DISTRIBUTED ADAPTIVE E-ASSESSMENT IN A HIGHER EDUCATION ENVIRONMENT

A Thesis submitted for the degree of Doctor of Philosophy

By

Xiaobin Lin

Faculty of Enterprise & Innovation,

Buckinghamshire New University

Brunel University

July, 2011

Abstract

The rapid growth of Information Communication Technology (ICT) has promoted the development of paperless assessment. Most of the e-Assessment systems available nowadays, whether as an independent system or as a built-in module of a Virtual Learning Environment (VLE), are fixed-form e-Assessment systems based on the Classical Test Theory (CTT). In the meantime, the development of psychometrics has also proven the potential for e-Assessment systems to benefit from adaptive assessment theories.

This research focuses on the applicability of adaptive e-Assessment in daily teaching and attempts to create an extensible web-based framework to accommodate different adaptive assessment strategies for future research. Real-data simulation and Monte Carlo simulation were adopted in the study to examine the performance of adaptive e-Assessment in a real environment and an ideal environment respectively. The proposed framework employs a management service as the core module which manages the connection from distributed test services to coordinate the assessment.

The results of this study indicate that adaptive e-Assessment can reduce test length compared to fixed-form e-Assessment, while maintaining the consistency of the psychometric properties of the test. However, for a precise ability measurement, even a simple adaptive assessment model would demand a sizable question bank with ideally over 200 questions on a single latent trait. The requirements of the two categories of stakeholders (pedagogical researchers and educational application developers), as well as the variety and complexity of adaptive models, call for a framework with good accessibility for users, considerable extensibility and flexibility for implementing different assessment models, and the ability to deliver excessive computational power in extreme cases. The designed framework employs a distributed architecture with cross-language support based on the Apache Thrift framework to allow flexible collaboration of users with different programming language skills. The framework also allows different functional components to be deployed distributedly and to collaborate over a network.

Acknowledgements

There are several people I would like to thank for supporting me throughout my research and during my thesis-writing process. Firstly, I am most grateful for the intellectual and spiritual support I received from my three supervisors, Dr. Richard Mather, Dr. Roger Dalrymple, and Dr. Huixin Chen. With their guidance I not only gained an understanding of statistics and UML design; more importantly, I also learned how to think philosophically behind the research. They also encouraged me to overcome all the unforeseen difficulties during my studies. My external examiner, Dr. Trevor Barker, provided a lot of detailed and valuable comments on this dissertation too. I will always bear in mind their continuous help and the friendships that emerged.

Throughout my research I received a lot of help from Laura Bray, John Boylan, Yiming Qiao, and Weihang Jiang. I wish to thank them for their understanding, friendship, and selfless support. I dedicate my dissertation to my parents, to whom I owe everything. They have given me all the love and support that I could ever hope for.

Author's Declaration

This is to certify that the work presented in the thesis is my own work and has not been submitted for any other degree.

Please notice the fact that some of the preliminary research results have been published. Some contents in chapter 4 ("System Analysis") and chapter 5 ("System Design and Implementation") have been published as:

> Lin, X., Qiao, Y., Weatherburn, G. and Chen, H., 2009b. AMTES: An Adaptive Tele-education System for Pressure Ulcer Education. International Conference on Education Technology and Computer 2009, Singapore 17th – 20th April 2009. IEEE: USA.
>
> Qiao Y., Lin X., Weatherburn G., Chen H., Guo L., Fletcher H., 2009. Adaptive Tele-education for Pressure Ulcer Prevention using Rich Internet Application, British Computing Society Health Computing 2009, Harrogate, UK 28th – 30th April 2009.

Some contents in chapter 6 ("Experiments and Data Analysis") have been published as:

> Lin, X. B., Fletcher, H., Mather, R., 2007. Key Issues Surrounding the Implementation of Computerised Adaptive Testing Tools. *The HEA 2006 eLearning and Teaching Workshop, 6th June, 2007*.
>
> Lin, X. B., 2007. Efficient Diagnosis of Examinee's Proficiency Using Adaptive Testing, *Conference proceedings of the BCUC 2007 e-Learning Conference*, 7th June 2007.
>
> Lin, X., Chen, H., Mather, R. & Fletcher, H., 2009a. Adaptive Assessment – A Practice of Classification on Small-Size Training Sets. In C. Crawford et al. (Eds.), *Proceedings of Society for Information Technology and Teacher Education International Conference 2009* (pp. 3168-3181). Chesapeake, VA: AACE

Contents

Table of Figures

List of Tables

A Glossary of Terms and Abbreviations

AfL – Assessment for Learning

BLOBs – Binary Large Objects

CAT – Computerised Adaptive Testing

CORBA – Common Object Request Broker Architecture

CTT – Classical Test Theory

IRT – Item Response Theory

IIF – Item Information Function

IRF – Item Response Function

ITS – Intelligent Tutoring System

JDBC – Java Database Connectivity

JVM – Java Virtual Machine

MDT – Measurement Decision Theory

MIRT – Multidimensional Item Response Theory

MRCMLM – Multidimensional Random Coefficients Multinomial Logit Model

MVC – Model View Controller

PDF – Probability Density Function

RMSE – Root Mean Square Error

SEM – Standard Error of Measurement

SOAP – Simple Object Access Protocol

RMI – Remote Method Invocation

RPC – Remote Procedure Call

UDDI – Universal Description Discovery and Integration

UIRT – Unidimensional Item Response Theory

URI – Uniform Resource Identifier

VLE – Virtual Learning Environment

W3C – World Wide Web Consortium

WSDL – Web Service Description Language

WWW – World Wide Web

XML – Extensible Markup Language

CHAPTER 1 INTRODUCTION

1.1 BACKGROUND

Assessment has been central to pedagogic practice since the birth of education. Among the earliest documented examples of assessment were practices in ancient China in about 2200 BCE where the purpose of those assessments was to test civil servants to ascertain their eligibility for the appointed duties (Gregory, 2007)[1]. This method was soon extensively adopted as a criterion to measure whether students were eligible to progress to the next educational stage, as well as a criterion for measuring instructors' performance in delivering knowledge. After long centuries of evolution, a multitude of assessment approaches and methodologies had evolved to serve various designated purposes such as filtering contestants, diagnosing learners' problems, helping learning and encouraging deep learning.

Recent advancements in Information and Communication Technology (ICT) and pedagogical psychology have had a profound impact on assessment. The World Wide Web is a growing international phenomenon with a particularly strong presence in North America, Western Europe and East Asia, and, as a result, ICT is now used in higher education as a mainstream method of learning communication (Bach *et al.*, 2008). Many assessments have been migrated to electronic platforms with the popularity of computers. In the late 1990s, the term "e-Assessment" began to be widely adopted by researchers and educationalists. E-Assessment can be defined (Ofqual, 2010) as,

[1] The history of assessment is obviously a very broad topic: recent treatments include Banta & Associates, 2002, Black and William, 1998a, Linn and Gronlund, 2000.

> the use of electronic systems for the development, operation and delivery of accredited qualification assessment or the collection of performance evidence, which contributes to the awarding of a unit or an accredited qualification.

To date, there have already been several ramifications of e-Assessment. Some of them are still at an early development stage, while others have been widely used as standard evaluation and learning tools (see Figure 1-1).

Over the past two decades, certain organisations in the USA such as Cisco and Microsoft started to adopt e-Assessment to deliver examinations in their qualification courses, and this was probably the earliest application form of e-Assessment. However, the spread of e-Assessment in public education in the UK has been advancing slightly slower. Until recently, the use of e-Assessment in the UK remained as more of the exception than the norm (Mansell, 2009).

The Driving Standards Agency offers e-Assessments to a large number of learner car drivers and motorcyclists every year while the Home Office requires applicants for British citizenship to take the e-Assessment "Life in the UK tests". Additionally, many professional qualification-awarding bodies in the UK have designed and developed e-Assessments for their applicants. As far as the public education system in the UK is concerned, the use of e-Assessment is gradually becoming more and more popular, especially in post-compulsory education (Winkley, 2010). Many HE institutions offer e-Assessment to evaluate students' mastery of course units (see Figure 1-1). For example, the University of Sunderland employs WebCT as a web-based e-Assessment tool while Buckinghamshire New University uses Blackboard™ and Moodle for their e-Assessment activities.

	No.	Total	Pre-92	Post-92	Coll	Eng	Wal	Sco
Moodle	41	55%	56%	50%	67%	56%	57%	57%
Blackboard	37	50%	41%	62%	56%	51%	71%	29%
WebCT	23	31%	33%	35%	11%	29%	0%	71%
Other VLE developed in house	17	23%	26%	23%	11%	19%	14%	71%
Other intranet-based products developed in house	9	12%	13%	12%	11%	10%	29%	14%
FirstClass	7	10%	13%	4%	11%	9%	0%	29%
Commercial intranet-based product	4	5%	3%	8%	11%	3%	14%	14%
Sakai	4	5%	8%	4%	0%	3%	0%	29%
Other open source	4	5%	8%	4%	0%	5%	14%	0%
Other commercial VLE	3	4%	3%	8%	0%	2%	0%	29%
Other open source VLE	3	4%	8%	0%	0%	5%	0%	0%
No VLE	3	4%	8%	0%	0%	3%	14%	0%
Desire2Learn	2	3%	3%	4%	0%	2%	14%	0%
Bodington	2	3%	5%	0%	0%	3%	0%	0%
Merlin	1	1%	3%	0%	0%	0%	0%	14%
COSE	1	1%	0%	4%	0%	2%	0%	0%

Figure 1 - 1: statistics about VLEs currently used across UK institutions (Browne *et al.*, 2008)

One of the earliest attempts at a systematic application of e-Assessment was the "Automatic Grader" project (Hollingsworth, 1960), which automated the process of assessing students' programming assignments. Subsequently, Forsythe and Wirth (1965) created an e-Assessment system to automatically assess programming exercises and this was adopted by the University of Stanford as the assessment tool for their numerical analysis course. The pioneers of this form incorporated some advantages of new technology, such as flexibility of assessment delivery in terms of time and location, and the diversity of content presentation. The introduction of interactive multimedia greatly enriched the user experience while taking assessment and web-based technologies lifted the time and location limitations imposed on participants (Brooks, 2000). In spite of these features

brought by the new technology wrappers, the kernel theoretical foundation for many e-Assessment systems at that period of time still remained unchanged compared to their pencil-and-paper-based predecessors, which resulted in the recycling of some of the drawbacks of traditional practice when presenting the new features. In the meantime, the achievements in pedagogical psychology have expanded understanding of the role and implementation of assessment. As a new role of assessment, many educationists believe the contemporary assessment systems should act as a reference and provide guidance in the planning of teaching and learning rather than merely being an evaluation tool (Wiggins, 1998; O'Connor, 2002; Cooper, 2006; Manitoba Education, Citizenship and Youth, 2006). Additionally, the development of modern psychometrics and computing has opened up new possibilities to administer more intelligent e-Assessment.

As the centrepiece of many educational improvement efforts, the focus of interest for assessment practice has recently shifted from evaluation tool to learning assistant. Many educationists state that assessment can and should play a vital part in the learning process and that it gave birth to the concept of Assessment for Learning (Snyder, 2006). Assessment for Learning (AfL) is defined as the process of seeking and interpreting evidence for use by learners and their teachers to identify the knowledge gap of students. Black and William's (1998) research indicates that the learning gains of students were significantly improved when classroom assessments were used for learning purposes.

AfL has been given high priority by successive UK governments as being at the heart of attaining the aim of creating a world-class education system for all. It improves teaching and learning efficiency in the following aspects. First of all, AfL emphasises a learning orientation rather than a performance orientation (Dweck, 2000). This kind of assessment can effectively guide the setting of teaching and learning goals. The information obtained from assessments is a valuable reference for both teachers and learners to determine and objectively appraise their goals. Secondly, AfL can be used to monitor the progress of teaching and learning. Periodic assessments are ideal reflections of current progress in teaching and learning. Last but not least, AfL can help teachers understand

learners' weaknesses and hence provide personalised tutoring accordingly. Diagnosing learners' weaknesses is one of the primary objectives in AfL, and it is also a crucial step leading towards more effective learning (Stacey *et al.*, 2009). Therefore, AfL has been widely adopted as the main purpose of Intelligent Tutoring Systems (ITS).

Innovative E-Assessments

A large proportion of current e-Assessments are based on fixed-order multiple-choice questions due to the easy realisation of computer marking. This practice has limited the diversification of e-Assessments and hence limited the potential of e-Assessments. As technology continues to advance, a rich community of e-Assessment researchers and developers are attempting to go beyond this simple form of e-Assessment. These efforts mainly focus on improving the following three aspects of e-Assessment: examinees' assessment experience, computer marking for different question types, approaches to administering assessments and evaluating examinee performance.

The incorporation of video, audio and interactive media into e-Assessments is the main approach to enriching examinees' assessment experience. This incorporation has been proven to be able to improve motivation and engagement from examinees. The car and motorcycle theory tests ask examinees to watch a video clip of everyday traffic scenes to identify potential traffic hazards. Many online foreign language proficiency assessments use audio as a means to assess examinees' listening and spelling skills. Bioscope is an interactive e-Assessment project run by Professor Don Mackenzie and a Cambridge-based assessment team, which attempts to develop a full-functioning virtual microscope to support A and AS level examinations (Cambridge University Press, 2010). Simulations and serious gaming[2] are also playing a more and more important role in education and assessment.

[2] Serious gaming here refers to games designed for educational purposes.

Figure 1 - 2: a screenshot of the bioscope project

As well as the efforts to enrich test-takers' experience, there is also a community of researchers working on the improvement of computer marking. The many challenges of computer marking lie in the automatic marking for customised texts such as sentences, paragraphs and essays. The possibility of immediate feedback and releasing examiners from a heavy marking load has made this area highly attractive despite its considerable technological difficulty. An incomplete solution could lead examinees towards composing a fixed structure of articles to cater to the system's taste to achieve high scores. Furthermore, linguistic analysis requires heavy computational power which makes it unaffordable for standalone computers. Currently, researchers are attempting to overcome this difficulty through the collaboration of distributed computers over networks.

There are already a number of software artifacts which can mark text-based responses. Intelligent Assessment Technologies (Jordon and Mitchell, 2009) and Openmark (Ekins, 2008) are two e-Assessment systems which can automate marking for short-answer questions. The latter is an open source project and has been adopted by Open University. ETS' e-rater (Attali and Burstein, 2006) is perhaps the most famous application for essay marking. E-rater intelligently compares the contents of the essay to be marked with a group of pre-graded scripts to identify the best match and then mark according to the correlation.

Adaptive e-Assessments focus on the intelligent selection of questions and estimation of examinees' abilities. In general, adaptive e-Assessment attempts to collect more information about the current examinee by using the examinee's responses throughout the assessment as sources of continuous feedback. As the examinee progresses through the e-Assessment, the system tends to comprehend more about the examinee's ability and ask more precise questions. Adaptive e-Assessments have the following three typical advantages:

- Within the same constrained time, adaptive e-Assessments can achieve a much more accurate measurement of an examinee's ability than conventional fixed-form e-Assessments. In other words, compared to fixed-form e-Assessments they can yield the same measurement accuracy within a significantly shorter period of time. The reason is that adaptive e-Assessments always use questions close to the examinee's ability level and omit those inappropriate questions. This reduces examinee fatigue, but, more importantly, it also improves examinees' motivation by providing befittingly challenging questions. As motivation is deemed by Salmon (2002, 2004) to be an important element of online learning in her five-stage model, adaptive e-Assessment could be integrated in online learning to foster students' interest in study.
- Adaptive e-Assessments also clearly differentiate examinees with similar ability among a large group of examinees with a wide ability range. In this case, conventional fixed-form e-Assessments inevitably tend to pitch questions around the average level of the target examinees, which weakens the differentiation at the two extreme ends of the ability scale. Therefore, adaptive e-Assessments can do well in personalised learning planning.
- Adaptive e-Assessments are sample- and test-independent, which means the results for different examinees measured with different questions are completely comparable. This makes adaptive e-Assessments significantly more flexible than conventional fixed-form e-Assessments.

Whether an assessment is designed for an evaluation purpose or a learning purpose, the accuracy of the result is always the basic requirement of a good assessment. There are other characteristics that are expected of a good assessment too, such as psychometrical invariance and efficient measurement, and these are challenges to assessment systems based on the traditional theory – Classical Test Theory (CTT) (Weiss and Yoes, 1991). Adaptive assessment systems based on modern psychometric theories like Item Response Theory (IRT) (Lord, 1991) and Measurement Decision Theory (MDT) (Rudner, 2001) have the potential to overcome the drawbacks of CTT; however, these theories failed to become popular among educationists due to the complexity of implementation.

Although the modern e-Assessment theories have existed for some time, they still have a long way to go before coming into public view and becoming a practical tool. Many obstacles are challenging the possibility for these theories to become popular. These modern test theories are usually strong assumption theories, which mean there are a number of prerequisites to meet before application. In the meantime, they are heavily dependent on the computational power of modern computers. In addition, little research has been conducted to investigate the performance of these theories with pragmatic considerations for teachers in practice.

A number of commercial and non-commercial e-Assessment tools have been developed to deliver paperless tests. These tools can be roughly divided into four types according to the purpose they serve. The first type of e-Assessment tools is the non-adaptive authoring tool, which entitles users to create and deploy electronic tests. This type of tool usually supports a variety of question types but only provides limited test delivery flexibility. For example, IntraLearn (2007) is capable of generating randomised questions, while Blackboard™ (2007) allows test designers to define different questions to be presented according to an examinee's performance in previous tests. Some systems are more sophisticated. For example, Questionmark Perception (2007) allows users to define certain questions to be presented based on how examinees answer specific questions or score

on particular topics. In general, these systems are usually not supported by well-established theoretical foundations for implementing the adaptability.

The second type of e-Assessment tool is the built-in e-Assessment modules of assessment-led ITS (Intelligent Tutoring System) systems such as e-Grammar (Yang *et al.*, 2006) and Assistment (Feng *et al.*, 2008). This type of e-Assessment tool is usually integrated tightly with the hosting e-Learning system and works as a supporting component for achieving a certain learning goal. This class of e-Assessment tool is usually preset to meet the teaching and learning needs of a certain subject and works closely with the supportive environment.

A third type of e-Assessment tool is the professional tools exclusively created for researchers in the relevant fields. This type of e-Assessment tool includes a number of commercial and non-commercial packages such as BILOG[3] (Zimowski *et al.*, nd.), MULTILOG (Thissen *et al.*, n.d.), ConQuest (Wu *et al.*, 1998), POSTSIM[4] (Assessment Systems Corporation, 2010) and FastTEST (Assessment Systems Corporation, 2010). They are primarily designed to complete certain professional tasks in a psychometric theoretical framework. For example, BILOG estimates item parameters for the one-, two-, or three-parameter IRT models. FastTEST is a tool to create and deliver conventional tests or adaptive tests based on the one-, two-, or three-parameter IRT models on independent or networked workstations. These tools are mainly subassemblies to be used together to accomplish a complete e-Assessment task in a certain theoretical framework. For instance, BILOG, POSTSIM and FastTEST are designed for the three basic processes of creating an IRT-based adaptive assessment, which are item calibration, post-hoc simulation and test delivery, respectively.

[3] BILOG works only under the Microsoft DOS environment. The new version of BILOG that works under the Microsoft Windows environment was renamed BILOG-MG.

[4] POSTSIM was renamed CATSim in 2010 (Assessment Systems Corporation, 2010).

The last type of e-Assessment tool is web-based adaptive e-Assessment systems. SIETTE (Conejo *et al.*, 2004) is one of the typical representatives. SIETTE is a web-based adaptive testing tool based on Unidimensional Item Response Theory (UIRT) which allows users to create and deliver web-based adaptive e-Assessments. The limitations of this type of e-Assessment tool include the irreplaceable underlying adaptive strategies and the exclusion of item calibration and model goodness-of-fit testing functionalities.

In conclusion, to the researcher's knowledge there is currently no such thing as an open web-based framework that can offer e-Assessment designers both freedom and support in assessment strategy development. What an e-Assessment designer can currently do with the existing e-Assessment systems is pretty much limited to selecting from a number of preset schemes. The advantages of a free and supportive e-Assessment framework are multi-faceted. First, such a framework should help researchers quickly develop and evaluate their own adaptive e-Assessment strategies by providing the necessary infrastructure. Secondly, this framework should also support communication among distributedly deployed system modules in preparation for the possible heavy computational load incurred by a complex algorithm. Thirdly, it should support scalability so as to allow the systems built upon it to grow from small to large. Additionally, little effort has been invested to study the performance of adaptive e-Assessment methods under a pragmatically attainable environment.

1.2 RESEARCH QUESTIONS

This research can be divided into two parts in response to two concrete questions:

1. What is the performance of a three-parameter-logistic-item-response-model-driven e-Assessment system in a pragmatic environment, e.g. a regular class with about 30 students[5]?

[5] A regular class is defined by Blatchford (2009) as a class with around 23 students

2. Can a distributed architecture-based framework support the good characteristics of adaptive e-Assessment applications, especially in terms of scalability and affordability?

The first question is the primary research question of this study, while the second question is derived from the process of answering the primary research question. It has been found that adaptive e-Assessment systems can be highly demanding in terms of computational power (Eggen, 2004). The high demand for computational power could be caused by the complexity of adaptive assessment models, the need for real-time ability estimation and question selection, or the delivery of adaptive assessment to a large group of candidates simultaneously. In order to balance the cost and computational power, it is assumed that a distributed framework with the potential to share the tasks among multiple computers within a network would be a feasible solution. In this situation, the second research question was brought up to form the second part of this research.

The primary research question is an attempt to explore the feasibility and performance of IRT-based adaptive assessments in a normal classroom environment in higher education. Out of all of the studies and applications on adaptive assessment, Item Response Theory is an increasingly popular approach to adaptive e-Assessment because it provides a theoretical justification for assessment implementation and analysis (Reise *et al.*, 2005). However, the application of this type of adaptive assessment system is limited to large-scale organisations and does not occur in a normal classroom environment (Frick, 1992; Olson, 1990; ETS, 2001; Taiwan Education Testing Centre, 2007). The main cause of this situation is the complexity and rigorous requirements in the implementation of IRT-based adaptive assessment. Examinee issues, item pools, test administration and test security are the four general areas of practical issues regarding IRT-based adaptive assessment (Wise and Kingsbury, 2000). These practical issues are actually surrounding the areas of item pools or question banks, which includes the main aspects to yield good ability measurements such as pool size, dimensionality of an item pool, item response model and item calibration. To ensure an item is rigorously calibrated, i.e. the parameters of the item are accurately

estimated, there may need to be 200 to 1,000 examinees involved in the pilot test and this would restrict small organisations from using IRT-based adaptive assessment (Frick, 1992).

The essential meaning of the first research question is to explore the feasibility and performance of introducing IRT-based adaptive assessment in small-scale environments in higher education. Therefore it would be ideal to approach it by using a real-life case. The two pivotal aspects in seeking a resolution to this question are: establishment of a common pragmatic environment and setting up a comparison reference for evaluation purposes. In the following research, a middle-sized class registered on a Cisco standard course will be selected as the research subject, and the performance of the proposed adaptive e-Assessment strategy will be evaluated against fixed-form e-Assessments with an identical question bank.

The performance here in the research is interpreted as efficiency. To be more specific, it refers to the index that synthetically measures the accuracy of the test and the testing time. The test accuracy was measured in the research via an objective function[6] formed based on the three-parameter IRT response function. Based on the assumption of the correctness of IRT, an examinee's opportunity to give a certain category of response can be worked out based on the examinee's ability and the parameters of the given question. Substituting the ability obtained from one e-Assessment into the objective function generated a result that highlighted the inconsistency between the measurement and the theoretic model, which was defined in the research as the error generated in the process of ability measurement. The more errors generated in the measurement, the less accurate the measurement was and vice versa. Unlike the Cisco fixed-form e-Assessment, the adaptive e-Assessment implemented in the research used non-fixed length tests which terminated the test when the system believed the measurement was reliable enough. In this research, the average number of questions used in a test was deemed to be the indicator of the testing time used.

[6] The objective function (equation 4.18) can be found in chapter 4 on page 70.

For the second research question, it is helpful to review the previous work on e-Learning and e-Assessment frameworks and analyse the requirements in applying contemporary e-Assessment theories to practical cases. Khaskheli (2004) suggested that a good e-Learning or e-Assessment system should feature the following seven characteristics:

- Interoperability
- Reusability
- Manageability
- Accessibility
- Durability
- Scalability
- Affordability

As defined by Bull, J. & McKenna, C. (2001), interoperability is the ability of different systems to share and exchange information via universal formats. Reusability, as a standard in object-oriented programming, refers to the ability of a component to be reused in other projects. Manageability is the ability of a system to keep track of the learning experience and activities of individual learners. Accessibility refers to the ability of the system to be accessed from anywhere at any time. Durability is similar to downward compatibility, which means the old learning materials can still be used in the updated version of the system. Scalability means the system's ability to expand from small to large while affordability here indicates whether the resource required to implement this system can be reasonably procured.

So far, many efforts have been invested to achieve interoperability and reusability of e-Learning and e-Assessment systems. These efforts are mainly embodied in the different educational standards published. Organisations and consortia such as Dublin Core (DC) (2010), The Instructional Management System Global Learning Consortium (IMSGLC) (2010), The Aviation Industry CBT (Computer Based Training) Committee (AICC) (2010), and The Alliance of Remote

Instructional Authoring and Distribution Networks for Europe (ARIADNE) (2010) have defined a number of standards for learning elements including learning objects, metadata and learner information.

However, little research has been done to enhance the affordability of e-Learning and e-Assessment systems. The affordability issue does not appear to be a big problem with traditional e-Assessment systems due to their relatively low demand for computational power. Nevertheless, complex model-based adaptive assessments have intensified the dilemma between affordability and adaptability.

The key to the solution of this dilemma can be retrieved by studying the common architecture and central components of a typical adaptive e-Assessment system. The factors extracted from the above analysis would provide good evidence for the requirements of an adaptive e-Assessment system, and these requirements could eventually evolve into the core components of the solution framework.

1.3 AIM AND OBJECTIVES

The broad aim of this research is to firstly investigate the performance of adaptive e-Assessment in a normal class environment in higher education, and secondly create a scalable and affordable framework for the development of adaptive e-Assessment. When it came to the achievement of this aim, the three-parameter-logistic-item-response-model-driven e-Assessment was selected as the representation of adaptive e-Assessments due to the representativeness of this item response adaptive model amongst various models. Distributed architecture was considered as a potential scheme for constructing an adaptive assessment framework to offer both scalability and affordability. Thus, the achievement of this research aim was converted into answering the two aforementioned research questions.

To explain in more detail how to achieve this research aim, the following eight sequential objectives were listed as the major milestones:

1. Investigate assessment theories with a focus on adaptive assessment theories.
2. Conduct a literature review on frameworks for web applications.
3. Investigate the pragmatic problems in the implementation of adaptive e-Assessment.
4. Design and implement an adaptive e-Assessment algorithm for a normal classroom environment.
5. Evaluate the performance of the algorithm implemented in objective 3 via real data simulation. The test length and the precision of ability estimation are used as the two indicators of the performance of the adaptive e-Assessment algorithm. Fixed-form e-Assessments were used as a comparison in this objective.
6. Investigate the impact of the size of the question bank on the efficiency of assessment via Monte Carlo simulation.
7. Propose and implement an e-Assessment framework which supports the 7 characteristics of good e-Assessment systems (Khaskheli, 2004) and provides a solution to the pragmatic problems identified in objective 2.
8. Evaluate whether the implemented framework achieves the aims outlined in objective 5.

1.4 SCOPE OF RESEARCH

The IRT-based adaptive e-Assessment approach provides a tailored assessment experience and, in the meantime, improves the accuracy of assessment. It hence potentially brings about many new features and can exhibit significant advantages in some applications. However, it is usually difficult to implement and maintain such an adaptive e-Assessment system due to the strong assumption of IRT. The aim of this research was firstly to benchmark the performance of adaptive e-Assessment

in a normal classroom environment in higher education[7], and then propose and implement a framework to support the development of adaptive e-Assessment in connection with the pragmatic issues that occur during its application.

Cisco is the authority in training and assessing computer networking knowledge and skills. In order to improve the credibility of the research, two fixed-form online chapter tests from Cisco's CCNA exploration course were selected as the comparative object. Fixed-form assessment is the main alternative form to adaptive e-Assessment and it differs from adaptive e-Assessment mainly in two aspects: item calibration and administration of assessment items. The purpose of comparison was to use the measurement of fixed-form e-Assessment as a baseline and investigate the performance difference between fixed-form e-Assessment and adaptive e-Assessment in a normal classroom environment. Each of these two Cisco chapter tests contains 24 multiple-choice questions and 60 students were recruited to attend the tests in the research. The responses from the 60 students as well as the questions from the two chapter tests were used as the source data in the implemented adaptive e-Assessment to obtain the experimental results[8]. The experiments were conducted in MATLAB, which provides a programming environment for computationally intensive work (MATLAB, 2011).

Web-based e-Assessment systems have advantages in flexibility, accessibility, data collection and analysis compared to standalone e-Assessment systems (Doe, 2004). After the empirical data indicated the positiveness of the performance of adaptive e-Assessment, a web-based adaptive e-Assessment application was implemented using Java Server Pages (JSP) to deliver the adaptive form of Cisco chapter tests to candidates. During the development of the web-based adaptive e-Assessment application, two important issues related to the implementation of adaptive e-

[7] This research was carried out in the context of higher education, but it can be potentially applied to a broader range of environments.

[8] Please refer to chapter 4 for a detailed description of the implementation of the experiments.

Assessment applications were identified: the various ramifications of adaptive e-Assessment approaches and the potential demand for computational power with the increase of model complexity, the number of users and the size of the question bank. An extensive literature review on the implementation of adaptive e-Assessment systems was conducted to verify the identified issues. Later a user requirement analysis was performed which collected a number of e-Assessment developers' and researchers' expectations on a framework for rapid implementation of adaptive e-Assessment systems. The analysis based on the information collected led to the idea of designing and implementing a distributed framework for adaptive e-Assessment systems. The distributed framework boasts three main features: distributed architecture for a low-cost solution to computationally intensive tasks, peer-to-peer architecture for easy extension and flexible deployment, and the Apache Thrift framework for lightweight and cross-language collaboration. The distributed framework potentially supports the balancing of computational load over a number of computers in a network so as to allow a computer cluster to work collaboratively to provide higher computational power. The peer-to-peer architecture enables each computer in the network to act as both client and server. A peer-to-peer service node can be flexibly deployed on any computer that has access to the naming service server[9].

The framework aims to make an attempt to separate the implementation of adaptive e-Assessment strategies from other parts of the system to improve the reusability of the application. It also attempts to even out the computational load by supporting distributedly deployed e-Assessment services. As mentioned above, this framework is defined as a general web-based framework for online e-Assessments. Therefore, the framework consists of a number of libraries which can be used as the basis from which to quickly contruct an e-Assessment application. Sun Java was used as the programming language for the framework because of its platform-independent feature. Currently the framework included in this research used Java as the programming language.

[9] A naming service server associates the name of the service provider with the location and information of the service.

However, some key elements in the framework such as the structure of the question unit supports cross-programming-languages extensibility and can be converted into libraries of other programming languages like c++, c#, vb, and python. For this reason there have been a lot of universal Java-based presentation frameworks such as Struts, Struts 2, and Stripes. The user interface libraries were not included in this framework, which makes it a completely back-end development framework for adaptive e-Assessment systems.

The three-parameter IRT model was used as the adaptive model in the research, which is an approach to adaptive e-Assessments consisting of unidimensional dichotomous items. The term "unidimensional" here refers to the fact that there is only one latent trait that affects the examinee's answer to the item. The term "dichotomous" refers to items with only two possible answer categories such as yes/no or true/false. Therefore, the findings of this research are limited to the generalisations that can be made. However, the adaptive e-Assessment framework implemented is able to support other adaptive assessment models.

1.5 CONTRIBUTION TO KNOWLEDGE

In relation to the research aim and research questions, this study claims a number of points as original contributions to knowledge. The contributions relating to the application of three-parameter logistic IRT-model-based adaptive e-Assessment, which is the primary research question in this study, are as follows:

- In the study, an adaptive e-Assessment algorithm based on the three-parameter logistic IRT model and Fisher's information matrix (Lehmann and Casella, 1998), which was optimised to adapt to the small-scale candidates environment, was implemented.
- The performance of the implemented e-Assessment algorithm was evaluated within a normal classroom environment in higher education, with comparisons to Cisco fixed-form e-Assessment.

- The reasonable question pool size for an ideal performance was evaluated through Monte Carlo simulation[10].

The contributions related to the framework for the development of web-based adaptive e-Assessment, which is the secondary research question in this study, are as follows:

- A peer-to-peer distributed framework for web-based adaptive e-Assessment was designed and implemented, allowing individual service components to be deployed on different computers in the network. This architecture is advantageous for the flexible extension of application components and the various needs of different adaptive e-Assessment models. The distributed nature of the framework also provides a potentially low-cost solution to computationally intensive tasks. However, the algorithm for balancing computational load over a number of computers within the network was not implemented in the framework.
- The implemented framework used the Apache Thrift framework (Slee *et al*., 2007) for cross-language services development. This enables developers with different language skills to collaboratively develop an application.
- By default, the implemented framework stores questions in a MySQL® database (MySQL, 2010) in the form of Binary Large Objects (BLOBs). This design frees developers from designing a new database table for each new question object. In practice, different question properties need to be defined due to the variety of adaptive approaches and models, which could result in each developer creating their own database table for their own question object. The design and implementation of storing question objects into BLOBs regardless of their structures reduces the effort of developing adaptive e-Assessment. Developers would only need to write a structure or class for their own question objects and then deflate them and store them in a database as BLOBs. The stored questions can be retrieved from the database and inflated into the original structure or class form.

[10] For details of Monte Carlo simulation, please refer to chapter 3, Methodology.

The above points contributed to the research and application of adaptive e-Assessment in a normal classroom environment in higher education.

1.6 THE STRUCTURE OF THE DISSERTATION

This research comprises two main parts: the adaptive assessment engine and the distributed framework. Therefore, the dissertation was also accordingly composed of two major divisions. The first division describes the theoretical background of adaptive assessment and the experimental results of this research. The experiment of the practical performance of IRT-based adaptive assessment was implemented within a MATLAB environment due to the rich mathematical functions supported. Considering the need to benefit a large range of users, a web-based adaptive assessment system was later developed using Java Server Pages (JSP). Then it was realised that a web-based framework is required so as to benefit more developers and researchers in this field. Therefore, the second division moved on to depict a web-based distributed framework for adaptive e-Assessment.

This dissertation is subdivided into eight chapters. The second chapter starts with a broad review of the literature in assessment theories, then gradually focalises on adaptive assessment theories, and ends with a review of distributed computing. The third chapter describes the research methodology used to analyse and tackle the two aforementioned research questions. Chapter 4 addresses the theoretical and implementation details of the proposed e-Assessment strategy as well as its experimental performance in a pragmatic environment. Chapter 5 details the design and implementation of the framework. The sixth chapter analyses the requirements to be considered in the design and implementation of the e-Assessment framework. Chapter 7 discusses the findings in relation to this research. The last chapter concludes the research, appraises the value of this research, and brings up the intended future work.

1.7 SUMMARY OF THE CHAPTER

This chapter introduced a brief history of assessment and the background information about e-Assessment and adaptive e-Assessment. The main branches of e-Assessment theories and research were also discussed to elicit the research aim and research questions. Finally, in this chapter, the contributions to knowledge and the structure of this dissertation were presented. The next chapter will conduct the literature review on assessment theories with a focus on adaptive e-Assessment theories, as well as the theories and common practices relating to web-based applications and frameworks.

CHAPTER 2 LITERATURE REVIEW

2.1 INTRODUCTION

This chapter firstly gives a brief review on assessment theories and then moves on to elaborate upon adaptive e-Assessment theories. IRT and MDT are the two main adaptive assessment approaches introduced in this chapter. IRT is the emphasis of the literature review conducted as well as the theoretic foundation employed in this research. This chapter comprehensively introduces the main categories of IRT such as the dichotomous IRT, the polytemous IRT, the unidimensional IRT, and the multidimensional IRT. MDT, as another main adaptive e-Assessment approach, which was proposed by Rudner (2001) and employs Bayesian Theorem and statistical

decision theory to categorise candidates into a number of ability groups, is also reviewed. Subsequently, this chapter also briefly reviews some technologies related to web application frameworks such as the model view controller architectural pattern, distributed architecture and web services. Distributed architecture and the model view controller architectural pattern were followed in the design and implementation of the e-Assessment framework.

2.2 ASSESSMENT THEORIES

Assessment is an inseparable part of the pedagogic process and a complex procedure in educational activities. The purposes of deploying an assessment may include identifying a person's learning difficulties, judging a person's mastery of skills and knowledge, and ranking a person's capabilities among a group of candidates (Price *et al.*, 2008). Despite the various purposes assessments may serve, they can be categorised into two broad groups: formative and summative. Summative assessments are used to evaluate students' learning outcomes and frequently have far-reaching impacts on students' future careers, while formative assessments are usually used to improve the quality of teaching and learning by building up students' profiles (Ecclestone, 2007; Sadler, 2008).

Whether an assessment is effective or not is measured by three elements: validity, reliability and explicitness (UKCLE, 2010). The validity of an assessment is determined by consistency of the practical measurement outcome and the designed measurement purpose of the assessment. The reliability of an assessment refers to the accuracy of the measurement and the repeatability of the assessment. The explicitness of an assessment refers to the clarity and transparency of the information given regarding the assessment.

In order to support assessment activities with different purposes, a number of theories have been developed and researched. The following section of this chapter briefly introduces some of these theories.

2.2.1 CLASSICAL TEST THEORY

The theoretical framework for Classical Test Theory (CTT) is the true score model, in which observed score is equal to the sum of the true score and the error score, namely X = T + E. CTT is based on the weak assumption that an examinee's ability is reflected by the average of many tests rather than a single test. There are four indicators in CTT: item difficulty, item discrimination, reliability and the Standard Error of Measurement (SEM).

Item difficulty is typically referred to as the p value (for percentage correct), which indicates the percentage of test takers who answered a given item correctly (Shultz and Whitney, 2004). Item discrimination is usually defined as the point-biserial correlation coefficients between how the test takers answered a given item and overall test performance (Shultz and Whitney, 2004). In CTT, reliability is mathematically defined as the ratio of the variation of the true score and the variation of the observed score, or one minus the ratio of the variation of the error score and the variation of the observed score:

$$\rho_{xx'} = \frac{\sigma_T^2}{\sigma_X^2} = 1 - \frac{\sigma_E^2}{\sigma_X^2}, \qquad (2.1)$$

where $\rho_{xx'}$ is the symbol for the reliability of the observed score, *X*, and σ_T^2, σ_X^2 and σ_E^2 are the variances on the true, measured and error scores respectively. SEM is not related to the accuracy of the measurement; it is the difference between an examinee's true score (which cannot be directly measured) and his/her highest or lowest observed score.

CTT is the most common theoretic foundation for traditional paper-and-pencil assessments, fixed-form e-Assessments and random e-Assessments. Paper-and-pencil assessments and fixed-formed e-Assessments are administered in the same linear way, that is, they deliver a fixed set of questions in a fixed order. The random e-Assessment approach is a variation on the fixed-form e-Assessment approach which operates slightly differently. Instead of administering a fixed set of questions in a fixed order, the random e-Assessment approach delivers the same fixed set of questions in a

random order to improve the security of assessments. It is noticed that the Cisco chapter tests, which were selected as the comparative object in this research, were instances of fixed-form e-Assessments based on CTT. Therefore the comparison studies in this research can be viewed as the comparisons between the performance of CTT-based e-Assessments and the performance of IRT-based adaptive e-Assessments in a higher education environment with small-scale candidates.

2.2.2 CRITERION-REFERENCED TESTING

Norm-referenced testing is an assessment paradigm where an individual's performance is compared to the performance of other examinees. Norm-referenced testing aims to provide the information about an individual's relative position among a norm group of his/her peers. Therefore, it is used to discriminate between high and low achievers within a defined field (Popham, 1975).

In contrary to norm-referenced testing, criterion-referenced testing measures an individual's ability by comparing it to a specific standard or learning objective (Bond, 1996). In other words, a criterion-referenced test is intended to assess an individual's absolute ability rather than his/her ranking among a group of people. Criterion-referenced testing is commonly used as the measurement for high-stake tests. In criterion-referenced testing, there may be such a situation in which all the examinees fail to meet the threshold of an acceptable mark or grade.

Norm-referenced testing and criterion-referenced testing reflect an individual's performance with different emphases. Therefore, when diagnosing an assessment instrument, it is vital to determine which measurement method is to be adopted. For example, suppose a pupil scored 95 and 70 respectively in two consecutive arithmetic tests and his percentile ranks in the class were respectively 70^{th} and 85^{th}. From a criterion-referenced perspective, the pupil's performance during the second test declined compared to the previous test; however, from a norm-referenced point of view, he made some progress in the second test. To give another example, a high-jump coach wanted to pick out some of his students to compete in a sports event at national level. If he adopted norm-referenced testing, he would not know the exact percentile as threshold. Thus he could set up

a criterion as the threshold score for selecting qualified contestants. He might refer to the past records of this event to determine which students could jump over 1.8 metres and therefore qualify for selection.

2.2.3 ADAPTIVE ASSESSMENT AND ITEM RESPONSE THEORY

Adaptive assessment, also known as computerised adaptive testing, is a computer-controlled system where computers make dynamic decisions about which questions should be presented to a student based on the estimate of his/her proficiency level in the selected topic. The computers also question attributes such as difficulty level, discrimination power and guessing factor. It is a method of assessment which employs computers to dynamically administer appropriate items to the examinee. The basic notion of an adaptive assessment system is to automatically mimic what an experienced examiner would do (Wainer *et al*., 2000). An adaptive assessment system differs from a non-adaptive assessment system in that the former offers an intelligent approach to assessments whilst the latter is essentially a computerised delivery of a traditional paper-and-pencil-based assessment. Modern psychometrical theories and mathematical models are commonly used in an adaptive assessment system as the underpinning theoretical foundations to overcome those innate drawbacks of Classical Test Theory (CTT).

With the advancement and popularity of ICT, traditional paper-and-pencil-format assessments are increasingly being replaced by e-Assessment. The initial form of e-Assessment is the fixed-form e-Assessment, which is an evolution and synthesis of classical examination media and presentation method rather than the core theory. In such an examination system, questions are administered to students in a fixed order as in the conventional paper-and-pencil-format examination.

Adaptive e-Assessment differs from fixed-form e-Assessment in the way of test administration. The dynamic selection of questions in adaptive e-Assessment can lead to shorter tests and more accurate ability estimation (Sands, Waters, & McBride, 1997; van der Linden & Glas, 2000; Wainer, 2000). However, the complex requirements of adaptive e-Assessment sometimes offset its

advantages. The sample-size sensitivity characteristic of Item-Response-Theory-models-based adaptive e-Assessment also restrains its application in large-scale examination environments (Tao *et al.,* 2008). The empirical study involved in this chapter modified an existing Cisco fixed-form computer-based test into an Item-Response-Theory-models-based adaptive e-Assessment to compare the performance difference within a 30-student class.

One of the basic functions of assessment is to infer students' level of understanding at a given point in time. As pointed out by Pellegrino *et al*. (2001), assessment is a process of reasoning from evidence. Since one cannot directly perceive students' mental processes, one must rely on less direct methods such as testing to make judgements on what they know. The goal of testing is to maximally reduce random errors contained in the results so as to reveal more facts.

An Item-Response-Theory-models-based adaptive e-Assessment views every single question on a test as an item and attempts to avoid unnecessary questions for the test-taker to maximise the testing efficiency. An adaptive e-Assessment can be typically shortened by 50% and still maintain a higher level of precision than fixed-form computer-based tests (Weiss and Kingsbury, 1984). However, the precision of testing results of an adaptive e-Assessment largely relies on the precision of its item calibration procedure. The item calibration procedure, which is similar to the "training" procedure of an Artificial Neural Network (ANN), is the prerequisite of an Item-Response-Theory-models-based computerised adaptive test and requires a large number of historical responses from different respondents. Various minimum sample sizes have been suggested by different researchers. Downing (2003) suggests 200 students to be a minimum sample size to fit the simplest IRT model whilst Reeve and Fayers (2005) consider 100 as perhaps the smallest sample size for Rasch models and Linacre (1994) suggests 50 for the simplest IRT model. Theoretically, the more respondents involved, the more accurate the calibration results will be. As most educationists only have access to a limited number of students, it is difficult to apply Item-Response-Theory-models-based adaptive e-Assessment in small- and medium-sized classes. Little research has been done to

investigate whether the possible advantages of an IRT model can still be obtained in such an environment.

In conventional paper-and-pencil assessments, examinees are evaluated under a true score model which assumes that the final score is influenced by true score and random error (Breakwell *et al*., 2006). The true score is an examinee's real capability measured without any errors. Psychologically, there is no way to acquire a true score but only an observed score, which is usually measured by an assessment associated with some extent of error. In CTT, all the examinees on the same test are exposed to an identical set of items and the results are based on the number of correct answers provided. This has posed a problem to examiners: for a large group of examinees, there must be a large set of testing items to cover all the possible ability levels among the examinee group in order to get the most precise classification. Such a lengthy assessment will inevitably result in fatigue which has an impact on examinees' performance. A potential solution to this problem is maximally to improve the efficiency of assessments by skipping those inappropriate items. This is where adaptive assessment can help.

Another major pitfall of CTT is that the observed score acquired using this theory is sample-dependent (Weiss and Yoes, 1991). In other words, the indicators adopted by CTT such as difficulty, discrimination and reliability vary as the examinee sample changes. Therefore, it is very difficult for the same assessment to obtain an identical level of difficulty, discrimination or reliability across different cohorts. For instance, the same assessment administered to a group of year one students and a group of year two students would probably lead to a different difficulty estimate for the questions in the assessment. Likewise, two assessments with different difficulty levels administered to the same group of students would also lead to a different estimate for the students' abilities. Hence the statistics associated with CTT are dependent on the group of examinees and the content of the assessments.

In order to overcome the above drawbacks of CTT, some modern psychometric theories have been adopted in adaptive assessment systems to improve the performance of assessments. The most prominent two among them are Item Response Theory (IRT) and Measurement Decision Theory (MDT).

IRT is the primary theoretical foundation for adaptive assessment. It is a body of related psychometric theory that provides a foundation for scaling persons and items based on responses to assessment items, which is gradually becoming popular (Reise *et al.*, 2005). Unlike Classical Test Theory (CTT), in which the main indicators such as difficulty, discrimination and reliability are sample-dependent, IRT has the property of invariance since it correlates item parameters to examinees' abilities.

The most prominent feature of IRT is that it explicitly defines the mathematical relationship between the latent traits and some item parameters. The latent trait here is also known as ability in many educational applications and the item here refers to a single question. IRT derives the probability of each response as a function of the latent traits and some item parameters. The same model is then used to obtain the likelihood function of abilities as a function of the actually observed responses and, again, the item parameters (Partchev, 2004). The ability values that maximise the likelihood become the ability estimates. In statistical terms, the probabilistic function can be viewed as a probability density function (PDF) and the abilities can be viewed as parameters. The ability estimation is the process of establishing the parameters to maximise the likelihood for the actually observed responses to occur.

In order to make IRT theory work, item calibration must precede all the other procedures. In the process of item calibration, the items are given to a sufficient number of examinees and their responses are then used to estimate the item parameters. This is an iterative process for the reason that both examinees' abilities and item parameters must be taken into account in order to achieve reasonably accurate calibration. The most important advantage of the iterative calibration is that it

builds a universal platform to compare the abilities of examinees who attempt different items. Therefore, it also provides a foundation based upon which adaptive testing can be applied while still maintaining the objectivity of measurement (Embretson and Reise, 2000).

Take a simple assessment consisting of only one item. Assume, for example, that the only item is a multiple-choice question, "What is the area of a rectangle that has a length of 3cm and a width of 2cm?" with three possible options (1) $9cm^2$, (2) $10cm^2$, and (3) $6cm^2$. Among the three options, the first one is the one that deviates furthest from the correct answer; the students who selected this option exhibit no knowledge in the area of rectangles. The second one is still incorrect but implies better knowledge; the students who chose this option may understand the concepts of area and circumference but tend to confuse them. The last one is the correct answer. Assume that this item has been correctly calibrated and its psychometric properties are plotted as below:

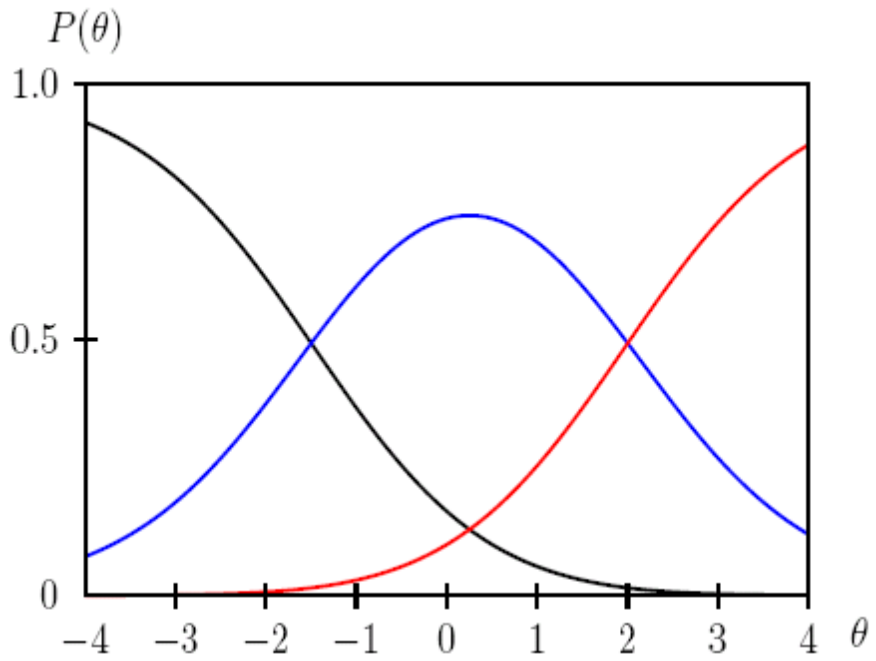


Figure 2 - 1: Item Response Function for a Partial Credit Multiple-choice Question (Partchev, 2004)

The above figure shows the relationship between an examinee's ability θ and the probabilities for choosing the three options $P(\theta)$. The black curve, blue curve and red curve respectively represent

the variance of the probabilities for choosing option one, option two and option three. Because one person can only choose one option for this item, the probabilities for three options to be chosen at one ability point should sum up to 1. As the examinee's ability increases, the probability for him/her to select option one gradually drops, the probability to select option two climbs up first then goes down, and the probability to select option three gradually increases.

In the above case, the two incorrect responses are considered separately (Kolen and Brennan, 2004). This is a so-called partial credit model or polytomous model, in which partial credit scoring is allowed for those partially correct answers. Those IRT models which only allow for two alternative responses are known as dichotomous models. In dichotomous models, all the responses are considered to be either correct or incorrect.

For both dichotomous IRT models and polytomous IRT models, there are three basic components which are Item Response Function (IRF), Item Information Function (IIF) and Standard Error of Measurement (SEM).

IRF is the fundamental assumption of the IRT theory. It is the formula which models the mathematical relationship between an examinee's response and other determinant parameters. The form of IRF varies according to different IRT models. Nonetheless, the IRFs in all of the known IRT models associate the probability of giving a certain response with the examinee's relevant abilities and item parameters.

