompetence.getLevelEntity();
int maxHp = gameLevel.getHp();
int maxMp = gameLevel.getMp();
int maxExp = gameLevel.getExp();
player.getPlayerListener().sendMove(Command.initializePlayerCommand(
    player.getId(),
    player.getPlace().getId(),
    player.getPlace().getMapFileRef(),
    pCompetence.getGameLevel(), pCompetence.getHP(),
    maxHp,
    pCompetence.getMP(), maxMp,
    pCompetence.getMyExp(), maxExp,
    pCompetence.getMyMoney(),
    player.getInventory().getDialogueHistory().toArray(  
        new String[0]), pArmor.getOutfitCode())
    );
}
```

Listing E.2: Server main program

```java
package ca.uregina.thg.server.authenticator;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.security.auth.login.CredentialException;
import javax.security.auth.login.LoginException;
import com.sun.sgs.auth.Identity;
import com.sun.sgs.auth.IdentityAuthenticator;
import com.sun.sgs.auth.IdentityCredentials;
import com.sun.sgs.impl.auth.NamePasswordCredentials;

/**
 * This authenticator will connect to a specific table in the
 * MySQL database and try to retrieve a password field from a row
 */
```
in that table. Be sure to set the constants at the start
of the source code so that the Database, Table, and Column
Names match the ones in your MySQL database.

@Author Chuck Liddell
@Author Dong Won Kim

public class THGAuthenticator implements IdentityAuthenticator {

    private static final Logger logger = Logger.getLogger(
        THGAuthenticator.class.getName());

    /**
     * @param prop Reference properties (unused in this implementation)
     */
    public THGAuthenticator(Properties prop){
        logger.info("Custom Authenticator instantiated.");
    }

    @Override
    public Identity authenticateIdentity(IdentityCredentials credentials) throws LoginException
    {
        if (! (credentials instanceof NamePasswordCredentials))
            throw new CredentialException("Unsupported credentials");

        NamePasswordCredentials npc = (NamePasswordCredentials) credentials;

        // Pull the data that we need from the credentials
        String userName = npc.getName().toLowerCase(); // don't worry about case
        String password = new String(npc.getPassword());

        // Call a method that will check the MySQL database for this user
        String matchPassword = getPasswordForUser(userName);

        if (matchPassword == null){ // This user was not found in the database
            throw new CredentialException("User does not exist");
        }
        else {
            try {
                String[] part = matchPassword.split(":");
                String salt = part[1];
                MessageDigest m = MessageDigest.getInstance("MD5");
byte b[] = m.digest(new String(password + salt).getBytes());
java.math.BigInteger bi = new java.math.BigInteger(1, b);
String encryptedPasswd = bi.toString(16);
while (encryptedPasswd.length() < 32)
    encryptedPasswd = "0" + encryptedPasswd;
encryptedPasswd = encryptedPasswd + ":" + salt;
if (!matchPassword.equals(encryptedPasswd)) { // User found, but wrong password
    throw new CredentialException("Invalid password");
}
} catch (NoSuchAlgorithmException ex) {
    logger.log(Level.SEVERE, null, ex);
}
}

// IdentityImpl is an extremely simple class that implements the Identity interface.
// If we reach this point in the method code we are acknowledging that
// the user credentials we were given are acceptable.
//return new IdentityImpl(userName);
return new THGIdentity(userName, password);

/**
 * Searches a MySQL database for a record that matches the given username.
 * @param userName The username to search for in the database
 * @return The matching password for the given username, or null if no valid record was found.