IIF is the mathematical equation to model the amount of information carried by an item for a certain examinee. The statistical meaning of information is defined as the reciprocal of the precision with which a parameter could be estimated (Lord, 1980). Therefore, the more precisely the parameter is estimated, the more information is to be gained. Statistically, the precision with which a parameter is estimated is measured by the variability of the estimates around the value of the parameter (Baker, 2001). Thus, if the variance of the estimators is denoted by σ^2, the amount of information, denoted by I, can be given by the following formula:

$$I = \frac{1}{\sigma^2} \tag{2.2}$$

For an examinee with ability estimate θ, the total amount of information carried by a test, denoted as TI, is simply the sum of the information of all the items contained in the test, namely

$$TI(\theta) = \sum_{i=1}^{N} I_i(\theta) \tag{2.3}$$

, where N is the number of questions.

The Standard Error of Measurement (SEM) is defined as the reciprocal of the square root of the total test information regarding the current examinee's ability, which is the sum of the information of all the attempted test items (Baker, 2001). If the test information is denoted as TI, the SEM can be given by the formula below:

$$SEM(\theta) = \frac{1}{\sqrt{TI(\theta)}} \tag{2.4}$$

According to the dimensionality of the data that the theory can be applied to, IRT can be further broken down into two subcategories: Unidimensional Item Response Theory (UIRT) and Multidimensional Item Response Theory (MIRT).

UNIDIMENSIONAL ITEM RESPONSE THEORY (UIRT)

In IRT, the probability for an examinee to give a correct response to an item is affected by the examinee's ability and the item's parameters. It has two basic assumptions: the first assumption is unidimensionality, namely, all the items to be tested are to measure the same dimensionality. The second assumption is local independency, namely, for examinees with the same ability level, the probability of giving a correct response is irrelevant to the probability of giving correct responses to other items.

Based on scoring schemes, UIRT models can be further broken down into dichotomous UIRT models and polytomous UIRT models. Dichotomous models classify all the responses from examinees into either "right" or "wrong", whereas polytomous models award partial credit for those examinees who provide partially correct answers.

DICHOTOMOUS UIRT MODELS

Among dichotomous UIRT models, the IRF is usually associated with the examinee's ability and three other relevant parameters, which are discrimination, difficulty and the guessing factor. However, not all of the dichotomous UIRT models use all three parameters. In light of the number of parameters utilised, dichotomous UIRT models can be further broken down into one-parameter models, two-parameter models and three-parameter models.

The one-parameter model, also known as the Rasch model, is the most parsimonious IRT model which merely considers the difficulty parameter b and ignores the other influences (Wright, 1968; Rasch, 1980). The IRF of this model is given by the formula:

$$P(U_{ij}=1|\theta_j,b_i)=\frac{1}{1+e^{-1.7(\theta_j-b_i)}}, \qquad (2.5)$$

where θ_j is the ability of examinee j, b_i is the difficulty parameter of item i and 1.7 is the constant used to approximate the normal ogive function using the logistic function (Weiss and Yoes, 1991).

The two-parameter model is based on the assumption that both the item discrimination and item difficulty are variable (Birnbaum, 1968). The IRF of this model is given by the formula:

$$P(U_{ij}=1|\theta_j,a_i,b_i)=\frac{1}{1+e^{-1.7a_i(\theta_j-b_i)}}, \qquad (2.6)$$

where a_i is the discrimination parameter of item i, and θ_j and b_i are the same as they are in the one-parameter model.

All three parameters – the discrimination parameter, difficulty parameter and guessing factor parameter – are changeable under the three-parameter model. The IRF for examinee j to give a correct answer to item i can be represented with a logistic function (Birnbaum, 1968):

$$P(U_{ij}=1|\theta_j,a_i,b_i,c_i)=c_i+(1-c_i)\frac{1}{1+e^{-1.7a_i(\theta_j-b_i)}}, \qquad (2.7)$$

where c_i is the guessing factor of item I, and θ_j, a_i and b_i are the same as they are in a two-parameter model.

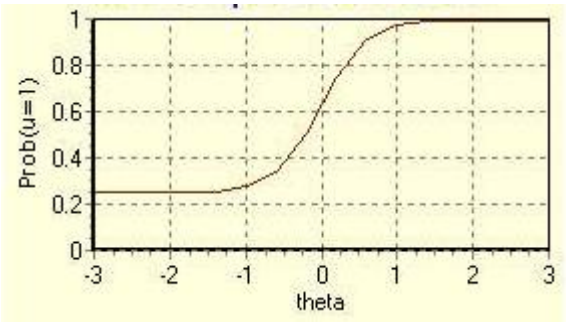


Figure 2 - 2: Item Response Function with parameters: a_i = 2.0, b_i = 0.0, c_i = 0.25 and θ_j varying from -3.0 to 3.0 (Rudner, 1998)

The IIF for the above 3PL IRT models is formulated as:

$$I_i(\theta)=\frac{P'_i(\theta)^2}{P_i(\theta)(1-P_i(\theta))}, \qquad (2.8)$$

where $P_i(\theta)$ denotes the probability for the current examinee to give a correct answer to item i and $P'_i(\theta)$ is the first derivative of $P_i(\theta)$.

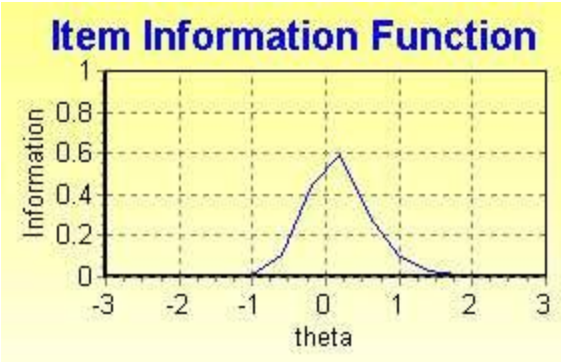


Figure 2 - 3: Item Information Function with parameters: a_i = 2.0, b_i = 0.0, c_i = 0.25 and θ_j varying from -3.0 to 3.0 (Rudner, 1998)

POLYTOMOUS UIRT MODELS

In addition to the aforementioned dichotomous models, there are polytomous UIRT models which allow for partial credit scoring for examinees with partially correct answers. These models are intended to differentiate between examinees with no knowledge and examinees with various degrees of knowledge. For example, in an essay-writing test, better writing styles usually score more points. The graded response model, the partial credit model and the generalised partial credit model are the three most widely employed polytomous models.

The graded response model is suitable for polytomous items whose responses are ordered based on the indication of knowledge level (Ostini and Nering, 2006). These responses are referred to as graded responses.

TESTLET RESPONSE MODELS

Lee, Brennan, and Frisbie (2000) broadly defined a testlet as "a subset of the items in a test form that is treated as a measurement unit in test construction, administration, and/or scoring." However, most of the literature delimits a testlet as a group of items that relate to a single stimulus (Wang and Wilson, 2005). The stimulus-based testlets refer to a subgroup or "bundle" of items which is a part of a test. These items are interrelated by some common stimulus beyond the latent trait being

assessed. Reading comprehension, analytical story questions, and interpretation of tables or figures are the most common applications of testlets. Reading comprehension usually consists of a short article followed by a subset of items; this represents a typical testlet since, in addition to the latent trait being measured, these items are interconnected by the information in the passage. The analytical-story and table-interpretation-type questions are also comprised of a subset of items related to the story and the text.

MULTIDIMENSIONAL ITEM RESPONSE THEORY (MIRT)

UIRT aims at tests in which all the items are measuring the same individual ability due to its unidimensional assumption. As a matter of fact, there are many real-life questions which cannot be solved by merely using a single ability or latent trait (Kelderman, 1996). The unidimensional assumption in UIRT is already compromised when the probability to generate the correct answer to a question is influenced by more than one latent trait. In such a case, UIRT is no longer applicable. The research conducted by Ackerman (1991) indicates that unidimensional calibration of multidimensional data tends to enlarge the influence of more discriminating dimensions and "filter out" those less discriminating dimensions.

As the successor of UIRT, Multidimensional Item Response Theory (MIRT) was developed as an extension of UIRT to cope with the measurement of multidimensional data (Reckase, 1997). The major difference between UIRT and MIRT is that MIRT takes more than one latent trait into account while assessing an examinee's performance.

Two distinct categories of multidimensionality, between group and within group, have been defined (Steinberg *et al*., 2000). According to Wang, Chen and Cheng (2004), an item is identified as a unidimensional item when it is intended to evaluate a single latent trait, while an item is identified as a multidimensional item when it is intended to evaluate multiple latent traits. With the aforementioned definitions, a between-group multidimensional test can be described as a test containing unidimensional items that assess different latent traits, whereas a within-group

multidimensional test can be described as a test containing multidimensional items. Figures 2-1 and 2-2, as shown below, depict between-group and within-group multidimensionality.

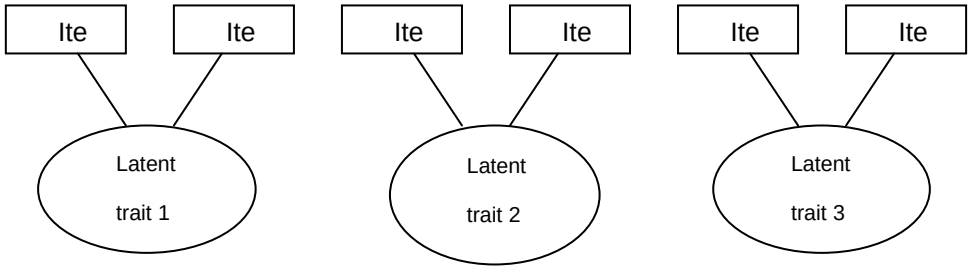


Figure 2 - 4: graphical description of between-group multidimensionality (adapted from Wang, Chen and Cheng, 2004)

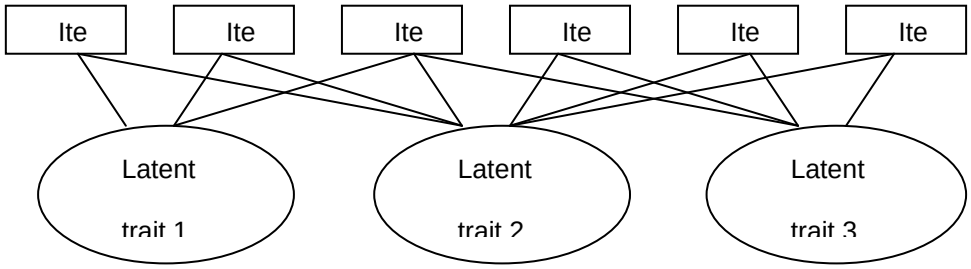


Figure 2 - 5: graphical description of within-group multidimensionality (adapted from Wang, Chen and Cheng, 2004)

Most of the MIRT models are derived from UIRT models. For example, the multidimensional two-parameters model suggested by Mckinley and Reckase (1983) is an extension of the two-parameters IRT model, in which the item response function is defined as below:

$$P_i(x_{ij}=1 \mid a_i, b_i, \theta_j) = \frac{1}{1+\exp[-(a'_i \theta_j - b_i)]}, \qquad (2.9)$$

where x_{ij} is examinee j's response to item i, and it equals 1 if the response is correct or 0 if the response is incorrect; a_i is the discrimination vector of item i, b_i is the difficulty level of item i

and θ_j is the ability vector of examinee j. This multidimensional model extends item discrimination parameter a_i and examinee ability parameter θ_j from scalars in the unidimensional model to vectors in the new multidimensional model. In this way, the new model takes the influence from multiple latent traits into consideration.

The multidimensional models developed by Sympsom (1978) and Hattie (1981) are similar, as they are both built on the three-parameters UIRT model proposed by Birnbaum (1968). The item response function is modelled as follows:

$$P_i(x_{ij}=1 \mid a_i, b_i, c_i, \theta_j) = c_i + \frac{1-c_i}{1+\exp[-a'_i(\theta_j - b_i 1)]}, \qquad (2.10)$$

where a_i is the discrimination vector of item i, θ_j is the ability vector of examinee j, and 1 is a D by one vector used to convert the difficulty parameter b_i from scalar to vector so as to make the subtraction between θ_j and b_i valid.

The above two MIRT models are quite similar in the sense that both of them are the extensions of the corresponding UIRT models and both of them can only cope with dichotomous data – and, because of the restrictions of the corresponding UIRT models, these two MIRT models are only applicable to mutually independent multidimensional data. In other words, these two UIRT models can only be applied to between-group multidimensional adaptive tests.

Adams *et al*. (1997) suggested a multidimensional random coefficients multinomial logit model (MRCMLM) which is a generalised MIRT model that can be used to adapt both between-group and within-group multidimensional tests. In the MRCMLM model, the probability for examinee j to give a response in category k of item i is formulated as below:

$$P_i(x_{ijk}=1,\xi \mid \theta_j)=\frac{\exp(b'_{ik}\,\theta_j + a'_{ik}\,\xi)}{\sum_{u=1}^{K_i}\exp(b'_{iu}\,\theta_j + a'_{iu}\,\xi)}, \tag{2.11}$$

where x_{ijk} is 1 if examinee j's response to item I is in category k or is 0 otherwise, ξ is the parameter vector to describe the items, θ_j is the ability vector of examinee j, K_i is the number of categories in item I, b'_{ik} is a scoring vector given to category k of item I across all the dimensions of latent traits, and a'_{ik} is a design vector given to category k of item i which defines the linear relationship among the elements of ξ.

The Item Information Function in MRCMLM is given as follows (Segall, 1996):

$$[I(\theta,\widehat{\theta}_j)+I(\theta,x_{k'})+\Sigma^{-1}], \tag{2.12}$$

where

$$I(\theta,\widehat{\theta})=\sum_{i\in V}\left\{\left[\sum_{k=1}^{K_i} b_{ik}b'_{ik}\; f_{ik}(\theta)\right]-E_i(\theta)E_i(\theta')\right\} \tag{2.13}$$

Σ is the covariance matrix of *X*, and

$$E_i(\theta)=\sum_{k=1}^{K_i} b_{ik}\,f_{ik}(\theta) \tag{2.14}$$

Based on IRT models and the assumption of local independency, the likelihood function for an examinee's response set to an assessment can be computed and hence each examinee's ability after attempting different items can be estimated. Adaptive assessment systems utilise the high speed of modern computers to iteratively estimate an examinee's ability and present the most appropriate question. Therefore, it does not need the interference of human beings and is able to achieve the

same measurement accuracy with a smaller number of test questions compared to conventional tests (Sand, Water and McBride, 1997; Wainer *et al*., 1990; Weiss, 1985).

The early adaptive assessment systems based on the UIRT theory have several restrictions which hinder the application of this modern test approach. First of all, the UIRT theory is built upon the assumption that there is a single ability which has an impact on examinees' performance on one test, which is not the case in many real-life tests. Secondly, the UIRT theory does not take into account the correlations between different individual abilities. Therefore, the test reliability is lower compared to more sophisticated models.

In order to overcome the drawbacks of UIRT, based on Multidimensional Item Response Theory (MIRT), researchers have proposed the concept of multidimensional adaptive assessment (Li and Schafer, 2005; Wang and Chen, 2004). Multidimensional adaptive assessment breaks through the limitation that a test can only include items measuring a sole ability. Compared to UIRT-based adaptive assessment, it requires fewer numbers of items to achieve the same reliability and precision of ability estimate by taking the correlativity into consideration. The study conducted by Wang, Chen and Cheng (2004) indicates that, in the condition of high correlation between different latent traits, multidimensional adaptive assessment can greatly improve the estimation reliability of each latent trait. However, Chen's study (2006) also suggests that multidimensional adaptive assessments impose an enormous computational burden on the assessment-providing server.

In adaptive assessments, one of the basic components is the ability estimation. Maximum likelihood (ML) method, Bayesian maximum *a posteriori* (MAP) method and Bayesian expected *a posteriori* (EAP) method are the three most frequently used estimation methods.

THE MAXIMUM LIKELIHOOD METHOD

The Maximum Likelihood (ML) method, also known as maximum likelihood estimation (MLE), is a general method to estimate parameters for a probability distribution to make it best fit the

observed data (Myung, 2003). The ML method is a standard parameter estimation method and one of the three most commonly used ability estimation methods in adaptive e-Assessment (Keller, 2000). The estimation of an examinee's ability is an important procedure in adaptive e-Assessment, which has a direct impact on the measurement precision of adaptive e-Assessment. The ML method was used in this research to estimate the examinee's ability due to its advantage of fast convergence (Keller, 2000).

To illustrate the ML method, consider the following scenario. A footballer scored 6 goals out of 10 trials and we need to calculate the probability of him scoring a goal in any one trial. There are only two possibilities in this scenario, goal or miss, therefore it conforms to Bernoulli distribution whose probability density function is:

$$f(y \mid n, w) = \frac{n!}{y!(n-y)!} w^{y} (1-w)^{n-y} (0 \le w \le 1; y = 0,1,\cdots,n), \tag{2.15}$$

where w is the probability of a success on any one trial, n is the total number of trials, and y is the total number of successes. In this case, the likelihood function can be given by:

$$L(w \mid n = 10, y = 6) = f(y = 6 \mid n = 10, w) = \frac{10!}{6!4!} w^{6} (1-w)^{4} (0 \le w \le 1). \tag{2.16}$$

In order to quickly find out the w to maximise the likelihood function, a general way is first to compute the logarithm of the likelihood function and then solve the w for the following first differential equation which is known as the log-likelihood equation:

$$\frac{\partial \ln L(w \mid y)}{\partial w} = 0 \tag{2.17}$$

Another condition which must also be satisfied is that the second differential equation must be negative at w. That is,

$$\frac{\partial^2 \ln L(w \mid y)}{\partial w^2} < 0 \tag{2.18}$$

In this case, the log-likelihood function of the aforementioned one-parameter Bernoulli model is calculated as:

$$\ln L(w \mid n = 10, y = 6) = \ln \frac{10!}{6!4!} + 6 \ln w + 4 \ln(1 - w) \tag{2.19}$$

The first derivative of the log-likelihood is calculated as:

$$\frac{d \ln L(w \mid n = 10, y = 6)}{dw} = \frac{6}{w} - \frac{4}{1 - w} = \frac{6 - 10w}{w(1 - w)} \tag{2.20}$$

Next, the second derivative of the log-likelihood is calculated as:

$$\frac{d^2 \ln L(w \mid n = 10, y = 6)}{dw^2} = -\frac{6}{w^2} - \frac{4}{(1 - w)^2} \tag{2.21}$$

According to the two differential equations above, it can be computed that the likelihood is maximised at w = 0.6. The relationship between likelihood and parameter w can be plotted as below:

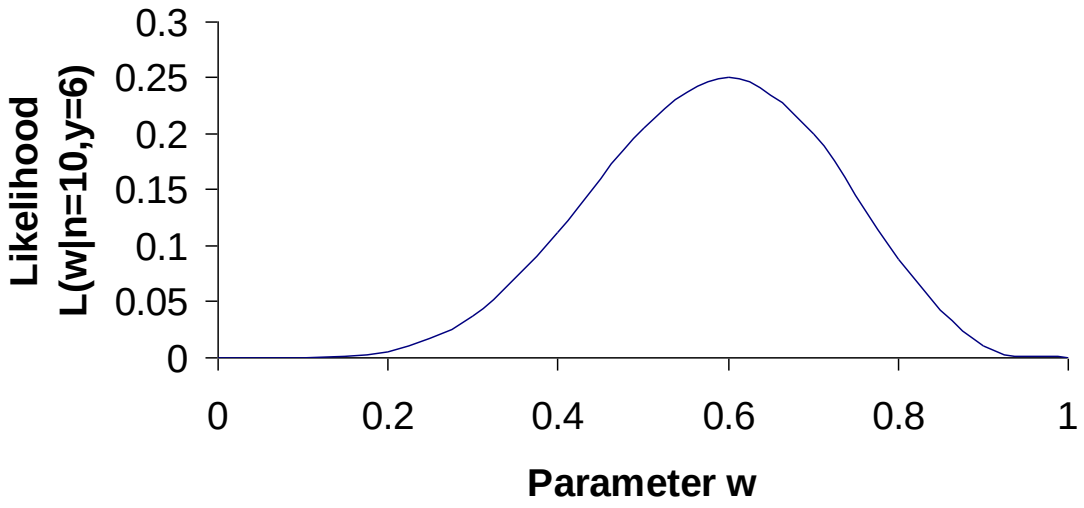


Figure 2 - 6: the likelihood function given observed success y = 6 and the number of trials n = 10 for the aforementioned one-parameter Bernoulli model

According to Lord (1980), an examinee's likelihood on a number of items can be formulated as:

$$L(u \mid \theta) = L(u_1, \cdots, u_n \mid \theta) = \prod_{i=1}^{n} P_i(\theta)^{u_i} (1 - P_i(\theta))^{(1-u_i)}, \tag{2.22}$$

where $u_1, \cdots, u_n$ is the vector of an examinee's attempts to n items, u_i is 1 if the answer to item i is correct and 0 otherwise, and $P_i(\theta)$ is the probability of generating a correct answer to item i when the ability level is θ.

Based on the above likelihood function, ML method can be applied to find out the θ which maximises the likelihood, i.e. the most likely ability of the current examinee. The calculation becomes more complicated when θ becomes a vector under MIRT models, which involves the measurement of multiple latent traits. Based on the MRCMLM model, the following procedures illustrate how to estimate θ using ML method. First of all, compute the logarithm of the likelihood function and find out the first differential vector for the log-likelihood function.

$$\frac{\partial \ln f(\theta \mid u)}{\partial \theta_k} = \frac{\partial \ln L(\theta \mid u)}{\partial \theta_k} = \sum_{i \in V} [b_{ik} - E_i(\theta)] \tag{2.23}$$

where $E_i(\theta) = \sum_{k=1}^{K} b_{ik} f_{ik}(\theta)$ and $f_{ik}(\theta) = \frac{\exp(b'_{ik}\theta_j + a'_{ik}\xi)}{\sum_{u=1}^{K_i} \exp(b'_{iu}\theta_j + a'_{iu}\xi)}$, and V is the set for items selected in the test,

$$\frac{\partial \ln L(\theta \mid u)}{\partial \theta} = \begin{bmatrix} \frac{\partial \ln L(\theta \mid u)}{\partial \theta_1} \\ \frac{\partial \ln L(\theta \mid u)}{\partial \theta_2} \\ \vdots \\ \frac{\partial \ln L(\theta \mid u)}{\partial \theta_k} \end{bmatrix} \tag{2.24}$$

The second differential for the likelihood function is a matrix as below:

$$\frac{\partial^2 \ln L(\theta \mid u)}{\partial \theta \partial \theta'} = \begin{bmatrix} \frac{\partial^2 \ln L(\theta \mid u)}{\partial \theta_1 \partial \theta_1} & \frac{\partial^2 \ln L(\theta \mid u)}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 \ln L(\theta \mid u)}{\partial \theta_1 \partial \theta_p} \\ \frac{\partial^2 \ln L(\theta \mid u)}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 \ln L(\theta \mid u)}{\partial \theta_2 \partial \theta_2} & \cdots & \frac{\partial^2 \ln L(\theta \mid u)}{\partial \theta_2 \partial \theta_p} \\ \vdots & \cdots & \cdots & \vdots \\ \frac{\partial^2 \ln L(\theta \mid u)}{\partial \theta_p \partial \theta_1} & \cdots & \cdots & \frac{\partial^2 \ln L(\theta \mid u)}{\partial \theta_p \partial \theta_p} \end{bmatrix} \tag{2.25}$$

The element of row k and column i is

$$\frac{\partial^2 \ln f(\theta \mid u)}{\partial \theta \partial \theta'} = -\sum_{i \in V} \left\{ \sum_{k=1}^{K_i} b_{ik} b'_{ik} f_{ik}(\theta) - E_i(\theta) E_i(\theta') \right\} - \sum^{-1} \tag{2.26}$$

Because there is no closed-form solution for the following first differential equation,

$$\frac{\partial \ln f(\theta \mid u)}{\partial \theta} = 0 \tag{2.27}$$

the Newton-Raphson iteration can be used to approximate (Wang and Chen, 2004). The Newton-Raphson method is an iterative procedure to find the root for $f(x) = 0$. It defines a closer root x_2 can be approximted from the previous root x_1 through the following equation: $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$

(2.28)

Let $\theta^{(j)}$ be the jth approximation to the value of θ that meets the above first differential equation. The closer approximation $\theta^{(j+1)}$ can be calculated as follows:

$$\theta^{(j+1)} = \theta^{(j)} - \delta^{(j)} \tag{2.29}$$

where

$$\delta^{(j)} = [\frac{\partial^2 \ln f(\theta \mid u)}{\partial \theta \partial \theta'}]^{-1} \times \frac{\partial \ln f(\theta \mid u)}{\partial \theta} \tag{2.30}$$

THE BAYESIAN EXPECTED A POSTERIOR (EAP) METHOD

The EAP method is another widely adopted method for ability estimation in adaptive e-Assessment (Keller, 2000). The EAP method uses the mathematical expectation of the posterior probability density function as the optimal estimate for the ability vector. The posterior probability density function $f(\theta \mid U)$ for ability distribution can be formulated as:

$$f(\theta \mid U) = \frac{L(U \mid \theta) f(\theta)}{f(U)}, \tag{2.31}$$

where $f(\theta)$ is the prior probability distribution of examinees' abilities, $L(U \mid \theta)$ is the likelihood function for the observed response at the ability level θ, and $f(U)$ denotes the marginal

probability of the response vector which is the integral of $L(U|\theta)f(\theta)$ at all ability intervals. For the tests with predefined discrete ability intervals, the mathematical expectation θ_{EAP} is as follows:

$$\theta_{EAP} = \sum_{q=1}^{k_q} \theta_q f(\theta_q | U) = \sum_{q=1}^{k_q} \theta_q \frac{L(U|\theta_q) f(\theta_q)}{\sum_{q=1}^{k_q} [L(U|\theta_q) f(\theta_q)]}, \quad (2.32)$$

where q is the number of ability intervals. As far as the multidimensional ability vector is concerned, the value of q increases exponentially as the dimensionality of the abilities to be estimated increases. For example, if the scale for a single ability dimension is divided into n discrete intervals and the number of dimensionality is 4, then the q is equal to n^4. With the increase of the number of discrete ability intervals, the precision of measurement increases as well. Meanwhile the computational power consumed in the process of ability estimation also increases at a great speed.

THE BAYESIAN MAXIMUM A POSTERIOR (MAP) METHOD

The MAP method is also a popular ability estimation method for adaptive e-Assessment (Keller, 2000). Segall (1996) depicted the MAP ability estimation method. Similar to the EAP method, the MAP method uses the Bayesian posterior probability formula as well. Assume $L(x|\theta)$ denotes the likelihood function for the observed response at the ability level θ, $g(\theta)$ denotes the prior distribution of θ, and $f(x)$ denotes the marginal probability of the response vector. The posterior density function of θ can be given as:

$$f(\theta | x) = L(x|\theta) \frac{g(\theta)}{f(x)} \quad (2.33)$$

Take the MRCMLM model. For example, assume the prior distribution $g(\theta)$ conforms to multivariate normal distribution, which can be given as:

$$g(\theta) = (2\pi)^{-D/2} \mid \Sigma \mid^{-1/2} \exp[-\frac{1}{2}(\theta - \mu)'\Sigma^{-1}(\theta - \mu)], \quad (2.34)$$

According to maximum likelihood estimation, the first differential of $\ln f(\theta \mid x)$ is

$$\frac{\partial}{\partial \theta} \ln f(\theta \mid x) = 0 \quad (2.35)$$

Expand the above formula to yield the following equation:

$$\frac{\partial}{\partial \theta} \ln f(\theta \mid x) = \frac{\partial}{\partial \theta} \ln L(x \mid \theta) - \frac{1}{2}\frac{\partial}{\partial \theta}[(\theta - \mu)'\Sigma^{-1}(\theta - \mu)] = \sum_{i \in V}[b_{ix_i} - E_i(\theta)] - [\frac{\partial}{\partial \theta}(\theta - \mu)']\Sigma^{-1}(\theta - \mu)$$
, (2.36)

where

$$E_i(\theta) = \sum_{k=1}^{K_i} b_{ik} f_{ik}(\theta) \quad (2.37)$$

Then calculate the second differential of $\ln f(\theta \mid x)$, which gives

$$H(\theta) = \frac{\partial^2 \ln f(\theta \mid x)}{\partial \theta \partial \theta'} = -\sum_{i \in V}\left\{\sum_{k=1}^{K_i} b_{ik} b'_{ik}\ f_{ik}(\theta) - E_i(\theta)E_i(\theta')\right\} - \sum^{-1} \quad (2.38)$$

The ML method, EAP method, and MAP method are all commonly used ability estimation methods in adaptive e-Assessment (Keller, 2000). Generally speaking, the two Bayesian-based ability estimation methods have better estimation reliability and smaller error compared to the ML method (Bock and Mislevy, 1982; Weiss and Mcbride, 1984), while the ML method can converge more quickly (Keller, 2000). In consideration of the small-sized question bank available, the ML method was adopted in this research to quickly obtain an estimation of an examinee's ability.

ITEM CALIBRATION IN IRT-BASED ADAPTIVE E-ASSESSMENTS

To obtain precise measurement of examinees' abilities, IRT-based adaptive e-Assessments require well-calibrated item pools (question banks). The purpose of item calibration is to assign numerical values to describe the psychometrical properties of questions. There are usually two approaches to item calibration: expert-based calibration and statistical calibration.

Expert-based calibration is a method to employ experts in the subject area to give their personal estimation of the parameter of the items. One way to obtain expert-estimated item parameters is via computer-based or paper-and-pencil-based questionnaires (Arruabarrena and Perez, 2005). Due to the fact that an expert's opinions can be subjective, it is recommended that a number of experts should be approached for an unbiased estimation. The main disadvantage of expert-based calibration is that in most circumstances experts can only estimate the difficulty parameter for items. It is usually difficult for experts to estimate parameters such as discriminative power and guess factor.

Statistical calibration is another common method used to estimate the numerical value of the parameters of the items and it was employed in this research. Given an IRT model, this calibration method can be used to obtain all the parameters of an item. However, a large group of participants may need to be recruited in the pilot test to obtain precise calibration outcomes. An issue with statistical calibration is that the security of the future assessment cannot be guaranteed because there will be a large number of individuals who have previewed the items (Kolen and Brennan, 2004).

TERMINATION CRITERION

Depending on the purpose of the assessment, the termination criterion for an adaptive assessment system may be based on a fixed-length rule or a variable-length rule. An adaptive assessment system based on fixed-length termination criteria administers a pre-defined number of items. Once all of these items are administered the e-Assessment is terminated and the examinee's ability is estimated. A fixed-length termination criterion is easy to implement because the adaptive e-

Assessment system only needs to decide whether an assessment should be terminated by counting the number of questions that have been administered. An assessment based on a fixed-length termination criterion also tends to make examinees feel they are being fairly assessed. The main disadvantage of an assessment based on fixed-length termination criterion is that it is difficult to obtain the same measurement precision for all the examinees. Frequently the examinees at the extremes of the ability distribution receive ability estimations based on a less precise assessment than examinees near the middle point of the ability distribution (Thissen & Mislevy, 2000).

A variable-length termination criterion stops an assessment when the pre-defined measurement precision is reached. In this case, measurement precision is often assessed based on the standard error of the estimation with a given ability. After each round of the item administration, the standard error of ability estimation is calculated to determine whether the assessment can be terminated. Adaptive e-Assessments based on a variable-length termination criterion can often lead to assessments with different number of questions, which makes it difficult to convince the examinees of the fairness of the assessments. The main advantage of this termination criterion is that all the examinees attending the adaptive assessment have their abilities estimated with the same precision (Thissen & Mislevy, 2000).

2.2.4 ISSUES WITH ADAPTIVE E-ASSESSMENTS

The history of adaptive assessments can be traced back to the early 20th century, when Alfred Binet (1905) made a milestone contribution to this area with his intelligence tests. Due to Mr. Binet's concern regarding the diagnosis of individual examinees rather than group examinees, he innovatively designed a tailored test for his test candidates. This type of test ranked questions by difficulty and adjusted the questions to be presented to the examinee based on the examinee's previous performance. For example, if the examinee frequently fails the current subset of questions, then he/she would be given relatively simpler questions.

Subsequently, Binet's method was refined to form the Flexilevel testing procedure (Lord, 1980; Betz & Weiss, 1975; Kocher, 1974) and a number of variants such as the Step procedure (Henning, 1987) and Testlets (Lewis & Sheehan, 1990). These methods can be easily implemented by using computers. Gradually more interest was attracted by this new evaluation paradigm (Weiss, 1974; Weiss, 1985; Flaugher, 1990; Carlson, 1994; Thiessen & Mislevy, 2000; Wise & Kingsbury, 2000; de Ayala, 2009; Thompson, 2009; Thompson, 2010). During many years' development of research in adaptive e-Assessment, many researchers have been attempting to bring the research further. For instance, Lilley *et al.* (2004) described the development and evaluation of a software prototype for adaptive e-Assessment, while Barker (2009) extended adaptive e-Assessment to an e-Learning system which can provide automated feedback to the candidates.

Among all the approaches to adaptive e-Assessment, IRT is the most commonly adopted one due to its systematicness and soundness, and it was adopted in this research as well. The following section of this chapter highlights some important issues associated with IRT and IRT-based adaptive e-Assessments, explains the reason why the three-parameter logistic IRT model was selected in this research and elicits the idea of designing and implementing a framework for adaptive e-Assessments.

USES OF IRT-BASED ADAPTIVE E-ASSESSMENT IN DIFFERENT CONTEXTS

Over the past few decades, adaptive e-Assessment has been mainly IRT based and has been used primarily in the areas of large-scale education, licensure and certification for processes such as Graduate Record Examinations (GRE) and Graduate Management Admission Tests (GMAT) (ETS, 2007). The only adaptive e-Assessment for employment purposes is the Armed Services Vocational Aptitude Battery (ASVAB) (ASVAB, 2007). The ASVAB assesses vocational abilities associated with a variety of positions in the military and the adaptive e-Assessment version has been implemented since 1982. SHLPreVisor (2011) is an adaptive e-Assessment centre which

delivers several hundred different adaptive e-Assessments for hiring and developing employees in organisations and industries.

ISSUES OF SUBJECT DOMAIN WITH IRT-BASED ADAPTIVE E-ASSESSMENTS

Little research has been conducted regarding the subject domain in which an adaptive e-Assessment can be applied. So far, adaptive e-Assessment has been widely applied in subject areas such as English, mathematics, logic, psychology, and computer science (ETS, 2001; ASVAB, 2007; Microsoft, 2011). The current adaptive approaches, especially IRT-based models, are mainly dependent on the form of items to be presented and the dimensionality of knowledge involved, rather than the subject domain. However, the application of adaptive e-Assessment has been restricted due to the limitation of auto marking. For example, it can be very difficult to apply adaptive e-Assessment in a subject where the main assessment form is writing. This is because there is great difficulty in implementing computerised auto marking for writing tasks.

BENEFITS AND LIMITATIONS OF IRT AND IRT-BASED ADAPTIVE E-ASSESSMENTS

IRT is the most common theoretic foundation for adaptive e-Assessments. The main benefits of IRT stem from the fact that this theory provides a foundation to achieve efficient assessments in terms of items used, time, and resources. The main benefits of IRT and IRT-based adaptive e-Assessments include item efficiency, feedback, time, flexibility in presentation forms, test-independent and sample-independent measurement, and better security.

The items in adaptive e-Assessment have better testing efficiency compared to the items used in paper-and-pencil assessments. Therefore, fewer items are needed in an adaptive e-Assessment to achieve the same measurement accuracy when compared to conventional assessments/e-Assessments. A typical paper-and-pencil assessment is created for a large group of candidates and therefore contains a large proportion of items with an average difficulty level. This is because a

large proportion of candidates are around the middle point on the ability scale. This has led to candidates in high-ability groups and low-ability groups having less chance of being assessed properly. Difficult items may cause candidates in low-ability groups to lose confidence in assessments, which in turn may affect the performance of these candidates. In addition, by administering items that are too easy for high-ability candidates, extraneous variables may be entered into their ability estimate, such as careless errors caused by boredom (Wainer *et al.*, 2000).

Another benefit of IRT and IRT-based adaptive e-Assessment is the immediate feedback for candidates (Wise and Plake, 1990; Barker, 2009). With conventional paper-and-pencil assessments, there is usually a time lag between assessment and feedback. As a result, most of these conventional assessments end up being a tool to assign scores to candidates with. Pedagogically this is not good enough since this practice does not allow candidates to obtain timely feedback to support their learning process.

Although it takes more time and effort to create an adaptive e-Assessment, it is more time efficient from the perspective of educationalists and candidates. In an adaptive e-Assessment, candidates tend to answer fewer questions compared to a fixed-form e-Assessment or paper-and-pencil assessment. Therefore the performances of candidates are less prone to the effects of fatigue. From the perspective of educationalists, adaptive e-Assessments are more time efficient as well. Educationalists do not need to worry about creating specific assessments that suit their students. All they need to do is select the right subject in an adaptive e-Assessment question bank and leave the question selection issue to the adaptive e-Assessment system. In addition, educationalists can also save time by giving the marking tasks for selection-type items and supply-type items to computers (Bennett, 1999).

Adaptive e-Assessments, as with other forms of computer-based assessments, have the advantage of presenting assessment items with more flexibility and creativity compared to conventional paper-and-pencil assessments (Parshall, Stewart & Ritter, 1996). Adaptive e-Assessments can

include pictures, movies, as well as other multimedia materials as assessment items (Parshall, Spray, Kalohn & Davey, 2002; Wainer et. al., 2000). These multimedia materials may help elicit positive attitudes from the candidates and motivate them to complete the assessment better (Parshall, Stewart & Ritter, 1996).

Other benefits of IRT and IRT-based adaptive e-Assessments include test-independent and sample-independent measurement and better security. IRT-based adaptive e-Assessments are sample-independent and test-independent due to the nature of IRT, which means the ability estimates of candidates measured with different questions are completely comparable. This means the ability estimates from IRT-based adaptive e-Assessments are invariable to the items and the group of candidates involved in the assessment. Therefore, the performances of candidates can be conveniently compared. On the other hand, IRT and IRT-based adaptive e-Assessments have improved security since the sequence of item administration is not fixed. It helps reduce the possibility of plagiarism and hence improves the reliability of measurement.

LIMITATIONS OF IRT AND IRT-BASED ADAPTIVE E-ASSESSMENT

Besides the aforementioned benefits of IRT and IRT-based adaptive e-Assessments, there are also some limitations associated with them. These limitations include the strict requirements in the application of IRT-based adaptive e-Assessments, the cost of setting up an IRT-based adaptive e-Assessment, and some unresolved issues in IRT-based adaptive e-Assessments.

Among all of the studies and applications on adaptive e-Assessments, IRT is the most commonly adopted underlying theoretic support due to its systematicness and soundness. However, the application of this type of adaptive assessment system is limited within large-scale organisations rather than a normal classroom environment (Frick, 1992; Olson, 1990; ETS, 2001; Taiwan Education Testing Centre, 2007). The main cause for this situation is the complexity and rigorous requirements in the implementation of IRT-based adaptive e-Assessments.

Because IRT makes strong assumptions about the assessment, to create an IRT-based adaptive e-Assessment one needs to select the right IRT model according to the type of question (dichotomous or polytomous) and the latent traits to be assessed (unidimensional or multidimensional). Then these questions need to be given away to a number of candidates in the form of a pilot test to collect observed responses (alternatively, these questions can be passed to experts in the subject area for parameter estimation in cases where there is only the difficulty parameter needed.). Subsequently, the values of parameters, which represent the psychometrical properties of the questions, need to be estimated and the goodness-of-fit index of each question needs to be calculated to ensure the right IRT model is used (Baker, 2001). Due to the aforementioned complex requirements of IRT-based adaptive e-Assessments, the cost of setting up such an e-Assessment can be high (Meijer & Nerling, 1999).

In additional, the shorter and more efficient characteristics of IRT have also caused some problems in testing. The three most noticeable issues are navigation of questions, content balancing and item exposure. The issue with navigation of questions in adaptive e-Assessments refers to the fact that examinees are usually not allowed to skip questions or change the answers to previous questions. Rudner (1998) pointed out that an examinee familiar with the working principles of an adaptive e-Assessment can firstly give wrong answers to the first few questions to bring down the difficulty level of the future questions and then go back to correct the answers to the previous questions. By using this strategy, an examinee can easily score highly in an adaptive e-Assessment. However, a more recent study conducted by Lilley and Barker (2004) indicated that allowing examinees to modify previously answered questions does not have a significant effect on the ability measurement provided the candidates do not intentionally exploit this vulnerability of adaptive e-Assessment.

Content balancing is a famous issue associated with adaptive e-Assessment. It concerns the assessment contents presented to the examinees with different proficiency levels (Kingsbury & Zara, 1991). An example to describe content balancing is language proficiency assessment for

foreign language learners. The proficiency of a language (reading, writing, listening and speaking) can be scaled on a single latent trait, but it covers different assessment contents and patterns. Because of educational pattern and environment, students from some countries tend to do better in reading and writing than listening and speaking in terms of foreign language study. This could result in students with lower proficiency being given more reading- and writing-related questions, while students with higher proficiency are given more listening- and speaking-related questions. Content balancing is research to ensure students with different proficiency levels receive a similar proportion of test contents.

Item exposure is another popular research topic in the adaptive assessment domain. Depending on the ability distribution of the examinees and the information structure of the question bank in an adaptive assessment, each question in the question bank has a different opportunity to be exposed to the examinees which is known as item exposure rate. Similar to content balancing, the over-exposure rate of some questions may be unwanted in some assessment settings. For example, a high-stakes examination whose question bank will be repeatedly used may not want some questions in its question bank to be disclosed to the students beforehand. In order to meet the requirement of these assessments, some exposure rate controlling methods have been proposed (Hetter & Sympson, 1997; Revuelta & Ponsada, 1998).

VALIDITY AND RELIABLITY OF ADAPTIVE E-ASSESSMENT

Although adaptive e-Assessments represent a more advanced assessment methodology, the validity and reliability issues are still the two main concerns when it comes to application, just like any other assessment methodology. Reliability refers to the extent to which an assessment accurately estimates the latent trait of the given individual. Validity is the subject-relatedness of the adaptive e-Assessment. In other words, it refers to whether the adaptive e-Assessment measures the right latent trait for the proficiency of the assessed subject area. Content and criterion-related validity are the two widely accepted methods for establishing validity (Fetzer *et al.*, 2011).

Content validity is the evaluation criterion for justifying the use of the assessment for the curriculum. It concerns the relatedness of the test content of the adaptive assessment to the requirements of the curriculum. For example, an adaptive e-Assessment for proficiency in a computer programming course can be validated by showing that the understanding of data structure, as measured by the adaptive e-Assessment, is a key knowledge point in the curriculum. The objective of the content validity is to demonstrate the correspondence between the curriculum and the competencies or tasks required by the adaptive e-Assessment (Fetzer *et al.*, 2011). The criterion-related validity indicator, as a complement to the content validity indicator, refers to the extent to which the measurement of an adaptive e-Assessment is related to the task or the organisation (Fetzer *et al.*, 2011).

The reliability of adaptive e-Assessment is the degree to which the measurement of an adaptive e-Assessment is free from error. In other words, the difference in scores among examinees should only be a reflection of the difference in skill or latent trait being measured rather than the functionality of the adaptive e-Assessment (Fetzer *et al.*, 2011). Due to the adaptive nature of adaptive e-Assessments, some indicators for the reliability of assessments based on CTT are no longer applicable to them. However, Standard Error of Measurement (SEM) is inherited to be the main indicator for the reliability of adaptive e-Assessment. For CTT-based assessments, the SEM obtained in the sample assessment is typically used for any given assessment. However, this single value is inappropriate for representing the SEM for all the assessments because the CTT-based assessments tend to be more reliable for measuring the candidates at the midpoint of the ability scale and less reliable for measuring the candidate at the extremes. In adaptive e-Assessments, the SEM is often used as a termination rule to ensure the same assessment reliability for every candidate. SEM was used as the termination rule in this research.

ITEM POOL SIZE AND PRECISION OF MEASUREMENT

An item pool or question bank is the source of questions in an adaptive e-Assessment. Traditionally, it is believed that an item pool with as small as 100 items could allow an adaptive e-Assessment to outperform a wide-range paper-and-pencil test in terms of measurement accuracy (Urry, 1977). However, most modern adaptive e-Assessment systems have item pools with 1,000 or more items (Wise and Kingsbury, 2000). This is because a large-size item pool increases the measurement accuracy of the adaptive e-Assessment under circumstances where constraints such as content constraints are imposed. A large-size item pool also improves the security of an adaptive e-Assessment since there is less chance of a candidate being presented the same item that someone else has attempted.

2.2.5 MEASUREMENT DECISION THEORY

IRT is not the only approach to adaptive e-Assessment. A number of adaptive approaches have been suggested as alternatives to IRT. A hierarchical content system proposed by Trentin (1997) starts an assessment with items of high difficulty to avoid candidates with high efficiency being presented low-difficulty items. Rudner (2001) proposes an adaptive assessment approach known as Measurement Decision Theory (MDT), which classifies examinees into finite ability groups based on statistical decision theory. Lutticke (2004) proposes an adaptive e-Assessment system in combination with a tutoring component for a computer science course. In the system, an incorrect response from a candidate would trigger a tutoring component and then the question would be presented to the candidate again until a correct response is provided. There are also adaptive e-Assessment systems based on if-then rules (Steven & Hesketh, 1999; Tzanavari *et al.*, 2004) and tree data structure (Kaburlasos, 2004; Cristea & Tuduce, 2005).

Amongst the above adaptive e-Assessment approaches, the MDT proposed by Rudner (2001) is another major approach to adaptive assessment after IRT. MDT has a sound theoretic foundation and is less demanding in terms of the preparation work required for implementing an adaptive e-Assessment. MDT is particularly useful in adaptive e-Assessments where there is a need to

categorise examinees into a finite number of discrete classes such as passed and failed. Unlike IRT, MDT does not concern itself with the distribution of examinees' abilities, the dimensionality of tested domain and the fit of data to the model. The only assumption in MDT is item independence, that is to say, an attempt to answer any one item will not affect the answers to other items in the same item bank. When implementing MDT, researchers need to have prior knowledge about the approximate proportion of masters among the examinee population and the conditional probability for examinees to respond correctly to each item in the item bank given their mastery states. This knowledge can be obtained through a simple pilot test with few examinees and items. Therefore, it is attractive in the situations where quick classification of examinees is needed.

An MDT-based test can be described with the following pseudocode:

1. Determine the possible mastery states for an examinee;

2. Create a calibrated item bank, which can be done via a pilot test;

3. Present the calibrated questions and obtain the examinee's response vector;

4. Form a decision about the examinee's mastery state.

Let K denote the number of possible mastery states, m_k denote different mastery states, z denote response vector, and D denote possible decisions. Given the following scenario, the judgement on whether an examinee masters a subject or not is to be made based on his responses to three items. In this case, there are two possible mastery states, that is, K = 2, and m_1 and m_2 are used to denote masters state and non-masters states respectively; there are only three items in the case, hence z comprises of z_1, z_2 and z_3, where $z_i \in (0,1)$. There are three possible decisions: the current examinee being a master, the current examinee being a non-master, and insufficient evidence for decision-making (test to be continued).

Testing starts with the proportion of the examinee population in each of the K categories and the proportion of the examinee population in each category that gives correct responses. The population proportions can be determined by the transformation or classification of the existing data, or alternatively, they can be derived from prior testing. Treat pilot test proportions as probabilities and let $P(m_k)$ and $P(z_n \mid m_k)$ be the notations:

$P(m_k)$ – the probability of a random examinee falling in the kth mastery category

$P(z_n \mid m_k)$ – the probability of response z_n occuring given the responder is in m_k

According to Bayes' theorem, the posterior mastery state of an examinee can be obtained using the priors:

$$P(m_k \mid z) = cP(z_n \mid m_k)P(m_k), \tag{2.38}$$

where c is a normalising constant and is given as follows:

$$c = \frac{1}{\sum_{i=1}^{K} P(z \mid m_k)P(m_k)}, \tag{2.39}$$

Assuming all the items contained in the test are independent of each other, or local independence,

$$P(z \mid m_k) = \prod_{i=1}^{N} P(z_i \mid m_k), \tag{2.40}$$

That is to say, the probability of the response vector can be obtained by calculating the product of the conditional probability of all the individual item responses.

To better illustrate this procedure, assume a simple case where there are only two mastery states and three dichotomous items. The prior probabilities of masters and non-masters among the

examinee group are 0.2 and 0.8 respectively, and the conditional probabilities for masters and non-masters to generate correct responses to the three items are shown as follows:

Items Mastery states	Item 1	Item 2	Item 3
Masters (m_1)	0.7	0.8	0.7
Non-masters (m_2)	0.2	0.4	0.3

Figure 2 - 7: conditional probabilities of generating correct responses

Assume an examinee's response vector z = [1,0,1], where 1 represents a correct response and 0 otherwise. The essence of the decision is to make a best judgement about the examinee's mastery state based on the available data. By formula (3), the probability for the examinee being a master is $P(z \mid m_1) = 0.7 \times (1-0.8) \times 0.7 = 0.098$, and the probability for the examinee being a non-master is $P(z \mid m_2) = 0.2 \times (1-0.4) \times 0.3 = 0.036$. Comparing the two conditional probabilities, a judgement can be made intuitionally that the examinee is more likely to be a master.

A sufficient condition for judgement to be made is the likelihood ratio $L(z) = \frac{p(z \mid m_2)}{p(z \mid m_1)}$, which for the example is $L(z) = 0.036 / 0.098 = 0.37$. This condition is sufficient because the ultimate decision rule can be regarded as a test comparing $L(z)$ against a reference value λ.

$$\begin{cases} d_2 - L(z) > \lambda \\ d_1 - L(z) < \lambda \end{cases} \qquad (2.41)$$

The value of λ is usually determined by the selected approaches and the cost of different types of classification error.

In general, the MDT is an excellent approach to adaptive e-Assessment on the conditions that - 1) there are only dichotomous questions involved; 2) the candidates are meant to be classified into

finite categories. Therefore, it suits the circumstances where there is a need to quickly obtain the categorical information about a candidate. For instance, it can be used to test whether a candidate has mastered a set of skills or not. Rudner (2001) concluded the advantages of the MDT approach as follows:

- generates accurate classifications of the mastery state for candidates
- can be integrated with a question bank
- is relatively simple in terms of implementation
- requires small-scale data from the pilot test
- can be applied to criterion-referenced tests
- can be extended to generate classifications on multiple skills
- can be used for sequential testing, and
- is relatively easy to explain to non-statisticians

However, in this research, MDT was not adopted due to relatively little research conducted in relation to it. For example, there is no literature on the detection and measurement of the bias generated. There is no literature to support the questions from one test to be used in another test either. Bearing in mind that there is more solid research literature to support every aspect of the IRT theory, MDT was adopted in the research.

2.3 WEB APPLICATION FRAMEWORK

"A framework is a set of prefabricated software building blocks or libraries which provide common functionalities that programmers can reuse, extend, or customise for specific computing tasks" (Taligent Inc, 1993). The purpose of frameworks is to allow software developers, rather than starting from scratch, to leverage a codebase which provides common logic, thus limiting the time required to build an application and reducing the possibility of introducing new bugs (Angelov *et al.*, 2006).

A well-known architecture pattern for creating web applications is the Model View Controller (MVC) (Singh and Johnson, 2001), which separates the user interface from the data model with business rules. The definitions and functions of MVC are as follows.

- **Model** - The model represents objects and properties of application which normally represent a real-world entity or process. This allows methods developed for real-world modelling applicable in web application development.
- **View** -The view acts as the interface to a model. It accesses enterprise data through the model and specifies how that data should be presented. It is the view's responsibility to maintain consistency in its presentation when the model changes. This can be achieved by using a push model, where the view registers itself with the model for change notifications, or a pull model, where the view is responsible for calling the model when it needs to retrieve the most current data.
- **Controller** – The controller is the infrastructure that defines the flow of the application. The controller translates interactions with the view into actions to be performed by the model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a web application, they appear as GET and POST HTTP requests. The actions performed by the model include activating business processes or changing the state of the model. Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view.

The application of the MVC architecture has the following advantages and disadvantages.

- **Re-use of model components.** The separation of model and view enables different views to use the same enterprise model. Therefore, it is easier to implement, test and maintain an enterprise application’s model components due to the fact that all access to the model is through these components.

- **Easier support for new types of clients.** A new type of client can be easily integrated into the existing enterprise application by implementing a new view and its controller logic.
- **Increased design complexity.** This design pattern may introduce some extra classes due to the separation of model, view, and controller.

This practice not only enables multiple views to use the same data model, which allows better support for multiple clients in implementation, testing, and maintenance, but also modularises the source code, promotes the reuse of code and allows multiple user interfaces as plug-ins. It is therefore recommended by many software engineers (Broemmer, 2003).

A large proportion of MVC frameworks are push-based (El-Bakry and Mastorakis, 2009). This architecture invokes the corresponding module to process the request, and then "push" the results to the user interface for rendering. For example, Struts, Struts 2 and Spring are representatives of push-based MVC presentation frameworks (Shan and Hua, 2006; McClintock, 2008; Pack, 2008). As a contrast to push-based architecture, another type of framework architecture is pull-based which enables the user interface to "pull" results from controllers. Pull-based architecture allows a single view to contact multiple controllers. A well-known example of a pull-based framework is Stripes (Fennell, 2009).

Another widely-used designed pattern for web application frameworks is the three-tier architecture. It is a client-server architecture which advocates the mutual independence of a user interface module, business logic module, and data storage and access modules (Gorton, 2011). This architecture not only inherits the advantages of modular software design; more importantly, it enables any of the three tiers to be modified independently as technology or requirements update. For instance, a change of database system would only affect the code in the data tier.

The three tiers in the three-tier architecture are commonly nominated as the Presentation tier, the Business Logic tier or the Application tier, and the Data tier, with their functionalities defined as follows. Presentation tier: as the topmost layer in the three-tier architecture, this layer is responsible

for displaying contents to the users. These contents could be assessment subjects, questions, student scores and anything else needed to inform the user. The Presentation tier serves other tiers by retrieving and outputting the information to the user. Business Logic tier (Application tier): this tier maintains the business rules within an application by implementing detailed operations. It communicates with the presentation tier and provides computational outcomes to the latter for outputting purpose. Data tier: this tier is responsible for the operations associated with databases such as database connection, database pool, and database management. The reason for making this tier independent and neutral from the other two tiers is because of the importance of databases in an application. In addition to the general advantages of the three-tier architecture, creating a tier exclusively for database manipulation improves database performance and the scalability of the application.

DISTRIBUTED COMPUTING AND WEB SERVICES

Distributed computing is the approach of using multiple autonomous computers so as to benefit from the collaborative effort within the network to achieve a common goal (Keidar, 2008). The computers within the network interact with each other through communication protocols. Distributed computing differs from parallel computing in that each computer in the network has its own private memory, while in parallel computing all the processors access a shared memory. Client-server and peer-to-peer are two typical architectures of distributed computing[11].

Client-server: this architecture is characterised by client machines contacting a server machine for data processing and then presenting it to the users (Berson, 1996). The client machines also capture user inputs and send them to the server machine for processing.

Peer-to-peer: in peer-to-peer architecture, each computer manages its own resources and there is not such a specific computer which is appointed to be the service provider (Schollmeier, 2002).

[11] Distributed computing is a vague and broad area. For more information please refer to Keidar, 2008,

Any computer in the network can serve as client or server and the responsibilities are shared among them. The peer-to-peer architecture has good scalability but consumes a lot of bandwidth (Schollmeier, 2002).

A web service is a typical application of distributed computing which is service oriented. It is difficult to give a complete and precise definition of service-oriented architecture. Jammes and Smit (2005) defined service-oriented architecture (SOA) as a set of architectural tenets for building autonomous yet interoperable systems. This definition is incomplete but embodies the two most important features of an SOA: autonomy and interoperability.

The basic unit in a service-oriented system is service. Autonomy means services are created independently of each other and they operate independently of their environment. Meanwhile, autonomy also means that the functionality offered by a service should be self-contained, that is to say, the functionality should be useful even if the service is independent of any high-level systems.

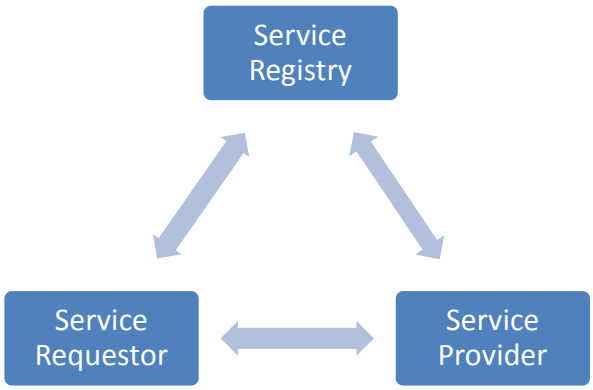


Figure 2 - 8: the working principle of web service architecture

A broad definition of web service is an application accessible to other applications over the Web (Sun Microsystems, 2006; Menasce and Almeida, 2001). This definition is so broad that it can refer to almost anything with a URL. In order to better distinguish web services from other web-based resources, the UDDI consortium (UDDI, 2001) provides a more precise definition which characterises web services as "self-contained, modular business applications that have open,

Internet-oriented, standards-based interfaces". The W3C (World Wide Web consortium) has further narrowed down the scope of web services as "a software application identified by a URI (Uniform Resource Identifier), whose interfaces and bindings are capable of being defined, described, and discovered as XML (Extensible Markup Language) artifacts. A web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols" (W3C, 2004). In the definition given by W3C, XML is included as a part of the standard solution. As a matter of fact, there are even more specific definitions for web services, with more standards forced in the implementation such as the one given by Webopedia, which strictly defines a web service as "a standardized way of integrating Web-based applications using the XML, SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language), and UDDI (Universal Description Discovery and Integration) open standards over an Internet protocol backbone. XML is used to tag the data; SOAP is the protocol used for exchanging structured data; WSDL is used for describing the services available, and UDDI is used for listing what services are available" (Jupitermedia, n.d.).

In a standard W3C standard web service, the three most important implementation protocols are SOAP, WSDL and UDDI. These three standards coordinate all the information exchange between service provider, service requester and service directory throughout the entire communication process. The role they play in the essential three-part communication in service-oriented architecture is depicted as follows:

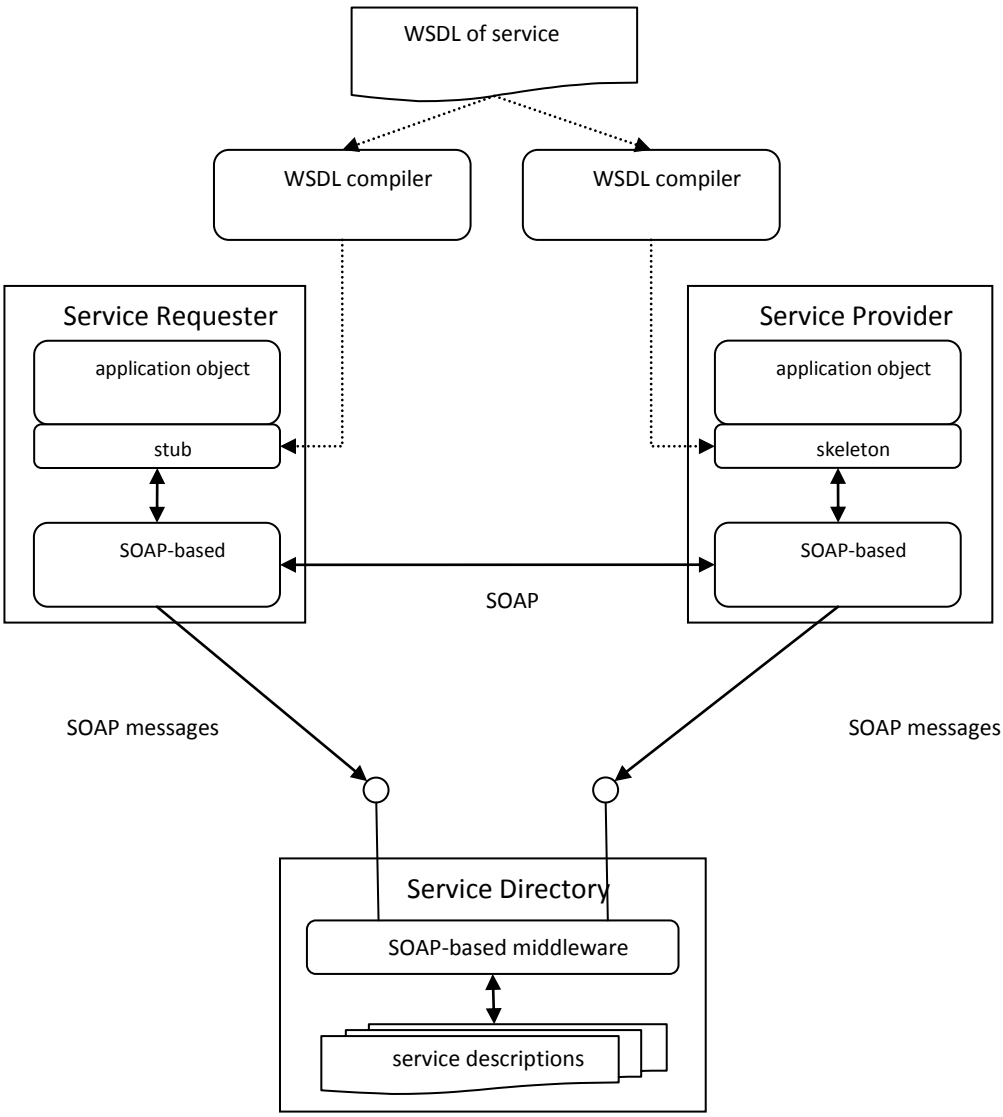


Figure 2 - 9: SOAP, WSDL, UDDI and their roles in the essential three-part communication in service-oriented architecture (adapted from Alonso *et al.*, 2004)

A FRAMEWORK FOR ADAPTIVE E-ASSESSMENT

A web application framework is a framework foundation designed to support the development of framework-compatible applications. It aims to minimise the overheads for the implementation of common tasks and the development of extended functions associated with the framework functionalities. For example, it can provide a library for database access, framework compatible

components development and so on. In this way, the designed framework enhances code reusability and reduces the effort and cost in application development. It is believed that a good web application framework should follow the common architecture patterns for web application development.

The literature review on adaptive e-Assessment in the former section of this chapter investigated various approaches to adaptive e-Assessment. Different approaches to adaptive e-Assessment or even different combinations of adaptive e-Assessment components often lead to different efficiencies in ability measurement. In consideration of this, a framework for adaptive e-Assessment should support a collection of different adaptive e-Assessment components as plug-ins so as to allow flexible combination or re-use of these components to save repeated work for researchers and developers. A web framework is preferable for implementing adaptive e-Assessment. It eliminates the geographical restriction of delivery of assessments. As the number of concurrent assessments and size of the question bank increases, the computational demand will increase considerably, especially when MIRT models are used (Chen, 2006). In conjunction with the seven characteristics of a good e-Learning system mentioned in chapter 1 (Khaskheli, 2004), these characteristics can be supported by the framework as follows:

- Interoperability: applications based on the framework should use a standard information description format such as XML.
- Reusability: components developed based on the framework can be easily used again in another project developed based on the same framework. This characteristic can be fulfilled via object-oriented programming and cross-language middle ware[12].
- Manageability: this issue is more related to the realisation of application rather than the framework.
- Accessibility: this characteristic can be fulfilled through a web framework.

[12] Please refer to the thrift framework in chapter 6.

- Durability: just like manageability, this issue is more related to the realisation of application rather than the framework.
- Scalability: this characteristic can be realised by implementing a framework with architecture similar to web service.
- Affordability: this characteristic can be realised via a framework based on distributed architecture.

2.4 SUMMARY

From the content point of view, this chapter reviewed the literature in two subject areas – assessment theories and web application frameworks. This chapter reviewed assessment theories starting from classical test theory, and then focused on the two main theorectical foundations in adaptive assessment – Item Response Theory and Measurement Decision Theory.

The adaptive e-Assessments implemented in this research were all based on the three-parameter logistic IRT model. Although IRT is not the only approach to adaptive e-Assessment, it has outstanding test efficiency compared to other adaptive e-Assessment approaches for the reason that it always presents the item with most information to the examinee (Lord, 1980; Weiss, 1983; Hambleton and Swaminathan, 1990; Wainer and Mislevy, 2000). IRT is also the most commonly used approach to adaptive e-Assessment and therefore it was used in this research. Considering the purpose of the research is to find out the performance of adaptive e-Assessment in the context of a higher education environment with small-scale candidates, the MIRT models were not appropriate to be employed in this research due to its demand for large-scale data in item calibration. The three-parameter logistic IRT model was designed for multiple-choice, single-answer questions, which is the majority type of question in the Cisco chapter tests. In order to avoid multiple latent traits being involved in the assessments, only the chapter 1 and chapter 2 tests in the Cisco course were used. Only these two chapter tests contain questions measuring the understanding of fundamental

concepts of networking. For the above reasons, the three-parameter logistic IRT model was chosen to be the adaptive model in the research.

The item calibration and ability estimation are two important procedures in adaptive e-Assessment. As discussed in this chapter, there are different approaches to these two procedures and each of them has both advantages and disadvantages. The statistical item calibration method was adopted in this research because it is believed that it better reflects the psychometric properties of the items from the students' point of view. The maximum likelihood estimation was used in the research to adopt the examinees' abilities due to the fact that it converges faster than EAP and MAP. In view of the small size of the question bank in the research, it requires an estimation method which can quickly converge at one ability point before the questions are depleted. Different combinations of item calibration methods and ability estimation methods have direct impacts on the efficiency of adaptive e-Assessment. However, they were not exhaustively experimented on in this research due to the limitations of the research's resources. Therefore, a framework for adaptive e-Assessment was designed and implemented to support the integration of different adaptive e-Assessment methods for the future research purpose.

The MIRT models and MDT approach are particularly useful in some other circumstances and they reflect the variety and complexity of adaptive e-Assessment. For example, the MDT approach may be the best option when there is only a need to evaluate the mastery state of candidates. Therefore, the idea of designing and implementing a universal framework for adaptive e-Assessment was brought up to extensively contribute to the adaptive e-Assessment research and application communities. The latter section of this chapter non-exhaustively reviewed MVC-based web frameworks and distributed web frameworks. The next chapter will discuss the methodology used in this research.

CHAPTER 3 RESEARCH METHODOLOGY

3.1 THE PURPOSE OF THIS RESEARCH

As stated in chapter 1, the aim of this research is to firstly investigate the performance of adaptive e-Assessment in a normal class environment in higher education, and secondly create a scalable and affordable framework for the development of adaptive e-Assessment. The current chapter will respectively describe the methodology adopted in the performance study for the proposed adaptive e-Assessment strategy and the framework construction.

3.2 ADAPTIVE E-ASSESSMENT ALGORITHM AND ITS PRAGMATIC PERFORMANCE

As mentioned in the section that discussed the scope of the research, this part of the research is a selective study based on the unidimensional three-parameter IRT model rather than the exhaustive study of the whole family of IRT. Thus this part of the research will establish an adaptive e-Assessment algorithm and apply it in a pragmatic environment to study its performance. The term pragmatic here refers to the conditions of the research that are easily achievable in higher education, which includes 60 students and 48 multiple-choice questions; it also denotes real-life cases rather than the ideal cases simulated in a laboratory environment, which refer to the fact that the responses of the students were collected in a real examination[13].

[13] The Cisco fixed-form e-Assessment was used to collect student responses in a real examination environment and the responses (real-data) collected were used to evaluate the performance of the implemented adaptive e-Assessment. Please refer to Chapter 4 for more details.

In this research, one adaptive e-Assessment algorithm was implemented based on the Item Response Theory (IRT). IRT is a body of psychometrical theory which formulates the probability for an examinee to give a certain type of answer as the function of the examinee's abilities and the parameters specific to the item[14]. According to the invariant psychometric property of IRT, IRT-based adaptive assessment makes it possible to compare the abilities of the examinees who attempt different assessments. Adaptive e-Assessment systems built upon IRT can administer different items to examinees based on their abilities, thereby reducing test length and improving measurement reliability.

Bearing in mind that all of the items available to this study were multiple-choice items with dichotomous scored answers, a decision on the selection of a dichotomous IRT model was made in the research. There are mainly three IRT models developed for dichotomous scored items: the One-Parameter Logistic Model (1-PL), the Two-Parameter Logistic Model (2-PL), and the Three-Parameter Logistic Model (3-PL) (Lord, 1980; Hambleton and Swaminathan, 1990; Wainer and Mislevy, 2000). The 1-PL model, which is also known as the Rasch Model in honour of its developer George Rasch (1901 – 1980), assumes that the psychometric property of an item is only affected by its difficulty. The 2-PL model assumes that the psychometric property of an item is determined by both difficulty and discrimination. The 3-PL model assumes there are three elements that affect the psychometric property of an item: difficulty, discrimination and guessing. The 3-PL IRT model was adopted in this research. The reasons for the selection of the 3-PL IRT model rather than the 1-PL IRT model and the 2-PL IRT model are as follows:

1) The 3-PL IRT model is widely used. Many adaptive e-Assessment systems such as SIETTE (Conejo *et al.*, 2004) and Barker's auto feedback system (Barker, 2009) adopted this model in their adaptive testing.

[14] The term "item" here refers to a single question.

2) The 3-PL IRT is designed for dichotomous questions with the probability of being guessed correctly. The questions in the Cisco chapter test used in the research were all multiple-choice questions. The 1PL model and the 2PL IRT model do not reflect the impact of the guessing factor.
3) The 3-PL IRT model is relatively simple amongst various models but offers adequate parameters to describe a unidimensional question. The relatively simple characteristic of the model makes the calibration of questions relatively easy to perform, i.e., a smaller number of observed responses is required.

The research methodology involved in this part of the study covers six aspects: the procedures of the adaptive e-Assessment strategy used in this part of the research, evaluation methods, examinee source, question source, item calibration, and Monte Carlo simulation. The procedures of the adaptive e-Assessment strategy used in this part of the research can be listed as follows:

1. Give an initial estimate of the examinee's ability.
2. Find the best question to ask.
3. Present the question and wait for a response.
4. Re-estimate the examinee's ability.
5. Check if the termination rule is met: if yes, terminate the assessment; if not, repeat procedure 2 to 4 until the termination rule is met.

The above 5-step procedures adopted in the proposed adaptive e-Assessment strategy are straight-forward. However, there are some techniques behind these procedures which need to be emphasised. Firstly, Fisher's information matrix was used as the measurement criterion to find out the best question. The Fisher information is an indicator for the amount of information that an observable random variable X carries about an undetermined parameter θ upon which the likelihood function of θ depends (Lehmann and Casella, 1998). Secondly, re-estimating the examinee's ability was done through Newton-Raphson method to continuously approach the true

ability of the examinee. The Newton-Raphson method is an iterative procedure to find the approximate root for $f(x) = 0$ (Wang and Chen, 2004). Thirdly, the termination rule was based on either of the following two conditions:

1. The system runs out of questions.
2. The measurement reliability reaches a threshold, which is 0.9 in this research. The 0.9 or 90% reliability means 90 out of 100 adaptive assessments will give the same measurement on the examinee's ability. The measurement reliability was chosen as the termination rule in this research to guarantee that every examinee was measured at the same reliability level, though they were given different questions.

The performance of an adaptive e-Assessment strategy is usually evaluated using response sets and personal data. Literature review shows that the three possible methods of obtaining response sets and personal data in an adaptive assessment are live testing, real-data simulation and Monte Carlo simulation.

The live-testing approach administrates real test items to live examinees (Weiss and Bock, 1983). To conduct live testing, researchers can obtain a calibrated item bank using IRT parameter estimation software such as XCALIBRE (Gierl and Ackerman, 1996) or Conquest (Wu *et al.*, 1998). These software artefacts usually work with some specific IRT models and researchers need to write programmes on their own for those IRT models without ready-made parameter estimation software. Additionally, researchers need to design applications or use tools like FastTest (Wainer *et al*., 2000) to adaptively deliver items to examinees. As live-testing uses live test administration and live examinees, it is therefore expensive and time-consuming. Another problem is that the responses from real examinees may have an indeterminate degree of noise due to the possible inconsistency of human beings' behaviour. However, it is often used to confirm the results of simulations. Therefore, it is common practice to run a number of simulations prior to the implementation of live testing.

Real-data simulation is also known as post hoc simulation. It is based on the real examinees' responses and real item bank obtained from conventional tests. The objective of real-data simulation is to maximally reduce the length of tests through re-administering items adaptively, in the premise of maintaining psychometrical properties of the scores. In a real-data simulation, the item response data available from a conventional test are used as input for parameter estimation. Then the calibrated item bank is used to adaptively administer items to examinees and the test ends when the termination criterion is met. The termination criterion can be based on measurement precision or test length. In the end, the simulations which result in similar ability estimates as the conventional test will be studied to find out the maximum length that can be saved by using an adaptive assessment.

Monte Carlo simulation is a class of computational algorithm which relies on repeatedly generated random samples to compute the results (Kalos and Whitlock, 2008). It can be employed to evaluate the potential performance of an approach to adaptive assessment with real or hypothetical item banks and various populations. It differs from real-data simulation in that simulated examinees are generated by the simulation process to have a specified ability distribution, and it can use either a real item bank or simulated item bank with parameters conforming to a specified distribution. Monte Carlo simulation was used in this research to simulate some conditions that were unable to be satisfied such as unlimited numbers of students and items. In Monte Carlo simulation, the probabilities for a simulated examinee to generate different responses to an item are calculated according to the IRT model adopted in the experiment. Currently, there is software that can automate Monte Carlo simulation. The most widely used programming languages to implement Monte Carlo simulation are FORTRAN (Ian and Sleightholme, 2006*)* and MATLAB (Gilat, 2004).

Considering the purposes of this part of the research are (i) to study the performance of a three-parameter-IRT-model-based adaptive e-Assessment strategy in a pragmatic environment, and (ii) the optimised item pool size for the above strategy in a theoretically ideal environment, both the real-data evaluation and the Monte Carlo simulation were adopted in the research. The reason for

using Monte Carlo simulation is the insufficient question numbers.

In the real-data simulation, the question pool and examinee source were from Cisco's course chapter test questions and the real students registered on the Cisco IT Essentials II course provided by Buckinghamshire New University. Cisco is one of the biggest technology companies in the world that provides networking certificate courses at various levels (Behrens *et al.*, 2004). Buckinghamshire New University is one institution in the UK which provides training for Cisco courses. In the Monte Carlo simulation, randomly generated virtual questions and students were used. The item calibration procedure in the real-data simulation was conducted using MLE and LSE. All of the above six aspects about this part of the research will be given further clarification in chapter 4.

3.3 APPROACH TO THE DESIGN OF FRAMEWORK FOR E-ASSESSMENT SYSTEMS

To minimise the modification and maintenance difficulty of software modules, many e-Assessment systems follow the three-tier architecture design pattern. The design of the framework in this research began with a summary of the basic elements of typical adaptive e-Assessment systems and an attempt to fundamentally support the common requirements of adaptive e-Assessment systems.

A typical adaptive e-Assessment system follows the three-tier architecture from top to bottom consisting of a presentation tier, business logic tier, and a data tier. The synoptical model for such an e-Assessment system can be depicted as follows:

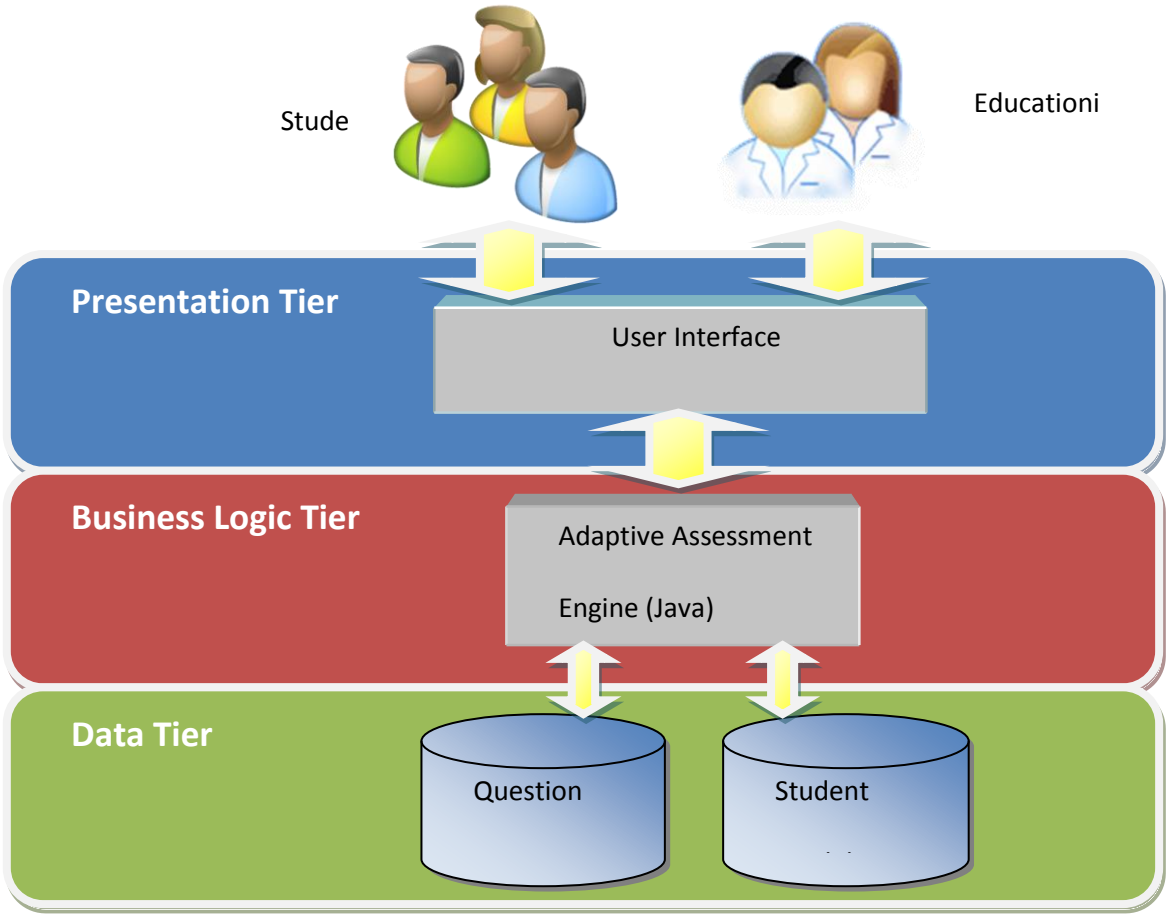


Figure 3- 1: the synoptical model of three-tier adaptive e-Assessment systems

In the above model, the presentation tier is responsible for presenting the system user interface to the user. This part of the system may include displaying a dashboard, capturing user requests, presenting questions and options to the user and receiving user responses. These display-related modules may be realised via client-end programmes or web browsers such as Microsoft Internet Explorer (IE), Google Chrome and Mozilla FireFox.

The business logic tier contains the core of the system – the adaptive assessment engine of the system. It acts as a middle layer to transfer data between the presentation tier and the data tier. It is also responsible for performing detailed processes inside the system. This tier verifies user operations received from the presentation tier and provides computational results to the presentation tier. The adaptive e-Assessment engine, which is the core of the system, determines

which question is to be presented, when to terminate the test and the estimated student ability based on the actions captured for the presentation tier.

Finally, the data tier keeps student profiles, published assessments and classified questions to be presented to the user. This tier receives queries relayed by the business logic tier, implements them and returns the results to the business logic tier. The backend database server used to store assessments, questions and student information can be changed as needed.

The above system inherits the typical advantages from the three-tier software architecture and modular programming. In this manner, it reduces the overheads in system maintenance and upgrades. However, this architecture still lacks the capability of tackling two common yet important issues in adaptive e-Assessment. Firstly, due to the reason that every psychometric model has its scope, adaptive e-Assessment developers may need to customise their own models in some cases. This requires an e-Assessment framework to have the flexibility to accommodate customised e-Assessment models. In the meantime, it also requires the framework to support flexible definitions of question unit structure, i.e. the properties that a question can have, to meet the requirements of corresponding adaptive algorithms. Secondly, some complex adaptive e-Assessment models are demanding on computational power. Therefore, it is ideal to spread the computational load across a number of servers via distributed architecture.

To provide a solution to the above two issues and support the characteristics of good e-Assessment application (Khaskheli, 2004), the framework needs to address these issues from the angle of architecture. In general, the framework should provide the libraries to support the development related to the following four categories of activities: user agent construction, communication between distributed user agents, database management, file management and utilities. Amongst them, the database management libraries include the support for the flexible question unit structure definition. As for the user interface libraries, they were not included in the framework for the simple reason that there have already been many well-established presentation frameworks such as

Stripes, Struts and Struts 2. E-Assessment developers can choose one of the presentation frameworks to develop their user interface, while using the framework provided in this research to develop backend functionalities such as business logic and data management.

With reference to the 7 characteristics of good e-Assessment application (Khaskheli, 2004) and the two aforementioned issues in adaptive e-Assessment, the following aspects were considered in the design and implementation of the framework.

- Interoperability: information description standards such as XML and Thrift were complied with in the implementation of the framework. Thrift (Slee *et al*., 2007) is a lightweight cross-language framework, originally developed at Facebook, which supports efficient and reliable cross-language communication among computer programming languages such as C++, Java, Python, PHP, Ruby, Perl, and C#. The implementation of the Thrift framework combines a language-neutral software stack implemented across numerous programming languages and an associated code generation engine that transforms a simple interface and data definition language into client and server remote procedure call libraries (Slee *et al*., 2007). XML was used to store and retrieve the framework-related configuration files. Important and frequently used data structures such as question and test class were saved in the language-neutral Thrift format so that they could be reused or imported into projects written in different programming languages.
- Reusability: the complete framework was written in Java language, which is a pure object-oriented language (Sun, 2006). With Java programming language and Thrift, the components constructed based on the framework can be easily reused.
- Manageability: this issue is more related to the realisation of application rather than the architecture of framework, therefore it was not taken into consideration in the design and implementation of the framework.
- Accessibility: this is a web framework which provides access free from geographic and time restrictions.

- Durability: just as manageability, this issue is also more related to the realisation of application rather than the framework.
- Scalability: the framework is based on the peer-to-peer distributed architecture and therefore inherits its excellent scalability. The good scalability of the framework allows different adaptive e-Assessment components to be flexibly plugged into or removed from an established application.
- Affordability: framework is based on the peer-to-peer distributed architecture. Therefore it has the potential[15] to provide cost-effective solutions to increased demand for computational power.

The framework was also based on the above seven characteristics. In the implementation of the framework, unit testing and integration testing were performed. The unit testing was performed using JUnit and the integration testing was performed using sample application. The JUnit is an open-source unit testing the framework for Java programming language which tests whether an expected output is generated given a specific function and input (JUnit, n.d.).

[15] The word potential was used here because distributed architecture provides the fundamental support for collaborative computation (Schollmeier, 2002). However, the functionalities of monitoring computational load and automatically coordinating computational load were not implemented in the current framework.

CHAPTER 4 EXPERIMENTS AND DATA ANALYSIS

4.1 INTRODUCTION

This chapter describes the process and results of the study on the performance of adaptive e-Assessment in a normal class environment in higher education, which is the first aim of this research.

4.2 EXPERIMENT ENVIRONMENT AND DESIGN

This experiment investigated the feasibility and performance of Item-Response-Theory-models-based adaptive e-Assessment with small-size training sets for the item calibration procedure. Real student responses and questions from the Cisco IT Essentials II course offered by Buckinghamshire New University were used in this experiment as data source. Cisco is one of the biggest technology companies in the world that provides networking certificate courses (Behrens *et al.*, 2004). IT Essentials II is a Cisco course designed to introduce multi-user and multi-tasking network operating systems such as Linux, Windows 2000, NT and XP (Cisco, 2008).

The first two chapter tests of Cisco IT Essentials II course were used as the question source, each of which contains 24 mixed-type questions, most of which being multiple-choice questions and a couple being multiple-answer questions. These questions by Cisco were originally used in fixed-form computer-based tests with each question carrying a score of one point. A student's proficiency with the contents of the chapter was indicated by a score from 0 to 24. All of the 48

questions were concept questions with 3 of them being multiple-answer questions and the other 45 being multiple-choice questions.

In the research, 60 real students' examination records, which included the examination records of 48 male students and 12 female students, were used to analyse and compare the performance of fixed-form e-Assessment and Item-Response-Theory-models-based adaptive e-Assessment. In the pilot test, all of the multiple-choice questions were administered to the students via the Cisco online assessment system[16] and the students' responses to every question were recorded for use in the later real-data simulation. The aforementioned questions were firstly administered to the students using the fixed-form e-Assessment originally designed by Cisco and the student scores obtained were then recorded as the control group data. Subsequently the same questions were administered to the same students in a real-data adaptive e-Assessment simulation and the student ability estimates obtained were recorded as the measurement by the adaptive e-Assessment algorithm. 60 students were considered in this research as the size of two common classes.

The two main aims of testing are Estimation and/or Classification: to estimate a person's competence and/or classify a person into the right proficiency group (Eggen, 2007). Therefore the first evaluation criterion for the performance of the two testing patterns was the precision of ability estimation. The second evaluation criterion was the length of tests, i.e. the number of questions used in each testing method to establish a student's ability.

Three experiments were designed to study the performance of the two testing patterns.

Experiment 1: The 60 students were evenly divided into two groups, group A and B, with each group consisting of 30 students. These 60 students were recruited from two classes on the Cisco IT Essentials course at Buckinghamshire New University. Class one had 34 students and class two had

[16] The Cisco online assessment system was used as the platform from which to provide fixed-form e-Assessments.

26 students. The last four students, by the descending order of student numbers from class one, were transferred into class two to balance out the number of students in these two groups. Then the remaining 30 students in class one were denoted as group A, and the newly formed class two was denoted as group B. The final distributions of gender in these two groups were, respectively, group A - 27 male students and 3 female students; and group B - 26 male students and 4 female students. The two chapter tests were respectively administered to these two groups of students to generate four groups of experimental data: group A students' test results on the chapter 1 test, group B students' test results on the chapter 1 test, group A students' test results on the chapter 2 test, and group B students' test results on the chapter 2 test. These data were collected together to enable a comprehensive comparison of test errors, test lengths and difference of results.

Experiment 2: The 60 students were also evenly grouped into group A and group B in exactly the same way as they were in experiment 1. However, the two chapter tests were integrated into one test to generate two groups of experimental data: group A students' test results on the integrated test, and group B students' test results on the integrated test. These data were also collected together for comprehensive comparisons as in experiment 1. The purpose of integrating two tests is to study the impact of the question bank size on the precision of ability estimation.

Experiment 3: The Monte Carlo simulation method was used in this experiment to generate virtual students and questions in the computerised adaptive test for evaluation purposes. This experiment was designed to study the impact of question bank size on the adaptive e-Assessment algorithm by simulation.

DATA COLLECTION AND ANALYSIS METHODS

This section describes how the data were collected and analysed, as well as what tools were used in this procedure. The raw data were collected by administering the two Cisco online chapter tests (each test containing 24 multiple-choice questions) to the 60 students in a real examination environment in November 2007. All of the 60 candidates were gathered in a computer lab and each

student was given a computer with internet access so that he/she could log into the Cisco online academy to complete the online chapter tests. The two Cisco online chapter tests are fixed-form e-Assessments which consist of 45 multiple-choice, single-answer questions and 3 multiple-choice, multiple-answer questions (1 in the chapter 1 test and 2 in the chapter 1 test). Each of these questions carries a score of one point and the candidate needs to check all the correct options for multiple-answer questions to get the score of one point. No partial mark was given for the multiple-answer questions.

In order to reduce the bias with human subjects, the 60 candidates were all recruited from the classes on the Cisco IT Essentials II course at Buckinghamshire New University. These 60 students were from the same intake and they had just completed their studies on the first two chapters of the Cisco IT Essentials II course by the time they attended the online chapter tests. They were given instructions about Cisco's online chapter tests and support regarding the operation and user interface of the Cisco system was given throughout the examination.

Cisco's online chapter tests automatically score the students' marks and record their responses to the questions. These recorded marks and responses were regarded as raw data in the experiments. They were then entered into MATLAB v7.04 beta for data analysis. MATLAB is a programming language mainly for mathematical computations and simulations (MATLAB, 2011). It has a great number of built-in mathematical functions which can greatly simply maths-related programming. For example, the following MATLAB code shows the Maximum Likelihood Estimation for the difficulty level of one question in the experiment.

mleb.m

```
function [] = mleb (nq)
%this function estimates the parameter b of
%1PL IRT model using maximum likelihood estimation
%nq is the number of questions

data = DataLoader('Chap1.dat'); %read data from .dat file
[m, n] = size(data);
studmark = sum(data');%calculate students' marks
```

```
studabil = zeros(1, m);
for i = 1 : m %convert marks into abilities
   if (studmark(i)<=7)
      studabil(i)=-2;
   elseif (studmark(i)<=9)
      studabil(i)=-1;
   elseif (studmark(i)<=11)
      studabil(i)=0;
   elseif (studmark(i)<=16)
      studabil(i)=1;
   else
      studabil(i)=2;
   end
end
%syms b; %difficulty level
b = -3:.1:3;
sigma = 0;
for i = 1 : m %expand derivative of log-likelihood function
   theta = studabil(i);
   if (data(i,nq)==1)
      sigma = sigma + 17/10./(1+exp(-17/10*theta+17/10*b)).*exp(-17/10*theta+17/10*b);
   elseif (data(i,nq)==0)
      sigma = sigma + 1 - 17/10./(1+exp(-17/10*theta+17/10*b)).*exp(-17/10*theta+17/10*b);
   end
end
%b= solve(sigma, b); %find the solution for b
plot (b, sigma); %plot sigma against b
```

Text Box 4 - 1: MATLAB code for ML estimation for difficulty level of a question in the experiments

The item calibrations, chi-square goodness-of-fit indices, and real-data simulations in these experiments were all implemented in MATLAB.

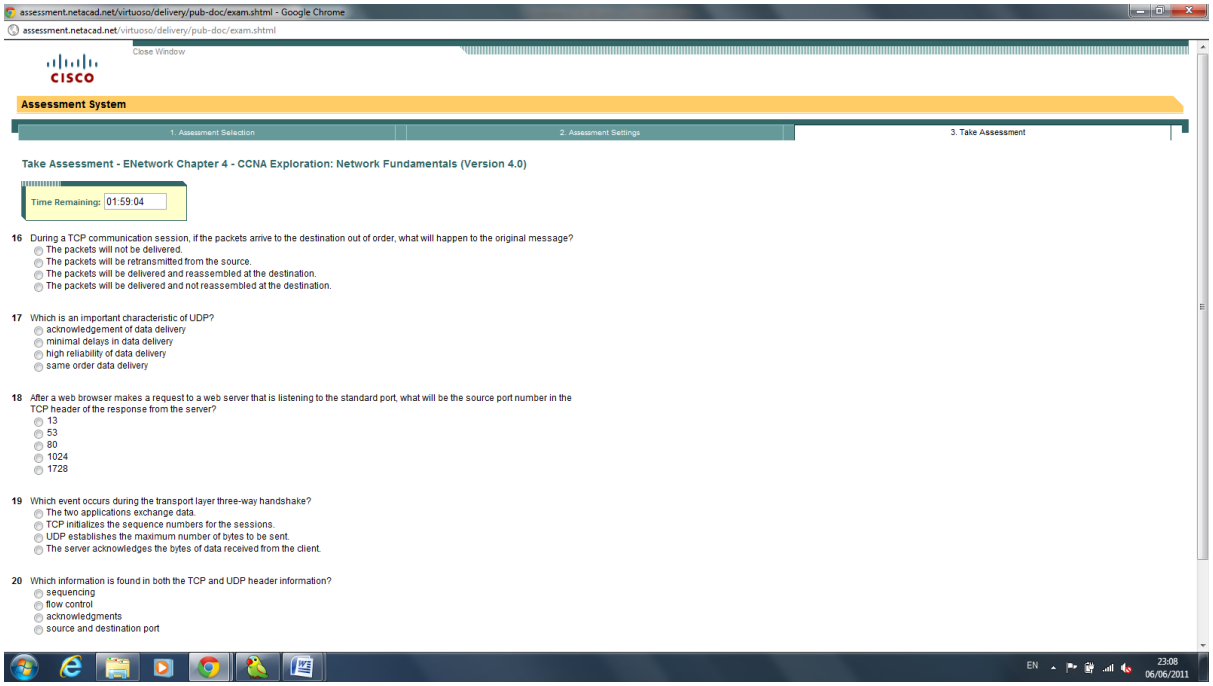


Figure 4- 1: a sample of Cisco's online chapter tests

4.2.1 SETTINGS AND IMPLEMENTATION OF EXPERIMENT 1 AND EXPERIMENT 2

In all of the fixed-form computer-based tests involved in the research, the questions were delivered to the students via the Cisco online testing platform and the results were recorded afterwards. The adaptive e-Assessments in the research required more complex settings which included three basic elements: the IRT model, the item calibration procedure, and the ability estimation strategy.

Considering all of the questions involved are dichotomous items and they are merely testing one latent trait – the understanding of concepts – the Three-Parameter Logistic (3PL) IRT Model (Birnbaum, 1968) was chosen to be the IRT model to support the adaptive e-Assessment experiments in the research. The model was used to calibrate the Cisco questions and obtain student ability estimation. The feature of the 3PL IRT Model can be described by its Item Response Function (IRF), which is given as follows:

$$p_{i,j}(\theta_j) = c_i + (1-c_i)\frac{e^{1.7a_i(\theta_j - b_i)}}{1+e^{1.7a_i(\theta_j - b_i)}}, \quad (4.1)$$

where $P_{i,j}$ (θ_j) is the probability of examinee j answering item i correctly; θ_j is the capability of examinee j; a_i is the discrimination index of item i; b_i is the difficulty parameter of item i; c_i is the guessing factor of item i; and 1.7 is the constant used to approximate the normal ogive function using the logistic function (Weiss and Yoes, 1991)

In the item calibration procedure of the adaptive e-Assessment, the Least Squares Estimation (LSE) and Maximum Likelihood Estimation (MLE) were used respectively to estimate the parameters for each question based on the observed student responses. Then the Chi-square goodness-of-fit index was employed to examine the model-fit between the observed data and the 3PL IRT model. An adaptive assessment strategy was established in the real-data simulation stage to infer each student's capability. All the responses adopted in this stage were extracted from the recorded real student responses.

The purpose of item calibration is to find out the values of parameters a, b and c in the equation (6.1) for each question in the question bank. As parameters θ, a, b and c are correlated with each other, their values should be estimated simultaneously. The specific measure is to list a group of conditional equations and then solve them for θ, a, b and c. Take LSE method, for example. Firstly the objective function is given as follows:

$$F(a_i, b_i, c_i) = \sum_{j=1}^{m} (p_{i,j}(\theta_j) - X_{i,j})^2, \quad (4.2)$$

where a_i, b_i, c_i are the parameters a, b and c of item i; m is the number of students; $P_{i,j}$ (θ_j) is the IRF for student j and item i; θ_j is the capability of student j; and $X_{i,j}$ is the observed response to item i from student j, which has value 1 if the response is correct or 0 otherwise. Then the observed student responses should be substituted into the above objective function to generate the

conditional equation. Subsequently the partial derivative for each parameter is computed to find out the numerical values that minimise the objective function. If the MLE method is employed, the following objective function is used:

$$L(a_i, b_i, c_i) = \prod_{j=1}^{m} p_{i,j}(\theta_j)^{\sum_{j=1}^{m} X_{i,j}} (1 - p_{i,j}(\theta_j))^{(m - \sum_{j=1}^{m} X_{i,j})} \tag{4.3}$$

where the definitions of parameters are the same as they are in equation 6.2. Subsequently the log-likelihood function should be worked out to simplify the computation before substituting observed responses into the objective function. In the last step, the partial derivative method is also used to find out the numerical values that maximise the objective function.

Insofar as this research is concerned, the small size of observed student performances made it difficult to estimate the values for four parameters simultaneously. Therefore a separate parameter estimation method was adopted to reduce the demand for training data sets. This parameter estimation method firstly determined the value of ability parameter θ for every student, and then substituted the values of θ into the objective function to estimate the values of parameter a, b and c for every question.

The determination of the values of ability parameter θ was based on the observed student performance. An approximate θ in {-3, -2, -1, 0, 1, 2, 3} was given to a student according to the student's overall score on 24 questions (experiment 1) or 48 questions (experiment 2). The purpose of this setting was twofold. Firstly, it integrated the scattered individual scores into 7 ability groups which made the calculation of the observed proportion of correct response at a certain capability point possible. The observed probability of correct response at a capability point can hence be calculated as

$$P(\theta_j) = \frac{r_j}{m_j}, \qquad (4.4)$$

where $P(\theta_j)$ is the observed proportion of correct response for capability group j; r_j is the observed number of correct responses from capability group j; and m_j is the total number of students in ability group j. Secondly, it made the range of parameter θ fall in between a symmetric value interval [-3, +3], which is a typical setting for θ in practice (Baker, 2001). According to credit level descriptors for higher education (SEEC, 2010), 7 ability groups were obtained and the students were then assigned to the ability groups according to their true scores in experiment 1 and experiment 2. The corresponding relationship among the 7 ability groups, the 24-point system in experiment 1, the 48-point system in experiment 2, the centesimal system and the distribution of students in each category are listed in the following table:

Table 4- 1: corresponding relation between ability parameter and true score in different scoring systems

Ability parameter θ	-3	-2	-1	0	+1	+2	+3
Corresponding mark interval in experiment 1 (24-point system)	[0, 4]	[5, 7]	[8, 9]	[10, 13]	[14, 16]	[17, 21]	[22, 24]
Corresponding mark interval in experiment 2 (48-point system)	[0, 9]	[10, 14]	[15, 19]	[20, 26]	[27, 34]	[35, 43]	[44, 48]
Counterpoint in centesimal grade system	[0, 19]	[20, 29]	[30, 39]	[40, 54]	[55, 69]	[70, 89]	[90, 100]

Subsequently, the values of parameter a and c were also determined separately in the research, which left difficulty level parameter b as the only indeterminate parameter needed to be resolved using LSE and MLE. In view of the fact that all the 48 questions are concept questions, it is assumed that they have the same discrimination index (parameter a) which was set as 1 in the research. The guessing factor c was defined as the probability of picking up a correct answer randomly with knowing the body of a question. For example, a multiple-answer question with 3

correct answers out of 4 options has the probability of C_4^3 being guessed. Finally, the two values of parameter b were estimated by LSE and MLE respectively.

In order to test how the student response data fit the 3PL model under the above environmental settings, the traditional chi-square goodness-of-fit index was computed for each question and for two values of parameter b. The traditional chi-square goodness-of-fit computation formula is as follows (Baker, 2001):

$$\chi^2 = \sum_{i=1}^{K} \frac{(O_i - E_i)^2}{E_i}, \qquad (4.5)$$

where K is the number of ability groups (7 in this research); O_i is the observed proportion of correct responses to the item from group i, which was calculated in the research as: the number of students giving a correct response to the question in the ability group / the number of students in the ability group; E_i is the expected proportion of correct responses to the item from group I, which was calculated based on the IRF of the 3PL IRT model given the parameters of the question and the ability group.

In the calculation of the chi-square goodness-of-fit index, the responses from group A students were used to calibrate the questions and then these calibrated questions were used to measure the abilities of the students from group B. The purpose of separately using these two groups of students was to keep these two procedures (the item calibration procedure and the ability measurement procedure) independent so as to avoid data interference. For each question, there were two different values of the chi-square goodness-of-fit index depending on if MLE or LSE was used to estimate parameter b. Therefore, there were originally two groups of chi-square goodness-of-fit indices and the parameter estimates by LSE method were eventually accepted due to their better goodness-of-fit chi-square indices. The chi-square goodness-of-fit indices for the 48 questions are listed in the table below.

Table 4- 2: the chi-square goodness-of-fit indices for the questions in the Cisco chapter tests

Chapter 1 questions			Chapter 2 questions		
Question NO.	chi-square goodness-of-fit index based on MLE	chi-square goodness-of-fit index based on LSE	Question NO.	chi-square goodness-of-fit index based on MLE	chi-square goodness-of-fit index based on LSE
1	1.344	1.122	1	1.185	0.996
2	1.712	1.535	2	1.673	1.437
3	1.175	1.033	3	1.083	1.225
4	1.66	1.471	4	2.24	2.102
5	1.822	1.61	5	1.765	1.568
6	1.385	1.042	6	1.866	1.618
7	2.042	1.71	7	1.714	1.62
8	1.272	1.179	8	1.802	1.775
9	1.125	0.975	9	1.294	1.325
10	1.67	1.34	10	1.488	1.417
11	1.912	1.802	11	1.59	1.306
12	2.011	1.618	12	1.342	1.298
13	1.214	1.083	13	1.762	1.68
14	1.563	1.475	14	1.442	1.445
15	1.732	1.552	15	1.676	1.572
16	1.672	1.645	16	1.583	1.48
17	1.592	1.63	17	1.89	1.812
18	1.432	1.096	18	1.76	1.298
19	1.308	1.371	19	1.657	1.38
20	1.825	1.603	20	1.149	1.102
21	1.677	1.578	21	1.158	1.045
22	1.966	1.67	22	1.626	1.602
23	2.12	1.58	23	1.608	1.597
24	1.624	1.246	24	1.472	1.38

The lower critical value of chi-square distribution at significance level $\alpha = 0.05$ with 6 degrees of freedom (there were 7 ability groups in the research), which is 1.635, was adopted as a benchmark. The chi-square goodness-of-fit tests in the above table showed that the LSE method generally produced better-fit parameter b estimations than MLE in this case. This is because LSE is less restricting and hence has relatively better performance on small sample size data, while the performance of MLE relies more on the sample size and therefore tends to be biased with small sample size data (Pratt, 1976). The results also revealed that there were 4 questions in either chapter test (questions 7, 11, 16 and 22 in the chapter 1 test and questions 4, 8, 13 and 17 in the chapter 2 test) whose chi-square goodness-of-fit indices were greater than 1.635, even by the LSE method. This may have been caused by the undersized set of sample responses or the nature of these questions which tend to baffle students with high proficiency. These 4 questions were not removed from the question bank due to the fact that the size of the question bank would be far too

small without these 4 questions. It could also reflect the size-sensitivity of the item calibration procedure.