 */
private String getPasswordForUser(String userName) {
    String password = null; // value we will return at the end of the method
    try {
        // Attempt to connect to the database
        Connection conn = DataBase.getConnection();

        // Build an SQL query that will locate the correct username / password record
        String query = "SELECT " + DataBase.PASSWORD_COLUMN_NAME + " FROM " + DataBase.JOS_USER_TABLE_NAME + " WHERE " + Database.JOS_USER_ID_COLUMN_NAME + " = " + userName;

        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        if (rs.next()) {
            password = rs.getString("" + Database.PASSWORD_COLUMN_NAME);
        }
        rs.close();
        stmt.close();
        conn.close();
    } catch (SQLException exc) {
        logger.log(Level.SEVERE, null, exc);
    }
    return password;
}
" WHERE " + Database.JOS_USERNAME_COLUMN_NAME + " = ?;"
PreparedStatement stmt = conn.prepareStatement( query );
stmt.setString(1, userName);
logger.info("Query: " + stmt.toString());

// The time statements let you know how long your MySQL queries are taking
long startTime = System.nanoTime();

// This line applies your SQL query to the database
// The database returns the results as a ResultSet
ResultSet results = stmt.executeQuery();

long estimatedTime = System.nanoTime() - startTime;
logger.info("Execution time : " + estimatedTime / Math.pow(10, 6) + " ms");

// Check to see if the ResultSet has any entries
// If it does, take the first one and grab the password from the 'Password' database column
if( results.next() ) { // results.next() returns true if there is at least one result
    password = results.getString( Database.PASSWORD_COLUMN_NAME );
}

stmt.close();
} catch (com.mysql.jdbc.exceptions.jdbc4CommunicationsException e) {
    logger.log(Level.SEVERE, "MY SQL NOT STARTED!", e);
} catch (SQLException e) {
    logger.log(Level.SEVERE, "SQL ERROR!", e);
}

// Return the password, or null if no records were returned by the database
return password;

@Override
public String[] getSupportedCredentialTypes() {
    return new String[] {"NameAndPasswordCredentials"};
}
import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Logger;
import java.util.logging.Level;

/**
 * Database definition
 * @author Dong Won Kim
 */

public class DataBase {
    private static final Logger logger = Logger.getLogger(DataBase.class.getName());

    // These constants are used to connect to the MySQL database
    private static final String SQL_HOST_NAME = "localhost";
    private static final String SQL_DATABASE_NAME = "wss";
    private static final String SQL_USERNAME = "wss";
    private static final String SQL_PASSWORD = "Wlss003";
    public static final String JOS_USER_TABLE_NAME = "jos_users";
    public static final String JOS_USERNAME_COLUMN_NAME = "username";
    public static final String PASSWORD_COLUMN_NAME = "password";

    // OTHI Tables
    public static final String OTHI_USER_TABLE_NAME = "jos_othi_users";
    public static final String COURSE_TABLE_NAME = "jos_othi_courses";
    public static final String OTHI_USER_COLUMN_NAME = "username";
    public static final String COURSE_ID_COLUMN_NAME = "course_id";
    public static final String COURSE_NAME_COLUMN_NAME = "course_name";
    public static final String BLOGCATEGORY_ID_COLUMN_NAME = "blog_category_id";
    public static final String BLOG_ID_COLUMN_NAME = "blog_id";

    // Blog Tables;
    public static final String BLOGCATEGORYMETADATA_TABLE_NAME = "blog_CategoryMetadata";
    public static final String CATEGORYMETADATA_ID_COLUMN_NAME = "category_metadata_id";
    public static final String CATEGORY_ID_COLUMN_NAME = "category_id";
public static final String METADATA_KEY_COLUMN_NAME = "metadata_key";
public static final String METADATA_VALUE_COLUMN_NAME = "metadata_value";

public static Connection getConnection () {
    Connection conn = null;
    try {
        DriverManager.registerDriver((Driver) Class.forName("com.mysql.jdbc.Driver").newInstance());
        // Build the URL we will use to connect to the database
        String connURL = "jdbc:mysql://" + SQL_HOST_NAME + "/" + SQL_DATABASE_NAME + "?user=" + SQL_USERNAME + "&password=" + SQL_PASSWORD;
        // Attempt to connect to the database
        conn = DriverManager.getConnection(connURL);
    } catch (com.mysql.jdbc.exceptions.jdbc4CommunicationsException e) {
        logger.log(Level.SEVERE, "MySQL NOT STARTED!", e);
    } catch (SQLException e) {
        logger.log(Level.SEVERE, "SQL ERROR!", e);
    } catch (ClassNotFoundException e) {
        logger.log(Level.SEVERE, "JDBC Driver Not Found", e);
    } catch (InstantiationException e) {
        logger.log(Level.SEVERE, "JDBC Driver Instantiation Error ", e);
    } catch (IllegalAccessException e) {
        logger.log(Level.SEVERE, null, e);
    }
    return conn;
}

Listing E.4: Database definition server program
Bibliography