The adaptive testing strategy employed in the research was designed to estimate student ability quickly and accurately. The logic of the ability estimation method was to infer the ability that most likely generated the student responses collected in the process of testing. Let X denote the response set, θ denote the unknown ability, and this problem can be converted into an MLE problem, that is, to find out the parameter θ that maximises the probability of X . The likelihood function here can be denoted as:

$$L(\theta|X) = f(X|\theta) \tag{4.6}$$

In a case where all the questions a student attempted were dichotomous (i.e. every attempted question only accepted either of two mutually independent responses such as correct or incorrect), the above likelihood function could be further expanded. Let θ denote this student's ability, $P_i(\theta)$ denote the probability for the student to give a correct response to question i, and x_i denote the student's response to question i where x_i =1 if the response is correct or x_i = 0 otherwise. Then, for m questions the student attempted, the likelihood function can be written as:

$$L(\theta|x_1,...,x_m) = \prod_{i=1}^{m} p_i(\theta)^{x_i}(1-p_i(\theta))^{(1-x_i)} \tag{4.7}$$

The $P_i(\theta)$ in the equation (4.7) is where dichotomous IRT Models fit in. Dichotomous IRT Models are psychometric models that describe the frequency of the correct response given a dichotomous item and a student. The Item Response Function (IRF) of the 3PL IRT model (4.1) was then substituted into the equation (4.7) to form the specific objective function for solving the parameter θ.

All the above procedures so far have been describing the basic logic about student ability and the underlying IRT model. However, the above method is not optimised to guarantee fast convergence of ability estimates. Therefore, instead of directly estimating the true ability based on a set of responses, a step-by-step approximating estimation method was introduced in the research. This method firstly gives an initial ability estimate for the current student, then it selects the best question based on this ability estimate to obtain a single response, and then it updates the ability estimate for the student. This process is reiterated until the confidence level of the estimation reaches a threshold.

The question selection criterion used in the research, i.e. the adaptive rule, is based on the Fisher Information. The Fisher Information is an indicator of the amount of information that an observable random variable X carries about an undetermined parameter θ upon which the likelihood function of θ depends. It can be written as follows (Lehmann and Casella, 1998):

$$I(\theta) = -E[\frac{\partial^2}{\partial \theta^2} \ln f(x|\theta)] \tag{4.8}$$

The $f(x|\theta)$ in the equation (4.8) is the Probability Density Function (PDF) of X. As far as this research is concerned, the $f(x|\theta)$ is defined as follows:

$$f(x|\theta) = p^x(1-p(\theta))^{(1-x)}, x = 0,1 \tag{4.9}$$

Substituting (4.9) into (4.8) gives equation (4.10):

$$I(\theta) = \frac{[p'(\theta)]^2}{p(\theta)[1-p(\theta)]} \tag{4.10}$$

Once the ability estimate for the current student was updated, the equation (4.10) was used to compute the $I(\theta)$ for every available question and the question with the highest $I(\theta)$ was chosen

as the best question to be presented to the student. The student response to the selected question was then used to update the ability estimate. The Newton-Raphson method was adopted to update the ability estimate to guarantee better successive approximations to the root. Let $F(\theta)$ denote the first derivative of the equation (4.7), then the purpose of using the Newton-Raphson method is to find the θ that makes $F(\theta) = 0$ so as to maximise the equation (4.7). Thereby the following equations can be obtained:

$$F(\theta) = \frac{\partial L(\theta | x_1, ..., x_m)}{\partial \theta} = 0 \tag{4.11}$$

$$\theta_{k+1} = \theta_k - \frac{F(\theta)}{F'(\theta)}, \tag{4.12}$$

Substituting (4.12) into (4.11) gives equation (4.13).

$$\theta_{k+1} = \theta_k - \frac{\sum \frac{[x_i - p_i(\theta)]p'(\theta)}{p(\theta)[1-p(\theta)]}}{\sum \frac{[p'(\theta)]^2}{p(\theta)[1-p(\theta)]}}, x = 0,1 \tag{4.13}$$

The equation (4.13) was used in the research to obtain an increasingly close approximation to the root of the equation (4.11).

The ability estimation procedure terminated when there were no unused questions available or the reliability (R) reached a certain threshold. The threshold for reliability in this research was set to 0.90 and the student abilities were assumed to conform to normal distribution ranging from -3 to +3 with a Standard Deviation (SD) of 1. The Standard Error (SE) can be calculated as:

$$SE = SD\sqrt{1-R} = 0.32 \tag{4.14}$$

Therefore reaching reliability over 0.90 can be interpreted as achieving an SE under 0.32. According to the definitions, the Test Information (TI) is the sum of question information and the question information is the reciprocal of the variance with which θ is estimated.

$$TI(\theta) = \sum I(\theta) \tag{4.15}$$

$$SE = \sqrt{Var(X)} = \frac{1}{\sqrt{TI(\theta)}} \tag{4.16}$$

From equation (6.14), (6.15) and (6.16) it can be inferred that a TI(θ) greater than 9.77 will lead to an SE smaller than 0.32. In this research, the termination rule of achieving reliability over 0.90 was interpreted as achieving a TI(θ) greater than 9.77.

4.2.2 THE RESULTS AND ANALYSIS OF THE EXPERIMENT 1

In the item calibration procedure of experiment 1, group A students' responses were used as the observed responses for the adaptive test for group B students; group B students' responses were used as the observed responses for the adaptive test for group A students. In this manner, the data set used to calibrate the questions is independent from the data set used in real-data simulation. It hence improved the reliability of the results of the computerised adaptive test. The results of this experiment are summarised in table 4-3.

Table 4- 3: comparison of the adaptive e-Assessment and the Cisco fixed-form assessment in experiment 1

		Chapter 1 Test			Chapter 2 Test		
Group	**Student No.**	Scores by fixed-form e-Assessment (24 point grading system)	Scores by adaptive e-Assessment (converted to 24 point grading system)	No. of questions used by adaptive e-Assessment	Scores by fixed-form e-Assessment (24 point grading system)	Scores by adaptive e-Assessment (converted to 24 point grading system)	No. of questions used by adaptive e-Assessment
A	1	6	8	16	11	10	19
	2	6	10	12	10	10	15
	3	10	13	19	6	9	20
	4	10	12	24	13	14	22
	5	10	9	16	14	12	20
	6	11	11	22	8	7	21
	7	11	8	19	20	16	17

	8	12	11	21	16	13	20
	9	12	15	18	24	24	20
	10	12	12	19	14	13	19
	11	12	14	20	13	15	20
	12	13	19	16	12	11	19
	13	16	21	19	16	18	19
	14	16	20	17	19	17	20
	15	16	15	22	9	10	21
	16	17	21	19	22	19	18
	17	18	18	20	14	16	18
	18	19	21	18	17	16	20
	19	19	20	23	23	22	22
	20	19	23	19	15	17	17
	21	20	14	18	18	19	20
	22	20	18	20	17	15	16
	23	21	17	17	15	14	20
	24	21	22	19	22	22	19
	25	21	22	19	20	21	19
	26	21	17	20	17	15	20
	27	20	23	19	19	21	19
	28	22	23	19	20	20	19
	29	22	14	20	16	17	17
	30	21	23	19	18	17	19
	31	22	13	18	24	24	20
	32	23	23	19	20	18	18
	33	23	22	16	17	18	19
	34	24	24	20	20	21	20
	35	6	9	16	10	14	16
	36	10	12	12	19	15	14
	37	10	11	19	7	8	20
	38	13	11	20	12	10	20
	39	12	12	16	10	11	18
	40	14	13	22	15	13	20
	41	15	12	19	12	12	19
	42	15	16	20	14	12	21
	43	16	15	20	10	11	16
	44	15	14	19	12	13	19
B	45	14	11	20	11	9	22
	46	15	16	19	14	12	20
	47	16	18	19	19	17	18
	48	17	18	20	15	15	18
	49	17	15	20	13	14	20
	50	18	17	19	16	15	19
	51	20	21	17	17	14	15
	52	20	22	19	23	20	20
	53	20	20	20	18	17	20
	54	20	18	19	22	22	24
	55	22	19	20	21	20	16
	56	22	20	24	20	17	20
	57	24	24	20	23	22	18
	58	24	24	19	22	20	23
	59	16	15	19	19	17	17
	60	12	13	20	14	13	22

The ability estimation by the adaptive e-Assessment system returned to values ranging from -3 to +3, while the student scores obtained from the Cisco fixed-form e-Assessment belonged to a 24-point grading system. In order to place the two ability notations on the same scale, the formula below was adopted to convert student scores from a 24-point grading system to the ability notation system in the adaptive e-Assessment:

$$\theta_{-3,3} = (\theta_{24pt} - 12)/4, \quad (4.17)$$

where $\theta_{-3,3}$ is the ability range from -3 to +3 obtained from the adaptive e-Assessment, and $\theta_{24\,pt}$ is the score in the 24-point grading system. For example, if an examinee's score in the 24-point grading system is 24, then his/her ability in the adaptive e-Assessment system is denoted as (24 - 12)/4 = 3. If an examinee's score in the adaptive e-Assessment system is denoted as -3, then his/her ability in the 24-point grading system will be (-3) x 4 + 12 = 0.

After placing the ability estimates obtained from both forms of test on the same scale, the precision comparison was established based on the objective function shown as follows:

$$E(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{m}(p_{i,j}(\theta_j) - X_{i,j})^2, \quad (4.18)$$

where $p_{i,j}(\theta_j)$ is the expected probability of student j answering item i correctly; $X_{i,j}$ is an observed response from student j to item i, which has the dyadic Boolean value 1 if the response is correct or 0 otherwise; m is the total number of students; n is the total number of questions. The testing model which generated less error is deemed to have better precision. Owing to the insufficient number of questions in this experiment, an information-driven termination criterion was adopted which asked the system to stop the test when none of the unused questions could provide item information greater than 0.05 for the current ability estimate. The item information is determined by the gradient of the Item Characteristic Curve for the question at the current ability estimate. The questions that provide information less than 0.05 means the Item Characteristic Curves at the current ability estimate is really flat. In other words, these questions are not good at distinguishing the students with similar abilities around the current ability estimate.

The comparison result indicated that the adaptive e-Assessment reduced the total error (E(θ)) by 29.3% compared to the fixed-form e-Assessments.

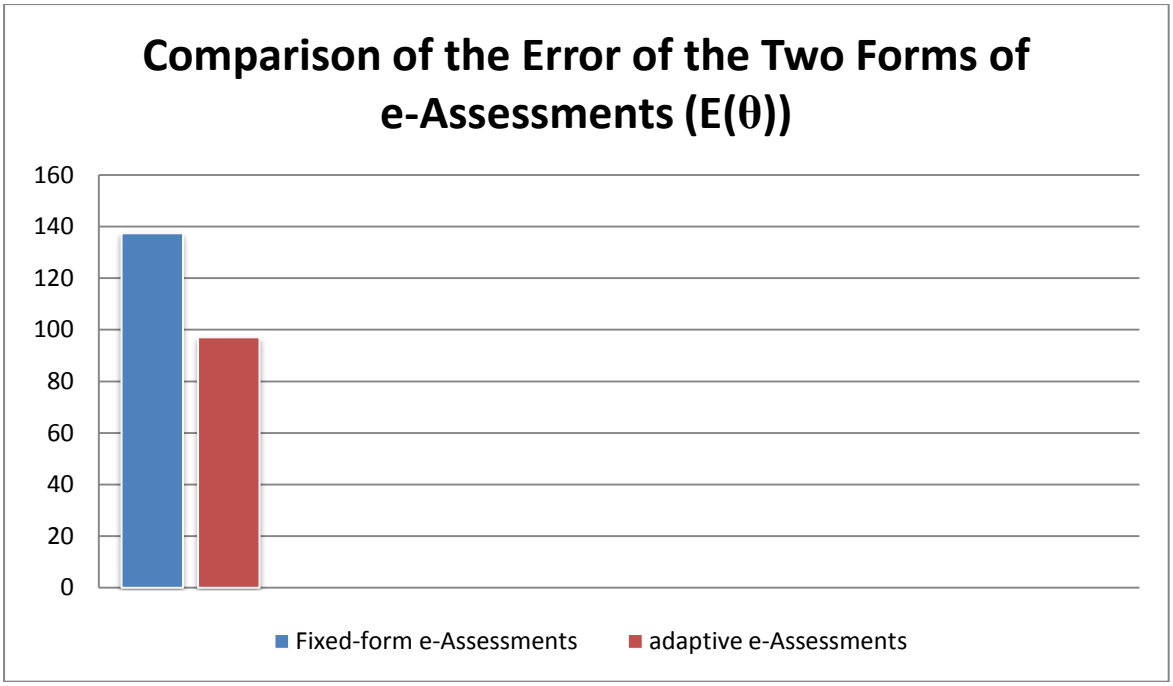


Figure 4- 2: comparison of the Error of the Two Forms of e-Assessments with 24 questions (E(θ))

The comparison results also indicated that, while achieving less total error, the adaptive e-Assessments also reduced, on average, the number of questions used by 20.8% compared to the fixed-form e-Assessments.

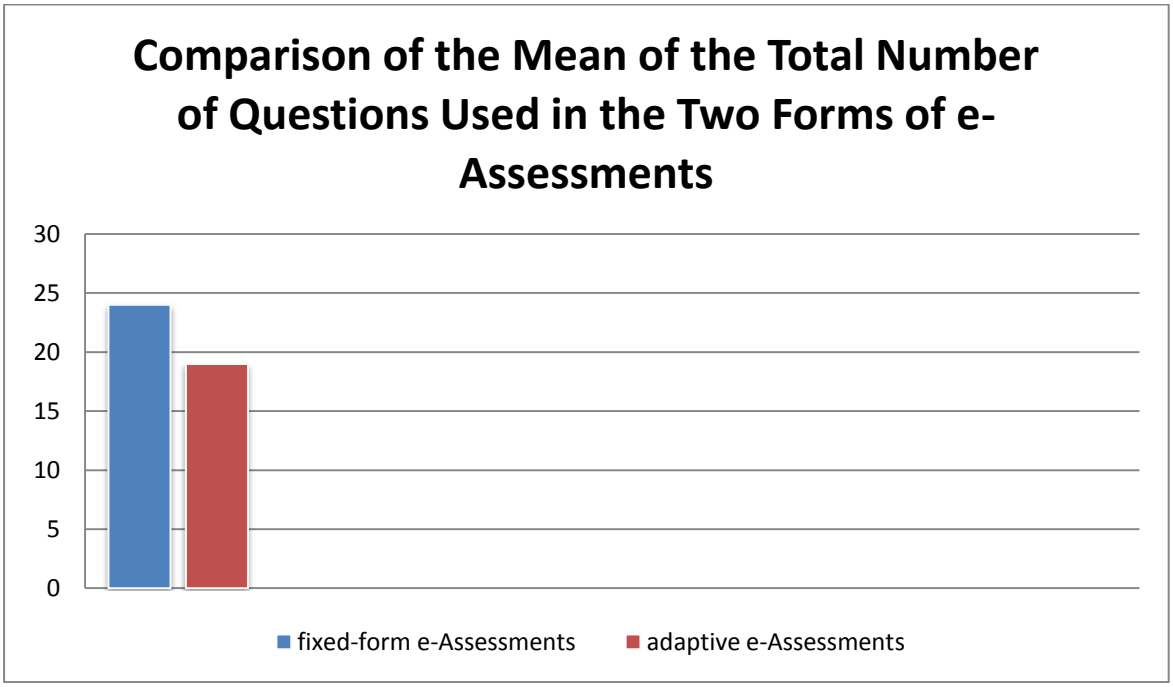


Figure 4- 3: comparison of the Mean of the Total Number of Questions Used in the Two Forms of e-Assessments with 24 questions

Subsequently, the statistical BIAS and Root Mean Square Error (RMSE) were also calculated to indicate the difference between the student ability estimates by the fixed-form e-Assessment and the proposed adaptive e-Assessment.

$$BIAS(\hat{\theta}_i) = \frac{\sum_{i=1}^{n}(\hat{\theta}_i - \theta_i)}{n}, \quad (4.19)$$

where θ_i is the score of student i obtained in the fixed-form e-Assessment, and $\hat{\theta}_i$ is student i's score equivalence in the 24-point system estimated by the adaptive e-Assessment.

$$RMSE(\hat{\theta}_i) = \sqrt{\frac{\sum_{i=1}^{n}(\hat{\theta}_i - \theta_i)^2}{n}}, \quad (4.20)$$

where θ_i and $\hat{\theta}_i$ have the same definitions as in equation (6.19) .

The results of BIAS and RMSE computation were concluded in the following table:

Table 4- 4: the BIAS and RMSE of the student ability estimates under the 24-point scoring notation by two e-Assessment methods

BIAS	RMSE
-0.25	2.38

The above results indicate that both the BIAS and RMSE are small. The consistency of the results suggests that the overall difference between the measurements yielded by two different e-Assessment methods is relatively small.

Three student's t-tests were carried out to compare the significant level of the difference between the assessments. The first student's t test was designed to measure whether the results of Cisco's

chapter 1 test from the two forms of assessment methods were significantly different. The second student's t test was designed to measure whether the results of Cisco's chapter 2 test from the two forms of assessment methods were significantly different. The third student's t test was designed to measure whether the results from the two chapter tests via adaptive assessment were significantly different, which would be discussed in experiment 2. The two-tails independent t-test was used in the research.

Student's t-test A: this test studies whether the test results of chapter 1 acquired from two different forms of e-Assessments are significantly different. In this t test, sample size n = 60; the degree of freedom v = 60 -2 = 58. Let μ_0 be the mean of the scores obtained from fixed-form e-Assessment, and μ_1 be the mean of the scores obtained from adaptive e-Assessment. The hypothesis test can be written as follows:

Null hypothesis $H_0 : \mu_1 = \mu_0$

Alternative hypothesis $H_1 : \mu_1 \neq \mu_0$

$t = 0.954$

Let $\alpha = 0.05$. 0.954 is smaller than the critical value $t_{0.05}$（58）= 1.671, therefore accept H_0. The difference between the two groups of results is insignificant.

Student's t test B: this test studies whether the test results of chapter 2 acquired from two different forms of e-Assessments are significantly different. The conditions and hypothesis are the same as the previous test.

Null hypothesis $H_0 : \mu_1 = \mu_0$

Alternative hypothesis $H_1 : \mu_1 \neq \mu_0$

$t = 0.491$

Let $\alpha = 0.05$. 0.491 is smaller than the critical value $t_{0.05}(58) = 1.671$, therefore accept H_0. The difference between the two groups of results is insignificant.

4.2.3 THE RESULTS AND ANALYSIS OF EXPERIMENT 2

In this experiment, the size of the question bank was increased to 48. As for the fixed-form e-Assessment, the number of total questions used was 48, which was the sum of the questions from the original Cisco chapter 1 and chapter 2 tests. The total score for these two tests was therefore 48. The score of an examinee by the Cisco fixed-form e-Assessment is therefore the sum of his/her scores in the chapter 1 test and the chapter 2 test. As in experiment 1, the precision and test length were compared for the adaptive e-Assessment and the fixed-form e-Assessment. The results were shown as below:

Table 4- 5: comparison of the adaptive e-Assessment and the Cisco fixed-form assessment in experiment 2

Combined Chapter Test				
Group	**Student No.**	Scores by fixed-form e-Assessment (48 point grading system)	Scores by adaptive e-Assessment (converted to 48 point grading system)	No. of questions used by adaptive e-Assessment
A	1	17	**15**	22
	2	16	32	20
	3	16	27	25
	4	23	32	30
	5	24	33	28
	6	19	23	27
	7	31	26	28
	8	28	41	26
	9	36	30	23
	10	26	33	24
	11	25	20	25
	12	25	18	25
	13	32	37	24
	14	35	42	22
	15	25	18	27
	16	39	35	23
	17	32	23	29
	18	36	42	23
	19	42	40	28
	20	34	37	24
	21	38	22	23
	22	37	31	25
	23	36	29	26
	24	43	45	24
	25	41	40	24
	26	38	34	30
	27	39	32	31
	28	42	35	21
	29	38	32	25

	30	39	30	24
B	31	46	42	23
	32	43	38	24
	33	40	44	26
	34	44	37	25
	35	16	29	22
	36	29	26	17
	37	17	32	24
	38	25	34	25
	39	22	16	32
	40	29	30	27
	41	27	21	24
	42	29	19	25
	43	26	20	25
	44	27	32	24
	45	25	28	27
	46	29	38	24
	47	35	31	24
	48	32	22	26
	49	30	40	23
	50	34	39	24
	51	37	35	25
	52	43	40	25
	53	38	32	31
	54	42	36	22
	55	43	40	27
	56	42	46	24
	57	47	46	25
	58	46	39	23
	59	35	25	28
	60	26	27	24

As in experiment 1, the ability estimation by the adaptive e-Assessment system still returned to values ranging from -3 to +3, while the student scores obtained from the Cisco fixed-form e-Assessment belonged to a 48-point grading system in this experiment. In order to place the two ability notations on the same scale, the formula below was adopted to convert student scores from a 48-point grading system to the ability notation system in the adaptive e-Assessment:

$$\theta_{-3,3} = (\theta_{48pt} - 24)/8, \quad (4.21)$$

where $\theta_{-3,3}$ is the ability ranged from -3 to +3 obtained from the adaptive e-Assessment and θ_{48pt} is the score in the 48-point grading system. For example, if an examinee's score in the 48-point grading system is 48, then his/her ability in the adaptive e-Assessment system is denoted as (48 - 24) / 8 = 3. If an examinee's score in the adaptive e-Assessment system is denoted as -3, then his/her ability in the 48-point grading system will be (-3) x 8 + 24 = 0.

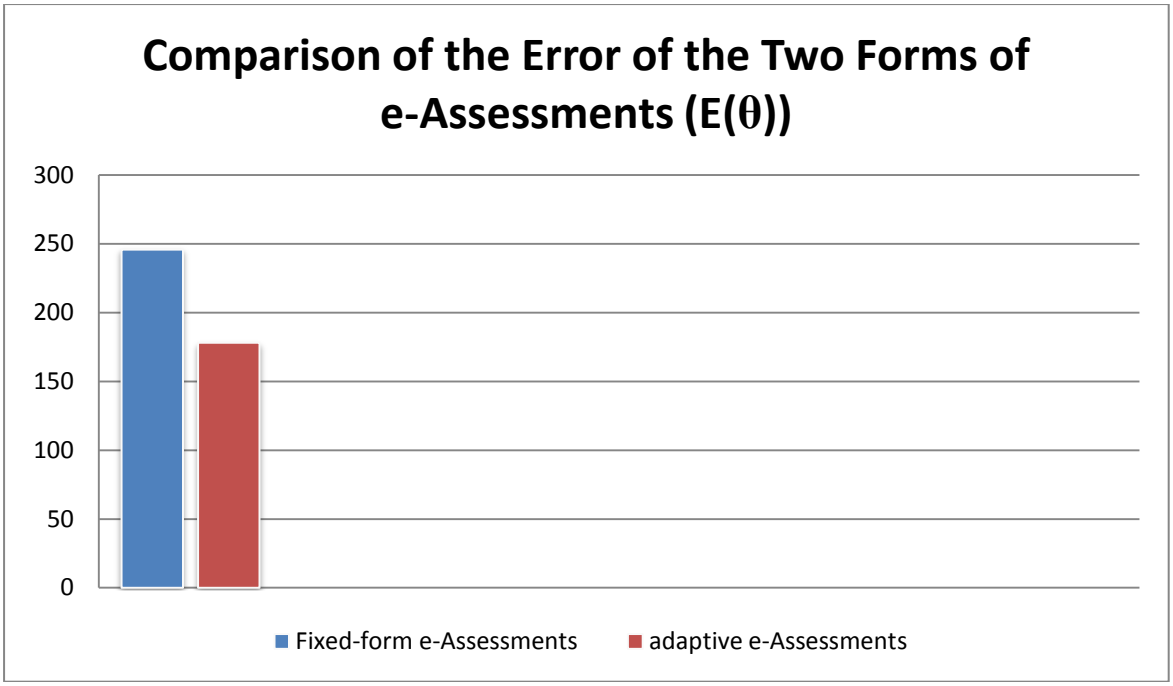


Figure 4- 4: comparison of the Error of the Two Forms of e-Assessments (E(θ))

This time the comparison results indicated that the adaptive e-Assessment reduced the total error by 27.5% compared to fixed-form e-Assessment.

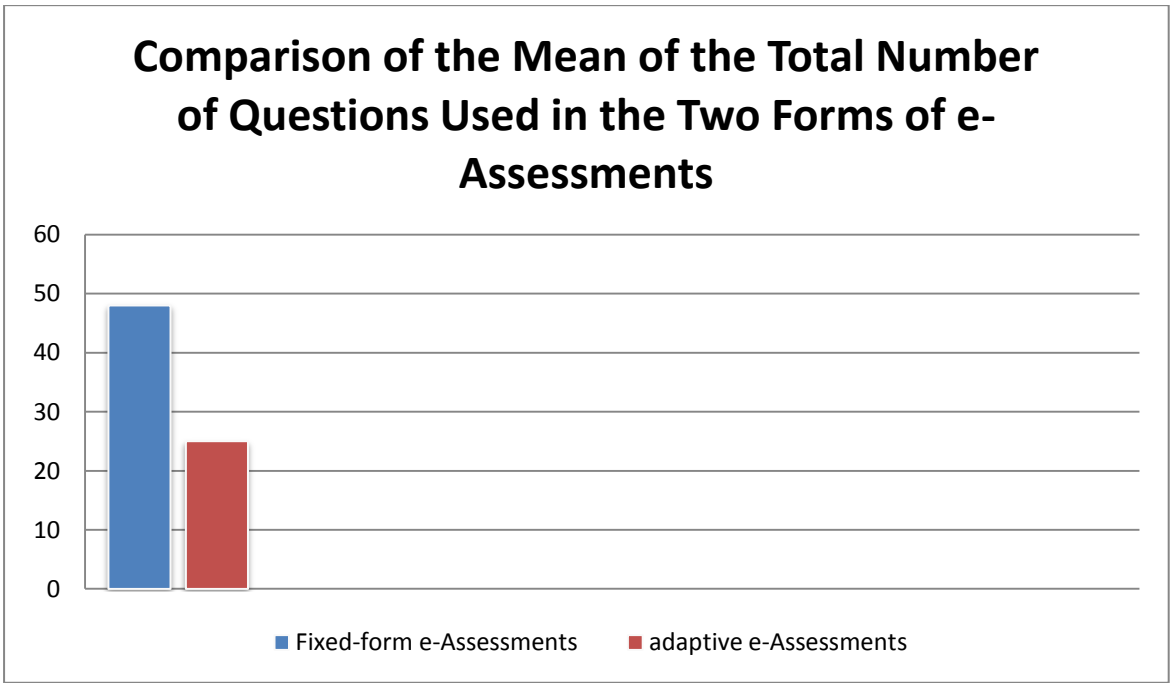


Figure 4- 5: comparison of the Mean of the Total Number of Questions Used in the Two Forms of e-Assessments with 48 questions

The comparison results also indicated that, while achieving less total error, the adaptive e-Assessments also reduced, on average, the number of questions used by 47.9% compared to the fixed-form e-Assessments.

As in experiment 1, the statistical BIAS and Root Mean Square Error (RMSE) were also calculated to indicate the difference between the student ability estimates by two e-Assessment methods.

Table 4- 6: the BIAS and RMSE of the student ability estimates under the 48-point scoring notation by two e-Assessment methods

BIAS	RMSE
-0.633	7.273

Student's t-test C: this test studies whether the results from the two chapter tests obtained via adaptive assessment are significantly different. In this t test, sample size n = 60 while the degree of freedom v = 60 -2 = 58. Let μ_0 be the mean of the scores of the chapter 1 test, and μ_1 be the mean of the scores of the chapter 2 test. The hypothesis test can be written as follows:

Null hypothesis $H_0 : \mu_1 = \mu_0$

Alternative hypothesis $H_1: \mu_1 \neq \mu_0$

$t = 0.676$

Let $\alpha = 0.05$. 0.676 is smaller than the critical value $t_{0.05}$（58）= 1.671, therefore accept H_0. The difference between the two groups of results is again insignificant.

4.2.4 THE IMPLEMENTATION, RESULTS, AND ANALYSIS OF EXPERIMENT 3

As mentioned earlier, the aim of the experiment was to study the impact of question bank size on the adaptive e-Assessment algorithm by simulation. The Monte Carlo simulation was used due to

the difficulty of obtaining a large-size question bank in a real-life situation. In this simulation, it is assumed that all of the questions were perfectly calibrated. Virtual questions and students were generated in this experiment. An adaptive e-Assessment based on the same algorithm used in experiment 1 and experiment 2 was simulated based on the virtual questions and students. The word 'virtual question' is used here for the reason that these questions have neither question body nor options. They are just tags attached with three parameters: a, b, and c. Each virtual question generated using Monte Carlo simulation was assigned with three randomly generated values respectively representing parameters a, b and c. In this experiment, the value ranges for parameters a, b, and c were [-1, +1], [-3, +3], and [0, 0.25] respectively. Virtual students are also tags generated by a computer program. Each virtual student was assigned with a randomly generated ability parameter θ ranged in [-3, +3]. The generation of ability parameter θ conformed to Gaussian distribution.

The pseudocode which describes the Monte Carlo simulation is as follows:

1. Generate virtual students and virtual questions.
2. For each assessment, test if the termination criterion is satisfied. Terminate the current assessment and move onto the next assessment if the termination criterion is met, otherwise continue with the current assessment.
3. Select and present the best virtual question for the current virtual student.
4. Compute the possible response.
5. Update the ability estimation for the current virtual student.
6. Repeat 2 - 5 until all the assessments are completed (the number of assessments are dependent on the number of virtual students).
7. Compute the average number of questions used in all the assessments and the percentage of correctly estimated abilities.

The response to a question from a virtual student in step 4 was determined by the comparison of a random fraction and the expected probability of getting the correct response. For example, if the virtual student's expected probability of giving a correct response to the question computed using the IRF is 0.8, and the random fraction generated has the value 0.76 (considering 0.8 > 0.76), then it is assumed that this time the student would give the correct response. The simulation was run to find out how the adaptive e-Assessment engine's classification performance is affected by the number of questions in question bank *N*. The simulation was conducted based on 1,000 virtual students (1,000 assessments) and question banks with 30, 50, 100, and 200 questions. The results of the simulations were summarised in table 4-4 for the reliability level 0.9.

Table 4- 7: results of the Monte Carlo simulations with various question bank sizes

I	30 questions available in the question bank *N*		50 questions available in the question bank *N*		100 questions available in the question bank *N*	
Reliability	**Percentage of correctly estimated students (acceptable error ±5%)**	**Average number of questions used**	**Percentage of correctly estimated students (acceptable error ±5%)**	**Average number of questions used**	**Percentage of correctly estimated students (acceptable error ±5%)**	**Average number of questions used**
90	**30**	**30**	**42.1**	**50**	**43.2**	**100**
II	200 questions available in the question bank N					
Reliability	**Percentage of correctly estimated students (acceptable error ±5%)**	**Average number of questions used**				
90	**71.2**	**102**				

From the above results, it can be seen that it is not easy to give correct ability estimation for all students. Due to the variance in random value generation every time, the observed results of these Monte Carlo simulations vary to some extent. However, the general performance of the estimation was mostly dependent on two impact factors: the acceptable error range and the number of questions in question bank *N*. Obviously, the wider the acceptable error range is, the higher the percentage of correctly estimated students will be; moreover, the more virtual questions there are,

the better chance of informative questions that suit each virtual student. When there were only 30 questions in the question bank, there was little chance of every student having suitable questions. As the size of the question bank reached 200, there was a huge boost of percentage of the correctly estimated students and it was the first time in the simulation that not all of the questions in the question bank were used. Additionally, the guessing factor c acts as an interfering factor in the estimation procedure. The higher the guessing factor is, the more difficult it is for the system to estimate correctly.

The results of the simulation suggest that the size of the question bank has an important impact on the precision of ability measurement in this implemented adaptive e-Assessment algorithm. This is because this algorithm used item information as the criterion for question selection, and the item information is computed based on the examinee's ability and the parameters of the question. If there are a good number of informative questions for the current examinee, then the adaptive e-Assessment always has appropriate questions to administer. In this situation, the assessment terminates quickly and the measurement is more reliable. The Monte Carlo simulation uses randomly generated questions. Therefore the more questions generated in the simulation, the more opportunity there is for a virtual candidate to have appropriate questions. This explains why modern adaptive e-Assessment systems (Wise and Kingsbury, 2000) usually have item pools with more than 1,000 questions. It is because an adaptive e-Assessment system needs to prepare a variety of questions for candidates at all ability levels so that there are always appropriate questions to administer.

4.3 CONCLUSION

The above three experiments revealed the following three factors. First, adaptive e-Assessment is a good substitute for fixed-form e-Assessment, featuring both reliable measurement of latent traits and short test length. In experiment 1 and 2, an overall error indicator was used to compare how the student ability estimates by the two different testing methods deviated from the true ability. The

comparison indicated that the adaptive e-Assessments were less error-prone than the fixed-form e-Assessments. Nonetheless, it may be argued that the above comparison was unfair because it was based on the assumption that the 3PL IRT model is the true underlying influence factor for the performance difference on the tests. Considering that there is no way to directly and reliably perceive a test-taker's real ability, there is therefore no adequate evidence to determine which e-Assessment methods provided more accurate estimates. However, it can be seen from the BIAS indicators and the RMSE indicators that there was no significant difference between the ability estimates obtained by the adaptive e-Assessments and the fixed-form e-Assessments, while at the same time the adaptive e-Assessments used fewer questions to draw the conclusion.

Secondly, the response sample used in the item calibration procedure has a profound impact on the subsequent test reliability and accuracy. It affects not only the selection of IRT model, but also the parameter estimation. The selection of an inadequate response sample may be biased and may result in the selection of the wrong IRT model and large errors in parameter estimation, which in turn has a great impact on the precision of ability estimation.

Thirdly, a reliable adaptive e-Assessment requires a relatively large question bank. The experimental results from the real-data simulation differed considerably from the results obtained from the Monte Carlo simulations. From the results of experiment 3 it can be seen that even for an ideal circumstance it would require a question bank with more than 200 questions to get acceptable estimation precision. This is because the more questions there are, the more chance there is for an adaptive e-Assessment system to find appropriate questions for candidates at all ability levels. Therefore, in experiment 1 and experiment 2, the comparison of the results obtained by the two testing methods can only indicate that computerised adaptive assessments can achieve a similar ability estimation precision with fewer questions compared to fixed-form computer-based assessments. This suggests that the questions in the Cisco chapter tests do not appear to be at the same difficulty level for the students and hence it is inappropriate to assign equal marks to all of the

questions. However, there was inadequate evidence to indicate that the student ability estimates or student scores were the reflection of real student abilities.

CHAPTER 5 REQUIREMENTS ELICITATION

5.1 INTRODUCTION

This chapter elicits the common requirements from the stakeholders of e-Assessment as the preparation for framework design and implementation. Since the purpose of a framework is to enable software developers to utilise a codebase, which offers common logic and functionalities, to reduce the time and effort required to build an application (Angelov *et al.*, 2006), it is important to capture the common requirements from the two groups of stakeholders of e-Assessment: e-Assessment researchers and e-Assessment developers.

5.2 APPROACH

The purpose of creating the adaptive e-Assessment framework is to provide a chassis for adaptive e-Assessment strategy implementation and adaptive e-Assessment system development to support the research in this area.

The impetus for developing such a framework comes from both the adaptive e-Assessment strategy research community and adaptive e-Assessment system developers. The researchers and the developers have different foci when it comes down to the design and implementation of e-

Assessment. The researchers focus on the assessment method itself and would expect minimal programming experience whilst the developers focus more on the architecture of the framework and would expect a maximal efficiency boost.

In order to better understand the expectation of both stakeholders, a requirement elicitation was conducted to collect and analyse the true requirements. The major areas of requirements that framework design would take into consideration are architecture design requirements, module communication requirements, configuration requirements, adaptive e-Assessment strategy implementation requirements, and database management requirements. As for the mapping of user interface and controller, it is not of concern in this part of the research due to the availability of various presentation frameworks such as Struts, Spring and Stripes.

The subsequent requirements analysis is based on the composite of stakeholders' requirements, additional experience, and the author's own viewpoint. For the reason that user-centred design has been shown to lead to more usable satisfying designs (Abras *et al*, 2004), the overarching methodology used in this part of the research was to elicit requirements from different classes of stakeholders and summarise user requirements via consultation. One-to-one interviews and focus group interviews were adopted to elicit and re-affirm the requirements.

One-to-one interview is a common technique for eliciting and gathering user requirements (Mcdonald and Welland, 2001). In the one-to-one interviews, a number of interview questions were presented to the interviewees and the answers were recorded by the interviewer. The interview questions were prioritised from the general aspect of adaptive e-Assessment to specific functionalities. The one-to-one interviews were semi-structured to allow interviewees to bring up their own concerns about adaptive e-Assessment. A focus group interview is defined as 'a technique involving the use of in-depth group interviews in which participants are selected because they are a purposive, although not necessarily representative, sampling of a specific population, this group being 'focused' on a given topic' (Thomas et al., 1995). Focus group interviews were used in

the research to provide a range of ideas that individual interviewees have about certain issues, as well as to re-affirm the user requirements elicited and collected from one-to-one interviews. Interviews were adopted in the research rather than questionnaires or other information collection techniques because it is easier to explain the research background to the interviewees and clarify certain umbigous concepts like modern psychometrical theories in such interactive environments.

5.2.1 ONE-TO-ONE INTERVIEWS

The gathering of user requirements was firstly conducted through informal one-to-one interviews with stakeholders. These stakeholders were e-Assessment users, e-Assessment researchers, and e-Assessment system developers. These interviews discussed the features of adaptive e-Assessment and the stakeholders' expected activities and means of engagement with such a system. The gaps in the current e-Assessment systems were also discussed with the stakeholders in the interviews.

A Java Server Pages (JSP) implementation of adaptive e-Assessment systems based on the algorithms used in the experiments in chapter 4 was employed to allow the stakeholders to gain a better understanding of an adaptive e-Assessment prototype.

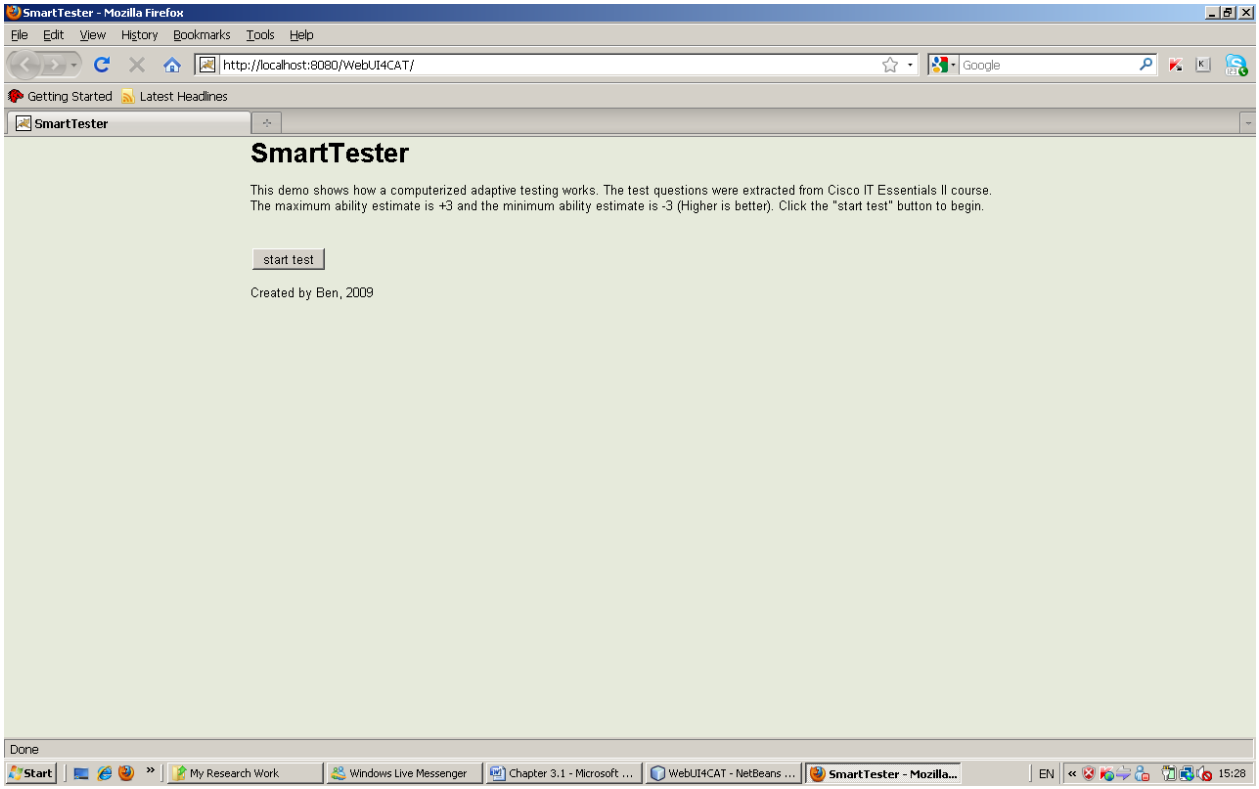


Figure 5- 1: a screenshot of the adaptive e-Assessment system implemented by JSP

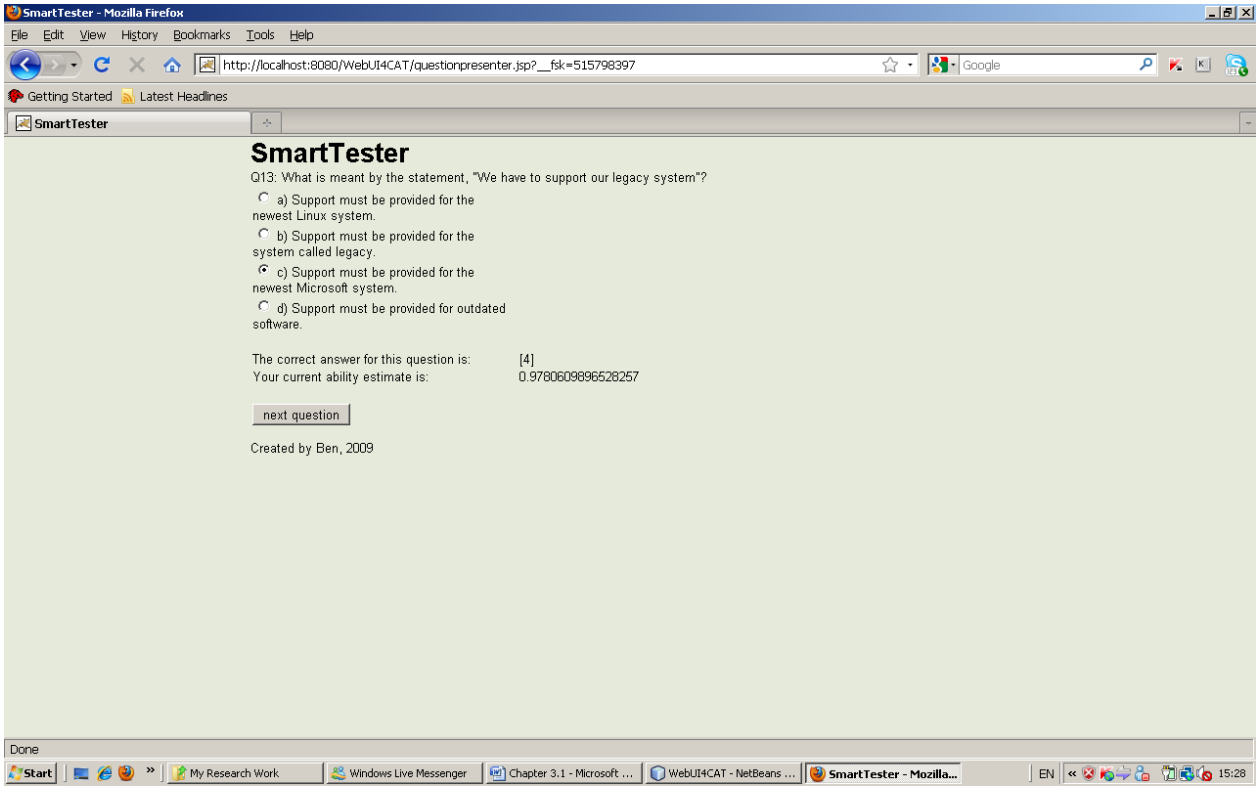


Figure 5- 2: another screenshot of the adaptive e-Assessment system implemented by JSP

5.2.2 FOCUS GROUP INTERVIEWS

Two levels of focus group meetings were organised to validate the information collected from the one-to-one interviews. The first level of focus group meetings were structured to validate the viewpoints summarised from one-to-one interviews. In the first level of focus group meetings, the JSP implementation of adaptive e-Assessment was presented to the attendants to help explain the main characteristics of adaptive e-Assessment. The second level of meetings presented the design and functionality of the proposed e-Assessment framework to the focus group.

The focus group was made up of e-Assessment users, e-Assessment researchers, and e-Assessment system developers. The contents of the meetings were based on the major concerns of the above two categories of stakeholders.

5.2.3 BACKGROUND OF INTERVIEWEES

The majority of interviewees were from the Faculty of Design, Media and Management (DMM) and the Flexible and Distributed Learning Centre (FDLC) of Buckinghamshire New University, UK, and the School of Software Engineering of Fujian Agriculture and Forestry University, P.R. China. These interviewees were selected because the nature of their teaching and working has brought them closer to e-Assessment. In the one-to-one interviews, 11 interviewees were involved, which included six interviewees from Buckinghamshire New University, UK and five interviewees from Fujian Agriculture and Forestry University, China. Amongst these interviewees, six are software engineers with experience of the development of e-Assessment systems, three of them are researchers in e-Assessment, and two of them are developers of e-Assessments. Both the level one and level two focused group meetings were conducted at Fujian Agriculture and Forestry University, China. The two interview meetings were co-moderated by the author and the director of the teaching and researching section of the school of software engineering, Fujian Agriculture and

Forestry University, China. Five lecturers with e-Assessment development experience and three software engineers with experience on the development of e-Assessment systems were invited to the interviews. The backgrounds of the interviewees are listed as below.

Table 5- 1: the background of the one-to-one interviewees

Organisation	Background of interviewee	Number of interviewees
Buckinghamshire New University, UK	E-Learning developer	2
Buckinghamshire New University, UK	E-Learning researcher	3
Buckinghamshire New University, UK	Software engineers with development experience on e-Learning	1
Fujian Agriculture and Forestry University, P.R. China	Software engineers with development experience on e-Learning	5

Table 5- 2: the background of the level one and level two focus group meeting attendants

Organisation	Background of interviewee	Number of interviewees
Fujian Agriculture and Forestry University, P.R. China	Lecturers with e-Assessment development experience	5
Fujian Agriculture and Forestry University, P.R. China	Software engineers with development experience on e-Learning	3

5.2.4 INTERVIEW DETAILS

The one-to-one interviews were semi-structured to introduce the context of the research and capture user requirements from the stakeholders. Questions could therefore be brought up during the interview. Notes were made during these interviews. Five questions were used in the one-to-one interviews and level 1 focused group interview and they were formulated based on the practical issues in adaptive e-Assessment (Wise and Kingsbury, 2000; Chen, 2006), the characteristics of

good e-Learning systems (Khaskheli, 2004), and the purpose of software frameworks (Taligent Inc, 1993; Angelov et al., 2006). The five questions are as follows:

1. What do you know about Adaptive e-Assessment?
2. Have you used any adaptive e-Assessment applications?
3. What do you consider to be the main advantages and disadvantages of these applications?
4. If you were asked to develop an adaptive e-Assessment application, what software functionality would you most expect to have?
5. Is there any other characteristic you consider to be important for an adaptive e-Assessment system that I have not covered during the discussion?

5.2.5 LIMITATIONS OF THE APPROACH

The major limitation of the requirement-gathering approach used in this part of the research is the relatively small sample size. The limited timescale and constraints on other resources have hindered the collection of more data. However, the increasingly repetitive agreements from the focus group indicate that some of the key requirements have been identified.

5.3 INTERVIEW RESULTS

In the one-to-one interviews and focus group interview, software engineers with experience on the development of e-Assessment systems showed concerns about issues on software engineering such as scalability and extensibility. Five of the six software engineers expected that the future framework would be able to accommodate different functional modules such as automarking components for different types of questions. All of the six software engineers expected that the libraries and application programming interfaces of the future framework could be easily reused. These requirements were consistent with the general requirements of a well-designed framework and they were taken into consideration in the design and implementation of the framework. The expectations of e-Assessment researchers and developers were more focused on the functionalities

provided by the framework such as automarking for gap-filling questions and easy creation of e-Assessments. The requirements of e-Assessment researchers and developers can be fulfilled by adding solution packages/libraries to the framework. However, these requirements were rejected as they are requirements for applications rather than the framework. It was determined that the priority of the framework was to provide the basic architectural support for future applications and in the meantime maintain the five good characteristics of the e-Learning/e-Assessment system mentioned in chapter 2: interoperability, reusability, accessibility, scalability, and affordability. These requirements were reconfirmed in the focus group interview.

5.4 REQUIREMENT ANALYSIS

The requirements collected can be concluded as researcher requirements and developer requirements. Researchers and developers pose different requirements from the standpoints of their different needs. However, there are still some overlaps between these two categories of requirements.

5.4.1 RESEARCHER REQUIREMENTS

The results of the interview showed that the researchers' main interest lies in the design and evaluation of adaptive assessment strategy. Therefore, the adaptive algorithm itself and the associated peripheral facilities become the concern of researchers. However, some researchers may have extended requirements to implement their customised e-Assessment strategies.

The design of adaptive assessment strategy is the main focus of e-Assessment researchers. The algorithm used to describe an adaptive assessment strategy can be ever-changing depending on the researchers' needs. Therefore the first research requirement is that the proposed framework can be adapted to various customised adaptive algorithms, namely, designing and implementing a standard interface for different plug-in algorithms.

A well-designed framework must also be prepared for the possible change of peripheral facilities varying with adaptive assessment strategy. The most noticeable one among these possible changes is the structure change of questions and tests. In a conventional software design, questions and tests are often stored in tables of a backend relational database with the different fields of these tables indicating the different properties of the questions and tests. However, different adaptive assessment strategies may require different fields to be created in database tables. For example, a Rasch-model-based adaptive strategy may need to reserve one field in the data table to record the difficulty level of each question while a two-parameter-logistic-IRT-model-based adaptive strategy may need to reserve two fields in the data table to respectively represent the difficulty level and discrimination power of each question. The change of the structure of the question unit and test unit will result in the re-design of data tables and the structure of the corresponding objects in the business logic layer. Therefore, this potential requirement can be defined as synchronisation of the objects in the business logic layer and data table.

Considering the fact that many adaptive assessment strategies may involve calculus computation, another potential requirement of e-Assessment researchers would be the numerical computation support library for calculus.

5.4.2 DEVELOPER REQUIREMENTS

From the developers' point of view, a good e-Assessment framework should help them develop adaptive e-Assessment systems more efficiently. This general requirement was supported by the following three detailed developer requirements: short learning curve, collaborative form and extensible scope.

The interviewed developers welcome a light-weight, loosely coupled and easy-to-learn framework. Whether the framework requires only a short learning curve directly determines the popularity of the framework among developers. Most developers would expect mastery of a framework within several hours' hands-on experience. Secondly, the interviewed developers also expect the

framework to collaborate well with popular tools they are using, such as Struts 2, log4j, and jUnit. Last but not least, the developers expect the framework to not only allow them to complete simple tasks easily, but to allow them to complete complex tasks.

5.4.3 COLLABORATIVE REQUIREMENTS

The collaborative requirements here are two-fold: the development collaboration between researchers and developers, and the sharing of computational resources among computers. For achieving complex tasks or functionalities which consist of multiple objectives, there may be a need for collaboration between researchers and developers. Due to the growing complexity of adaptive algorithms, it would also be helpful if multiple computers could share their computational resources when necessary.

5.5 REQUIREMENT SUMMARY

The stakeholders' requirements are two-fold. Researchers have relatively high-end requirements with their focus being the assessment strategy per se. Developers have relatively low-end requirements with their focus being an efficient development cycle. To meet the requirements from both categories of stakeholders, a reasonable approach is to design the framework based on developers' requirements and then extend the framework further to meet researchers' requirements.

The requirements listed above can be categorised by functionality as module communication, adaptive e-Assessment strategy, database support, and utilities. The following chapter will state how these functionalities were achieved in the framework.

CHAPTER 6 FRAMEWORK DESIGN AND IMPLEMENTATION

6.1 APPROACH TO DESIGN-CLASS MODELLING AND IMPLEMENTATION

Ambler (2004) suggests that, "the purpose of design-class modelling is to model the static structure of how your software will be built." The proposed framework of the system was accordingly constructed in two stages. The first stage focused on the architecture of the framework and fulfilling the requirements of e-Assessment developers. Therefore the work at this stage generated a loosely-coupled framework which is better illustrated functionality by functionality. The second

stage added some extension to the work of the first stage to cater to the requirements of e-Assessment researchers. The work of this stage can be illustrated using a typical scenario – the test-taking scenario – which can then be illustrated with a class design model and other UML diagrams.

Sun Java was selected to be the programming language for implementation. Java was chosen because it is a free and open source programming language with powerful network communication capability and rich third-party libraries. MySQL was chosen to be the backend database management system for the reason that it is a free but full-featured database management system. However, the use of any other database system compatible with JDBC is available and easy to achieve.

6.2 THE CORE COMMUNICATION MECHANISM

This section describes the design pattern of the communication mechanism and the constitution. Distributed architecture was adopted as the fundamental communication pattern to facilitate the collaboration and sharing of scattered resources across the network. This architecture provides a good foundation for scalable systems and enables remote collaboration among humans and computers. Researchers and developers working remotely are able to collaborate via implementing a mutually agreed information exchange protocol. Under this architecture, overall computational load can also be dispersed across the network.

There are four basic objects which are the cornerstones of the proposed distributed architecture. They are *Actor*, *Behavior*, *Communication*, and *Message*. *Actor* is an independent thread of control which receives asynchronous *Message*s and processes them one at a time. *Behavior* is an object bound to an *Actor* to specify what the *Actor* does in response to a *Message*. Under distributed deployment of the framework based system there will be multiple groups of logically-related *Actors* residing in different Java Virtual Machines (JVM) across the network. *Communication* is the management unit for such a group of logically-related *Actor*s and the connection maps between JVMs. *Message* is media used for the purpose of communication between different JVMs.

Apart from the above four basic objects, there are also a few other components which are useful for the construction of a distributed e-Assessment system. *DynamicDispatchBehavior* is a *Behavior* adapter which performs dynamic dispatch on the class of a *Message* object. Objects of this class wrap another target *Behavior*. When *DynamicDispatchBehavior.call(Message)* receives a *Message* object, it looks at the actual dynamic class of the *Message*, and dispatches the *Message* to the most specific matching *call* method within the target *Behavior*. For example, say we have the class *A* extends *Message* and class *B* extends *A*. If a *Message* of class *B* is received, the instance of *DynamicDispatchBehavior* first checks whether its target has a *call(B)* method. If not, it then checks whether the target has a *call(A)* method. If so, the *Message* is dispatched to *call(A)*. The principle is elaborated in the code below:

DynamicDispatchBehavior.java

```
public void call(Message message) {
    // Try to find a more specific "call" method which accepts Message.
    Class<? extends Object  >  messageClass = Message.class;
    Class<? extends Object  > dispatchClass = message.getClass();
    Class<? extends Behavior> behaviorClass = behavior.getClass();
    while (!messageClass.equals(dispatchClass)) {
        try {
            Method method = behaviorClass.getMethod("call", dispatchClass);
            method.invoke(behavior, message);
            return;
        } catch (Exception e) {
            // Method not found.
        }
        dispatchClass = dispatchClass.getSuperclass();
    }
```

```
        // If we got here, we couldn't find a more specific method.
        // Only call the general case if it won't be a recursive.
        if (behavior != this) {
            behavior.call(message);
        }
    }
```

Text Box 6- 1: Excerpt from *DynamicDispatchBehavior.java*

The framework also includes a number of services as off-the-shelf tools for rapid development. *PeerToPeerBroadcastNamingService* is the implementation of one node/peer of a fully distributed peer-to-peer naming service. The general idea is that one naming peer operates as a naming service in each JVM instance. Different instances of the naming service communicate, so that services registered in one JVM instance are also known about by others. The net effect is to maintain a global shared set of known services. *SimpleService* is a simple way to create a single data source service. It automatically registers an *Actor* for the given *Behavior*, and registers the *Actor* with the peer-to-peer naming service.

Although adopting the universal W3C Web Service standards is beneficial to Enterprise Application Integration (EAI) across multiple organisations, it was decided in this research not to implement the off-the-shelf protocols but design a peer-to-peer distributed architecture from scratch. The reason is two-fold. Firstly, it is believed that the essence of web-service-based architecture lies in the standardisation of communication protocols on service description, service discovery and service interaction. These standards enable the excellent interoperability of web-service-based architecture and represent the main attraction of web-service-based architecture. However, as far as application is concerned, there is no need for the framework based e-Assessment systems to communicate with other web-based systems. Secondly, these standard "envelopes" used to wrap the information to achieve interoperability do not come free. They consume computational power and network bandwidth. Therefore, customised light-weight

distributed architecture would reduce the load of computers and network at the cost of reduced interoperability.

The services provided in the proposed framework are similar to web-services promoted by W3C in terms of general architecture: both architectures are distributed. However, there are some important differences. The following table shows the commonality and difference of these two architectures.

Table 6- 1: comparison of the design pattern used in the framework with the standard web service promoted by W3C

		design pattern used in the framework	**standard web service promoted by W3C**
commonality	purpose	make the functionality of an information system available online to provide fundamental support for distributed computing	
	general architecture	Both are based on the three-component implementation infrastructure, namely, the service provider, the service requester, and the service registry	
difference	communication standard	framework defined	SOAP
	service description	framework defined	WSDL

	standard		
	service registry standard	framework defined	UDDI

The framework proposed in the research is conceptually similar to a web service promoted by W3C but differs in implementation standards. The purpose of both architectures is to make the functionality of an information system available online to provide fundamental support for distributed computing. They are both based on the three-component implementation infrastructure, namely, the service provider, the service requester, and the service registry. However, in a W3C standard web service it is required to use standard web technology such as SOAP, WSDL and UDDI to alleviate heterogeneity and facilitate application integration across organisations. It is argued that inter-organisation application integration is not the purpose of the proposed framework. Therefore, framework-defined communication, description and management standards were adopted in this research to enable the light-weighted use of bandwidth and computational resources for the future distributed systems built upon this framework.

In contrast to the standard SOAP, WSDL, and UDDI protocols, the proposed framework adopts a simplified and light-weighted method to manage the communication between service requesters and service providers in conjunction with the features provided by the Java programming language. Unlike many web-service-based applications where service directories use a centralised approach, in the proposed framework the service directories were kept in a peer-to-peer approach, i.e., a directory of all the services residing in the current JVM is kept in the JVM. The *PeerToPeerBroadcastNamingService* provided by the framework can then be used to communicate among different JVMs so that every instance of naming service knows about the services register in the other JVMs. Each service provider contains a *StringServiceInfo* object to describe the service, which plays the role of WSDL protocol. A *Message* object is passed between the service requester

and the service provider and it resembles an SOAP message. The following diagram shows the basic architecture of this proposed framework.

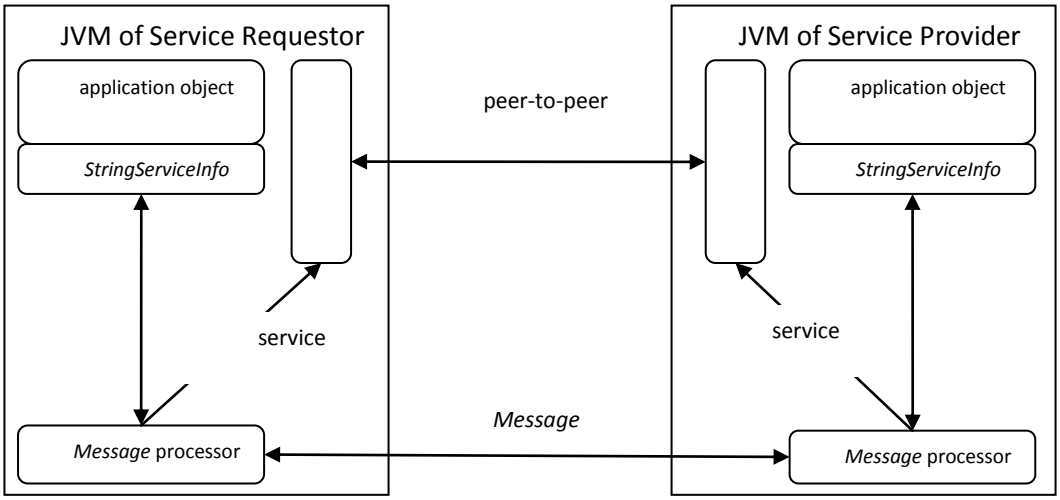


Figure 6- 1: the basic architecture of the proposed framework

6.3 DATABASE DESIGN AND IMPLEMENTATION

The proposed framework separates the Java Database Connectivity (JDBC) driver from database manipulation to enhance the flexibility of database connectivity. The configuration of the database is stored in "database-config.xml", which defines the database connection configuration. When a developer needs to configure his/her own database, he/she only needs to modify this configuration file accordingly.

database-config.xml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<properties>
  <category name="JDBC">
    <property name="DRIVER" value="com.mysql.jdbc.Driver"/>
    <property name="HOST" value="localhost"/>
```

```
    <property name="PORT" value="3306"/>
    <property name="USER" value="dbuser"/>
    <property name="PWD" value="dbuser"/>
    <property name="DATABASE" value="catdb"/>
  </category>
</properties>
```

Text Box 6- 2: database-config.xml

Many assessment strategies require the cooperation of a specific data structure, i.e. the structures of the question unit and test unit. This is determined by the underlying adaptive algorithm and will affect the structures of the data tables used to store relevant information. For instance, the adaptive algorithm A operates based on questions described using three parameters and adaptive B operates based on questions described using two parameters. Then two data tables need to be created to separately store these two types of questions, one of which has two fields for question parameters and the other three fields for question parameters. The repeated creation of data tables is a tedious yet error-prone task. Therefore, an innovative method was adopted in the proposed framework to store question information in a Binary Large Objects (BLOBS).

In the framework design, question units are serialised into bytes before being store in database tables and these bytes are deserialised when retrieving from the database. Java™ provides a serialisation and deserialisation method to deflate and inflate data objects. Object serialisation is the process of saving an object's state to a sequence of bytes, and object deserialisation is the process of rebuilding those bytes into a live object. Object serialisation and deserialisation guarantee object persistence in the process of storage and transmission (Hagimont and Louvegnies, 2009).

This research used the Thrift Remote Procedure Call (RPC) framework to define question unit class. Thrift (Slee *et al*., 2007) is a cross-language framework originally developed at Facebook which supports efficient and reliable cross-language communication among computer programming languages such as C++, Java, Python, PHP, Ruby, Perl, and C#. The implementation of the Thrift

framework combines a language neutral software stack implemented across numerous programming languages and an associated code generation engine that transforms a simple interface and data definition language into client and server remote procedure call libraries (Slee, *et al*., 2007). In this sense, the Thrift framework works similarly to SOAP, COM, and CORBA.

The purpose of introducing the Thrift Framework is to reserve the framework the ability to accommodate modules developed in different programming languages. The framework has been coded in Java programming language so far. However, modules developed in other programming languages may be incorporated into the framework in the future for better performance and development efficiency. Considering question unit class is a basic data element that will be manipulated by the future modules, the raw structure of question unit class was defined with the neutral language provided by the Thrift framework before transforming into the Java libraries.

6.4 OTHER SUPPORTING MODULES

Other supporting modules of this framework include environment support, logging support, management and utilities packages. Each package provides category-specific support Java classes for rapid development of the framework-based systems. For example, the *FileManager* class defines methods used to obtain directories and files from the installation directory and the repository.

6.5 CREATING A SIMPLET DISTRIBUTED SYSTEM BASED ON THE FRAMEWORK

This section gives an example to explain how to quickly create a simple distributed system using the common classes provided by the framework. In this example, two types of messages and three peer-to-peer nodes will be created to implement a simple distributed system.

In this example, firstly a service management module will be created to manage all the services connected to it. It is a standalone executable that runs and loads the port number to use from the s*ervice-config.xml* file specified in the *InstallationNames.java* class of the environment package.

ManagementService.java

```
package services.management;


import java.net.SocketAddress;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.UnknownHostException;


import org.apache.log4j.Logger;
import logging.LoggingConfig;


import communication.Actor;
import communication.Communication;
import communication.MutableBehavior;
import communication.services.PeerToPeerBroadcastNamingService;
import environment.ServiceConfig;
import environment.ServiceConfig.InvalidConfigurationException;


/**
 * Standalone executable that runs the management service.  Finds the port
 * number to use from ServiceConfig, and then launches the management
 * service.  Other services can then connect to the management service on
 * this port.
```

```
 */

public class ManagementService {

   private static final Logger logger = Logger.getLogger("management");



   /**

    * TODO: if we are passed a command line arg of an ip address, then we

    * should use that ip address as an existing network to join, as opposed

    * to creating a new network.

    */

   public static void main(String[] args) {

      try {

         ServiceConfig serviceConfig = new ServiceConfig();

         // New-style naming bootstrap actor

         // we must bind to the real ip address of this machine, so we load from the service-config

         if (serviceConfig.localInetAddress == null) {

            System.err.println("WARNING!: host ip must be defined as the actual ip of this machine -
distributed services may not work!");

         } else if (serviceConfig.localInetAddress.equals(InetAddress.getByName("localhost"))) {

            System.err.println("WARNING!: localhost is not a valid choice for ip, must use the actual ip of this
machine - distributed services may not work!");

         }

         InetSocketAddress localInetSocketAddress = new
InetSocketAddress(serviceConfig.localInetAddress, serviceConfig.namingBootstrapPort);

         MutableBehavior              mutableBehavior = new MutableBehavior();

         Communication                 communication = new Communication(localInetSocketAddress,
mutableBehavior);

         PeerToPeerBroadcastNamingService namingService =
PeerToPeerBroadcastNamingService.createNewNetwork(communication, null);

         mutableBehavior.behavior = namingService;
```

```
        } catch (UnknownHostException e) {
            System.err.println(e);
            logger.fatal("dying", e);
        } catch (InvalidConfigurationException e) {
            System.err.println(e);
            logger.fatal("dying", e);
        }
    }
}
```

Text Box 6- 3: ManagementService.java

After the *ManagementService.java* is loaded into the memory of one server in the network, other service modules can be created on different servers in the network and connected to the management service to form a distributed system. Subsequently, a service providing a module named *Master.java* and a service requesting module named *Worker.java* are to be created and deployed on different servers to verify the distributed processing ability of this simple system. When the *Worker.java* and *Master.java* are being booted up, they need to be provided with the IP address and the TCP port on which the *ManagementService.java* is running so that these two modules can be connected to the existing distributed network created by the *ManagementService.java.* The verification of the effectiveness of the communication between these three distributed modules is through the *Master.java* sending a *RequestMessage.java* object and the *Worker.java* responding with a *ReplyMessage.java* object.

Master.java

```
package services.examples.masterandworker;


import communication.services.SimpleService;
import communication.services.ServiceNameActorLocator;
```

```
import communication.Message;

import communication.Actor;

import communication.Behavior;



/**

 * Example class that sends a single RequestMessage and prints a response

 * when called.

 * Start one of these after starting the management service and a Worker.

 * ex: ~/examples$ java -cp ../dist/lib/commons.jar:. com.cat.services.examples.masterandworker.Master
localhost 31337

 */

public class Master implements Behavior {

    /**

     * Used for testing.

     */

    public static void main(String[] args) throws Exception {

        SimpleService self = new SimpleService(new Master());

        System.err.println("ServiceNameActorLocator is running");

        ServiceNameActorLocator actorLocator = new ServiceNameActorLocator(self.namingService);

        Actor actor = actorLocator.getActorFor("Master");

        Actor worker = actorLocator.getActorFor("Worker");



        long startTime = System.currentTimeMillis();

        int concurrent = new Integer(args[2]); // concurrent requests

        for (int i = 0; i < concurrent; i++) {

            (new Thread(new Sender(actor, worker))).start();

        }
```

```
        long totalTime = System.currentTimeMillis() - startTime;
        System.err.println("total time: " + totalTime);
    }

    private static class Sender implements Runnable {
        private Actor replyTo;
        private Actor target;
        public Sender(Actor replyTo, Actor target) {
            this.replyTo = replyTo;
            this.target = target;
        }
        public void run() {
            for (int i = 0; i < 1000000; i++) {
                Message request = new RequestMessage(String.valueOf(i), replyTo);
                try {
                    Thread.sleep(40);
                } catch (java.lang.InterruptedException e) {}
                target.send(request);
            }
        }
    }


    public void call(Message message) {
        System.out.println(message);
    }
}
```

Text Box 6- 4: Master.java

Worker.java

```
package services.examples.masterandworker;


import communication.DynamicDispatchBehavior;
import communication.services.SimpleService;


/**
 * Example class that listens for a RequestMessage and then responds with a
 * ReplyMessage.
 * Start one or more of these after starting the management service.
 * ex: ~/examples$ java -cp ../dist/lib/commons.jar:. masterAndWorkers.Worker localhost 31337
 */
public class Worker extends DynamicDispatchBehavior {


    public static void main(String[] args) throws Exception {
        new SimpleService(new Worker());
        System.out.println("Worker is running");
    }


    public void call(RequestMessage message) {
        System.out.println("received RequestMessage message: " + message);
        synchronized (this) {
            try {
                this.wait(50);
            } catch (java.lang.InterruptedException e) {}
```

```
        }

        message.replyActor.send(new ReplyMessage("re: " + message.toString()));

    }

}
```

Text Box 6- 5: Worker.java

RequestMessage.java

```
package services.examples.masterandworker;


import communication.Actor;

import communication.ReplyableMessage;


public class RequestMessage extends ReplyableMessage {

    public static final long serialVersionUID = 1L;

    public final String contents;


    public RequestMessage(String contents, Actor replyActor) {

        super(replyActor);

        this.contents = contents;

    }


    public String toString() {

        return contents;

    }

}
```

Text Box 6- 6: RequestMessage.java

ReplyMessage.java

```
package services.examples.masterandworker;
```

```
import communication.Message;
import communication.Actor;


public class ReplyMessage extends Message {
    public static final long serialVersionUID = 1L;
    public final String contents;


    public ReplyMessage(String contents) {
        this.contents = contents;
    }


    public String toString() {
        return contents;
    }
}
```

Text Box 6- 7: ReplyMessage.java

6.6 A DISTRIBUTED ADAPTIVE E-ASSESSMENT SYSTEM BASED ON THE FRAMEWORK

Traditionally, use-case diagrams are adopted to capture the user requirements of the system to be engineered. These diagrams are good at capturing entities, business units and the interaction between them at class level. As mentioned above, this proposed framework follows a distributed design pattern whose focuses are on distributed services and activities. The following sections of this thesis will use a distributed test-taking scenario as an example through which to discuss this design method and the modelling approaches adopted in this research.

6.6.1 ILLUSTRATIVE USE-CASE AND SEQUENCE DIAGRAM FOR TAKING A TEST

Conventionally use-cases are considered during the early stage of application development (Ambler, 2004). On this occasion the UML use-case and sequence model are used to illustrate the operational delivery and mechanism behind rest testing.

The framework focuses on providing an expandable and collaborative architecture rather than detailed implementation of functionalities. For the demonstration purpose, the following section explains how a test-taking scenario can be implemented based on the framework. Other functionalities may be easily constructed upon the basic components supplied by the framework. In the following use-case diagram, distributed modules are denoted respectively as user actor, resource actor, requester actor and service actor to better illustrate this scenario.

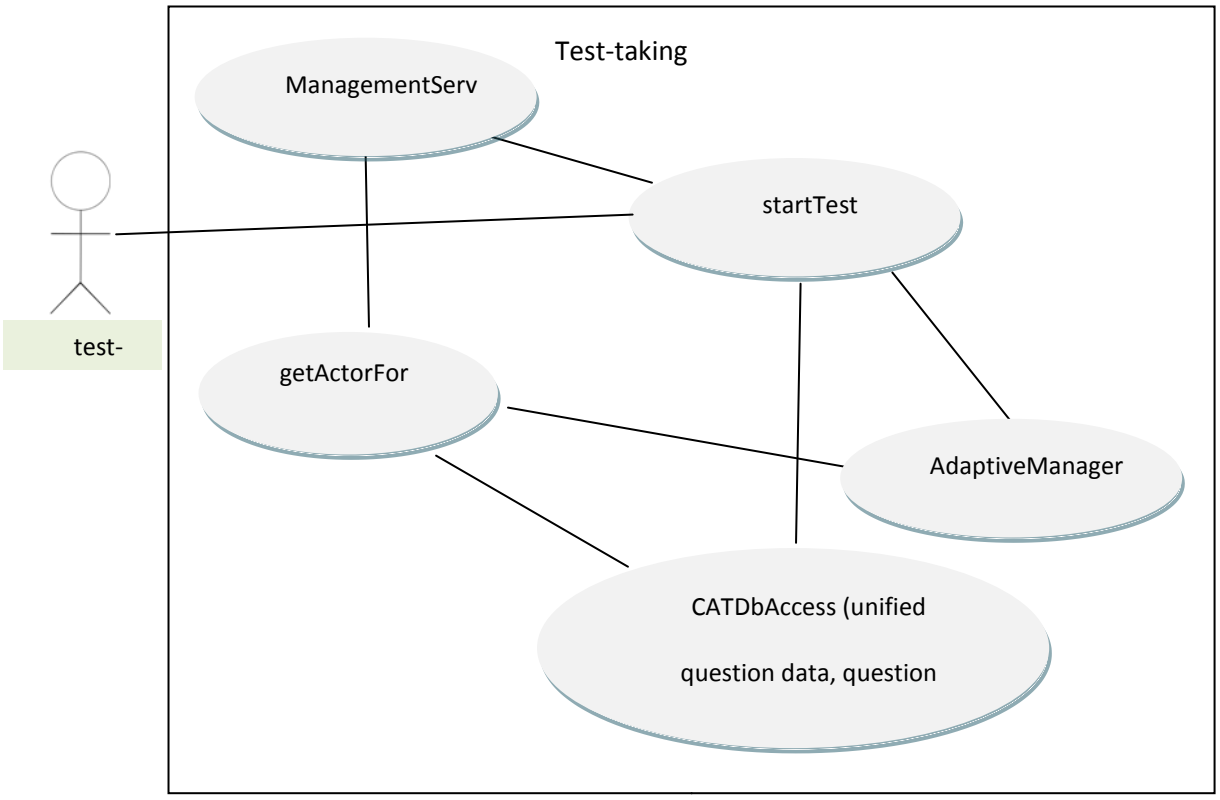


Figure 6- 2: the test-taking use-case

The above typical scenario involves one requester actor – test-taker, one user actor – *ManagementService*, one resource actor – *CATDbAccess* (providing access to unified question data, question calibration data, test data and user data), and one service actor – *AdaptiveTestManager*. The *ManagementService* was deployed on one server and is always active for listening to the connection request from the different services deployed on the distributed servers. When a distributed actor starts, it must be provided with the IP address of the server on which the *ManagementService* is running and the port number that the *ManagementService* is listening so that it can connect to the network. Then the peer-to-peer broadcasting module residing in each distributed node starts to communicate to ensure the registry information is passed to every server. In this way, a requester actor can always find the service actor it needs to contact.

Table 6- 2: test-taking use case

Use Case Name	Test-taking	
Goal in Context	A test-taker participates in an adaptive test administered by the system.	
Preconditions	n/a.	
Success End Condition	The test-taker goes through the test and is presented with the result.	
Failed End Condition	The test-taker is advised of the failure reason.	
Primary Actors	A test-taker.	
Trigger	A request for test-taking is generated.	
DESCRIPTION	**Step**	**Action**
	1	Having selected a test, the test-taker submits a request to the testing user interface to start the test.
	2	The testing user interface contacts the *ManagementService* to locate the *AdaptiveTestManager* service.
	3	The testing UI requests the *AdaptiveTestManager* service to send a question back.
	4	The *AdaptiveTestManager* service estimates the test-taker's current ability.
	5	The *AdaptiveTestManager* service contacts the *ManagementService*

		to locate the database repository service.
	6	The *AdaptiveTestManager* service contacts the *CATDbAccess* service to retrieve the question that it considers to be the most appropriate for the current test-taker.

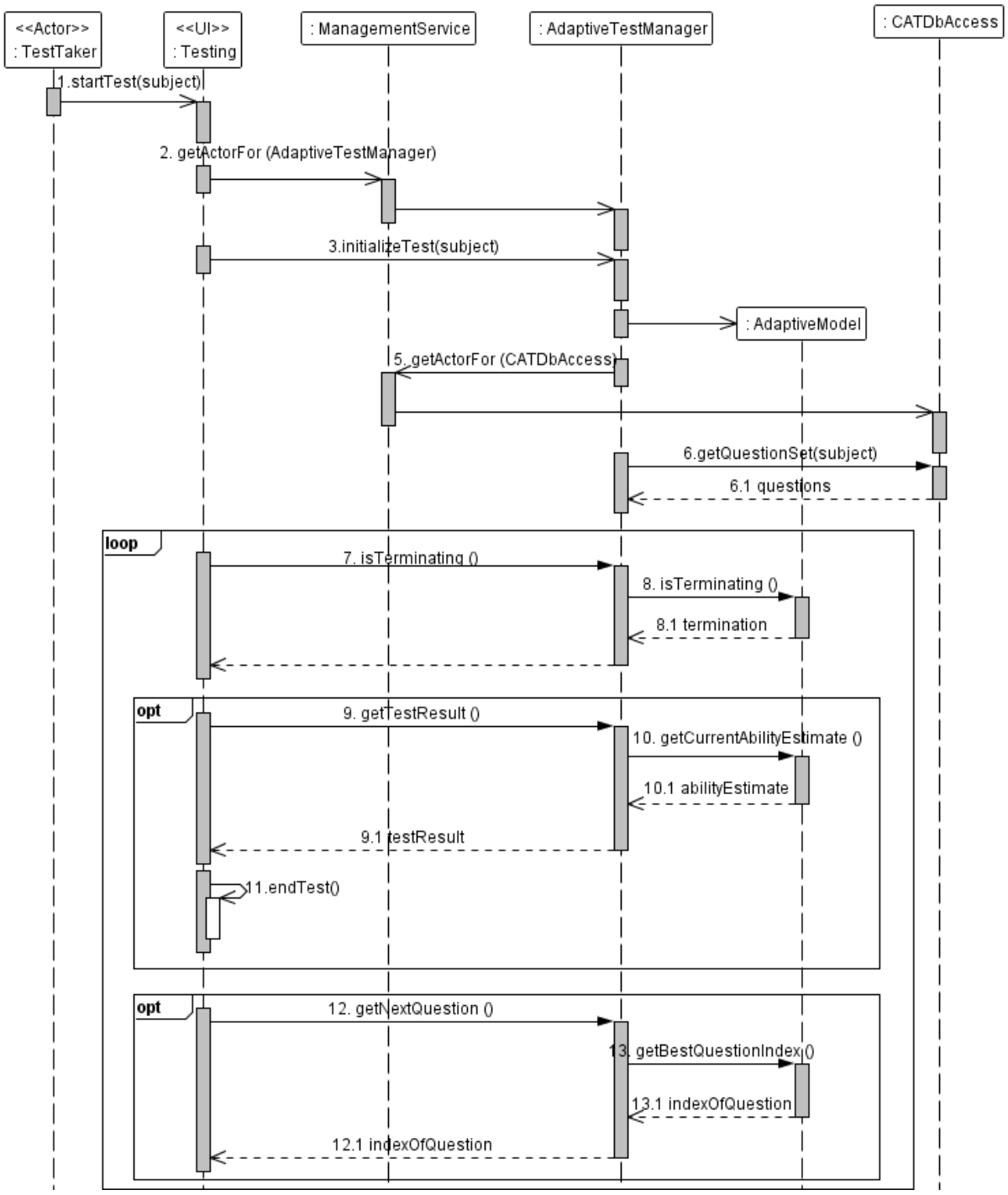


Figure 6- 3: Sequence diagram for the "Test-taking" system use-case

6.6.2 CLASS MODEL FOR THE TEST-TAKING USE-CASE

The testing-taking use case is the implementation of the proposed adaptive testing method. The implementation was achieved using the proposed distributed framework. The class model for this use case is as follows:

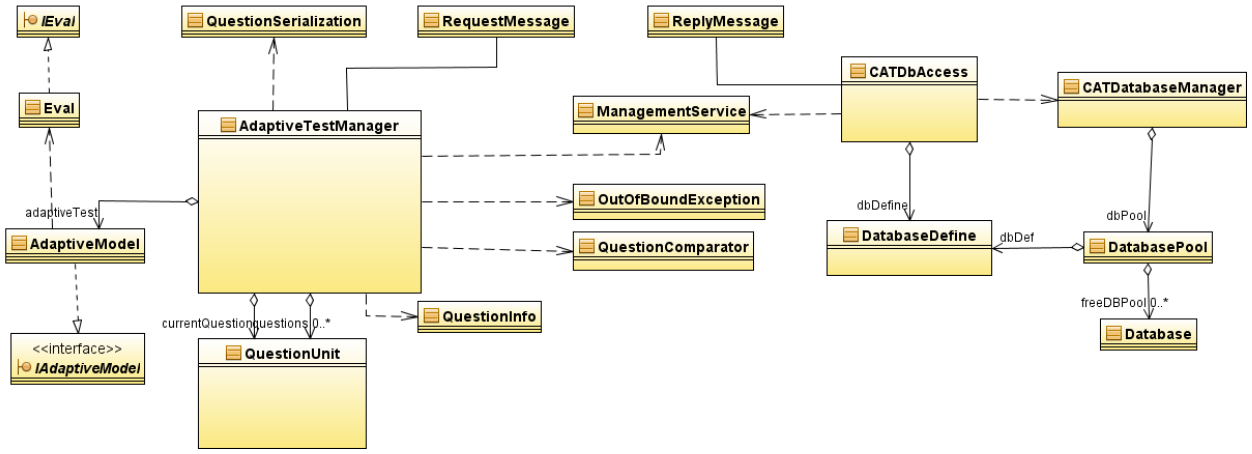


Figure 6- 4: class diagram for the "Test-taking" use-case.

Please note on the above class diagram the following two points. (1) The testing user interface was not included in the above class diagram for the reason that it is rendered using the Stripes framework, which will be introduced in the later sections. (2) The *AdaptiveTestManager* service and *CATDbAccess* service can be combined into one service to improve the system performance. They were divided into two services to demonstrate the concept of distributed services.

The above class diagram consists of three parts: two functional services and one management service. The two functional services are the *CATDbAccess* service and *AdaptiveTestManager* service. They communicate with each other via the management service.

CATDatabaseManager is the management class of *DatabasePool*. *DatabasePool* is a connection pool which maintains several *Database* instances. Due to the expensive cost of database connection, the accesses which would have required connections with the database are redirected to

the *DatabasePool*. This has greatly improved the performance by simplifying the database access procedure. It has also provided a means of resource sharing. *DatabaseDefine* is the class that encapsulates the database definition. *CATDbAccess* is the service that invokes the *CATDatabaseManager* to access the system database and reads the database configuration file for the parameters used for setting up a connection with the designated database.

RowSet was used as the interface to encapsulate database access and forward data from the *CATDbAccess* service to the *AdaptiveTestManager* service actor across the network. *RowSet* is a disconnected and serialisable version of the JDBC *ResultSet* interface. It extends the *ResultSet* interface and therefore inherits all the methods of *ResultSet*. The separation design of the *CATDbAccess* service and the *AdaptiveTestManager* service is simply a demonstration of the system's across-network communication capability. The use of the *RowSet* interface is simply because of its serialisable feature. However, the *RowSet* interface attempts to load and hold all the data in the memory at one time, which slows down the processing speed and consumes a considerable amount of memory resource. Therefore in consideration of system performance and resource optimisation, the above two services should be combined and deployed on the same server; while the *ResultSet* interface should be used to speed up processing and reduce memory usage.

The *AdaptiveTestManager* service dynamically estimates the test-taker's ability and picks up the most appropriate multiple choice question for him/her. It sends an instance of *RequestMesage* to the *CATDbAccess* service when it needs to access a data set. Then the *CATDbAccess* service responds with an instance of *ReplyMessge* which contains the requested data set encapsulated by a *RowSet*. Both *RequestMesage* and *ReplyMessge* extend from the *Message* class, which is a serialisable abstract class included in the system foundation class package.

The *Eval* class is a collection of numerical evaluation methods, which provides the support for derivative and integral computation. The *AdaptiveModel* class implements the proposed adaptive testing method using the computation components of the *Eval* class.

The *QuestionUnit* class represents the structure of a question. The *QuestionSerialization* class provides methods to serialise and deserialise a question. The structure of QuestionUnit was originally defined using the Apache Thrift framework.

6.6.3 PRESENTATION FRAMEWORK

The system uses Java Server Pages (JSP) as the scripting language and Stripes as the presentation framework. JSP is a server-side technology for creating dynamic web contents. Stripes is a high-quality and easy-to-use presentation framework created to overcome the drawbacks of Struts. It supports such functions as property binding, indexed properties, validation, and multi-event actions, with the requirements of simple setting. The following section briefly introduces the application of JSP and Stripes in this research.

To use Stripes, there are basically two configuration places to be made in the web application's web.xml file. Firstly, the Stripes filter needs to be configured to tell the system the root package for scanning the classpath of the web application's action beans. Secondly, the Stripes dispatcher servlet must be specified as to how the request to the Stripes framework should be processed. In this case, the Stripes configuration in the web.xml is as follows:

web.xml

```
<filter>
    <display-name>Stripes Filter</display-name>
    <filter-name>StripesFilter</filter-name>
    <filter-class>
      net.sourceforge.stripes.controller.StripesFilter
```

```
        </filter-class>
        <init-param>
            <param-name>ActionResolver.Packages</param-name>
            <param-value>actionbeans</param-value>
        </init-param>
    </filter>
    <servlet>
        <servlet-name>DispatcherServlet</servlet-name>
        <servlet-class>
            net.sourceforge.stripes.controller.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>DispatcherServlet</servlet-name>
        <url-pattern>*.action</url-pattern>
    </servlet-mapping>
```

Text Box 6- 8: web.xml

In the above configuration, the init-param tag specifies the action package root as actionbeans. All the subfolders under the actionbeans directory are automatically included. When a form is bound with an action bean url, the filter searches through the actionbeans folder and all of its subfolders to locate the action bean. The servlet tag and servlet-mapping tag specify the servlet class and the url pattern of the servlets.

Prior to using the Stripes framework, this project created a taglibs.jsp file which imports the Stripes tag library. Every jsp file in the SmartTester project which requires the use of Stripes functionality includes the taglibs.jsp at the beginning of the file. The script in the taglibs.jsp which is used to

import the Stripes tag library is listed below:

taglibs.jsp

```
<%@ taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
```

Text Box 6- 9: taglibs.jsp

To bind a form with an action bean, the action bean url must be specified in the form. The following script uses the stripes:form tag to bind the question presentation form with an action bean named Test.action.

questionpresenter.jsp

```
<stripes:form action="/Test.action">
   <stripes:errors/>
      <table width="400" border="0">
         <tr>
            ${actionBean.currentQuestion.description}
         </tr>

         <c:forEach items="${actionBean.currentQuestion.options}" var="option" varStatus="status">
            <tr>
               <td>
                  <stripes:radio name="userInformation.userChoice" value="${status.count}"
                                 id="${status.count}" />
                  <stripes:label for="${status.count}">${option}</stripes:label>
               </td>
            </tr>
         </c:forEach>
```

```
        <tr>
            <td colspan="2"> </td>
        </tr>
        <tr>
            <td colspan="2">The correct answer for this question is:</td>
            <td>${actionBean.currentQuestion.answers}</td>
        </tr>
        <tr>
            <td colspan="2">Your current ability estimate is:</td>
            <td>${actionBean.adaptiveTestManager.currentAbility}</td>
        </tr>
        <tr>
            <td colspan="2"> </td>
        </tr>
        <tr>
            <td colspan="2"><stripes:submit name="NextQuestion"
                                value="next question"></stripes:submit></td>
        </tr>
    </table>
</stripes:form>
```

Text Box 6- 10: questionpresenter.jsp

The above question presentation form is responsible for displaying a multiple-choice question and acquiring a user response. A new question is displayed every time the user selects a question option and clicks the “next question” button. The “next question” button is associated with the nextQuestion resolution in the action bean. A resolution is implemented when the binding action is triggered. The following excerpt from the action bean illustrates the content of the nextQuestion

resolution.

TestActionBean.java

```
@ValidationMethod(on = "nextQuestion")
        public void validateUserChoice(ValidationErrors errors) {
                if (null == userInformation.getUserChoice()) {
                        errors.add("userInformation.userChoice", new SimpleError(
                                        "Please select one answer!"));
                }
    FlashScope.getCurrent(getContext().getRequest(), true).put(this);
        }

  @HandlesEvent("StartTest")
        public Resolution startTest() {
    adaptiveTestManager = new AdaptiveTestManager();
    currentQuestion = adaptiveTestManager.getCurrentQuestion();
                return new RedirectResolution("/questionpresenter.jsp").flash(this);
        }

  @DefaultHandler
  @HandlesEvent("NextQuestion")
  public Resolution nextQuestion() {
    //acquires user choice
    Set<Integer> answers = new HashSet<Integer>();
    int userChoice = Integer.parseInt(userInformation.getUserChoice());
```

```
        answers.add(userChoice);
        //update test status
        try {
            adaptiveTestManager.getAnswersAndUpdate(answers);
        }catch (NullPointerException e) {
            System.out.println("adaptiveTestManager returned null");
            e.printStackTrace();
        }
        //Shall test finish?
        if (adaptiveTestManager.isTestInformationEnough()) {
            return new RedirectResolution("/testcompletion.jsp").flash(this);
        } else if ((currentQuestion = adaptiveTestManager.getCurrentQuestion())==null){
            return new RedirectResolution("/testabortion.jsp").flash(this);
        }

        return new RedirectResolution("/questionpresenter.jsp").flash(this);
    }
}
```

Text Box 6- 11: TestActionBean.java

A noticeable point is that there are two event handlers in the above script excerpt. The startTest resolution is responsible for the test initiation event. The Stripes framework allows multiple Stripes forms to be bound with multiple event handlers within a single action bean.

6.6.4 THE ADAPTIVE TESTING LOGIC

The adaptive testing service actor of the system follows a designated logic flow to implement adaptive e-Assessment. The logic flow is depicted as below:

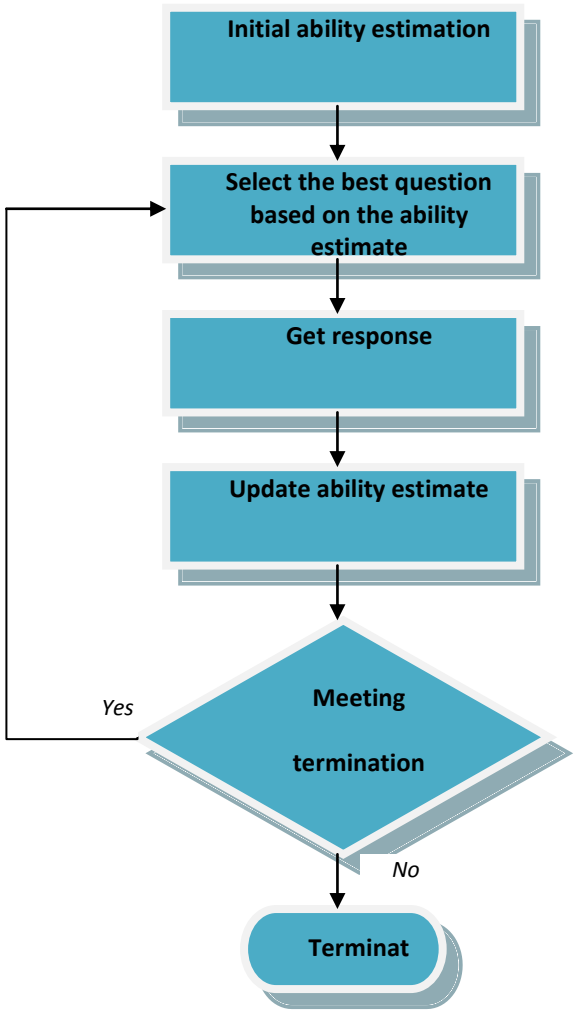


Figure 6- 5: the designated logic flow of the adaptive testing engine

In the above logic flow, three procedures – select the best question based on the ability estimate, update ability estimate and meeting termination criteria – are extended from the chosen adaptive model. In view of the fact that the designated adaptive models are a ramification of Item Response Theory models, a java interface was defined to achieve the compatibility of extended adaptive models.

IAdaptiveModel.java

```
package testengine;


/**
 *
 * @author Ben
 */
public interface IAdaptiveModel {


    /**
     * select the best question (i.e., the most informative question)
     * @return the index of the most informative question
     */
    public int selectBestQuestion ();


    /**
     * update currentAbility based on responseHistory and testHistory
     */
    public void updateEstimatedAbility ();


}
```

Text Box 6- 12: IAdaptiveModel.java

The adaptive models that implement the above interface must provide a concrete implementation for the defined methods to give instructions to the system about how to select the best question and the rules to update the estimated ability.

Since calculus may be frequently involved in the process of question selection and ability estimation, the finite difference approximation was used in the testing system.

6.6.5 EVALUATION OF THE FRAMEWORK

Since an application framework is in fact a set of prefabricated software building blocks or libraries, it is difficult to evaluate it from the aspect of functionalities. In this research, the evaluation of the implemented framework was based on the following five goods characteristics of e-Learning/e-Assessment systems discussed in chapter 2: interoperability, reusability, accessibility, scalability, and affordability. The interoperability and reusability of the applications developed based on the framework were fundamentally supported by the use of information description standards: XML and Thrift. XML stands for eXtensible Markup Language (XML), which is markup language for creating machine-readable documents. XML was used in the framework as the standard format to store and retrieve the configuration files. Thrift (Slee *et al*., 2007) is a cross-language framework originally developed at Facebook which supports efficient and reliable cross-language communication among computer programming languages such as C++, Java, Python, PHP, Ruby, Perl, and C#. The implementation of the Thrift framework combines a language neutral software stack implemented across numerous programming languages and an associated code generation engine that transforms a simple interface and data definition language into client and server remote procedure call libraries (Slee *et al*., 2007). In the framework, important and frequently used data structures such as question and test class were saved in the language neutral Thrift format so that they could be reused or imported into projects written in different programming languages. The use of XML and Thrift ensures the interoperability and reusability of the information. In addition, the

framework was coded in the Java programming language. The object-oriented feature of the Java programming language also contributes to the reusability of components of the framework.

The accessibility, scalability, and affordability characteristics of the applications using the framework are fundamentally supported by the peer-to-peer distributed architecture of the framework. The web-based feature of the framework removes the geographic and time restrictions and enhances accessibility. The framework provides libraries to support communication across the components deployed on different computers within a network. An application developed based on such a framework is scalable and can be easily extended. The framework is based on the peer-to-peer distributed architecture. Therefore it has the potential[17] to provide cost-effective solutions to increased demand for computational power.

In view that the accessibility, scalability, and affordability characteristics were actually supported by the peer-to-peer distributed architecture of the framework, there is a need to test the communication functionality between service providers and service requesters deployed on different computers. Testing a distributed framework is a highly complex and time-consuming task due to the difficulty of deploying the entire system in real settings. In order to simplify the configurations in the testing, the communication functionality provided by the framework was tested based on a simple example application deployed on three computers in a local area network. All three computers were installed with Java Runtime Environment 1.6.0_17, and their ip addresses were 10.57.10.246, 10.57.10.247, and 10.57.10.248, respectively.

In the testing, three web-based components were derived from the library provided in the framework and they were respectively deployed on the above three computers. The three components are respectively responsible for a) inter-connecting and managing distributed modules,

[17] The word potential was used here because distributed architecture provides the fundamental support for collaborative computation (Schollmeier, 2002). However, the functionalities of monitoring computational load and automatically coordinating computational load were not implemented in the current framework.

b) requesting service, and c) providing service. The inter-connecting and managing module was deployed on the computer with ip address 10.57.10.247. The service provider was deployed on the computer with ip address 10.57.10.246, which sends a message back to the service requester with "re:" prefix attaching to the incoming message. The service requester was deployed on the computer with ip address 10.57.10.248, which sends a message to the service provider and requests response. The startup sequence for these three distributed modules is 1) the inter-connecting and managing module; 2) service provider; 3) service requester. In the testing, the inter-connecting and managing module successfully received connections from the other two computers and the service requester successfully received the replied message from the service provider.

Due to time limitations, two indicators of the performance of an application framework – the development complexity (Gallagher *et al.*, 2008) and the performance of the framework – were not evaluated. The development complexity and the performance of the framework would be evaluated as part of future work. Also the library for monitoring and allocating computational load across the distributed components was not implemented.

CHAPTER 7 DISCUSSION

Adaptive assessment promises a potential to adminster e-Assessments in a more accurate and intelligent way. However, the birth of this assessment solution has also brought some new problems which did not exist in conventional e-Assessments. Much attention has been devoted to perfecting adaptive assessment. Some commonly recognised issues in adaptive assessment are content balancing, item exposure, multiple scales, termination and web-based adaptive assessment. Content balancing is an issue concerning the assessment contents presented to examinees with different proficiency levels (Kingsbury & Zara, 1991). An example is language proficiency assessment for foreign language learners. The proficiency of a language (reading, writing, listening and speaking) can be scaled on a single latent trait, but it covers different assessment contents and patterns. Because of educational pattern and environment, students from some countries tend to do better in reading and writing than listening and speaking in terms of foreign language study. This could result in students with lower proficiency being given more reading- and writing-related questions, while students with higher proficiency are given more listening- and speaking-related questions. Content balancing involves research to ensure students with different proficiencies receive a similar proportion of test contents.

Item exposure is another popular research topic in the adaptive assessment domain. Depending on the ability distribution of the examinees and the information structure of the question bank in an adaptive assessment, each question in the question bank has a different opportunity to be exposed to the examinees which is known as item exposure rate. Similar to content balancing, the over-

exposure rate of some questions may be unwanted in some assessment settings. For example, a high-stakes examination whose question bank will be repeatedly used may not want some questions in its question bank to be disclosed to the students beforehand. In order to meet the requirement of these assessments, some exposure rate controlling methods have been proposed (Hetter & Sympson, 1997; Revuelta & Ponsada, 1998).

Lilley and Barker (2004) studied the possibility of allowing modification of previously entered answers in adaptive assessments. Their research findings indicate that modifying the previously entered answers on an adaptive assessment usually does not have a significant effect on the performance of examinees. Barker, Lilley, and Britton (2006) also conducted a study on the automated feedback adaptive assessment using Bloom's taxonomy of the cognitive skills.

With the development of the World Wide Web (WWW), web-based adaptive assessment has gradually become a popular research topic. Different from many previous studies, this research focuses on the applicability of web-based adaptive assessment in normal classroom environments. It is believed in this research that the applicability of adaptive assessment in daily teaching environments is the decisive factor in the prevalence of this technology. Many previous studies have proven that adaptive assessment, especially IRT-based adaptive assessment, has advantages over conventional assessment in aspects such as accuracy and test length. However, these studies were mostly conducted in ideal environments, i.e., with a large number of observed responses and a large-sized question pool. Due to the complexity of applying adaptive assessment, it is usually only available in high-stakes examinations such as GRE and GMAT. The two research questions are:

1. What is the performance of a three-parameter-logistic-item-reponse-model-driven e-Assessment system in a pragmatic environment, i.e. a regular class with about 30 students[18]?

[18] A regular class is defined by Blatchford (2009) as a class with around 23 students

2. Can a distributed architecture based framework support the good characteristics of adaptive e-Assessment applications, especially in terms of scalability and affordability?

The research started from using two groups of real students (30 students in each group) on Cisco courses to compare the performance of the implemented adaptive assessment to fixed-form computer-based assessments in such a setting. Then a web-based framework was designed and implemented to provide fundamental support for the good characteristics in the development of adaptive e-Assessment systems. The intention of this research was to study the feasibility of applying IRT-based adaptive assessment to low-stakes assessment and to propose a web-based framework which provides basic facilities to accommodate customised adaptive assessment strategies. In response to the aim and objectives brought up in the first chapter, the following eight aspects were investigated during the research:

1. Investigate assessment theories with a focus on adaptive assessment theories.
2. Conduct literature review on framework for web applications.
3. Investigate the pragmatic problems in the implementation of adaptive e-Assessment.
4. Design and implement an adaptive e-Assessment algorithm for normal classroom environments.
5. Evaluate the performance of the algorithm implemented in objective 3 via real-data simulation. The test length and the precision of ability estimation are used as the two indicators of the performance of the adaptive e-Assessment algorithm. Fixed-form e-Assessments were used as a comparison in this objective.
6. Investigate the impact of the size of the question bank on the efficiency of assessment via Monte Carlo simulation.
7. Propose and implement an e-Assessment framework which supports the 7 characteristics of good e-Assessment systems (Khaskheli, 2004) and provides solutions to the pragmatic problems identified in objective 2.
8. Evaluate whether the implemented framework achieves the objectives outlined in objective 5.

Investigate assessment theories with a focus on adaptive assessment theories.

Test administrators face two completing assessment strategies when implementing a computer-based test. They can either adopt adaptive e-Assessment or fixed-form e-Assessment, where the former takes advantage of modern test theories and the latter is a migration of conventional paper-and-pencil tests. Before comparing the difference of these two assessment strategies in application, chapter 2 briefly reviewed the theories and models of adaptive e-Assessment and CTT, where the latter is the theoretic foundation of fixed-form e-Assessment.

The major value of adaptive e-Assessment is that the ability scores estimated by it are invariant. In other words, the ability estimates by adaptive e-Assessment would not be dependent on the particular selection of test items. This makes student ability scores obtained via different test items comparable. It also makes test results more convincing. Besides, adaptive e-Assessment can theoretically bring shorter tests and more accurate ability estimates. However, these improvements are achieved at the expense of much stricter assumptions and more complex implementation. It requires in-depth empirical study to balance the gains and cost of adaptive e-Assessment in daily application.

Conduct literature review on frameworks for web applications.

A framework is a set of prefabricated software building blocks or libraries which provide common functionalities that programmers can reuse, extend, or customise for specific computing tasks (Taligent Inc., 1993). The purpose of frameworks is to allow software developers, rather than starting from scratch, to leverage a codebase which provides common logic, thus limiting the time required to build an application and reducing the possibility of introducing new bugs (Angelov *et al.*, 2006). The literature review on frameworks for web applications was conducted in this research to find out the common features of good web applications so as to support these features with a framework. Two issues with the implementation of adaptive e-Assessment were also investigated to allow solutions stemming from the framework.

Investigate the pragmatic problems in the implementation of adaptive e-Assessment.

Two pragmatic problems were identified in the research. Firstly, bearing in mind that there are different approaches to adaptive e-Assessment in different contexts, it is ideal for an e-Assessment framework to have the flexibility to accommodate customised e-Assessment models. Secondly, some complex adaptive e-Assessment models are demanding in terms of computational power. Therefore, it is useful to spread the computational load across a number of servers via distributed architecture.

Design and implement an adaptive e-Assessment algorithm for normal classroom environments.

IRT is a body of psychometric theories based on the application of mathematical models to measuring latent traits. It is often adopted as the theoretical basis in adaptive assessments and was used in this research too. The application of an adaptive assessment system can be broken down into two stages: the item calibration stage and the adaptive testing stage.

For the item calibration stage, MLE and LSE were used to estimate the parameters for every question and the Chi-Square goodness-of-fit index was used to examine whether a mathematical model fits a question. The real-data simulation results indicated that LSE yielded better parameter estimation than MLE in the context of the small data training set used in the research. However, the parameter estimation technique Expectation-Maximisation (EM) was not tested in the research due to time limitation. For the data set which is incomplete or has missing values, EM is generally regarded as a good method for better maximum-likelihood estimation of the parameters (Nodelman *et al.*, 2005).

For the adaptive testing stage, the Fisher Information was used as the criteria for the best question to be presented next. This technique together with the Newton-Raphson method greatly enhanced the speed of convergence in ability estimation. Moreover, Bayes' theorem has been proven to be an efficient classification method when the purpose of the e-Assessment is to classify the examinees.

Evaluate the performance of the algorithm implemented in objective 3 via real-data simulation. The test length and the precision of ability estimation are used as the two indicators of the performance of the adaptive e-Assessment algorithm. Fixed-form e-Assessments were used as a comparison in this objective.

The purpose of real-data simulation was to test how much shorter an e-Assessment can be when being presented by adaptive e-Assessment compared to fixed-form e-Assessment, while maintaining the psychometrical properties unchanged. The findings of this research suggest that in a pragmatical environment where only around 30 observed answers to one question is usable for question parameter calibration, IRT-based adaptive assessment may still exhibit certain but not significant advantages over fixed-form assessment. These advantages mainly lie in the aspects of measurement accuracy and test time.

Under the assumption of the correctness of the Item Response Theory, adaptive assessment used in this research did show some performance improvement over fixed-form assessment in terms of overall measurement accuracy and test length. The result of the student's t-tests also suggests that the improvement of measurement was insignificant. It is believed that the improvement could be further boosted with the increase of sample size at the item calibration stage. From the angle of question setting, the results suggest these questions are not of the universal difficulty level. It is therefore reasonable to allocate questions different scores according to their difficulty levels.

Due to the limitations of sample size and question bank, this research did not completely explore the psychometric properties of the questions such as the discrimination parameter and the guessing factor. Ideally, these two parameters should be estimated together with the difficulty parameter based on a group of conditional equations for the reason that all the three parameters collaboratively describe the psychometric properties of a question in the 3PL IRT model. However, the small sample size of the real student responses collected made it difficult to estimate all three parameters simultaneously. Therefore only the value of the most important parameter – difficulty

level – was statistically estimated based on the student responses, while the values of the other two parameters were acquired by approximation. The difficulty parameter was deemed the most important parameter because it is the only parameter in the Rasch model, which was the original form of the 3PL IRT model.

Investigate the impact of the size of the question bank on the efficiency of assessment via Monte Carlo simulation.

The use of Monte Carlo simulation in this research was designed to investigate the impact of question bank size on the efficiency of the adaptive e-Assessment algorithm. The randomly generated questions represented an unoptimised question bank, which is the case when the question designer has no knowledge of the future candidates. An optimised question bank here refers to a question bank with a large proportion of questions designed to suit the ability range of the prospective candidates. The results indicate that for the implemented adaptive e-Assessment algorithm, the accuracy of ability estimation increased greatly when the size of the question bank increased from 100 to 200.

Propose and implement an e-Assessment framework which supports the seven characteristics of good e-Assessment systems (Khaskheli, 2004) and provides a solution to the pragmatic problems identified in objective 2.

Because various testing models may have different assumptions and testing strategies, it is beneficial to design an e-Assessment application framework which can accommodate new testing modules as plug-ins. This framework ideally should allow local and remote testing service modules to connect to the system to provide service. In this consideration, the idea of service-oriented design was adopted in the implementation of the system framework.

The core of the system is the management module with a built-in registry mechanism. The management module can be started independently and other services can be connected to and

registered on it. This architecture design offers expandability for future test service modules and should be in favour of distributed deployment of test service modules. However, the distributed deployment of system services was not tested due to time limitation.

Evaluate whether the implemented framework achieves the objectives outlined in objective 5.

The evaluation of the implemented framework was based on the five goods characteristics of e-Learning/e-Assessment systems discussed in the literature review: interoperability, reusability, accessibility, scalability, and affordability. The framework supports interoperability and reusability by the use of information description standards XML and Thrift. The framework implements object transitions over networks based on the Java serialisation interface and uses the Apache Thrift framework to provide cross-language support for interoperability amongst programmers using different programming languages. The peer-to-peer distributed architecture of the framework provides support for accessibility, scalability, and potential support for affordability. Distributed adaptive e-Assessment applications can be easily constructed based on the libraries provided by the framework. In the evaluation of the framework, a simple distributed application was created to test the built-in communication functionality of the framework and prove the capability of the framework in terms of extensibility. The successful communication between the components in the sample application verified the communication mechanism adopted by the framework. As shown in the sample application, future components can be easily created and connected to the existing application and this architecture of the framework enhances the scalability of the future applications based on it.

Due to time limitations, only the peer-to-peer communicational mechanism of the framework was tested during the course of this research. The library for monitoring and allocating computational load across the distributed components was not implemented.

This chapter discussed the research work conducted with regard to the six research objectives. The overall research aim and the answers to the two research questions will be concluded and discussed in the next chapter.

CHAPTER 8 CONCLUSIONS AND FUTURE WORK

8.1 CONCLUSIONS

The research proposed an extensible distributed framework for e-Assessment systems and compared the performance difference between adaptive e-Assessment and computer-based fixed-form assessment with a small training data set. It made a contribution to the application of e-Assessment by proposing an innovative universal framework for e-Assessment applications and investigating the performance of the IRT-based algorithm with the availability of limited observed data. These research results meanwhile also reflect the fact that assigning equal marks to every question in Cisco chapter tests is sometimes not reasonable enough.

The results of this research attempted to answer the following two questions:

A. *Is it beneficial to introduce IRT-based adaptive e-Assessments to low-stakes assessments?*
B. *What is the most appropriate framework for web-based e-Assessment systems?*

The answer to the first question depends on the nature and the purpose of the assessment. Adaptive e-Assessments have advantages in terms of the test time and test precision, but require strong assumptions and complex test settings. In the meantime, IRT-based adaptive e-Assessments need a large amount of historical responses as a "training set". Additionally, adaptive e-Assessments require frequent updates of question banks in order to alleviate the overexposure of certain items. All the above requirements seem to have limited the application of adaptive e-Assessments to high-stakes examinations. However, the results of this research indicated that IRT-based adaptive e-Assessments may still have relative advantages in assessment efficiency compared to fixed-form e-Assessments. This advantage comes from the relatively objective method in determining the psychometric properties of questions.

The implemented framework provides a mechanism to facilitate the communication of modules distributed across the network, which is a key issue in the design of web applications. With the support of the framework adding new features to an existing system, a user only needs to extend from the basic service types and specify the new message-handling processor.

Subsequently, it is time to revisit the two research questions posed in the first chapter and conclude the research findings in light of them. The two research questions are as follows:

1. What is the performance of a three-parameter-logistic-item-reponse-model-driven e-Assessment system in a pragmatic environment, i.e. a regular class with about 30 students[19]?
2. Can a distributed architecture-based framework support the good characteristics of adaptive e-Assessment applications, especially in terms of scalability and affordability?

For the first question, the empirical results in the research indicated that a three-parameter-logistic-item-response-model-driven e-Assessment system can still outperform the conventional fixed-form e-Assessment even in a class with only 30 students. The implemented adaptive e-Assessment

[19] A regular class is defined by Blatchford (2009) as a class with around 23 students

generated ability measurements with fewer errors and less testing time (fewer questions) than those of fixed-form Cisco e-Assessment. The fact that the adaptive e-Assessment outperformed the fixed-form e-Assessment implied that those multiple choice questions were not equally difficult for the candidates. However, the improvement of the adaptive e-Assessment was not significant due to the small-scale pilot test data. The experiment indicated that adaptive e-Assessment relies heavily on the observed response from pilot tests. In order to improve the performance of adaptive e-Assessment when there are only small-scale pilot test data available, test administrators will need to use a simple IRT model and reduce the number of parameters to be estimated. The use of expert calibration (Barker, 2009) will be a good alternative solution in such a situation.

For the second question, the implemented framework proved the feasibility of sponsoring an adaptive e-Assessment system with a distributed framework. The implemented framework consists of a number of libraries, ranging from peer-to-peer distributed components to database connection components, which jointly provide the support for the development of a distributed adaptive e-Assessment system. The framework was especially designed to tackle two practical issues in the development of adaptive e-Assessment systems: scalability and affordability. In terms of scalability, the framework supports adding different distributed modules without modifying the core of the framework. The design of the framework defines the base classes and interface classes to implement the framework-compatible distributed services. A framework-compatible service can be easily customised and added to an existing network of the framework-compatible services based on the libraries provided. The framework also provided a good foundation for system affordability. With the possibility of deploying components of an adaptive e-Assessment system across the network, the framework has actually established the foundation for multiple computers to work collaboratively to provide increased computational power. However, the drawback of the framework is that the load balancing functionality was not implemented. The e-Assessment developers have to implement their own schemes of load balancing.

8.2 CONTRIBUTION TO KNOWLEDGE

Although IRT-based adaptive e-Assessment has been proved to outperform fixed-form e-Assessment in terms of test length and measurement precision, it has barely been introduced into real classroom applications due to poor usability. Two important underlying reasons for the poor usability of the IRT-based e-Assessment systems are the theoretically strict prerequisites and possibly high computational demand when delivering a testing service to a large group of people.

This research firstly explored the performance of a common IRT model – the three-parameter logistic model – in a pragmatic classroom environment in higher education. With merely 30 students' responses as observed data, the results revealed that IRT-based adaptive e-Assessment may still have advantages over fixed-form e-Assessment with a limited training set. Monte Carlo simulation was also employed to explore the theoretical idea size of the question bank for the three-parameter logistic IRT model. The research secondly addressed a web-based peer-to-peer distributed framework designed to provide expandability, scalability and cost-effective computational support for adaptive e-Assessment algorithm plug-ins.

8.3 APPLICATIONS OF THE RESEARCH

The recent developments in computer hardware and software have called for new visions in the application of computers as e-Assessment tools. Contemporary computers can be used not only as a medium for innovative and interactive multimedia such as sound, graphics, videos and flashes; they also provide possibilities for web-based, adaptive and intelligent e-Assessments.

For e-Assessment researchers, this research compared the performance of adaptive e-Assessment and computer-based fixed-form assessment in a daily teaching environment. For most researchers, it is not easy to access a large quantity of observed question responses. Therefore this research examined the performance of adaptive e-Assessment under the setting of a normal classroom environment, which has around 30 students in one class. The method used in this research can be applied to validate the structure of an assessment. Ideally a well-structured assessment should

maintain a prioritised proportion of questions of different difficulty levels. The analysis method can be used to find out if there is difficulty difference in a group of questions.

For e-Assessment tool developers, this research provides a web-based extensible framework to accommodate distributed testing modules. Most adaptive assessment models are derived from test theories with strong assumptions such as IRT. Therefore these models have different applicable scopes. Developers often want to integrate different assessment strategies into the system to fit for different assessment environments. In addition, some adaptive e-Assessment models like multidimensional IRT models require heavy computational power (Lin *et al*., 2007a). The proposed framework allows different modules to be connected to the service network from different servers, hence providing the potential to reduce the computational burden for each server and increase extensibility of the application.

8.4 LIMITATIONS OF THE RESEARCH

Constrained by restricted resources, there are several limitations in this research. First of all, due to the limited number of questions and real students, the research is based on real-data simulation using the unidimensional dichotomous IRT model. Therefore the results of this study are limited to the generalisations that can be made.

Secondly, the adaptive assessment method used in this research has a number of limitations. The training set used in the pilot test, the observed responses from real students, is relatively small. The small training set could have led to biased results. Barker's (2009) experiment shows that introduction of expert opinions at the stage of the pilot test will improve the measurement of examinees' abilities. Another limitation of the adaptive assessment method is the limited size of the question bank. A bigger question bank may help improve the precision of the measurement. The latter Monte Carlo simulation in chapter 4 indicated that an increase in the question bank would lead to a more precise ability measurement.

Thirdly, the proposed adaptive e-Assessment method was not tested within the proposed framework due to time limitations. Instead, the distributed module joining functionality, the message forwarding mechanism, and the database connection were tested, as they are considered as the core of the framework. Additionally, an optimised method about how to allocate the computational burden across the peer-to-peer network was not realised.

8.5 FUTURE RESEARCH

Adaptive e-Assessments employ the observed data obtained from pilot-tests to individualise and streamline the measurement process. By avoiding presenting irrelevant questions to an individual respondent, adaptive e-Assessments simultaneously reduce the number of questions needed in an assessment and increase the precision of measurement. In a Higher Education environment, individualising and streamlining assessments offer important benefits in both research and practice. Using fewer questions in assessments means improvement of efficiency. Plenty of time can be saved for learning and the fatigue caused by assessments can be reduced. Improved measurement precision is useful in ability diagnosis.

Unfortunately, the technology and infrastructure required to implement adaptive e-Assessment are currently not available for most researchers and educationalists. Existing adaptive e-Assessment tools are highly technical. Large banks of previously observed responses to questions, which are needed by implementation of adaptive e-Assessments, are not readily accessible. All the aforementioned downsides have made adaptive e-Assessment a technology beyond the reach of most educationalists. The current research attempted to investigate the performance of adaptive e-Assessment with limited accessible resources, and proposed a framework to facilitate the composition of web-based adaptive e-Assessment systems. However, the research is limited to the three-parameter logistic model and the integration of the adaptive assessment model and the framework has not been tested yet.

The future research will cover two major tasks. The first task is to extend the studies to other adaptive e-Assessment models to find a generalised adaptive strategy which has good performance and low requirements in practice. The second task is to integrate the tested adaptive e-Assessment strategies into the distributed framework to form a complete adaptive assessment system.

REFERENCES:

Abras, C., Maloney-Krichmar, D., Preece, J., 2004. User-Centered Design. In Bainbridge, W. *Encyclopedia of Human-Computer Interaction*. Thousand Oaks: Sage Publications.

Ackerman, T. A., 1991. *The use of unidimensional parameter estimates of multidimensional items in adaptive testing*. Applied Psychological Measurement, vol.13, pp.113-127.

Adams, R. J., Wilson, M. and Wang, W. C., 1997. *The multidimensional random coefficients multinomial logit model*. Applied Psychological Measurement, vol. 21, pp.1-23.

AICC, 2010. *The Aviation Industry CBT (Computer Based Learning) Committee*, available from http://www.aicc.org/index.html [accessed on 16th March, 2010].

Allen, M.J., & Yen, W. M., 2002. *Introduction to Measurement Theory*. Long Grove, IL: Waveland Press.

Alonso, G., Casati F., Kuno, H. and Machiraju, V., 2004. *Web Services*. Germany: Springer - Verlag Berlin Heidelberg.

Amber, S., 2004. *The Object Primer 3rd Edition - Agile Model Driven Development with UML*. USA: Cambridge University Press.

Anderson, J. R., 1990. *The adaptive character of thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., 1993. *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. & Lebi ère, C., 1998. *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Angelov, C., X. Ke, et al. (2006). A Component-Based Framework for Distributed Control Systems. *32nd EUROMICRO Conference on Software Engineering and Advanced Application,* pp.20-27

ARIADNE, 2001. ARIADNE Foundation. Available from http://www.ariadne-eu.org/ [Accessed on 16th March, 2010].

Arruabarrena, R. and Pérez, T.A., 2005. Una experiencia arbitrando incidencias producidas en pruebas de campo, *Proceedings of VI congreso nacional de Informática Educativa / I Simposio Nacional de Tecnologías de la Información y las Comunicaciones en la Educacion: SIntice-2005 (ADIE, CEDI'05)*, pp.161-166, Granada (Spain), September 2005, Thomsom Paraninfo, Granada (Spain).

ASVAB, 2007. *About ASVAB*. Available from http://www.military.com/ASVAB [Accessed on 17th May, 2011].

Attali, Y. and Burstein, J., 2006. Automated Essay Scoring With e-rater® V.2. *The Journal of Technology, Learning and Assessment*, 4(3), 2006

Bach, S., Haynes, P. & Smith, J.L., 2007. *Online Learning and Teaching in Higher Education*. Maidenhead: OU and McGraw Hill.

Baker, F., 2001. *The Basics of Item Response Theory 2nd edition*. Maryland, US: ERIC Clearinghouse on Assessment and Evaluation.

Banta, T.W. & Associates, 2002. *Building a Scholarship of Assessment*. San Francisco, US: Jossey-Bass, pp.3-25.

Barker, T., Lilley, M. and Britton, C. (2006). Computer Adaptive Assessment and its use in development of a student model for blended learning, Annual Blended Learning Conference 2006. pp.176-191.

Barker, T., 2009. An automated feedback system based on adaptive testing: extending the model, *Proceedings of the Interactive Computer Aided Learning Conference*, ICL2009, 23 - 25 September 2009, Villach/Austria.

Behrens, J.T., Mislevy, R. J., Bauer M., Williamson, D. M., Levy, R., 2004. Introduction to Evidence Centered Design and Lessons Learned from its Application in Global E-Learning Program, *International Journal of Testing*, 4(4), 2004, pp. 295 -301.

Bennett, R. E., 1999. *Using new technology to improve assessment*. Princeton, NJ:Educational Testing Service.

Berson, A., 1996. *Client/server architecture*. McGraw-Hill.

Betz, N.E. &Weiss, D. J., 1975. *Empirical and simulation studies of flexilevel ability testing* (Research Report 75-3). Minneapolis: University of Minnesota, Department of Psychology, Psychometric Methds Program, 1975.

Binet, A., Simon, and Th. (1905) Méthodes nouvelles pour le diagnostic du niveau intellectual desanormaux. *Année psychol.*, 1905, 11, pp.191-244.

Birnbaum, A., 1968. Some Latent Trait Models and Their Use in Inferring an Examinee's Mental Ability. In F. M. Lord, and M.R. Novick (Eds.) *Statistical Theories of Mental Test Scores*. Reading, MA: Addison-Wesley.

Black, P. and William, D., 1998a. Assessment and classroom learning. *Assessment in Education*, vol. 5, no. 1, March, 1998, pp.7-74.

Black, P. & William, D., 1998b. *Inside the black box : raising standards through classroom assessment*. London, Kings College.

Blackboard, 2007. *Blackboard*. Available from http://blackboard.com/products/Academic_Suite/index [Accessed on 17th Jan, 2007].

Blatchford, P., 2009. Class Size. In Anderman, Eric (ed.). *Psychology of Classroom Learning: An Encyclopedia*. Detroit: Macmillan Reference USA.

Bloom, B. S., 1984. The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring, *Educational Researcher*, vol.13, no. 6, 1984, pp.4-16.

Bock, R. D., and Mislevy, R. J., 1982. Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement*, 6, pp.431-444.

Bond, L. A., 1996. Norm- and criterion-referenced testing. *Practical Assessment, Research & Evaluation*, 5(2), Available from http://ericae.net/pare/getvn.asp?v=5&n=2 [Accessed on 3rd Jan, 2010].

Breakwell, G. M., Hammond S., Fife-shaw C. and Smith, J. A. (Eds.), 2006. *Research Methods in Psychology 3rd Edition*. London: Sage Publications, pp.190

Broemmer, D., 2003. *J2EE best practices: Java design patterns, automation, and performance*. Wiley.

Brooks, P. W. 2000. Internet Assessment: Opportunities and Challenges. *The 24th Annual IPMAAC Conference on Personnel Selection*. 4th June, 2000.

Browne, T., Hewitt, R., Jenkins, M. and Walker, R. 2008. 2008 Survey of Technology Enhanced Learning for higher education in the UK. Available from: http://www.ucisa.ac.uk/publications/~/media/290DD5217DA5422C8775F246791F5523.ashx [Accessed on 25th March, 2010]

Bull, J. & McKenna, C., 2001. *Blueprint for Computer Assisted Assessment*. London: RoutledgeFalmer.

Cambridge University Press, 2010. *Bioscope*. Available from http://www.cup.cam.ac.uk/uk/education/secondary_projectpage.asp?id=2500811 [Accessed on 24th March, 2010]

Carlson, R. D., 1994. *Computer-Adaptive Testing: a Shift in the Evaluation Paradigm*, Journal of Educational Technology Systems, 22(3), pp. 213-224

Chan, P., 2005. What learners learn through problem solving? *Teaching Methods*, vol. 9, no. 3, November 2005. Available from http://www.cdtl.nus.edu.sg/link/Nov2005/tm2.htm [Accessed on 24th July, 2006].

Chen, Po-His, 2006. The Influences of the Ability Estimation Methods on the Measurement Accuracy in Multidimensional Computerized Adaptive Testing. *Bulletin of Educational Psychology*, 2006, 38 (2), pp.195-211

Cisco, 2008. *IT Essentials II: Network Operating Systems*. Available from: http://www.cisco.com/web/learning/netacad/course_catalog/IT2.html [Accessed 30th March, 2008]

Condie, R., Livingston, K. and Seagraves, L., 2005. *Evaluation of the assessment is for learning programme: Final report and appendices*. Available from: http://www.scotland.gov.uk/Resource/Doc/47121/0020476.pdf. Accessed on 15th July, 2006.

Conejo, R., Guzmán, E., Millán, E., Trella, M., Pérez-De-La-Cruz, J.L. and Ríos, A., 2004. SIETTE: A Web-Based Tool for Adaptive Testing. *International Journal of Artificial Intelligence in Education*, vol. 14, 2004, pp.1-33.

Cooper, D. 2006. *Talk About Assessment: Strategies and Tools to Improve Learning*. Toronto, ON: Thomson Nelson.

Cristea, P., and Tuduce, R., 2005. 'Automatic Generation of Exercises for Selftesting in Adaptive E-Learning Systems: Exercises on AC Circuits', *Third 227 International Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia Amsterdam*, The Netherlands, 19 July 2005.

DC, 2010. *The Dublin Core Metadata Initiative*. Available from http://dublincore.org/ [Accessed on 16th March, 2010].

de Ayala, R.J., 2009 *The Theory and Practice of Item Response Theory*. New York: The Guilford Press.

Doe, C. L., 2004. A Look At … Web-Based Assessment. *MultiMedia and Internet at Schools*, 11 (2), March/April 2004. Available from http://www.infotoday.com/MMSchools/mar04/doe.shtml [Accessed on 18th May, 2011]

Downing, S. M., 2003. Item response theory: applications of modern test theory in medical education. *Medical Education*, 37(8), 2003

Dweck, C. S., 2000. *Self-theories: Their role in motivation, personality and development*. Philadelphia: Psychology Press.

Ecclestone estone estone, K., 2007 Commitment, compliance and comfort zones: the effects of formative assessment on 'learning careers' in vocational education, *Assessment in Education*. 14(3), pp.315-333.

Eggen, T., 2007. Choices in CAT Models in the Context of Educational Testing. *2007 GMAC Conference on Computerized Adaptive Testing*, 7 June 2007.

Ekins, J., 2008. Interactive mathematics e-quizzes using Open Mark. *MSOR Connections*, 8(3), 2008, pp. 21-24.

El-Bakry, H. M., and Mastorakis, N., 2009. *User interface for internet applications*. World Scientific and Engineering Academy and Society (WSEAS).

Embretson, S. E. and Reise, S. P., 2000. *Item response theory for psychologists.* Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

ETS, 2007. *ETS Website,* available from http://www.ets.org [Accessed 30th April, 2011].

Feng, M., Heffernan, N.T, Koedinger, K.R., 2006. Addressing the Testing Challenge with a Web-Based E-Assessment System that Tutors as it Assesses (Best Student Paper Nominee), *the 15th International World Wide Web Conference,* Edinburgh, Scotland.

Feng, M., Beck, J., Heffernan, N., and Koedinger, K., 2008. Can an Intelligent Tutoring System predict math proficiency as well as a standardized test? In Baker & Beck (Eds.). *Proceedings of the 1st International Conference on Education Data Mining.* Montréal, Canada, pp.20-21 June 2008.

Fennell, T., 2009. Stripes: an overview - guidin principles. Available from http://www.stripesframework.org/display/stripes/An Overview - Guiding Principles [Accessed on 5th June, 2011]

Fetzer, M., Dainis, A., and Lambert, S., 2011. Computerized Adaptive Testing in an Enployment Context, *SHLPreVisor White Paper 2011*. Available from http://central.shl.com/SiteCollectionDocuments/White Papers, Guidelines and Other/White Papers/SHL PreVisor white paper - CAT in an employment context - 2011.pdf [Accessed on 30th May, 2011]

Flaughter, R. 1990. *Item Pools* In: Wainer, H. (Ed) (1990) *Computerized Adaptive Testing: A Primer*. New Jersey: Lawrence Erlbaum Associates.

Forsythe G. E., and Wirth N., 1965. *Automatic grading programs, Communications of the ACM,* 8(5), pp.275-278.

Frick, T. W., 1992. Computerized adaptive mastery tests as expert, *Journal of Educational Computing Research Systems,* 8 (2), pp.187-213.

Frieden, B. R., 2004. *Science from Fisher Information: A Unification (2nd Revised edition)*. Cambridge, UK: Cambridge University Press, pp.*23 – 30*.

Gallagher, K., Hatch, A., and Munro, M., 2008. Software Architecture Visualization: An Evaluation Framework and Its Application, *IEEE Transactions on Software Engineering,* 34(2), pp.260 - 270.

Gierl, M. J. and Ackerman, T., 1996. XCALIBRE — Marginal Maximum-Likelihood Estimation Program, Windows Version 1.10. *Applied Psychological Measurement,* 20(3), pp. 303-307.

Gilat, A., 2004. *MATLAB: An Introduction with Application, 2nd Edition.* London: John Wiley and Sons Inc.

Gorton, I., 2011. *Essential Software Architecture*. Springer.

Gouli, E., Kornilakis, H., Papanikolaou, K.A. and Grigoriadou, M, 2001. Adaptive Assessment Improving Interaction in an Educational Hypermedia System. *Human Computers Interaction 2001, Panhellenic Conference with International Participation,* pp.217-222.

Gregory, R. J., 2007. *Psychological testing : history, principles, and applications*. Boston, Pearson/Allyn and Bacon.

Hambleton, R. K. and Swaminathan, H., 1990. *Item Response Theory: Principles and Applications*. Kluwer-Nijhoff Publishing.

Hagimont, D. and Louvegies, D., 2009. Javanaise: distributed shared objects for internet cooperative applications. *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing* 2009, pp.339-354.

Hattie, J., 1981. *Decision criteria for determining unidimensional and multidimensional normal ogive models of latent trait theory*. Armidale, New South Wales, Australia: The University of New England, Centre for Behavioral Studies.

Henning, G.,1987. *A guide to language testing*. Cambridge, Mass.: Newbury House

Hetter, R. D., & Sympson, J. B., 1997. *Item exposure control in CAT-ASVAB.* In W. A. Sands, B. K. Waters, & J. R. McBride (Eds.), Computerized adaptive testing: From inquiry to operation. Washington DC: American Psychological Association. pp.141-144.

Hollingsworth, J. 1960. Automatic graders for programming classes. *Communications of ACM,* 3(10), pp.528– 529.

Ian, C. and Sleightholme, J., 2006. *Introduction to Programming with Fortran, 1st edition*. London: Springer.

IMS GLC, 2010. *The IMS Global Learning Consortium*. Available from http://www.imsglobal.org/background.html [accessed on 16th March, 2010].

IntraLearn, 2007. *IntraLearn*. Available from: http://www.intralearn.com/1070_Intralearn.asp. Accessed on 17th Jan, 2007.

Jordan, S. and Mitchell T., 2009. E-Assessment for Learning? The Potential of Short-Answer Free-Text Questions with Tailored Feedback. *British Journal of Educational Technology,* 40 (2), pp.371-385.

JUnit, n.d.. *About JUnit.org*. Available from http://www.junit.org/about [Accessed on 3rd June, 2011]

Jupitermedia Corporation, n.d.. *Web Services*. Available from http://www.webopedia.com/TERM/W/Web_services.html. [Accessed on 9 Dec, 2009].

Kaburlasos, V. G., Marinagi, C. C., and Tsoukalas, V. S., 2004. 'PARES: A Software Tool for Computer-Based Testing and Evaluation Used in the Greek Higher Education System', *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT'04)*.

Kalos, M. H., and Whitlock, P. A., 2008. *Monte Carlo Methods*. Wiley.

Keidar, I. 2008. Distributed computing column 32 - The year in review, *ACM SIGACT News*, 39(4), pp.53-54.

Kelderman, H., 1996. Multidimensional Rasch models for partial-credit scoring. *Applied Psychological Measurement*, vol. 20, pp.155-168.

Keller, L. A., 2000. *Ability Estimation Procedures in Computerized Adaptive Testing*. Available from http://www.aicpa.org/BecomeACPA/CPAExam/PsychometricsandScoring/TechnicalReports/DownloadableDocuments/Keller_AbilityEstimation.pdf [Accessed on 3rd June, 2011]

Khaskheli, A.R., 2004. Intelligent Agents and e-Learning, in Roope Raisamo and Erkki Mäkinen (eds.), *Brief Studies in Computer Science (Fall 2003)*, pp. 28-45.

Kingsbury, G. G., & Zara, A. R. 1991. A comparison of procedures for content-sensitive item selection in computerized adaptive tests. *Applied Measurement in Education*, 4, pp.241-261.

Kocher, A., T., 1974. *An empirical investigation of the stability and accuracy of flexilevel tests*. Paper presented at the Annual Meeting of the National Council on Measurment in Education (chicago, Illinois, April 1974)

Kolen, M. J. and Brennan, R. L., 2004. *Test Equating, Scaling, and Linking: Methods and Practices 2nd edition*. New York: Springer-Verlag.

Lehmann, E. L. and Casella, G., 2003. *Theory of Point Estimation 2nd edition*. New York: Springer.

Lewis, C. and Sheehan, K., 1990. Using Bayesian decision theory to design a computerized mastery test. *Applied Psychological Measurement*, 14, pp.367-386.

Li, Y. H. and Schafer, W. D., 2005. Trait parameter recovery using multidimensional computerised adaptive testing in reading and mathematics. *Applied Psychological Measurement*, vol. 29, pp3-25.

Lilley, M. and Barker, T., 2004. A computer-adaptive test that faciliates the modification of previously entered responses: an empirical study. *Lecture Notes in Computer Science*, Vol. 3220/2004, pp.22-23.

Lilley, M., Barker, T. & Britton, C. 2004. The development and evaluation of a software prototype for computer-adaptive testing. *Computers & Education*, 43(1-2): 109-123.

Lin, X. B., Fletcher, H., Mather, R., 2007a. Key Issues Surrounding the Implementation of Computerised Adaptive Testing Tools. *The HEA 2006 eLearning and Teaching Workshop*, 6th June, 2007.

Lin, X. B., 2007b. Efficient Diagnosis of Examinee's Proficiency Using Adaptive Testing, *Conference proceedings of the BCUC 2007 e-Learning Conference*, 7th June 2007.

Lin, X., Chen, H., Mather, R. & Fletcher, H., 2009a. Adaptive Assessment – A Practice of Classification on Small-Size Training Sets. In C. Crawford et al. (Eds.), *Proceedings of Society for Information Technology and Teacher Education International Conference 2009* (pp. 3168-3181). Chesapeake, VA: AACE.

Lin, X., Qiao, Y., Weatherburn, G. and Chen, H., 2009b. AMTES: An Adaptive Tele-education System for Pressure Ulcer Education. *International Conference on Education Technology and Computer 2009*, Singapore 17th – 20th April 2009. USA: IEEE.

Linacre, J. M., 1994. Sample size and item calibration stability. *Rasch Measurement Transactions*, 37(4), pp.328

Linn, R.L. and Gronlund, N.E., 2000. *Measurement and Assessment in Teaching, 8th Edition*. NJ: Prentice-Hall, pp.1-50.

Lord, F. M., 1980. *Applications of item response theory to practical testing problems*. Hillsdale, NJ: Lawrance Erlbaum Associates.

Lütticke, R., 2004. 'Problem Solving with Adaptive Feedback', Adaptive Hypermedia and Adaptive Web-Based Systems 2004, *Lecture Notes in Computer Science*, 3137 , 2004, pp. 417-420.

Manitoba Education, Citizenship, and Youth, 2006. *Rethinking Assessment with Purpose in Mind: assessment for learning, assessment as learning, assessment of learning*. Winnipeg, Manitoba, Canada: MECY.

Mansell, W., 2009. Why hasn't e-Assessment arrived more quickly? *The Guardian*, 21 July.

MATLAB, 2011. *MATLAB - The Language Of Technical Computing*. Available from http://www.mathworks.com/products/matlab/ [Assessed on 19th May 2011]

McClintock, K., 2008. *"Web Framework Project Comparison Matrix"*. Available from http://olex.openlogic.com/wazi/2008/web-framework-comparison-matrix/ [Accessed on 5th June, 2011].

Mckinley, R. L. and Reckase, M. D., 1983. MAXLOG: A computer program for the estimation of the parameters of a multidimensional logistic model. *Behavior Research Methods and Instrumentation*, vol. 15, pp.389-390.

McDonald, A. and Welland, R., 2001. Web Engineering in Practice, *Proceedings of the Fourth WWW10 Workshop on Web Engineering*, 1 May 2001, pp.21 - 30.

Meijer, R. R. & Nerling, M. L., 1999. Computerized adaptive testing: Overview and introduction, *Applied psychological measurement*, 23(3), pp.187-194.

Menasce, D. and Almeida, V., 2001. *Capacity Planning for Web Services*. NJ: Prentice Hall.

Microsoft, 2010. *Microsoft Certification Exam Policies*. Available from http://www.microsoft.com/learning/en/us/certification/exam-policies.aspx [Accessed on 12th May, 2011]

MySQL, 2010. *Whiter Papers*. Available from http://www.mysql.com/why-mysql/white-apers/ [Accessed on 12th May, 2011]

Myung, J., 2003. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, vol. 47, pp. 90-100.

Nodelman U., Shelton C.R., and Koller, D., 2005. Expectation Maximization and Complex Duration Distributions for Continuous Time Bayesian Networks. *Proceedings of the Twenty-First International Conference on Uncertainty in Artificial Intelligence*, pp. 421-430.

O'Connor, K. 2002. *How to Grade for Learning*. Arlington Heights, IL: Skylight.

Ofqual, 2010. *Ofqual Publishes Reports into On-demand Testing*. Available at http://www.ofqual.gov.uk/news-and-announcements/130-news-and-announcements-press-releases/234-ofqual-publishes-reports-into-on-demand-testing [Accessed on 10 Apr, 2010]

Olson, J., 1990. Applying computerized adaptive testing schools, *Measurement and Evaluation in Counseling and Development*, 23 (1), pp.31-38.

Ostini, R. and Nering, M. L., 2006. *Polytomous Item Response Theory Models*. London: Sages Publications.

Pack, R., 2008. Choosing Your Java Based Web Framework: A Comparison. *JavaOne SM Conference*.

Parshall, C. G., Stewart, R. and Ritter, J., 1996. Innovations: Graphics, sound and alternative response modes. *Paper presented at the National Council of Measurement in Education*, April 9-11, 1996, New York.

Parshall, C. G., Spray, J. A., Kalohn, J. C. and Davey, T., 2002. *Practical considerations in computerbased testing*. NY: Springer

Partchev, I., 2004. *A visual guide to item response theory*. Available at http://www2.uni-jena.de/svw/metheval/irt/VisualIRT.pdf [Accessed on 9 Nov, 2006]

Pellegrino, J. W., Chudowski, N. and Gklaser, R. (Eds.), 2001. *Knowing What Students Know: The Science and Design of Educational Assessment*. Washington DC: National Academy of Sciences.

Phankokkruad, M. and Woraratpanya, K., 2009. Web Service Architecture for Computer-Adaptive Testing on e-Learning, *International Journal of Behavioral, Cognitive, Educational and Psychological Sciences*, 1(1), 2009, pp.43-47.

Pratt, J. W., 1976. "F. Y. Edgeworth and R. A. Fisher on the efficiency of maximum likelihood estimation". *The Annals of Statistics*, **4** (3): pp.501–514.

Price, M., O'Donovan, B., Rust, C. & Carroll, J, 2008. Assessment standards: a manifesto for change, *Brookes eJournal of Learning and Teaching*, 2(3), Available from http://bejlt.brookes.ac.uk/article/assessment_standards_a_manifesto_for_change/ [Accessed on 27 July, 2010]

Popham, J. W., 1975. *Educational evaluation*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Qiao Y., Lin X., Weatherburn G., Chen H., Guo L., Fletcher H., 2009. Adaptive Tele-education for Pressure Ulcer Prevention using Rich Internet Application, *British Computing Society Health Computing 2009*, Harrogate, UK 28th – 30th April 2009.

Questionmark Perception, 2007. *Questionmark Perception*. http://www.questionmark.com/uk/perception/authoring_windows.aspx#Create_Assessments [Accessed on 17th Jan, 2007].

Rae, S. and Whitelock, D., 2005. *E-Assessment: which way now?*. Available from: http://kn.open.ac.uk/public/getfile.cfm?documentfileid=8679 [Accessed on 31st May, 2006].

Rasch, G., 1980. *Probabilistic models for some intelligence and attainment tests*. Chicago: The University of Chicago Press.

Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Koedinger, K. R., Junker, B., Ritter, S., Knight, A., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T.E., Upalekar. R, Walonoski, J.A., Macasek. M.A., Rasmussen, K.P., 2005. The Assistment Project: Blending Assessment and Assisting. In C.K. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.) *Proceedings of the 12th Artificial Intelligence In Education*, pp.555-562. Amsterdam: ISO Press.

Reckase, M.D., 1997. The Past and Future of Multidimensional Item Response Theory, *Applied Psychological Measurement*, 21(1), 1997, pp.25-36.

Reeve B. B. and Fayer P., 2005. Applying item response theory modelling for evaluating questionnaire item and scale properties. In P. Fayers and R. Hays (Ed.). *Assessing quality of life in clinical trials 2nd edition*. Oxford: Oxford University Press.

Reise, S.P., Ainsworth, A. T., Haviland, M. G., 2005. Item Response Theory: Fundamentals, Applications, and Promise in Psychological Research. *Current Directons in Psychological Science*, 14(2), April 2005, pp.95-101.

Revuelta, J., & Ponsada, V., 1998. A comparison of item exposure control methods in computerized adaptive testing. *Journal of Educational Measurement*, vol.35, pp.311-327.

Ridgway, J., McCusker, S. and Pead, D., 2006. *Literature review of e-Assessment*. http://www.futurelab.org.uk/download/pdfs/research/lit_reviews/futurelab_review_10.pdf [Accessed on 10th of July, 2006].

Ripley, M., 2004. E-assessment question 2004 – QCA keynote speech e-assessment: an overview. *Delivering E-assessment – a Fair Deal for Learners*, 20 April 2004.

Roberts, T. S., 2006. Self, Peer, and Group Assessment in E-learning: An Introduction. Roberts, T. S. (Eds). *Self, Peer and Group Assessment in E-learning*. London: Information Science Publishing.

Rudner, L. M., 1998. *An On-line, Interactive, Computer Adaptive Testing Tutorial*. Available from: http://edres.org/scripts/cat/catdemo.htm. [accessed 9 Feb 2007].

Rudner, L. M., 2001. *Measurement Decision Theory*. Available from http://ericae.net/mdt/mdt11c.pdf, [accessed 9 Feb 2007].

Sadler, D.R., 2008. Formative assessment and the design of instructional systems. Republished in W. Harlen (Ed), 2008. *Student assessment and testing.* Ch. 14, Vol. 2, 3 28. London: SAGE.

Salmon, G. 2002. *E-tivities - The Key to Active Online Learning*. London: Kogan Page.

Salmon, G. 2004. *E-moderating: The Key to Teaching and Learning Online 2nd Edition.* London: Kogan Page.

Sand, W. A., Water, B. K. and McBride, J. R. (Eds.), 1997. *Computerized adaptive testing: from inquiry to operation*, Washington, DC: American Psychological Association.

Schollmeier, R., 2002. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, *Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE (2002)*, pp.101-102.

SEEC, 2010. *Southern England Consortium ofr Credit Accumulation and Transfer: Credit Level Descriptors for Higher Education (Draft)*. [Online]. Available from: http://www.seec.org.uk/sites/seec.org.uk/files/SEEC descriptors June10_1.pdf [Accessed 08 Jan 2011].

Segall, D. O., 1996. Multidimensional adaptive testing. *Psychometrika*, vol. 61, pp.331 – 345.

Shan, T. C., and Hua, W. W., 2006. "Taxonomy of Java Web Application Frameworks", *Proceedings of IEEE International Conference on e-Business Engineering (ICBE'06)*, Shanghai, China, 2006, pp.378-385.

SHLPreVisor, 2011. SHLPreVisor. Available from http://www.previsor.com/news/press/SHLPreVisor-Releases-2011-Global-Assessment-Trends-Report [Accessed on 5th May, 2011]

Shultz, K. S. and Whitney, D. J., 2004. *Measurement Theory in Action: Case Studies and Exercises*. London: SAGE Publications.

Singh, I., and Johnson, M., 2001. *J2EE Platform Design Patterns for Enterprise Applications*. BluePrints Sun Microsystems, Inc.

Slee M., Agarwal A. and Kwiatkowski M., 2007. *Thrift: Scalable Cross-Language Services Implementation*. Available from: http://incubator.apache.org/thrift/ [Accessed 30 Nov 2009]

Snyder, M. M., 2006. Self, peer, and group assessment in e-Learning, *Internet & Higher Education*, 9(4), pp.317-320.

Stacey, K., Price, B., Steinle, V., Chick, H. and Gvozdenko, E., 2009. SMART Assessment for Learning, Conference of International Society for Design and Development in Education , 28th Sep - 1st Oct 2009.

Steven, C., and Hesketh, I., 1999. Increasing learner responsibility and support with the aid of adaptive formative assessment using QM designer software, in S. Brown, J. Bull & P. Race (Eds.) (1999), *Computer-Assisted Assessment in Higher Education*, London: Kogan Page Ltd.

Sun Microsystems., 2006. *The Java™ Web Service Tutorial, Feb. 2006.* Available from http://java.sun.com/webservices/docs/2.0/tutorial/doc/. [Accessed on 21 June 2009].

Sympson, J. B., 1978. A model for testing with the multidimensional items. In D.J. Weiss (Ed.). *Item response theory and computerized adaptive testing conference proceedings*. MN, US: University of Minnesota press.

Taiwan Education Testing Center, 2007. *Taiwan Education Testing Center Website*, Available from http://www.edutest.com.tw/about/index.htm [Accessed 4th May, 2011].

Tao, Y. H., Wu, Y. L., and Chang, H. Y., 2008. A Practical Computer Adaptive Testing Model for Small-Scale Scenarios, *Educational Technology and Society*, 11(3), pp.259-274.

Taligent Inc., 1993. *Building Object-Oriented Frameworks*. Available from: http://lhcb-comp.web.cern.ch/lhcbcomp/ Components/ postscript/ buildingoo.pdf [Accessed on 30th April, 2011]

Thissen, D. & Mislevy, R. J., 2000. Testing Alogrithms In: Wainer, H. (Ed). *Computerized Adaptive Testing: A Primer (2nd ed.) (pp.101 - pp.103).* Mahwah, NH: Lawrence Erlbaum Associates.

Thissen D., Chen, W.H. and Bock D., n.d.. *Multilog-7*. Available from http://www.assess.com/xcart/product.php?productid=244&cat=26&page=1 [Accessed on 10th May 2011]

Thomas, L., MacMillan, J., McColl, E., Hale, C., and Bond S, 1995. Comparison of focus group and individual interview methodology in examining patient satisfaction with nursing care, *Social Sciences in Health*, 1(4), pp. 206–219.

Thompson, N.A., 2009. *Ability Estimation with Item Response Theory*. White Paper. Assessment Systems Corporation.

Thompson, N.A., 2010. *Adaptive Testing: Is it Right for Me?* White Paper Assessment Systems Corportation.

Trentin, G., 1997. 'Computerized adaptive tests and formative assessment', *Journal of Educational Multimedia and Hypermedia*, 6 (2), 1997, pp.201 -220.

Tzanavari, A. S., and Pastellis, P., 2004. 'Giving More Adaptation Flexibility toAuthors of Adaptive Assessments', Adaptive Hypermedia and Adaptive Web-Based Systems 2004, *Lecture Notes in Computer Science*, 3137/2004, 2004, pp. 340–343.

UDDI Consortium, 2001. *UDDI Executive White Paper*. Available from http://www.uddi.org/pubs/UDDI_Executive_White_Paper.pdf. [Accessed on 12 July, 2009].

UKCLE, 2010. *Assessment*. Availble from http://www.ukcle.ac.uk/resources/assessment-and-feedback/teaching-legal-research-assessment/ [Accessed on 15 July, 2010].

Urry, V., 1977. Tailored testing: A successful application of latent trait theory. Journal of Educational Measurement, 14, pp.181-196.

Van der Linden, W. J., & Glas, C. A. W. (Eds.)., 2000. *Computerized adaptive testing: Theory and practice*. Boston: Kluwer

W3C, 2004. *Web Services Architecture Requirements*. Available from http://www.w3.org/TR/wsa-reqs/. [Accessed on 10 Jan, 2009]

Wainer, H., Dorans, N. J., Flaugher, R., Green, B. F., Mislevy, R. J., Steinberg, L., et al. (Eds.), 1990. *Computerized adaptive testing: A primer*. Hillsdale, NJ: Lawrence Erlbaum Associates publish.

Wainer, H., Dorans, N. J., Eignor, D., Flaugher, R., Green, B. F., Mislevy, R. J., Steinberg, L. and Thissen, D., 2000. *Computerized Adaptive Testing: A Primer 2nd Edition*. Mahwah, New Jersey: Lawrence Erlbaum Associates, Inc. pp.9-11.

Wainer, H. and Mislevy, R. J,, 2000. Item Response Theory, Item Calibartion, and Proficiency Estimation, in H. Wainer (2000), *Computerized Adaptive Testing: A Primer*, Lawrence Erlbaum Associates Inc.

Wang, W. C., and Chen, P.-H., 2004. Implementation and measurement efficiency of multidimensional computerized testing. *Applied Psychological Measurement*, vol. 28, pp.295-316.

Wang, W. C., Chen, P. H., and Cheng, Y. Y., 2004. Improving measurement precision of test batteries using multidimensional item response models. *Psychological Methods*, vol. 9, pp.116-136.

Wang, W. C. and Wilson, M., 2005. The Rasch Testlet Model. *Applied Psychological Measurement*, 29(2), March 2005, pp126-149.

Webassessor , 2007. *Webassessor*. Available from: http://www.kryteriononline.com/testdelivery.htm [Assessed on 17th Jan, 2007]

Weiss, D. J., 1974. *Strategies of adaptive ability measurement (Research Report 74-5)*. Minneapolis: University of Minnesota, Department of Psychology, Psychometric Methods Program 1975.

Weiss, D. J., 1983. *New Horizons in Testing: Latent Trait Test Theory and Computerized Adaptive Testing*. Academic Press Inc.

Weiss, D. J., 1985. *Adaptive testing by computer*. Journal of Consulting & Clinical Psychology, 53, 774-789.

Weiss, D. J. (Ed.), 1985. *Item response theory and computerized adaptive testing conference proceedings*. MN: University of Minnesoda press.

Weiss, D. J. and Bock, R. D., 1983. *New horizons in testing: latent trait test theory and computerized adaptive testing*. NY: Academic Press.

Weiss, D. J., and Kingsbury, G. G., 1984. Application of computerized adaptive testing to educational problems. *Journal of Educational Measurement*, 21, pp.361-375.

Weiss, D. J., and Mcbride, J. R., 1984. Bias and information of Bayesian adaptive testing, *Applied Psychological Measurement*, 8 (3), pp.273-285.

Weiss, D. J. and Yoes, M., 1991. Item Response Theory. In R. K. Hambleton & J. Zall(Eds.), *Advances in educational and psychological testing*. Boston: Kluwer-Nijhoff

Wiggins, G. 1998. *Educative Assessment*. San Francisco, CA: Jossey Bass.

Winkley, J., 2010. *E-assessment and Innovation*. Available from http://emergingtechnologies.becta.org.uk/upload-dir/downloads/page_documents/research/emerging_technologies/e-assessment_and_innovation.pdf [Accessed on 3rd, May, 2010]

Wise, S. L. & Kingsbury, G. G. , 2000. Practical Issues in Developing and Maintaining a Computerized Adaptive Testing Program, *Psicologica*, vol. 21, pp.135-155.

Wise, S. L. & Plake, B. S., 1990. Computer-based testing in higher education. Measurement and evaluation in counseling and development, 23, 3-10. Wright, B. D., 1968. Sample-free test calibration and person measurement. *Proceedings of the 1967 Invitational Conference on Testing. Princeton,* NJ: Educational Testing Service.

Wu, M.L., Adams, R.L. and Wilson, M. R., 1998. *Acer ConQuest*. Melbourne, Victoria, Australia: Australian Council for Educational Research press.

Yang, M., Zapata-Rivera, D. & Bauer, M., 2006. e-Grammar: An Assessment-based Learning Environment for English Grammar. In E. Pearson & P. Bohman (Eds.), *Proceedings of World Conference on* Educational *Multimedia, Hypermedia and Telecommunications 2006,* Orlando Florida, USA, 26-30 June 2006.

Zimowski,M., Muraki, E., Mislevy, R.J., and Bock, R.D., n.d. *BILOG-MG.* Available from http://www.assess.com/xcart/product.php?productid=217 [Accessed on 10th May 2011]

BIBLIOGRAPHY:

Ambler, S. W. 2004. *The Object Primer: Agile Model Driven Development with UML 2*. Cambridge: Cambridge University Press.

Andrew, G. R. 2000. *Foundations of Multithreaded, Parallel, and Distributed Programming*. Reading, MA: Addison-Wesley.

Arora, S. and Barak, B. 2009. *Computational Complexity – A Modern Approach*. Cambridge: Cambridge University Press.

Assessment Systems Corporation, 2010. *Assessment Systems Corporation Newsletter*. Available from http://www.assess.com/docs/ASC_Newsletter_Sept_2010.pdf [Accessed on 10th May, 2011]

Attiya, H. and Welch J. 2004. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Hoboken, New Jersey: Wiley – Interscience.

Barker, T. & Lilley, M. 2004. "The development and evaluation of computeradaptive testing software in a UK university", *Proceedings of the 2004 Learning and Teaching Conference,* University of Hertfordshire, United Kingdom.

Cormen, T. H., Leiserson, S. E. and Rivest, R. L. 2001. *Introduction to Algorithms 2nd Edition*. Cambridge, MA: MIT Press.

Dolev, S. 2000. *Self-Stabilization*. Cambridge, MA: MIT Press.

Eggen, T. J. H. M., 2004. *Contributions to the theory and practice of computerized adaptive testing.* Available from: http://doc.utwente.nl/50759/1/thesis_Eggen.pdf [Access on the 5th May, 2011]

Elmasri, R. and Navathe, S. B. 2003. Fundamentals of Database Systems 4th Edition. Reading, MA: Addison - Wesley.

Faber, J. 1998. *Java Distributed Computing*. Cambridge, MA: O'Reilly Media, Inc.

Fowler, M. 2003. *UML Distilled: A Brief Guide to the Standard Object Modeling Language 3rd edition*. Addison-Wesley.

Garg, V. K. 2002. *Elements of Distributed Computing*. Wiley – IEEE Press.

Ghosh, S. 2007. *Distributed Systems – An algorithmic Approach*. Iowa: Chapman & Hall/CRC.

Godfrey, B. 2002. *A primer on distributed computing*. Available from: http://www.bacchae.co.uk/docs/dist.html [Accessed on 12th, August 2010]

Gregory, Robert J. , 2011. *Psychological Testing: History, Principles, and Applications, sixth edition*. Boston, USA: Allyn & Bacon.

Hambleton, R. K., Swaminathan, H., & Rogers, H. J., 1991. *Fundamentals of Item Response Theory*. Newbury Park, CA, USA: Sage Press.

Henderson-Sellers, B. and Gonzalez-Perez, C. 2006. "Uses and Abuses of the Stereotype Mechanism in UML 1.x and 2.0", *Model Driven Engineering Languages and Systems*. Springer Berlin/ Heidelberg, pp. 16 – 26.

Herlihy, M. P. and Shavit, N. N. 2008. *The Art of Multiprocessor Programming*. Oxford: Morgan Kaufmann.

John, H. 2000. *The Unified Process for Practitioners: Object-oriented Design, UML and Java*. Springer.

Keidar, I. 2008. "Distributed computing column 32 – The year in review", *ACM SIGACT News*, 39(4), pp.53 – 54.

Lilley, M. & Barker, T. 2006a. "Computerised adaptive testing: extending the range of assessment formats in a Computer Science course", *Proceedings of ICL2006 Conference*, September 27 -29, 2006 Villach, Austria.

Linial, N. 1992. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1), pp.193 – 201

Lynch, N. A. 1996. *Distributed Algorithms*. San Francisco, CA: Morgan Kaufmann.

Martin, R. C. 2003). *UML for Java Programmers*. Prentice Hall.

Naor, M. and Stockmeyer, L. 1995. "What can be computed locally?", *SIAM Journal on Computing*, 24(6), pp.1259 – 1277.

Papadimitriou, C. H. 1994. *Computational Complexity*. Reading, MA: Addison-Wesley.

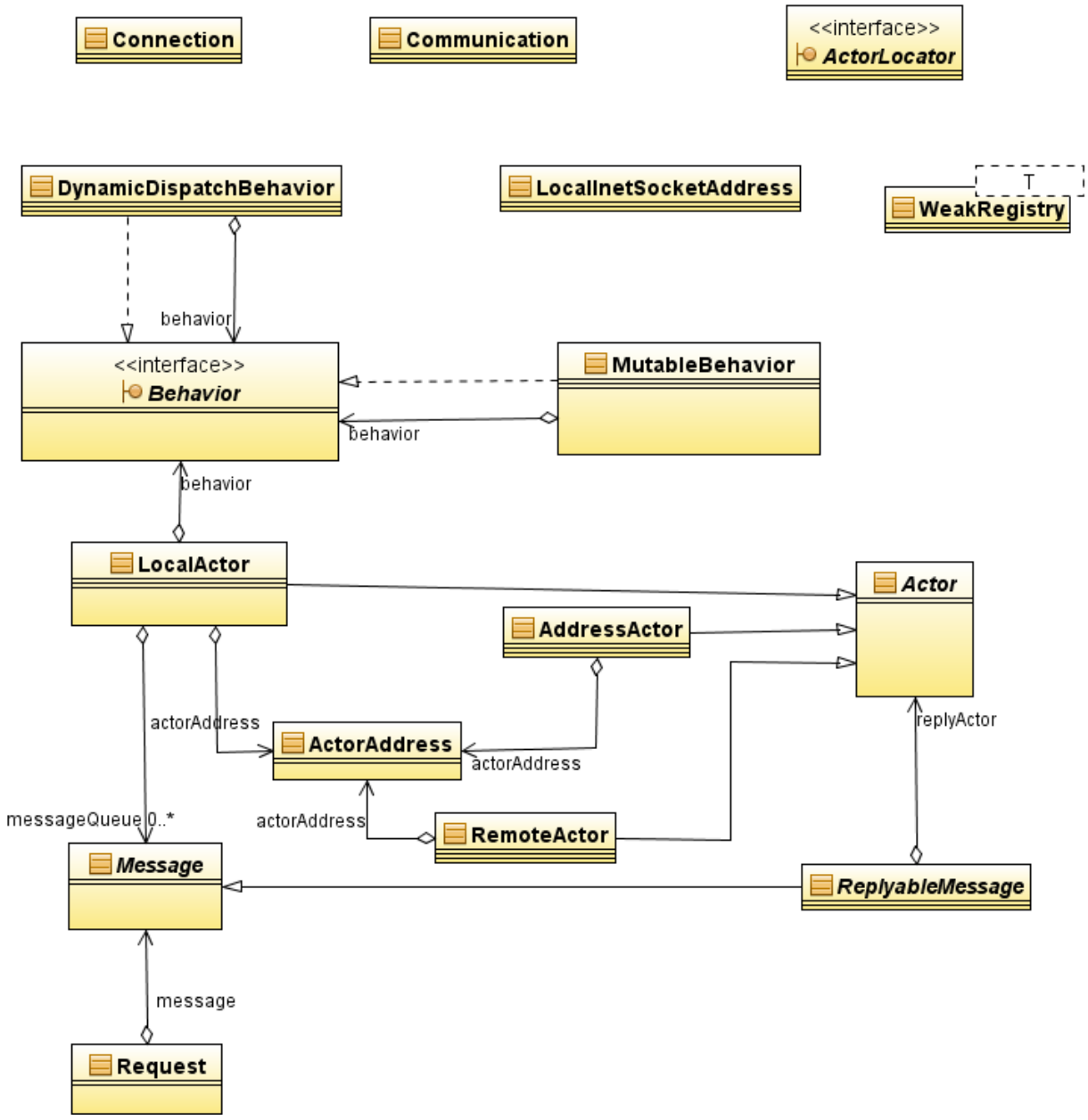
Peleg, D. 2000. *Distributed Computing: A Locality-Sensitive Approach*. Philadelphia, PA: SIAM.

Tel, G. 1994. *Introduction to Distributed algorithms*. Cambridge: Cambridge University Press.

Appendix - A1 Source Code of the Framework

This section includes the complete source code of the distributed adapitive assessment framework.

The communication package



Actor.java

Actor.java
package communication; import java.io.Serializable; /** * <p>An <tt>Actor</tt> is an active object: an independent thread

```
 * of control which receives <i>asynchronous</i> messages and processes
 * them one at a time.</p>
 *
 * <p>Actor references can be passed to other actors inside of messages.
 * For example, actor A can send a message to actor B, asking B to perform
 * some computation.  If A would like some response from B when it's done,
 * A should include a reference to itself within the message.</p>
 *
 * <p>The <tt>Actor</tt> class does not have a public constructor; use
 * Communication.newActor(Behavior) instead.</p>
 *
 * @see  Behavior
 * @see  Communication
 */
public abstract class Actor implements Serializable {
   /** Package-private constructor prevents other packages from extending
    * class Actor.
    */
   Actor() { }
   /**
    * <p>Sends an asynchronous message to this <tt>Actor</tt>.  The
    * <tt>send()</tt> method returns immediately.</p>
    *
    * <p>If the target <tt>Actor</tt> resides in another JVM instance, the
    * message will automatically be forwarded over the network.  Messages
    * may occasionally be delayed or dropped due to transient network or
    * host failures.</p>
    *
    * <p>Once the message is received by its target <tt>Actor</tt>, it is
    * enqueued if the <tt>Actor</tt> is busy processing another message.
    * Queued messages are processed sequentially.
    *
    * @param  message
    */
   public void send(Message message) { }
}
```

ActorAddress.java

ActorAddress.java

```
package communication;

import java.io.Serializable;
import java.net.InetSocketAddress;

/**
 * An <tt>ActorAddress</tt> encapsulates the IP address and TCP socket of a
 * <tt>Communication</tt> instance and a local actorId within that
 * instance.  This is used in the serialized form of an <tt>Actor</tt>
 * reference.
 *
 */
class ActorAddress implements Serializable {
   private static final long serialVersionUID = 1L;
   final InetSocketAddress   inetSocketAddress;
   final long                actorId;

   ActorAddress(InetSocketAddress inetSocketAddress, long actorId) {
      this.inetSocketAddress = inetSocketAddress;
      this.actorId           = actorId;
```

```
    }
    public boolean equals(Object object) {
        if (object instanceof ActorAddress) {
            ActorAddress actorAddress = (ActorAddress) object;
            return inetSocketAddress.equals(actorAddress.inetSocketAddress)
                && actorId == actorAddress.actorId;
        } else {
            return false;
        }
    }
    public int hashCode() {
        // Is this hashCode any good?!
        return inetSocketAddress.hashCode() ^ (int) actorId;
    }
    public String toString() {
        return inetSocketAddress.toString() + '#' + Long.toString(actorId);
    }
}
```

ActorLocator.java

ActorLocator.java

```
package communication;
/**
 * A way to find an Actor, given a simple service name string.  This is
 * used by the HTTP Gateway, and might be useful elsewhere.
 *
 */
public interface ActorLocator {
    Actor getActorFor(String serviceName);
}
```

AddressActor.java

AddressActor.java

```
package communication;

import java.io.ObjectStreamException;

/**
 * <p><tt>AddressActor</tt> is used solely as the serialized form of an
 * <tt>Actor</tt> reference.  When an <tt>AddressActor</tt> is
 * deserialized, it contacts the in-scope <tt>CommunicationManager</tt> to
 * construct a corresponding <tt>LocalActor</tt> or
 * <tt>RemoteActor</tt>.  A <tt>ThreadLocal</tt> variable is used to track
 * which <tt>CommunicationManager</tt> to use.</p>
 *
 * <p>An <tt>AddressActor</tt> cannot legitimately be sent a message.</p>
 *
 */
final class AddressActor extends Actor {
    private static final long  serialVersionUID = 1L;
    // ThreadLocal is used in lieu of a singleton/global.
    static final ThreadLocal<CommunicationManager> threadLocalCommunicationManager = new
ThreadLocal<CommunicationManager>();
    private final ActorAddress actorAddress;
    AddressActor(ActorAddress actorAddress) {
        this.actorAddress = actorAddress;
    }
    public void send(Message message) {
        // Not implemented.
```

```
    }
    Object readResolve() throws ObjectStreamException {
        CommunicationManager communicationManager = threadLocalCommunicationManager.get();
        if (communicationManager != null)
            return communicationManager.getActorFor(actorAddress);
        else
            return null;
    }
}
```

Behavior.java

Behavior.java

```
package communication;

/**
 * <p>A <tt>Behavior</tt> specifies what an <tt>Actor</tt> does in response
 * to a <tt>Message</tt>.</p>
 *
 * <p>As mentioned in the description of the <tt>Actor</tt> class,
 * <tt>Actor</tt>s dispatch <tt>Message</tt>s one at a time.  Internally,
 * each <tt>Actor</tt> has an associated <tt>Behavior</tt> object;
 * dispatching a message corresponds to invoking the <tt>Behavior</tt>'s
 * <tt>call(Message)</tt> method and waiting for it to return.</p>
 *
 * <p>Users of the communication library can provide custom
 * <tt>Behavior</tt>s by defining classes that implement the
 * <tt>Behavior</tt> interface.  An <tt>Actor</tt> with a given
 * <tt>Behavior</tt> is created by calling the <tt>newActor(Behavior)</tt>
 * method of a <tt>Communication</tt> object.
 *
 * @see  Actor
 * @see  Communication
 * @see  Message
 */
public interface Behavior {
    /**
     * Processes one <tt>Message</tt> and then returns.  The
     * <tt>Actor</tt> associated with a <tt>Behavior</tt> will only
     * dispatch one message at a time.  Hence unless other concurrent
     * constructs are involved, the body of the <tt>call(Message)</tt>
     * method can assume it has exclusive, single-threaded access to
     * the contents of the <tt>Behavior</tt> object.
     *
     * @param  message
     */
    void call(Message message);
}
```

Communication.java

Communication.java

```
package communication;

import java.net.InetSocketAddress;

/**
 * <p>Manages <tt>Actor</tt>s and network communication.</p>
 *
 * <p>Typically, a user creates a single instance of <tt>Communication</tt>
 * which is shared by a logically related group of <tt>Actor</tt>s within a
```

```
 * single JVM.  A <tt>Communication</tt> object listens on a given IP
 * address and TCP port, and dispatches each incoming message to the
 * corresponding target <tt>Actor</tt>.</p>
 *
 * <p>Each <tt>Communication</tt> object may include a <i>default actor</i>
 * which receives all messages for which the target actor cannot be found.
 * This is useful as a boot-up mechanism, when you know the IP address and
 * TCP port of a <tt>Communication</tt> instance, but don't yet have
 * references to any of its <tt>Actor</tt>s.</p>
 *
 * <p>Internally, the communication subsystem is able to tolerate dropped
 * TCP connections by reconnecting as necessary.  However, network or host
 * failures may occasionally lead to dropped or delayed
 * <tt>Message</tt>.</p>
 *
 * @see  Actor
 * @see  Behavior
 * @see  Message
 */
public class Communication {
  private final CommunicationManager communicationManager;

  /**
   * Creates a <tt>Communication</tt> system that listens on the given IP
   * address and TCP port.  This constructor includes a <tt>Behavior</tt>
   * for a default <tt>Actor</tt>.
   *
   * @param  localInetSocketAddress  The IP address and TCP port on which
   *                                 to listen.  This must be an IP
   *         address of the local host and an unused TCP port on which
   *         this process is allowed to listen.  If you would like to
   *         perform communication between multiple hosts, ensure that
   *         the IP address is not 127.0.0.1.
   *
   * @param  defaultBehavior  The <tt>Behavior</tt> to supply for the
   *                          default <tt>Actor</tt>.  The
   *         <tt>Communication</tt> object will internally create and
   *         hold a reference to the <tt>Actor</tt> associated with this
   *         behavior.
   */
  public Communication(InetSocketAddress localInetSocketAddress, Behavior defaultBehavior) {
    Actor defaultActor = new LocalActor(new ActorAddress(localInetSocketAddress, 0), defaultBehavior);
    this.communicationManager = new CommunicationManager(localInetSocketAddress, defaultActor);
  }

  /**
   * Creates a <tt>Communication</tt> system that listens on the given IP
   * address and TCP port.  This constructor does not include any
   * message-handling behavior for the default actor.
   *
   * @param  localInetSocketAddress
   *
   * @see  Communication#Communication(InetSocketAddress, Behavior)
   */
  public Communication(InetSocketAddress localInetSocketAddress) {
    Actor defaultActor = new LocalActor(new ActorAddress(localInetSocketAddress, 0), new
DiscardMessageBehavior());
    this.communicationManager = new CommunicationManager(localInetSocketAddress, defaultActor);
```

```
    }

    /**
     * Constructs a new <tt>Actor</tt> with the given <tt>Behavior</tt>.
     *
     * @param  behavior  The <tt>Behavior</tt> object used to process
     *                   messages sent to the <tt>Actor</tt>.
     *
     * @return  A reference to an <tt>Actor</tt>, which can be used locally
     *          our placed in a <tt>Message</tt> sent to other
     *          <tt>Actor</tt>s.
     */
    public Actor newActor(Behavior behavior) {
       return new LocalActor(communicationManager, behavior);
    }

    /**
     * Returns a reference to the default <tt>Actor</tt> of the
     * <tt>Communication</tt> instance listening on the given IP address
     * and TCP port.  If no <tt>Communication</tt> instance is actually
     * listening on the given IP:port, then messages sent to the
     * corresponding default actor are simply discarded.
     *
     * @param  inetSocketAddress  The IP address and TCP port of the
     *                            destination <tt>Communication</tt>
     *                            instance.
     *
     * @return  A reference to an <tt>Actor</tt>.
     */
    public Actor getDefaultActorFor(InetSocketAddress inetSocketAddress) {
       return communicationManager.getDefaultActorFor(inetSocketAddress);
    }

    private static class DiscardMessageBehavior implements Behavior {
       public void call(Message message) {
       }
    }

    public int getIncomingMsgCounter() {
       return communicationManager.getIncomingMsgCounter();
    }

    public int getOutgoingMsgCounter() {
       return communicationManager.getOutgoingMsgCounter();
    }

    public int getPort() {
       return communicationManager.getPort();
    }
}
```

CommunicationManager.java

CommunicationManager.java

```
package communication;

import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;
```

```
import utils.ThreadLocalStack;
import utils.UniqueID;

import org.apache.log4j.Logger;
import logging.CustomLogger;
import logging.LoggingConfig;

/**
 * <tt>CommunicationManager</tt> is the implementation of the public class
 * <tt>Communication</tt>.
 *
 */
class CommunicationManager implements Runnable {
    static {
        LoggingConfig.configForDistributedLogging();
    }
    private static final Logger logger = CustomLogger.getLogger("communication");
    private final InetSocketAddress                 localInetSocketAddress;
    private final Actor                             defaultActor;
    private final WeakRegistry<Actor>               actorRegistry;
    private final HashMap<InetSocketAddress, Connection> connectionMap;
    private int                                     incomingMsgCounter;
    private int                                     outgoingMsgCounter;
    private int                                     localPort;

    public CommunicationManager(InetSocketAddress localInetSocketAddress, Actor defaultActor) {
        this.localInetSocketAddress = localInetSocketAddress;
        this.defaultActor           = defaultActor;
        this.actorRegistry          = new WeakRegistry<Actor>();
        this.connectionMap          = new HashMap<InetSocketAddress, Connection>();
        this.incomingMsgCounter     = 0;
        this.outgoingMsgCounter     = 0;
        this.localPort              = -1;
        (new Thread(this)).start();
    }
    /**
     * Called by LocalActor.LocalActor() to register a new Actor.
     */
    ActorAddress getActorAddressFor(Actor actor) {
        if (actor instanceof RemoteActor) {
            return ((RemoteActor) actor).actorAddress;
        } else {
            return new ActorAddress(localInetSocketAddress, actorRegistry.put(actor));
        }
    }
    /**
     * Called by AddressActor.readResolve() to deserialize an Actor
     * reference.
     */
    Actor getActorFor(ActorAddress actorAddress) {
        if (localInetSocketAddress.equals(actorAddress.inetSocketAddress)) {
            return actorRegistry.get(actorAddress.actorId);
        } else {
            return new RemoteActor(this, actorAddress);
        }
    }
    /**
```

```
     * Implementation of Communication.getDefaultActorFor()
     */
    public Actor getDefaultActorFor(InetSocketAddress inetSocketAddress) {
        if (localInetSocketAddress.equals(inetSocketAddress)) {
            return defaultActor;
        } else {
            return new RemoteActor(this, new ActorAddress(inetSocketAddress, 0));
        }
    }
    /**
     * sendTo() is called by RemoteActor.send()
     */
    void sendTo(ActorAddress actorAddress, Message message) {
        Connection connection = getConnectionTo(actorAddress.inetSocketAddress);
        if (connection != null) {
            connection.send(new Request(actorAddress.actorId, message));
        }
    }
    Connection getConnectionTo(InetSocketAddress inetSocketAddress) {
        Connection connection;
        synchronized (connectionMap) {
            connection = connectionMap.get(inetSocketAddress);
            if (connection == null) {
                try {
                    connection = new Connection(localInetSocketAddress, inetSocketAddress, this);
                } catch (Exception e) {
                    logger.error("Failed to get a connection to: " + inetSocketAddress + "\n" +
                            "The Exception is: ", e);
                }
            }
        }
        return connection;
    }
    /**
     * addConnection() is called by Connection.Connection() when a
     * connection is opened.
     */
    void addConnection(Connection connection) {
        synchronized (connectionMap) {
            InetSocketAddress inetSocketAddress = connection.getInetSocketAddress();
            Connection existingConnection = connectionMap.get(inetSocketAddress);
            if (existingConnection != null && !existingConnection.equals(connection)) {
                existingConnection.close();
            }
            connectionMap.put(inetSocketAddress, connection);
        }
    }
    /**
     * removeConnection() is called by Connection.run() when a Connection
     * is closed.
     */
    void removeConnection(Connection connection) {
        synchronized (connectionMap) {
            InetSocketAddress inetSocketAddress = connection.getInetSocketAddress();
            Connection existingConnection = connectionMap.get(inetSocketAddress);
            if (existingConnection != null && existingConnection.equals(connection)) {
                existingConnection.close();
                connectionMap.remove(inetSocketAddress);
```

```
            }
        }
    }

    /**
     * Called by Connection.run() when a message is received along a
     * connection.
     */
    public void call(Request request) {
        // Install the caller's UniqueID into dynamic scope.
        ThreadLocalStack<UniqueID>.Key key =
UniqueID.threadLocalStack.push(request.message.uniqueID);
        try {
            Actor actor = actorRegistry.get(request.actorId);
            if (actor != null) {
                actor.send(request.message);
            } else {
                defaultActor.send(request.message);
            }
        } catch (Exception e) {
            logger.error("Exception: ", e);
            // Transient error, skip message.
        }
        // Remove the caller's UniqueID from dynamic scope.
        UniqueID.threadLocalStack.pop(key);
    }
    /**
     * implements Runnable
     */
    public void run() {
        try {
            while (true) {
                ServerSocket serverSocket = new ServerSocket(localInetSocketAddress.getPort());
                this.localPort = serverSocket.getLocalPort();
                try {
                    while (true) {
                        Socket socket = serverSocket.accept();
                        try {
                            new Connection(localInetSocketAddress, socket, this);
                        } catch (Exception e) {
                            logger.error("Exception: ", e);
                            // Maybe it was a bad connection.
                            try {
                                socket.close();
                            } catch (Exception e2) {
                                logger.error("Exception: ", e2);
                        }
                        }
                    }
                } catch (Exception e) {
                    logger.error("Exception: ", e);
                    // Try to listen again.
                }
            }
        } catch (Exception e) {
            logger.error("Exception: ", e);
            // Give up.
        }
```

```
    }

    void recievedOneMessage() {
        incomingMsgCounter++;
    }

    void sentOneMessage() {
        outgoingMsgCounter++;
    }

    /**
     * Return the incoming message counter.
     */
    int getIncomingMsgCounter() {
        return incomingMsgCounter;
    }

    /**
     * Return the outgoing message counter.
     */
    int getOutgoingMsgCounter() {
        return outgoingMsgCounter;
    }

    /**
     * Return the port which this communication is listenning to.
     */
    int getPort() {
        return localPort;
    }

}}
```

Connetion.java

Connection.java

```
package communication;

import java.io.IOException;
import java.io.ObjectInput;
import java.io.ObjectInputStream;
import java.io.ObjectOutput;
import java.io.ObjectOutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;

import org.apache.log4j.Logger;
import logging.CustomLogger;
import logging.LoggingConfig;

class Connection implements Runnable {
    static {
        LoggingConfig.configForDistributedLogging();
    }
    private static final Logger logger = CustomLogger.getLogger("communication");
    private final InetSocketAddress    localInetSocketAddress;
    private final InetSocketAddress    remoteInetSocketAddress;
    private final CommunicationManager communicationManager;
    private final Socket               socket;
    private final ObjectInput          in;
```

```
    private final ObjectOutputStream   out;
    private int resetPeriod;
    private int                        resetCounter;
    private static int MAX_RESET = 1024; // with 100M allocated for
            //textsearchservice, this settled at around 256 and 512. -don.hayes

    // Client-side constructor: open a connection to remoteInetSocketAddress
    Connection(InetSocketAddress localInetSocketAddress, InetSocketAddress remoteInetSocketAddress,
CommunicationManager communicationManager) throws IOException, ClassNotFoundException {
        this.resetPeriod = MAX_RESET;
        this.resetCounter = 0;
        this.localInetSocketAddress  = localInetSocketAddress;
        this.remoteInetSocketAddress = remoteInetSocketAddress;
        this.communicationManager    = communicationManager;
        this.socket                  = new Socket(remoteInetSocketAddress.getAddress(),
remoteInetSocketAddress.getPort());
        // Important: open "out", flush "out", and then open "in".
        // (other side opens "in" and then "out")
        this.out    = new ObjectOutputStream(socket.getOutputStream());
        out.writeObject(localInetSocketAddress);
        out.flush();
        this.in     = new ObjectInputStream (socket.getInputStream ());
        (new Thread(this)).start();
        communicationManager.addConnection(this);
    }
    // Server-side constructor: receive a connection on socket
    Connection(InetSocketAddress localInetSocketAddress, Socket socket, CommunicationManager
communicationManager) throws IOException, ClassNotFoundException {
        this.resetPeriod = MAX_RESET;
        this.resetCounter = 0;
        this.localInetSocketAddress  = localInetSocketAddress;
        this.communicationManager    = communicationManager;
        this.socket                  = socket;
        // Important: open "in" and then "out", and flush "out".
        // (other side opens "out" and then "in")
        this.in                    = new ObjectInputStream (socket.getInputStream ());
        this.remoteInetSocketAddress = (InetSocketAddress) in.readObject();
        this.out                   = new ObjectOutputStream(socket.getOutputStream());
        out.flush();
        (new Thread(this)).start();
        communicationManager.addConnection(this);
        System.out.println("received connection from " + remoteInetSocketAddress.toString());
    }
    InetSocketAddress getInetSocketAddress() {
        return remoteInetSocketAddress;
    }
    void send(Request request) {
        try {
            synchronized (out) {
                out.flush();
                out.writeObject(request);
                resetCounter++;
                this.communicationManager.sentOneMessage();
                adjustResetPeriod();
                if (resetCounter >= resetPeriod) {
                    out.reset(); // reset the stream to avoid accumulating references. -don.hayes
                    resetCounter = 0;
                }
```

```
            }
        } catch (Exception e) {
            logger.error("Exception: ", e);
            // Drop message.
        }
    }
    protected double getJvmMemUsagePercent() {
        Runtime r = Runtime.getRuntime();
        long max = r.maxMemory();
        long total = r.totalMemory();
        long free = r.freeMemory();
        long realUsage = total - free;
        return ((double) realUsage / (double) max) * 100.0;
    }
    /**
     * Adjust the reset period up or down based on the memory usage.
     * In practice this decreases the resetPeriod to 0, because the cache
     * is greedy. -don.hayes
     */
    protected void adjustResetPeriod() {
        double memUsage = getJvmMemUsagePercent();
        if (memUsage > 70.0) {
            resetPeriod = resetPeriod / 2;
            if (resetPeriod < 1) {
                resetPeriod = 1;
            }
            logger.debug("decreasing resetPeriod to: " + resetPeriod);
        } else if ( (memUsage < 50.0) && (resetPeriod < MAX_RESET) ) {
            resetPeriod = resetPeriod * 2;
            logger.debug("increasing resetPeriod to: " + resetPeriod);
        }
    }
    synchronized void close() {
        try {
            socket.close();
        } catch (IOException e) {
            logger.error("Exception: ", e);
            // Ignore.
        }
    }
    public void run() {
        try {
            AddressActor.threadLocalCommunicationManager.set(communicationManager);
            while (true) {
                try {
                    Object object = in.readObject();
                    if (object instanceof Request) {
                        Request request = (Request) object;
                        this.communicationManager.recievedOneMessage();
                        communicationManager.call(request);
                    }
                } catch (ClassNotFoundException e) {
                    logger.error("Exception: ", e);
                    // Transient error, skip message.
                }
            }
        } catch (Exception e) {
            logger.error("Exception: ", e);
```

```
            // Abort.
            synchronized (this) {
               if (communicationManager != null) {
                  communicationManager.removeConnection(this);
               }
            }
         } finally {
            try {
               out.close();
            } catch (IOException e) {
               logger.error("IOException when closing: ", e);
               // ignore
            }
         }
      }

   }
```

DynamicDispatchBehavior.java

DynamicDispatchBehavior.java

```
package communication;

import java.lang.reflect.Method;

/**
 * <p>Is a <tt>Behavior</tt> adapter which performs <i>dynamic dispatch</i>
 * on the <i>class</i> of a <tt>Message</tt> object.  Objects of this class
 * wrap another <i>target</i> <tt>Behavior</tt>.  When
 * <tt>DynamicDispatchBehavior.call(Message)</tt> receives a message, it
 * looks at the <i>actual dynamic class</i> of the message, and dispatches
 * the message to the most specific matching <tt>call</tt> method within
 * the target <tt>Behavior</tt>.</p>
 *
 * <p>For example, say we have the classes <tt>A extends Message</tt> and
 * <tt>B extends A</tt>.  If a message of class <tt>B</tt> is received, the
 * instance of <tt>DynamicDispatchBehavior</tt> first checks whether its
 * target has a <tt>call(B)</tt> method.  If not, it then checks whether
 * the target has a <tt>call(A)</tt> method.  If so, the message is
 * dispatched to <tt>call(A)</tt>.</p>
 *
 * <p><b>Usage notes:</b></p>
 * <ol>
 * <li>All <tt>call()</tt> methods must specify a <b>subclass of
 * <tt>Message</tt></b> as their argument type.  Dynamic dispatch
 * <b>ignores</b> <tt>call</tt> methods whose argument type is an
 * <b>interface</b>.  Single inheritance amongst classes ensures that
 * exactly one <tt>call</tt> method will be the most specific match.</li>
 * <li>The <tt>Behavior</tt> passed to the constructor <i>must</i> belong
 * to a <b>public</b> class, and the overloaded <tt>call()</tt> methods
 * must also be <b>public</b>.  If you derive a class from
 * <tt>DynamicDispatchBehavior</tt>, this class must be <b>public</b>.
 * Otherwise, <tt>DynamicDispatchBehavior</tt> will not be able to access
 * the <tt>call()</tt> methods.  <i>(Note that you can hide a public inner
 * class inside a package-private class.)</i></li>
 * </ol>
 *
 */
public class DynamicDispatchBehavior implements Behavior {
   private Behavior behavior;
```

```
    /**
     * <p>Constructor called by subclasses.  If you create a class that
     * extends <tt>DynamicDispatchBehavior</tt> and has several overloaded
     * <tt>call</tt> methods, the appropriate overloaded <tt>call</tt>
     * method will automatically be called by <tt>call(Message)</tt>.</p>
     *
     * <p>Note that the subclass cannot usefully specify its own catch-all
     * <tt>call(Message)</tt> method, since that would override
     * <tt>DynamicDispatchBehavior.call(Message)</tt>.</p>
     */
    protected DynamicDispatchBehavior() {
        this.behavior = this;
    }
    /**
     * <p>Constructor for wrapping another <tt>Behavior</tt>.  If you
     * already have an existing <tt>Behavior</tt> object with serveral
     * overloaded <tt>call</tt> methods, you can pass it to this
     * constructor to create a <tt>Behavior</tt> with a
     * <tt>call(Message)</tt> method that calls the appropriate overloaded
     * <tt>call</tt> method of <tt>behavior</tt>.</p>
     *
     * <p>Unlike when subclassing, <tt>behavior</tt> may specify its own
     * catch-all <tt>call(Message)</tt> method.</p>
     *
     * @param  behavior  A <tt>Behavior</tt> with several overloaded
     *                   <tt>call</tt> methods.
     */
    public DynamicDispatchBehavior(Behavior behavior) {
        this.behavior = behavior;
    }
    /**
     * Dispatches <tt>message</tt> to the appropriate overloaded
     * <tt>call</tt> method.  If no matching method is found, the
     * <tt>message</tt> is simply discarded.
     *
     * @param  message  The <tt>Message</tt> to dispatch.
     */
    public void call(Message message) {
        // Try to find a more specific "call" method which accepts Message.
        Class<? extends Object  >  messageClass = Message.class;
        Class<? extends Object  > dispatchClass = message.getClass();
        Class<? extends Behavior> behaviorClass = behavior.getClass();
        while (!messageClass.equals(dispatchClass)) {
            try {
                Method method = behaviorClass.getMethod("call", dispatchClass);
                method.invoke(behavior, message);
                return;
            } catch (Exception e) {
                // Method not found.
            }
            dispatchClass = dispatchClass.getSuperclass();
        }
        // If we got here, we couldn't find a more specific method.
        // Only call the general case if it won't be a recursive.
        if (behavior != this) {
            behavior.call(message);
        }
    }
```

```
}
```

LocalActor.java

LocalActor.java

```
package communication;

import java.util.LinkedList;
import java.util.Queue;

final class LocalActor extends Actor {
   private static final long serialVersionUID = 1L;
   private final ActorAddress actorAddress;
   private final Behavior behavior;
   private      ActorThread actorThread;
   private final Queue<Message> messageQueue;

   LocalActor(ActorAddress actorAddress, Behavior behavior) {
      this.actorAddress = actorAddress;
      this.behavior     = behavior;
      this.actorThread  = null;
      this.messageQueue = new LinkedList<Message>();
   }

   LocalActor(CommunicationManager communicationManager, Behavior behavior) {
      this.actorAddress = communicationManager.getActorAddressFor(this);
      this.behavior     = behavior;
      this.actorThread  = null;
      this.messageQueue = new LinkedList<Message>();
   }

   public void send(final Message message) {
      synchronized (messageQueue) {
         messageQueue.add(message);
         if (actorThread == null) {
            actorThread = new ActorThread();
            actorThread.start();
         }
      }
   }

   private Object writeReplace() {
      return new AddressActor(actorAddress);
   }

   private class ActorThread extends Thread {
      public void run() {
         while (true) {
            Message message;
            synchronized (messageQueue) {
               message = messageQueue.poll();
               if (message == null) {
                  // Optional enhancement: make this thread wait for
                  // some time in case more messages do arrive.
                  actorThread = null;
                  return;
               }
            }
            behavior.call(message);
         }
```

```
        }
    }
}
```

LocalInetSocketAddress.java

LocalInetSocketAddress.java

```
package communication;

import java.io.IOException;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.UnknownHostException;

import org.apache.log4j.Logger;
import logging.CustomLogger;
import logging.LoggingConfig;
import environment.ServiceConfig;

/**
 * Encapsulates a method to find an available IP address and TCP port on
 * the current host.  Useful for calculating the <tt>InetSocketAddress</tt>
 * passed to <tt>Communication</tt>'s constructor.
 *
 */
public final class LocalInetSocketAddress {
    // CustomLogger uses LocalInetSocketAddress, so to avoid infinite recursion use a standard logger
    private static final Logger logger = Logger.getLogger("communication");
    private LocalInetSocketAddress() { }
    /**
     * Finds an available IP address and TCP port on the current host.
     * Assuming the host is properly configured, the IP address should be
     * reachable from other hosts (not 127.0.0.1 or 0.0.0.0).
     */
    public static InetSocketAddress getLocalInetSocketAddress() throws IOException,
UnknownHostException {
        Socket socket = new Socket();
        socket.bind(new InetSocketAddress(getLocalInetAddress(), 0));
        InetSocketAddress localInetSocketAdddress = (InetSocketAddress) socket.getLocalSocketAddress();
        socket.close();
        return localInetSocketAdddress;
    }
    /**
     * Returns the host's canonical IP address.
     * Can be set in IInstallationNames.INSTALLATION_DIR_NAME/etc/service-config.xml
     */
    public static InetAddress getLocalInetAddress() throws UnknownHostException {
        InetAddress localInetAddress = null;
        try {
            localInetAddress = (new ServiceConfig()).localInetAddress;
        } catch (Exception e) {
            logger.error("Exception: ", e);
        }
        if (localInetAddress == null) {
            localInetAddress = InetAddress.getLocalHost();
        }
        return localInetAddress;
    }
}
```

Message.java

Message.java

```
package communication;

import java.io.Serializable;

import utils.UniqueID;

/**
  * <tt>Message</tt> is an abstract base class for all messages which may be
  * sent to <tt>Actor</tt>s and dispatched to actor <tt>Behavior</tt>s.
  * <tt>Message</tt> implements <tt>Serializable</tt>, since Java
  * serialization may be used to transmit messages across the network.</p>
  *
  * <p>In general, classes implementing <tt>Message</tt> should contain only
  * <tt>public final</tt> fields, and no methods other than constructors and
  * possibly overloaded <tt>Object</tt> methods like <tt>equals(Object)</tt>,
  * <tt>hashCode()</tt>, and <tt>toString()</tt>.</p>
  *
  * @see  Actor
  * @see  Behavior
  */
public abstract class Message implements Serializable {
    /**
     * A UniqueID is included to allow profiling and debugging at the
     * transaction level, across multiple message sends.  In CodeInsight, a
     * new UniqueID is generated for each HTTP request.  All code that runs
     * in response to a given HTTP request must inherit the UniqueID, hence
     * it is included in all Messages.
     */
    public final UniqueID uniqueID;
    protected Message() {
        this.uniqueID = UniqueID.threadLocalStack.peek();
    }
}
```

MutableBehavior.java

MutableBehavior.java

```
package communication;

/**
  * Encapsulates a mutable reference to another <tt>Behavior</tt>.  This
  * allows you to pass a <tt>Behavior</tt> to the <tt>Communication</tt>
  * constructor, a behavior adapter, or elsewhere; and then later define or
  * modify what the <tt>Behavior</tt> does.  Useful for breaking recursive
  * dependencies.
  *
  */
public class MutableBehavior implements Behavior {
    /** A mutable reference to a <tt>Behavior</tt> object. */
    public Behavior behavior;
    /**
     * Constructs a <tt>MutableBehavior</tt> with no <tt>Behavior</tt>
     * initially assigned.
     */
    public MutableBehavior() {
        this.behavior = null;
    }
```

```
    /**
     * If <tt>behavior</tt> is defined, forward <tt>message</tt> to it.
     * Otherwise discard <tt>message</tt>
     *
     * @param  message
     */
    public void call(Message message) {
        behavior.call(message);
    }
}
```

RemoteActor.java

RemoteActor.java

```
package communication;

final class RemoteActor extends Actor {
    private static final long serialVersionUID = 1L;
    private final CommunicationManager communicationManager;
    final ActorAddress        actorAddress;

    RemoteActor(CommunicationManager communicationManager, ActorAddress actorAddress) {
        this.communicationManager = communicationManager;
        this.actorAddress         = actorAddress;
    }

    public void send(Message message) {
        communicationManager.sendTo(actorAddress, message);
    }

    private Object writeReplace() {
        return new AddressActor(actorAddress);
    }

    public boolean equals(Object object) {
        if (object instanceof RemoteActor) {
            RemoteActor remoteActor = (RemoteActor) object;
            return communicationManager.equals(remoteActor.communicationManager)
                && actorAddress.equals(remoteActor.actorAddress);
        } else {
            return false;
        }
    }

    public int hashCode() {
        return actorAddress.hashCode();
    }

    public ActorAddress getAddress() {
        return actorAddress;
    }
}
```

ReplyableMessage.java

ReplyableMessage.java

```
package communication;


/**
 * A Message object which includes a replyActor field.  Any request message
```

```
  * that expects a reply should usually be a subclass of ReplyableMessage.
  *
  */
public abstract class ReplyableMessage extends Message {
    public Actor replyActor;
    public ReplyableMessage(Actor replyActor) {
        this.replyActor = replyActor;
    }
}
```

Request.java

Request.java

```
package communication;

import java.io.Serializable;

/**
  * A <tt>Request</tt> is what's actually exchanged between
  * <tt>Communication</tt> instances.  This identifies a message and a
  * target <tt>Actor</tt> associated with the receiving
  * <tt>Communication</tt> instance.
  *
  */
class Request implements Serializable {
    private static final long serialVersionUID = 1L;
    final long              actorId;
    final Message           message;

    Request(long actorId, Message message) {
        this.actorId = actorId;
        this.message = message;
    }
}
```

WeakRegistry.java

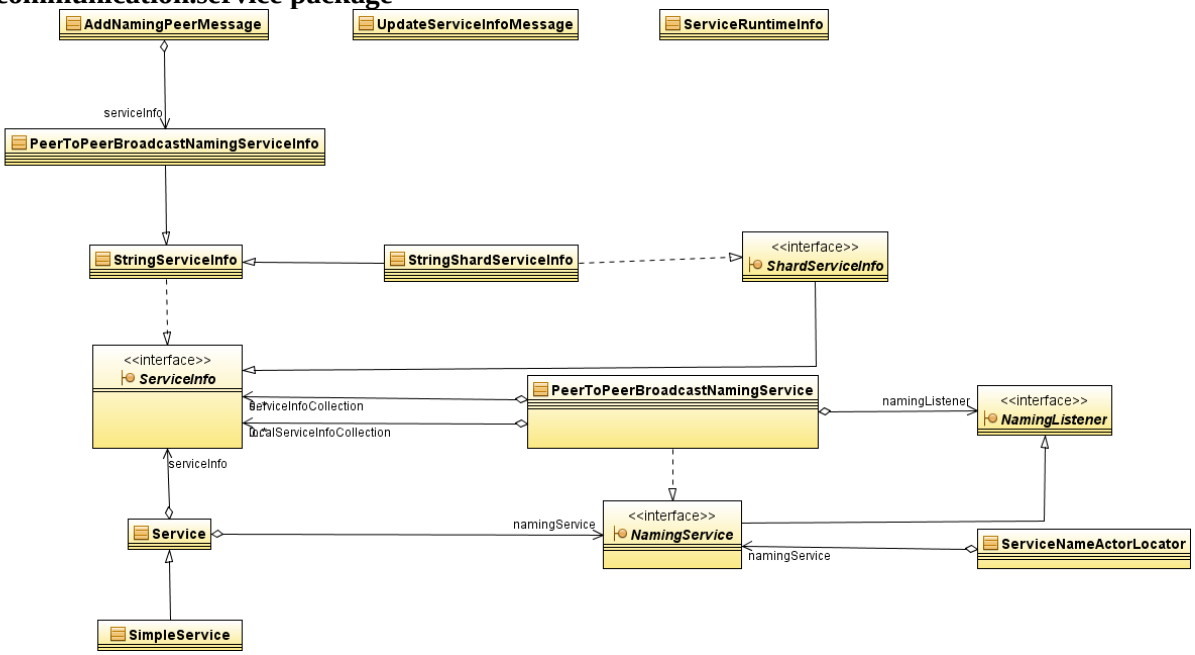
WeakRegistry.java

```
package communication;

import java.lang.ref.WeakReference;
import java.util.WeakHashMap;

/**
  * Maintains a weak set of objects, keyed by integer values.  Unlike
  * WeakHashMap, keys are simply "long" values, and if an object stored into
  * the WeakRegistry becomes only weakly-reachable, it will be discarded.
  * This is useful for maintaining a mapping between external ids and
  * objects, without the need to explicitly remove discarded objects from
  * the map.
  *
  * Keys may eventually be reused if you run out of positive longs.  But
  * that should take a while.
  *
  */
class WeakRegistry<T> {
    /**
      * Thrown in the highly unlikely case that every single key is
      * currently in use (as keys are signed longs, there are 2^63 possible
      * keys).
      */
```

```
private static class OutOfKeysError extends Error {
   private static final long serialVersionUID = 1L;
}

/** Unique id counter */
private long nextId;

/**
 * Theory of operation:
 *
 *  <p><tt>registry</tt> is a <tt>WeakHashMap</tt>, meaning entries
 *  will remain as long as someone else has a non-weak reference to
 *  the <tt>Long</tt> key object.</p>
 *
 *  <p><tt>inverseRegistry</tt> is also a <tt>WeakHashMap</tt>,
 *  meaning entries will remain as long as someone has a non-weak
 *  reference to the <tt>T</tt> key object.</p>
 *
 *  Since <tt>registry</tt> only contains <tt>WeakReference</tt>s to
 *  <tt>T</tt>s, an entry only remain in <tt>inverseRegistry</tt> as
 *  long as <i>THE APPLICATION</i> has a reference to the <tt>T</tt>.
 *
 *  Now as long as the actor-id <tt>Long</tt> key objects ARE <i>NOT
 *  REFERENCED ANYWHERE ELSE,</i> when the application discards a
 *  <tt>T</tt>, its entry is removed from <tt>inverseRegistry</tt>,
 *  which causes the <tt>Long</tt> key to be discarded, removing the
 *  mapping from <tt>registry</tt>.
 */
private final WeakHashMap<Long, WeakReference<T>> registry;
private final WeakHashMap<T, Long>                inverseRegistry;

public WeakRegistry() {
   this.nextId              = 1L;
   this.registry            = new WeakHashMap<Long, WeakReference<T>>();
   this.inverseRegistry     = new WeakHashMap<T, Long>();
}

/** Tries to allocate a new, unused  id.  The id returned is only
 *  guaranteed unique while in a "synchronized this" context.
 */
private long getId() throws OutOfKeysError {
   while (registry.containsKey(nextId) && nextId < Long.MAX_VALUE) {
      nextId++;
   }
   if (nextId == Long.MAX_VALUE) {
      nextId = 1L;
      while (registry.containsKey(nextId) && nextId < Long.MAX_VALUE) {
         nextId++;
      }
      if (nextId == Long.MAX_VALUE) {
         throw new OutOfKeysError();
      }
   }
   return nextId++;
}

/** Stores a reference to an item, returns a numeric id allowing access
 *  to that reference.  The reference is discarded when the rest of the
```

```
     *  application no longer refers to the item.
     */
    public synchronized long put(T item) throws OutOfKeysError {
        if (item == null)
            return 0L;
        Long id = new Long(getId());
        registry.put(id, new WeakReference<T>(item));
        inverseRegistry.put(item, id);
        return id.longValue();
    }

    /** Retrieves an item, given its numeric id (as returned by put()).  If
     *  id does not correspond to any item accesible by the application,
     *  then null is returned instead.
     */
    public synchronized T get(long id) {
        WeakReference<T> itemRef = registry.get(id);
        if (itemRef == null) {
            return null;
        } else {
            return itemRef.get();
        }
    }
}
```

The communication.service package



AddNamingPeerMessage.java

AddNamingPeerMessage.java

```
package communication.services;

import communication.Actor;
import communication.Message;

class AddNamingPeerMessage extends Message {
    private static final long serialVersionUID = 1L;
    public final PeerToPeerBroadcastNamingServiceInfo serviceInfo;
    public AddNamingPeerMessage(PeerToPeerBroadcastNamingServiceInfo serviceInfo) {
```

```
        this.serviceInfo = serviceInfo;
    }
    public AddNamingPeerMessage(Actor actor) {
        this.serviceInfo = new PeerToPeerBroadcastNamingServiceInfo(actor);
    }
}
```

NameListener.java

NamingListener.java

```
package communication.services;

/**
 * <p>Consists of methods called by the naming service to indicate changes
 * in the set of <tt>ServiceInfo</tt> objects.  Users may register a
 * <tt>NamingListener</tt> with the naming service's constructor in order
 * to receive notifications whenever a service's <tt>ServiceInfo</tt>
 * object is added to or retracted from the set of available services.</p>
 *
 * @see ServiceInfo
 */
public interface NamingListener {
    /**
     * Called when a new service is made available.
     *
     * @param  serviceInfo  A description of the new service
     */
    void add(ServiceInfo serviceInfo);
    /**
     * Called when a service is removed from the set of available services.
     *
     * @param  serviceInfo  The description of the removed service.  May
     *                      not be == to the <tt>ServiceInfo</tt>
     *         previously sent to <tt>add()</tt>; use <tt>equals()</tt> for
     *         comparison instead.
     */
    void remove(ServiceInfo serviceInfo);
}
```

NamingService.java

NamingService.java

```
package communication.services;

/**
 * <p>The abstract interface to naming services.  You can add or remove
 * information describing available services from a set of
 * <tt>ServiceInfo</tt> objects that are synchronized across naming
 * services running in different processes and on different hosts.  In
 * other words, <tt>NamingServices</tt> present the abstraction of a single
 * shared set of services known by all JVM instances.  You can also request
 * a copy of the set by calling <tt>getServiceInfos()</tt>.</p>
 *
 * <p>It is up to the user to decide what information is stored in a
 * <tt>ServiceInfo</tt> and how to choose services to communicate with.
 * Naming services simply synchronize this information across the
 * network.</p>
 *
 */
public interface NamingService extends NamingListener {
    /**
```

```
     * Add a <tt>ServiceInfo</tt> to the shared set of available service
     * information.  Automatically propagates the update to other instances
     * of the naming service.
     *
     * @param  serviceInfo
     */
    void add(ServiceInfo serviceInfo);
    /**
     * Remove a <tt>ServiceInfo</tt> from the shared set of available
     * service information.  Automatically propagates the update to other
     * instances of the naming service.  The <tt>serviceInfo</tt> provided
     * should be <tt>equal()</tt> to a <tt>ServiceInfo</tt> object
     * previously registered with <tt>add()</tt> (it need not be the same
     * exact object).
     *
     * @param  serviceInfo
     */
    void remove(ServiceInfo serviceInfo);
    /**
     * Returns a copy of all of the service information currently
     * available.  It is up to the caller to decide how to use these
     * <tt>ServiceInfo</tt> objects.
     */
    public Iterable<ServiceInfo> getServiceInfos();
}
```

PeerToPeerBroadcaseNamingService.java

PeerToPeerBroadcastNamingService.java

```
package communication.services;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;

import communication.Actor;
import communication.Communication;
import communication.DynamicDispatchBehavior;

/**
 * Implements one node/peer of a fully distributed peer-to-peer naming
 * service.  The general idea is that one naming peer operates as a naming
 * service in each JVM instance.  Different instances of the naming service
 * communicate, so that services registered in one JVM instance are also
 * known about by others.  The net effect is to maintain a global shared
 * set of known services.  The actual information exchange is instances of
 * the ServiceInfo class, which can contain any information&mdash;but in
 * general, should include some identifying description of a service and an
 * Actor reference that can be used to contact the service.
 *
 * @see ServiceInfo
 */
public class PeerToPeerBroadcastNamingService extends DynamicDispatchBehavior implements
NamingService {
    /**
     * The Actor that handles messages sent to us.  A reference is kept
     * here to prevent the actor from being prematurely garbage collected.
     */
    private final Actor                localPeerToPeerBroadcastNamingActor;
    /**
```

```
     * All ServiceInfo instances we know about across the network.
     */
    private final Collection<ServiceInfo> serviceInfoCollection;
    /**
     * All ServiceInfo instances which were created locally (not by a peer).
     */
    private final Collection<ServiceInfo> localServiceInfoCollection;
    /**
     * All naming peer Actors we know about.
     */
    private final Collection<Actor>       peerToPeerBroadcastNamingActors;
    /**
     * If namingListener is non-null, we will send it an event whenever a
     * ServiceInfo is added or removed.
     */
    private final NamingListener          namingListener;

    private PeerToPeerBroadcastNamingService(Communication communication, NamingListener
namingListener) {
        this.localPeerToPeerBroadcastNamingActor = communication.newActor(this);
        this.serviceInfoCollection               = new HashSet<ServiceInfo>();
        this.localServiceInfoCollection          = new HashSet<ServiceInfo>();
        this.peerToPeerBroadcastNamingActors     = new HashSet<Actor>();
        this.namingListener                      = namingListener;
    }

    /**
     * Create a new naming network which has no existing peers.
     *
     * @arg communication   An instace of the Communication system, which
     *                      allows us to create new Actors and communicate
     *                      with other services.
     * @arg namingListener  Register a NamingListener if you'd like to be
     *                      notified when a ServiceInfo is added or removed
     *                      by a peer.
     */
    public static PeerToPeerBroadcastNamingService createNewNetwork(Communication communication,
NamingListener namingListener) {
        PeerToPeerBroadcastNamingService ns = new PeerToPeerBroadcastNamingService(communication,
namingListener);
        ns.addLocalToCollection(new
PeerToPeerBroadcastNamingServiceInfo(ns.localPeerToPeerBroadcastNamingActor));
        return ns;
    }
    /**
     * Join an existing peer-to-peer naming network.
     *
     * @arg communication   An instace of the Communication system, which
     *                      allows us to create new Actors and communicate
     *                      with other services.
     * @arg initialPeerInetSocketAddress  IP:TcpPort of an existing peer, used
     *                                    to initialize peer-to-peer naming.
     * @arg namingListener  Register a NamingListener if you'd like to be
     *                      notified when a ServiceInfo is added or removed
     *                      by a peer.
     */
    public static PeerToPeerBroadcastNamingService joinExistingNetwork(Communication communication,
Actor initialPeerActor, NamingListener namingListener) {
```

```
        PeerToPeerBroadcastNamingService ns = new PeerToPeerBroadcastNamingService(communication,
namingListener);
        ns.addLocalToCollection(new
PeerToPeerBroadcastNamingServiceInfo(ns.localPeerToPeerBroadcastNamingActor));
        initialPeerActor.send(new AddNamingPeerMessage(ns.localPeerToPeerBroadcastNamingActor));
        // Wait until we know about our peers.
        synchronized (ns.peerToPeerBroadcastNamingActors) {
          try {
            ns.peerToPeerBroadcastNamingActors.wait();
          } catch (InterruptedException e) { }
        }
        return ns;
    }
    private void addLocalToCollection(ServiceInfo serviceInfo) {
        synchronized (localServiceInfoCollection) {
          localServiceInfoCollection.add(serviceInfo);
        }
        addToCollection(serviceInfo);
    }
    private void addToCollection(ServiceInfo serviceInfo) {
        synchronized (serviceInfoCollection) {
          serviceInfoCollection.add(serviceInfo);
        }
        if (serviceInfo instanceof PeerToPeerBroadcastNamingServiceInfo) {
          synchronized (peerToPeerBroadcastNamingActors) {
            peerToPeerBroadcastNamingActors.add(((PeerToPeerBroadcastNamingServiceInfo)
serviceInfo).actor);
          }
        }
        try {
          namingListener.add(serviceInfo);
        } catch (Exception e) { }
    }
    private void removeLocalFromCollection(ServiceInfo serviceInfo) {
        synchronized (localServiceInfoCollection) {
          localServiceInfoCollection.remove(serviceInfo);
        }
        removeFromCollection(serviceInfo);
    }
    private void removeFromCollection(ServiceInfo serviceInfo) {
        synchronized (serviceInfoCollection) {
          serviceInfoCollection.remove(serviceInfo);
        }
        if (serviceInfo instanceof PeerToPeerBroadcastNamingServiceInfo) {
          synchronized (peerToPeerBroadcastNamingActors) {
            peerToPeerBroadcastNamingActors.remove(((PeerToPeerBroadcastNamingServiceInfo)
serviceInfo).actor);
          }
        }
        try {
          namingListener.remove(serviceInfo);
        } catch (Exception e) { }
    }
    /**
     * A user can call add() to register a ServiceInfo, thus disseminating
     * it across the network.
     */
    public void add(ServiceInfo serviceInfo) {
```

```
        addLocalToCollection(serviceInfo);
        UpdateServiceInfoMessage updateServiceInfoMessage =
UpdateServiceInfoMessage.newAddMessage(serviceInfo);
        synchronized (peerToPeerBroadcastNamingActors) {
          for (Actor actor : peerToPeerBroadcastNamingActors) {
            try {
              actor.send(updateServiceInfoMessage);
            } catch (Exception e) { }
          }
        }
      }
      /**
       * A user can call remove() to revoke a previously registered
       * ServiceInfo, thus removing it from all naming service peers across
       * the network.
       */
      public void remove(ServiceInfo serviceInfo) {
        removeLocalFromCollection(serviceInfo);
        UpdateServiceInfoMessage updateServiceInfoMessage =
UpdateServiceInfoMessage.newRemoveMessage(serviceInfo);
        synchronized (peerToPeerBroadcastNamingActors) {
          for (Actor actor : peerToPeerBroadcastNamingActors) {
            try {
              actor.send(updateServiceInfoMessage);
            } catch (Exception e) { }
          }
        }
      }
      /**
       * Returns the set of ServiceInfo instances which the naming service
       * knows about at this instant in time.  If you need to wait for other
       * services to register themselves, you should provide a NamingListener
       * to the constructor rather than calling getServiceInfos().
       */
      public synchronized Iterable<ServiceInfo> getServiceInfos() {
        return new ArrayList<ServiceInfo>(serviceInfoCollection);
      }

      /**
       * When another peer wants to use us to bootstrap itself, it will send
       * us an AddNamingPeerMessage.  Since all naming peers can handle this
       * message, <i>any</i> service can act as a naming bootstrap.  It is
       * important that all peers which may act as a naming bootstrap have
       * access to the definitions of all ServiceInfo classes, otherwise some
       * ServiceInfos will be missing from their acquaintance set.
       *
       * Users: do not direcly invoke call().
       */
      public void call(AddNamingPeerMessage addNamingPeerMessage) {
        // Add the peer to our local view of the ServiceInfo set
        addToCollection(addNamingPeerMessage.serviceInfo);
        // Send our view of the ServiceInfo set back to the peer
        addNamingPeerMessage.serviceInfo.actor.send(new UpdateServiceInfoMessage(null,
getServiceInfos()));
        // Notify others (besides us and the peer) of the new peer.
        UpdateServiceInfoMessage updateServiceInfoMessage =
UpdateServiceInfoMessage.newAddMessage(addNamingPeerMessage.serviceInfo);
        synchronized (peerToPeerBroadcastNamingActors) {
```

```
            for (Actor actor : peerToPeerBroadcastNamingActors) {
               if (actor != addNamingPeerMessage.serviceInfo.actor
               &&  actor != localPeerToPeerBroadcastNamingActor) {
                  try {
                     actor.send(updateServiceInfoMessage);
                  } catch (Exception e) { }
               }
            }
         }
      }

      /**
       * When another peer adds or removes some services, it will broadcast
       * an UpdateServiceInfo message to all other peers.  This method
       * handles such messages.
       *
       * Users: do not direcly invoke call().
       */
      public void call(UpdateServiceInfoMessage updateServiceInfoMessage) {
         synchronized (serviceInfoCollection) {
            synchronized (peerToPeerBroadcastNamingActors) {
               if (updateServiceInfoMessage.removeServiceInfos != null) {
                  for (ServiceInfo serviceInfo : updateServiceInfoMessage.removeServiceInfos) {
                     removeFromCollection(serviceInfo);
                  }
               }
               if (updateServiceInfoMessage.addServiceInfos != null) {
                  for (ServiceInfo serviceInfo : updateServiceInfoMessage.addServiceInfos) {
                     addToCollection(serviceInfo);
                  }
               }
               // Wake up the constructor if it's waiting.
               peerToPeerBroadcastNamingActors.notify();
            }
         }
      }
   }
```

PeerToPeerBroadcaseNamingServiceInfo.java

PeerToPeerBroadcastNamingServiceInfo.java

```
package communication.services;

import communication.Actor;

class PeerToPeerBroadcastNamingServiceInfo extends StringServiceInfo {
   private static final long serialVersionUID = 1L;

   public PeerToPeerBroadcastNamingServiceInfo(Actor actor) {
      super("", actor);
   }
   public boolean equals(Object object) {
      if (object instanceof PeerToPeerBroadcastNamingServiceInfo) {
         return actor.equals(((PeerToPeerBroadcastNamingServiceInfo) object).actor);
      } else {
         return false;
      }
   }
   public int hashCode() {
      return actor.hashCode();
```

```
    }
}
```

Service.java

Service.java

```
package communication.services;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.lang.Runtime;

import communication.Actor;
import communication.Communication;
import communication.LocalInetSocketAddress;
import communication.MutableBehavior;

/**
 * Encapsulates some of the boilerplate for creating Actor-based services
 * that communicate with other services running in other JVM instances.
 *
 */
public class Service {
    /**
     * An initialized Communication instance which you can use to create
     * new Actors.
     */
    public final Communication communication;
    /**
     * A (hopefully bootstrapped) naming service that you can use to find
     * out about and contact other services running on the network.
     */
    public final NamingService namingService;

    /**
     * Each service needs to have <tt>ServiceInfo</tt>. It is needed to
     * register a service into the naming system.
     *
     * Make it not final, because it may be changed afterward.
     */
    public ServiceInfo serviceInfo;

    /**
     * Command line argument-based constructor.  Assumes args[0] is a
     * hostname or IP address and args[1] is a TCP port number of another
     * peer to use to bootstrap naming.
     */
    public Service(String[] args) throws IOException {
        this(args, null);
    }
    /**
     * Command line argument-based constructor.  Assumes args[0] is a
     * hostname or IP address and args[1] is a TCP port number of another
     * peer to use to bootstrap naming.
     */
    public Service(String[] args, NamingListener namingListener) throws IOException {
        this(new InetSocketAddress(args[0], Integer.parseInt(args[1])), namingListener);
    }
    public Service(InetSocketAddress initialPeerInetSocketAddress) throws IOException {
        this(initialPeerInetSocketAddress, (NamingListener) null);
```

```
    }
    public Service(InetSocketAddress initialPeerInetSocketAddress, NamingListener namingListener) throws
IOException {
        this(LocalInetSocketAddress.getLocalInetSocketAddress(), initialPeerInetSocketAddress,
namingListener);
    }
    public Service(InetSocketAddress localInetSocketAddress, InetSocketAddress
initialPeerInetSocketAddress) {
        this(localInetSocketAddress, initialPeerInetSocketAddress, null);
    }
    /**
     * Primary constructor.  Other constructors simply supply a subset of
     * this constructor's parameters.  Think of this as a function which
     * returns the two public fields, communication and namingService.
     *
     * @arg localInetSocketAddress  IP:TcpPort on which to listen
     * @arg initialPeerInetSocketAddress  IP:TcpPort of an existing peer, used
     *                              to initialize peer-to-peer naming.
     * @arg namingListener  Register a NamingListener if you'd like to be
     *                      notified when a ServiceInfo is added or removed
     *                      by a peer.
     */
    public Service(InetSocketAddress localInetSocketAddress, InetSocketAddress
initialPeerInetSocketAddress, NamingListener namingListener) {
        MutableBehavior mutableBehavior     = new MutableBehavior();
        this.communication                  = new Communication(localInetSocketAddress, mutableBehavior);
        Actor initialPeerActor              = communication.getDefaultActorFor(initialPeerInetSocketAddress);
        PeerToPeerBroadcastNamingService ns =
PeerToPeerBroadcastNamingService.joinExistingNetwork(communication, initialPeerActor,
namingListener);
        this.namingService                  = ns;
        mutableBehavior.behavior            = ns;
        this.serviceInfo                    = new StringServiceInfo("Unknown Name", null);
    }

    /**
     * Get runtime information of this <tt>Service</tt>.
     *
     * @return The runtime information, encapsulated in a
     *         <tt>ServiceRuntimeInfo</tt> object.
     */
    public ServiceRuntimeInfo getRuntimeInfo() {
        int memoryUsageInKB = (int)(Runtime.getRuntime().totalMemory() -
                              Runtime.getRuntime().freeMemory()) / 1024;
        int totalMemoryInKB = (int)(Runtime.getRuntime().maxMemory() / 1024);
        int inMsgCounter = communication.getIncomingMsgCounter();
        int outMsgCounter = communication.getOutgoingMsgCounter();
        int port = communication.getPort();
        return new ServiceRuntimeInfo(serviceInfo, memoryUsageInKB, totalMemoryInKB,
                                      inMsgCounter, outMsgCounter, port);
    }

    /**
     * Register a <tt>Service</tt>.
     *
     * Register a service to the naming system.
     *
     * @param serviceInfo The information about this <tt>Service</tt>.
```

```
     */
    public void register(ServiceInfo serviceInfo) {
        this.serviceInfo = serviceInfo;
        this.namingService.add(serviceInfo);
    }
}
```

ServiceInfo.java

ServiceInfo.java

```
package communication.services;

import java.io.Serializable;
import java.util.Date;

import communication.Actor;

/**
 * <p>Encapsulates information about an available service.  Typically, a
 * class implementing <tt>ServiceInfo</tt> will include some information
 * about a service's capabilities an <tt>Actor</tt> reference to allow
 * others to send messages to the service.</p>
 *
 * <p>The <tt>equals(Object)</tt> and <tt>hashCode()</tt> methods must be
 * properly implemented so that <tt>ServiceInfo</tt> objects can be
 * correctly added to and removed from a set data structure.  Service
 * information must also be serializable; hence a copy of a
 * <tt>ServiceInfo</tt> object should behave identically to the
 * original.</p>
 *
 * It is important that all peers which may act as a naming bootstrap have
 * access to the definitions of all ServiceInfo classes, otherwise some
 * ServiceInfos will be missing from their acquaintance set.
 *
 * @see com.patterninsight.lib.communication.Actor
 * @see Object
 */
public interface ServiceInfo extends Serializable {
    boolean equals(Object object);
    int hashCode();

    /**
     * Get the name string of this service.
     */
    String getName();

    /**
     * Get the actor of this service. Typically, a service should have an Actor.
     */
    Actor getActor();

    /**
     * Get the time when the service is started.
     */
    Date getStartingTime();

    /**
     * Get the location string of this service. This is more for the purpose
     * of displaying the location.
     */
```

```
    String getLocation();

    /**
     * Get the IP address of the host where this service is located.
     */
    String getHostAddress();
}
```

ServiceNameActorLocator.java

ServiceNameActorLocator.java

```
package communication.services;

import java.util.ArrayList;
import java.util.List;
import java.util.Collection;

import communication.Actor;
import communication.ActorLocator;

/**
  * An implementation of ActorLocator which works with a NamingService to
  * find matching StringServiceInfo instances.
  *
  */
public class ServiceNameActorLocator implements ActorLocator {
    private final NamingService namingService;
    public ServiceNameActorLocator(NamingService namingService) {
        this.namingService = namingService;
    }
    /**
     * Search for a StringServiceInfo with the given name, and return the
     * Actor field of the first match.  If there are multiple matches, whatever
     * is found first will be returned; if there are none, returns null.
     */
    public Actor getActorFor(String serviceName) {
        for (ServiceInfo serviceInfo : namingService.getServiceInfos()) {
            if (serviceInfo instanceof StringServiceInfo) {
                StringServiceInfo stringServiceInfo = (StringServiceInfo) serviceInfo;
                if (serviceName.equals(stringServiceInfo.name)) {
                    return stringServiceInfo.actor;
                }
            }
        }
        return null;
    }
    /**
     * Return all actors that match the given service name.
     * @author don.hayes@patterninsight.com
     */
    public Iterable<Actor> getAllActorsFor(String serviceName) {
        List<Actor> actors = new ArrayList<Actor>();
        for (ServiceInfo serviceInfo : namingService.getServiceInfos()) {
            if (serviceInfo instanceof StringServiceInfo) {
                StringServiceInfo stringServiceInfo = (StringServiceInfo) serviceInfo;
                if (serviceName.equals(stringServiceInfo.name)) {
                    actors.add(stringServiceInfo.actor);
                }
            }
        }
```

```
        return actors;
    }

    /**
     * Get a set of ServiceInfo according to a given name.
     *
     * @param serviceName The name of the service.
     * @return A set of ServiceInfo that match the name.
     */
    public Collection<StringShardServiceInfo> getAllServiceInfosFor(String serviceName) {
        List<StringShardServiceInfo> infos = new ArrayList<StringShardServiceInfo>();
        for (ServiceInfo serviceInfo : namingService.getServiceInfos()) {
            if (serviceInfo instanceof StringShardServiceInfo) {
                StringShardServiceInfo stringServiceInfo = (StringShardServiceInfo) serviceInfo;
                if (serviceName.equals(stringServiceInfo.name)) {
                    infos.add(stringServiceInfo);
                }
            }
        }
        return infos;
    }

    /**
     * Get a service info with a give name and is on the same machine as a given
     * ServiceInfo.
     *
     * @param serviceName The name of the service.
     * @param refServiceInfo The returned service should be on the same node as
     *                       refServiceInfo.
     * @return The service that match requirement.
     */
    public StringServiceInfo getServiceInfoOnSameNode(
                            ServiceInfo refServiceInfo,
                            String serviceName) {
        for (ServiceInfo serviceInfo : namingService.getServiceInfos()) {
            /* Only go through StringServiceInfo, because only StringServiceInfo
             * and its subclasses have names. */
            if (serviceInfo instanceof StringServiceInfo) {
                StringServiceInfo stringServiceInfo =
                                    (StringServiceInfo)serviceInfo;
                if (serviceName.equals(stringServiceInfo.getName()) &&
                    stringServiceInfo.isFromSameNode(refServiceInfo)) {
                        return stringServiceInfo;
                    }
            }
        }
        return null;
    }

}
```

ServiceRuntimeInfo.java

ServiceRuntimeInfo.java

```
package communication.services;

import java.io.Serializable;
import java.util.Date;

/** Encapsulates runtime information about a service.
```

```
 */
public class ServiceRuntimeInfo implements Serializable {
   private static final long serialVersionUID = 0L;
   public final int memoryUsage;
   public final int totalMemory;
   public final Date startingTime;
   public final String location;
   public final int port;
   public final String name;
   public int inMsgCounter;
   public int outMsgCounter;

   /** A constructor with data being initialized.
    */
   public ServiceRuntimeInfo (ServiceInfo serviceInfo,
                              int memoryUsage,
                              int totalMemory,
                              int incomingMessageCounter,
                              int outgoingMessageCounter,
                              int port) {
      this.name = serviceInfo.getName();
      this.memoryUsage = memoryUsage;
      this.totalMemory = totalMemory;
      this.inMsgCounter = incomingMessageCounter;
      this.outMsgCounter = outgoingMessageCounter;
      this.startingTime = serviceInfo.getStartingTime();
      this.location = serviceInfo.getLocation();
      this.port = port;
   }

   /** A constructor without data being initialized.
    */
   public ServiceRuntimeInfo() {
      this.memoryUsage = -1;
      this.totalMemory = -1;
      this.inMsgCounter = -1;
      this.outMsgCounter = -1;
      this.startingTime = null;
      this.location = "Unknown";
      this.name = "Unknown";
      this.port = -1;
   }

   /** Convert to a string.
    */
   public String toString() {
      return "[" + name + "] memory: " + memoryUsage;
   }
}
```

ShardServiceInfo.java

ShardServiceInfo.java

```
package communication.services;


/**
 * Represents a service that services up a piece of the entire data set (known as a shard).
 * For example, a keyword search worker might just serve up a small subset of all the projects,
 * and it broadcasts what shard it serves using this ServiceInfo
```

```
 *
 */
public interface ShardServiceInfo extends ServiceInfo {
    /**
     * A hash that uniquely identifies the shard.
     */
    public int shardHashCode();
}
```

SimpleService.java

SimpleService.java

```
package communication.services;

import java.io.IOException;
import java.net.InetSocketAddress;

import communication.Actor;
import communication.Behavior;
import environment.ServiceConfig;

/**
  * A simple way to create a single data source service.  Automatically
  * registers and Actor for the given Behavior, and registers the Actor with
  * the peer-to-peer naming service.  The naming service is provided a
  * StringServiceInfo, with name the same as the Behavior object's class's
  * short name (i.e., the name of the Java class, not including package).
  *
  * Loads the IP and port number for the naming server from service-config.xml
  *
  */
public class SimpleService extends Service {
    private Actor actor;

    public SimpleService(Behavior behavior) throws IOException,
ServiceConfig.InvalidConfigurationException {
        // Since call to super() needs to be first, can't share an instance of ServiceConfig
        super(new InetSocketAddress((new ServiceConfig()).namingBootstrapHost,
                                    (new ServiceConfig()).namingBootstrapPort));
        this.actor = communication.newActor(behavior);
        namingService.add(new StringServiceInfo(behavior.getClass().getSimpleName(), actor));
    }
}
```

StringServiceInfo.java

StringServiceInfo.java

```
package communication.services;

import java.net.InetAddress;
import java.util.Date;

import communication.Actor;

/**
  * A simple instantiation of ServiceInfo, useful for simple services that
  * don't have a better way to be located.  Simply associates a name String
  * with an Actor.
  *
  * This class is intended to be used in most of the cases. If there is
  * specific data for a certain type of services, you can write a subclass
```

```
 * of this to handle those data.
 *
 */
public class StringServiceInfo implements ServiceInfo {
   private static final long serialVersionUID = 0L;
   public final String name;
   public final Actor  actor;
   public final Date startingTime;
   public InetAddress location;

   public StringServiceInfo(String name, Actor actor) {
      this.name  = name;
      this.actor = actor;
      this.startingTime = new Date();
      try {
         this.location = InetAddress.getLocalHost();
      } catch (Exception e) {
         this.location = null;
         //pass
      }
   }

   // Methods required by the ServiceInfo interface:
   @Override
   public boolean equals(Object object) {
      if (object instanceof StringServiceInfo) {
         StringServiceInfo stringServiceInfo = (StringServiceInfo) object;
         return  name.equals(stringServiceInfo.name)
               && actor.equals(stringServiceInfo.actor);
      } else {
         return false;
      }
   }

   public int hashCode() {
      return name.hashCode();
   }

   public String toString() {
      return name;
   }

   public String getName() {
      return name;
   }

   public Actor getActor() {
      return actor;
   }

   public Date getStartingTime() {
      return startingTime;
   }

   public String getLocation() {
      if (location == null)
         return "Unknown location";
      else
```

```
            return location.getHostAddress() + "(" + location.getHostName() + ")";
        }

        /**
         * Get the IP address of the host where the service is located.
         */
        public String getHostAddress() {
            if (location != null) {
                return location.getHostAddress();
            }
            else {
                return null;
            }
        }

        /**
         * Tell if a given serviceInfo is from the same node.
         *
         * This method use IP address to test if two serviceInfos are from the same
         * machine(node) or not.
         *
         * @param serviceInfo The serviceInfo you want to test.
         * @return Tells if the given serviceInfo is from the same node or not.
         */
        public boolean isFromSameNode(ServiceInfo serviceInfo) {
            if (this.location == null) {
                return false;
            }
            else {
                return this.location.getHostAddress().equals(serviceInfo.getHostAddress());
            }
        }
    }
```

StringShardServiceInfo.java

StringShardServiceInfo.java

```
package communication.services;

import communication.Actor;

/**
 * A combination of the ShardServiceInfo and a StringServiceInfo
 *
 */
public class StringShardServiceInfo extends StringServiceInfo implements ShardServiceInfo {
    private static final long serialVersionUID = 0L;
    private int shard;

    public StringShardServiceInfo(String name, Actor actor, int shard) {
        super(name, actor);
        this.shard = shard;
    }

    public int shardHashCode() {
        return shard;
    }

    public String toString() {
        return name + ":" + shard;
```

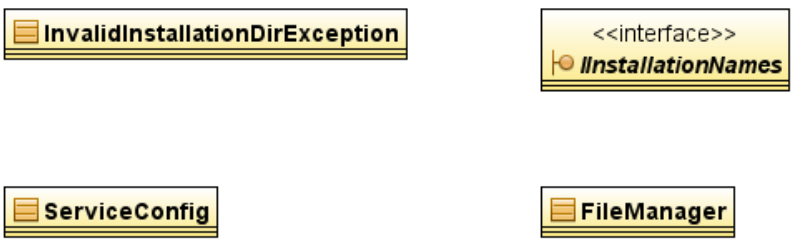
```
    }
}
```

UpdateServiceInfoMessage.java

UpdateServiceInfoMessage.java

```
package communication.services;

import java.util.LinkedList;

import communication.Message;

class UpdateServiceInfoMessage extends Message {
   private static final long serialVersionUID = 1L;
   public final Iterable<ServiceInfo> removeServiceInfos;
   public final Iterable<ServiceInfo> addServiceInfos;
   public UpdateServiceInfoMessage(Iterable<ServiceInfo> removeServiceInfos, Iterable<ServiceInfo>
addServiceInfos) {
      this.removeServiceInfos = removeServiceInfos;
      this.addServiceInfos    = addServiceInfos;
   }
   public UpdateServiceInfoMessage(ServiceInfo removeServiceInfo, ServiceInfo addServiceInfo) {
      if (removeServiceInfo != null) {
         LinkedList<ServiceInfo> serviceInfos = new LinkedList<ServiceInfo>();
         serviceInfos.add(removeServiceInfo);
         removeServiceInfos = serviceInfos;
      } else {
         removeServiceInfos = null;
      }
      if (addServiceInfo != null) {
         LinkedList<ServiceInfo> serviceInfos = new LinkedList<ServiceInfo>();
         serviceInfos.add(addServiceInfo);
         addServiceInfos = serviceInfos;
      } else {
         addServiceInfos = null;
      }
   }
   public static UpdateServiceInfoMessage newAddMessage(ServiceInfo serviceInfo) {
      return new UpdateServiceInfoMessage(null, serviceInfo);
   }
   public static UpdateServiceInfoMessage newRemoveMessage(ServiceInfo serviceInfo) {
      return new UpdateServiceInfoMessage(serviceInfo, null);
   }
}
```

The environment package



FileManager.java

FileManager.java

```
package environment;

import java.io.File;
import java.io.FileNotFoundException;

import utils.FileUtils;

/**
 * FileManager is used to obtain directories and files from the
 * installation directory and the repository of PI. It is the only
 * place where files and directories should be obtained from.
 *
 */
public class FileManager {

   private static String INSTALLATION_DIR_LOCATION =
      System.getenv(IInstallationNames.INSTALLATION_DIR_ENV_VARIABLE_NAME);

   /**
    * Return the installation location without checking if it exists
    */
   public static String unsafeInstallationLocation() {
      return INSTALLATION_DIR_LOCATION;
   }

   /**
    * The installation location. We make sure that it exists. If it does not
    * an InvalidInstallationDirException is thrown
    *
    * @return A String that represents the full path to the installation location
    * @throws InvalidInstallationDirException When the installation location is invalid
    */
   public static String installationLocation() throws InvalidInstallationDirException {
      return installationDir().getAbsolutePath();
   }

   /**
    * Returns the installation directory of PI. If it is invalid, i.e. it does
    * not exist, or it's not a directory or the environment variable that
    * points to it has not been defined, an InvalidInstallationDirException is
    * thrown
    *
    * @return The File object that represents the installation directory of PI
    * @throws InvalidInstallationDirException
    *            When the dir cannot be located or is invalid
    */
   public static File installationDir() throws InvalidInstallationDirException {
      try {
         return getSafeDirectory(INSTALLATION_DIR_LOCATION,
                  IInstallationNames.INSTALLATION_DIR_ENV_VARIABLE_NAME);
      } catch (Exception e) {
         throw new InvalidInstallationDirException(e.getMessage());
      }
```

```
    }

    /**
     * Get the File object that corresponds to the given subdirectory inside the
     * installation directory. The file is constructed by concatenating the give
     * path and the installation directory path:<br/>
     * <tt>File file = new File(installationDir, subDir);</tt>
     * <br/> If the file does not exist a FileNotFoundException is thrown. If
     * the installation dir is not know (i.e. env variable was not defined) a
     * InvalidInstallationDirException is thrown.
     *
     * @param subLocation
     *            The path of the file to create a File object for. It should
     *            represent the path of the file under the installation
     *            directory. E.g.: <i>etc/search-cache-config.xml</i>
     * @return The File object that corresponds to the file/directory
     * @throws FileNotFoundException
     *             When the file does not exist
     * @throws InvalidInstallationDirException
     *             When the installation dir is not known or it does not exist
     */
    public static File getFileInInstallation(String subLocation) throws FileNotFoundException,
            InvalidInstallationDirException {
        String location = FileUtils.join(installationLocation(), subLocation);
        File file = new File(location);
        if (!file.exists()) {
            throw new FileNotFoundException(file.getAbsoluteFile() + " does not exist");
        }
        return file;
    }

    /**
     * Get the absolute path that corresponds to the given subdirectory inside the
     * installation directory. The file is constructed by concatenating the give
     * path and the installation directory path<br/>
     * <br/> If the file does not exist a FileNotFoundException is thrown. If
     * the installation dir is not know (i.e. env variable was not defined) a
     * InvalidInstallationDirException is thrown.
     *
     * @param subLocation
     *            The path of the file inside the installation directory. It should
     *            represent the path of the file under the installation
     *            directory. E.g.: <i>etc/search-cache-config.xml</i>
     * @return The String that corresponds to the file/directory
     * @throws FileNotFoundException
     *             When the location does not exist
     * @throws InvalidInstallationDirException
     *             When the installation dir is not known or it does not exist
     */
    public static String getLocationInInstallation(String subLocation) throws FileNotFoundException,
            InvalidInstallationDirException {
        return getFileInInstallation(subLocation).getAbsolutePath();
    }

    /**
     * Get the path to the location of the given file in the installation directory.
     * It is unsafe because the file might not exist.
     *
```

```
     * @param The path of the file inside the installation directory. It should
     *        represent the path of the file under the installation
     *        directory. E.g.: <i>etc/search-cache-config.xml</i>
     *
     * @return The String that corresponds to the file/directory
     */
    public static String getUnsafeLocationInInstallation(String subLocation) {
        return FileUtils.join(unsafeInstallationLocation(), subLocation);
    }

    /**
     * Get the File object that represents the give location in the installation directory.
     * It is unsafe because the file might not exist.
     *
     * @param The path of the file inside the installation directory. It should
     *        represent the path of the file under the installation
     *        directory. E.g.: <i>etc/search-cache-config.xml</i>
     *
     * @return The File object that corresponds to the file/directory
     */
    public static File getUnsafeFileInInstallation(String subLocation) {
        return new File(getUnsafeLocationInInstallation(subLocation));
    }

    /**
     * A help function that checks if the installation/data directory exists and if it is
     * a valid directory. If the check is passed, return a Java <tt>File</tt> object of
     * that directory.
     *
     * @param location The location of the directory.
     * @param env The name of the environment variable that specifies the directory.
     * @return A java <tt>File</tt> of that directory.
     * @throw Exception, containing the message of why the checking hasn't passed.
     */
    private static File getSafeDirectory(String location, String env) throws Exception {
        if (location == null) {
            throw new Exception("Environment variable \"" + env + "\" is not set");
        }
        File dir = new File(location);
        if (!dir.exists()) {
            throw new Exception("Environment variable \"" + env + "\" points to " +
                                "a directory that does not exist.");
        }
        if (!dir.isDirectory()) {
            throw new Exception("Environment variable \"" + env + "\" points to " +
                                "a file instead of a directory.");
        }
        return dir;
    }

    /**
     * Reset the location of the installation directory, which was read from
     * the environment variable before. This method is used internally for
     * the purpose of testing.
     *
     * NOTE: This method has to be public, because the testing code in other
     *       packages also needs it.
     *
```

```
     * @param newLocation The new installation directory you want to set.
     */
    public static void resetInstallationDirLocation(String newLocation) {
        INSTALLATION_DIR_LOCATION = newLocation;
    }
}
```

IInstallationNames.java

IInstallationNames.java

```
package environment;

import utils.FileUtils;

/**
 * Interface with all the names in the installation directory of CAT.
 * Every name should only be hardcoded here.
 *
 */
public interface IInstallationNames {
    public static final String INSTALLATION_DIR_ENV_VARIABLE_NAME = "CAT_HOME";
    public static final String LOG4J_CONFIGURATION =  FileUtils.join("etc", "log4j.properties");
    public static final String SERVICE_CONFIGURATION = FileUtils.join("etc", "service-config.xml");
    public static final String DATABASE_CONFIGURATION = FileUtils.join("etc", "database-config.xml");
}
```

InvalidInstallationDirException.java

InvalidInstallationDirException.java

```
package environment;

public class InvalidInstallationDirException extends Exception {

    private static final long serialVersionUID = 1L;

    public InvalidInstallationDirException(String message) {
        super(message);
    }
}
```

ServiceConfig.java

ServiceConfig.java

```
package environment;

import java.io.File;
import java.net.InetAddress;

import org.jdom.Element;
import org.jdom.input.SAXBuilder;

/**
  * Encapsulates various configurable parameters that apply to all services.
  * The public final fields of this class correspond to the settings in a
  * configuration file.  Normally, this data is stored in
  *
<tt>IInstallationNames.INSTALLATION_DIR_NAME/IInstallationNames.SERVICE_CONFIGURATION
</tt>
  */
public class ServiceConfig {
    /**
      * Exception thrown to indicate a malformed or missing configuration file.
```

```
     */
    public static class InvalidConfigurationException extends Exception {
        private static final long serialVersionUID = 1L;
        public InvalidConfigurationException() { }
        public InvalidConfigurationException(String message) {
            super(message);
        }
        public InvalidConfigurationException(String message, Throwable cause) {
            super(message, cause);
        }
        public InvalidConfigurationException(Throwable cause) {
            super(cause);
        }
    };

    /**
     * The local IP address on which to listen for communication.
     */
    public final InetAddress localInetAddress;
    /**
     * The amount of time, in milliseconds, to wait for a reply to a
     * request message before considering the request as failed.
     */
    public final long synchronousRequestTimeout;
    /**
     * The host on which the service we should use as a naming bootstrap
     * resides.  For example, the management service can be used as a
     * naming bootstrap.
     */
    public final String namingBootstrapHost;
    /**
     * The TCP port number on <tt>namingBootstrapHost</tt> on which the
     * naming bootstrap is listening.
     */
    public final int namingBootstrapPort;

    /**
     * Load the service configuration from
     * <tt>IInstallationNames.INSTALLATION_DIR_NAME/etc/service-config.xml</tt>
     */
    public ServiceConfig() throws InvalidConfigurationException {
        this(FileManager.getUnsafeFileInInstallation(IInstallationNames.SERVICE_CONFIGURATION));
    }

    /**
     * Load a service configuration from an arbitrary file.
     */
    public ServiceConfig(File configFile) throws InvalidConfigurationException {
        try {
            Element configElement = new SAXBuilder().build(configFile).getRootElement();
            if (!configElement.getName().equals("service-config")) {
                throw new InvalidConfigurationException(configFile.toString() + ": <service-config> tag not
found.");
            }
            Element hostElement = configElement.getChild("host");
            if (hostElement != null) {
                this.localInetAddress = InetAddress.getByName(hostElement.getAttribute("ip").getValue());
            } else {
```

```
            this.localInetAddress = null;
          }
          Element synchronousRequestTimeoutElement = configElement.getChild("synchronous-request-
timeout");
          this.synchronousRequestTimeout =
Long.parseLong(synchronousRequestTimeoutElement.getAttribute("milliseconds").getValue());
          Element namingBootstrapElement = configElement.getChild("naming-bootstrap");
          this.namingBootstrapHost = namingBootstrapElement.getAttribute("host").getValue();
          this.namingBootstrapPort = Integer.parseInt(namingBootstrapElement.getAttribute("tcp-
port").getValue());
        } catch (InvalidConfigurationException e) {
          throw e;
        } catch (Exception e) {
          throw new InvalidConfigurationException(e);
        }
    }

    /**
     * Test case: read and print out the service configuration.
     */
    public static void main(String[] args) throws Exception {
        ServiceConfig serviceConfig = new ServiceConfig();
        System.out.println("synchronousRequestTimeout: " + serviceConfig.synchronousRequestTimeout);
        System.out.println("      namingBootstrapHost: " + serviceConfig.namingBootstrapHost);
        System.out.println("      namingBootstrapPort: " +
Integer.toString(serviceConfig.namingBootstrapPort));
    }
}
```

The logging package

CustomeLogger.java

CustomLogger.java

```
package logging;

import java.net.UnknownHostException;
import java.net.InetAddress;
import org.apache.log4j.Logger;
import utils.UniqueID;

/**
 * <p>Wraps a Logger, including additional information in the logging
 * message.  Since I don't know of a standard mechanism in log4j 1.2.14 to
```

```
 * include additional information, we simply construct a string that
 * extends the message's string.</p>
 *
 * <p>Currently includes the following information:
 * <ul>
 *     <li> caller's class/method/line number
 *     <li> current transaction's UniqueID
 *     <li> this host's IP address
 *     <li> the user-specified message
 * </ul></p>
 *
 * <p>Note that as currently implemented, the message string will be
 * constructed regardless of whether the wrapped Logger ends up logging the
 * event.</p>
 */
public class CustomLogger extends Logger {
   private Logger logger;

   /**
    * Create a new logger which modifies the messages sent to <tt>logger</tt>.
    *
    * @param  logger  The logger to wrap
    */
   public CustomLogger(Logger logger) {
      super(logger.getName());
      this.logger = logger;
   }

   public static Logger getLogger(String name) {
      return new CustomLogger(Logger.getLogger(name));
   }
   public void debug(Object message) {
      logger.debug(formatLogEntry(message));
   }
   public void debug(Object message, Throwable t) {
      logger.debug(formatLogEntry(message), t);
   }
   public void info(Object message) {
      logger.info(formatLogEntry(message));
   }
   public void info(Object message, Throwable t) {
      logger.info(formatLogEntry(message), t);
   }
   public void warn(Object message) {
      logger.warn(formatLogEntry(message));
   }
   public void warn(Object message, Throwable t) {
      logger.warn(formatLogEntry(message), t);
   }
   public void error(Object message) {
      logger.error(formatLogEntry(message));
   }
   public void error(Object message, Throwable t) {
      logger.error(formatLogEntry(message), t);
   }
   public void fatal(Object message) {
      logger.fatal(formatLogEntry(message));
   }
```

```
    public void fatal(Object message, Throwable t) {
        logger.fatal(formatLogEntry(message), t);
    }

    private String formatLogEntry(Object message) {
        String ipAddress = "unknown host";
        try {
            ipAddress = InetAddress.getLocalHost().toString();
        } catch (UnknownHostException e) {}
        return getCaller() + "] ["
            + UniqueID.currentString() + "] ["
            + ipAddress + "] ["
            + message.toString();
    }

    /**
     * Computes the class, method, and line number of the method that
     * called debug/info/warn/error/fatal above.
     */
    private String getCaller() {
        StackTraceElement ste = (new Throwable()).getStackTrace()[3];
        return ste.getClassName() + "." + ste.getMethodName() + "():" + ste.getLineNumber();
    }
}
```

LoggingConfig.java

LoggingConfig.java

```
package logging;

import java.io.File;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

import org.apache.log4j.Level;
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

import environment.IInstallationNames;
import environment.FileManager;

/**
 * <p>Configures log4j appenders. This class should be used before invoking any
<code>Logger.getLogger()</code>
 * to properly initialize logging appenders. It supports appenders for file loggging, remote logging,
 * console logging and file+remote logging.</p>
 *
 * <P>In order to use the logging mechanism in a process, one of the <code>config*()</code> methods in
this
 * class should be invoked at the very beggining, in main() methods usually.
 *
 * Afterwards, a logger can be obtained by: <br/>
 *
 * <pre> Logger logger = Logger.getLogger(MyClass.class.toString())</pre>
 *
 * <p>The String <code>MyClass.class.toString()</code> is always preprended at the logging messages.
 * It can be replaced by any other string, but is not recommended because it will cause confusion.</p>
 *
 */
```

```
public class LoggingConfig {
   /** The name of the default file appender. */
   private static final String FILE_LOGGER = "FILE";

   /** The name of the default console appender. */
   private static final String CONSOLE_LOGGER = "CONSOLE";

   /** The name of the default remote appender. */
   private static final String REMOTE_LOGGER = "REMOTE";

   /** The default logging level: DEBUG. */
   private static final Level DEFAULT_LOGGING_LEVEL = Level.DEBUG;

   /** The default delay time before try to reconnect. */
   private static final int DEFAULT_RECONNECTION_DELAY_IN_MILLIS = 5000;

   /** The default logging server port. */
   private static final int DEFAULT_PORT = 58778;

   /** The default logging server host. */
   private static final String DEFAULT_HOST = "localhost";

   /** The default log layout. */
   private static final String DEFAULT_LAYOUT = "org.apache.log4j.PatternLayout";

   /** The default layout conversion pattern. */
   private static final String DEFAULT_LAYOUT_PATTERN = "%d{ISO8601} [%t] %-5p %c %x -
%m%n";

   // No public constructor for this class. Just call static methods.
   private LoggingConfig() {}

   /**
    * This method configures the appenders using the configuration in the given file.
    * Should be avoided unless there is a need for doing something other than the default
    * functions provided in this class.
    *
    * @param file The file that contains log4j configuration
    */
   public static void configFromFile(File file) {
      PropertyConfigurator.configure(file.getAbsolutePath());
   }

   /**
    * This method configures the appenders using the configuration in the given property list.
    * Should be avoided unless there is a need for doing something other than the default
    * functions provided in this class.
    *
    * @param props The properties object containing the configuration
    */
   public static void configWithProperties(Properties properties) {
      PropertyConfigurator.configure(properties);
   }

   /**
    * Configure the appenders for remote logging. This can be useful to see the order of events
    * by all the services that do remote logging.
    *
```

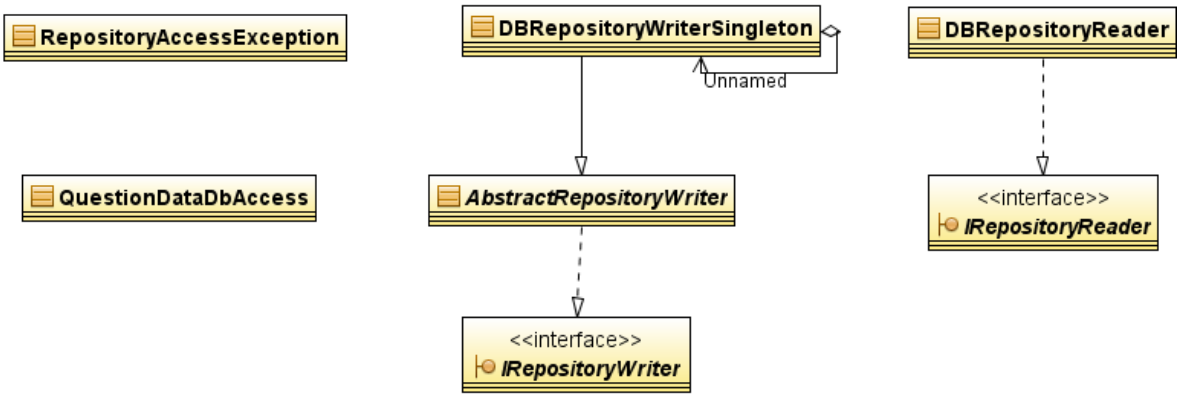
```
     * @param level The level of logging to be used (DEBUG &lt; INFO &lt; WARN &lt; ERROR &lt;
FATAL).
     *       It can be overriden by individual logger using logger.setLevel(Level);
     * @param host The host to which should send the logs. Should be an IP or localhost
     * @param port The port in which the host listens
     */
    public static void configForDistributedLogging(Level level, String host, int port) {
        Properties props = distributedLoggingProps(level, host, port);
        props.setProperty("log4j.rootLogger", appendersProperty(level.toString(), REMOTE_LOGGER));
        configWithProperties(props);
    }

    /**
     * Configure the appenders for remote logging using the given level. It does not require to
     * specify host and port, uses the defaults.
     *
     * @param level The level of logging to be used (DEBUG &lt; INFO &lt; WARN &lt; ERROR &lt;
FATAL).
     *       It can be overriden by individual logger using logger.setLevel(Level);
     */
    public static void configForDistributedLogging(Level level) {
        configForDistributedLogging(level, DEFAULT_HOST, DEFAULT_PORT);
    }

    /**
     * <p>Configure the appenders for remote logging using the default configuration file.
     * Currently, the file is located as <code>$CAT_HOME/etc/log4j.properties</code>.
     * Its location is up to change. If the file doesn't exist or
     * its content is not valid Log4j configuration content, configure the remote logging
     * by the default hostname/port and the default logging level of DEBUG.</p>
     *
     * <p><b>NOTE</b>: This method should be the prefered initialization method under normal
     * circumstances.</p>
     */
    public static void configForDistributedLogging() {
        try {
            File configFile = FileManager.getFileInInstallation(IInstallationNames.LOG4J_CONFIGURATION);
            configFromFile(configFile);
        } catch (Exception e) {
            // if an exception occurs, configure for distributed logging using DEBUG Level
            configForDistributedLogging(Level.DEBUG);
        }
    }

    /**
     * Configure the appenders for file logging in the given file using the given level.
     *
     * @param level The level of logging to be used (DEBUG &lt; INFO &lt; WARN &lt; ERROR &lt;
FATAL).
     *       It can be overriden by individual logger using logger.setLevel(Level);
     * @param file The path to the file that logs will be appended. If the file is the same
     *       across different sessions it appends at the end of it.
     */
    public static void configForFileLogging(Level level, String file) {
        Properties props = fileLoggingProps(level, file);
        props.setProperty("log4j.rootLogger", appendersProperty(level.toString(), FILE_LOGGER));
        configWithProperties(props);
    }
```

```
    /**
     * Configure the appenders for file logging in the given file using the default logging level
     * (DEBUG).
     *
     * @param file The path to the file that logs will be appended. If the file is the same
     *        across different sessions it appends at the end of it.
     */
    public static void configForFileLogging(String file) {
        configForFileLogging(DEFAULT_LOGGING_LEVEL, file);
    }

    /**
     * Configure the appenders for console logging using the given level.
     *
     * @param level The level of logging to be used (DEBUG &lt; INFO &lt; WARN &lt; ERROR &lt;
FATAL).
     *        It can be overriden by individual logger using logger.setLevel(Level);
     */
    public static void configForConsoleLogging(Level level) {
        Properties props = consoleLoggingProps(level);
        props.setProperty("log4j.rootLogger", appendersProperty(level.toString(), CONSOLE_LOGGER));
        configWithProperties(props);
    }

    /**
     * Configure the appenders for console logging using the default logging level.
     */
    public static void configForConsoleLogging() {
        configForConsoleLogging(DEFAULT_LOGGING_LEVEL);
    }

    /**
     * Configure the appenders for file logging in the given file and at the same time for remote
     * logging to the given host/port.
     *
     * @param level The level of logging to be used (DEBUG &lt; INFO &lt; WARN &lt; ERROR &lt;
FATAL).
     *        It can be overriden by individual logger using logger.setLevel(Level);
     * @param file The path to the file that logs will be appended. If the file is the same
     *        across different sessions it appends at the end of it.
     * @param host The host to which should send the logs. Should be an IP or localhost
     * @param port The port in which the host listens
     */
    public static void configForFileAndDistributedLogging(Level level, String file, String host, int port) {
        Properties fileProps = fileLoggingProps(level, file);
        Properties distributedProps = distributedLoggingProps(level, host, port);

        Properties union = new Properties();
        addAll(fileProps, union);
        addAll(distributedProps, union);

        union.setProperty("log4j.rootLogger",
                    appendersProperty(level.toString(), REMOTE_LOGGER, FILE_LOGGER));
        configWithProperties(union);
    }

    /**
```

```
     * Configure the appenders for file logging in the given file and at the same time for remote
     * logging.
     *
     * @param file The path to the file that logs will be appended. If the file is the same
     *        across different sessions it appends at the end of it.
     */
    public static void configForFileAndDistributedLogging(String file) {
        configForFileAndDistributedLogging(DEFAULT_LOGGING_LEVEL, file, DEFAULT_HOST,
DEFAULT_PORT);
    }


    /**
     * Provide properties for writing logs to a remote logging server.
     *
     * @param level The logging leve.
     * @param host The hostname of the logging server.
     * @param port The port of the logging server.
     */
    private static Properties distributedLoggingProps(Level level, String host, int port) {
        Properties props = new Properties();
        props.setProperty("log4j.appender." + REMOTE_LOGGER, "org.apache.log4j.net.SocketAppender");
        props.setProperty("log4j.appender." + REMOTE_LOGGER + ".Threshold", level.toString());
        props.setProperty("log4j.appender." + REMOTE_LOGGER + ".RemoteHost", host);
        props.setProperty("log4j.appender." + REMOTE_LOGGER + ".Port", String.valueOf(port));
        props.setProperty("log4j.appender." + REMOTE_LOGGER + ".ReconnectionDelay",
String.valueOf(DEFAULT_RECONNECTION_DELAY_IN_MILLIS));
        props.setProperty("log4j.appender." + REMOTE_LOGGER + ".LocationInfo", "true");
        return props;
    }

    /**
     * Provide properties for a file logger.
     *
     * @param level The logging leve.
     * @param file The name of the logging file.
     */
    private static Properties fileLoggingProps(Level level, String file) {
        Properties props = new Properties();
        props.setProperty("log4j.appender." + FILE_LOGGER, "org.apache.log4j.FileAppender");
        props.setProperty("log4j.appender." + FILE_LOGGER + ".File", file);
        addAll(getDefaultLoggingSchema(FILE_LOGGER, level), props);
        return props;
    }

    /**
     * Provide properties for writing logs to the console.
     *
     * @param level The logging leve.
     */
    private static Properties consoleLoggingProps(Level level) {
        Properties props = new Properties();
        props.setProperty("log4j.appender." + CONSOLE_LOGGER, "org.apache.log4j.ConsoleAppender");
        addAll(getDefaultLoggingSchema(CONSOLE_LOGGER, level), props);
        return props;
    }

    /**
```

```
     * Return a list of properties of default logging schema details.
     *
     * @param appender The name of the appender.
     * @param level The logging level.
     * @return A list of properties of default logging schema.
     */
    private static Properties getDefaultLoggingSchema(String appender, Level level) {
        Properties props = new Properties();
        props.setProperty("log4j.appender." + appender + ".Threshold", level.toString());
        if (!appender.equals(CONSOLE_LOGGER)) {
            props.setProperty("log4j.appender." + appender + ".Append", "true");
        }
        props.setProperty("log4j.appender." + appender + ".layout", DEFAULT_LAYOUT);
        props.setProperty("log4j.appender." + appender + ".layout.ConversionPattern",
DEFAULT_LAYOUT_PATTERN);
        return props;
    }

    /**
     * Add a property list to another.
     *
     * @param from The property list that need to add.
     * @param to The property list where <pre>from</pre> needs to add to.
     */
    private static void addAll(Properties from, Properties to) {
        for(Object k : from.keySet()) {
            String key = (String)k;
            to.setProperty(key, from.getProperty(key));
        }
    }

    /**
     * Generate a string from a logging level and a list of appenders. The string is used to
     * set the <pre>log4j.rootLogger</pre> property.
     *
     * For example, if the logging level is DEBUG and there are two appenders, say "FILE" and
     * "CONSOLE". The string this method will generate is something like:
     * <pre>DEBUG, FILE, CONSOLE</pre>.
     *
     * @param level The logging level.
     * @param appenders A list of appenders that thig logger will write logs to.
     */
    private static String appendersProperty(String level, String ... appenders) {
        StringBuffer property = new StringBuffer();
        property.append(level);
        for (String appender : appenders) {
            property.append(", " + appender);
        }
        return property.toString();
    }
}
```

The repository package



AbstractRepositoryWriter.java

AbstractRepositoryWriter.java

```
package repository;

import parsedquestion.Constants;

/**
 * Abstract class for writing questions to question repository, which provides some common
 * variables and functionalities.
 *
 */

public abstract class AbstractRepositoryWriter implements IRepositoryWriter {
    /**
     * Whether to compress data in repository.
     */
    protected boolean compressEnabled = true;

    /**
     * Next question ID to be added into the repository.
     */
    protected long nextQuestionUnitId = Constants.INVALID_QUESTION_ID;
    protected long nextTestUnitId = Constants.INVALID_TEST_ID;

    /**
     * Return the next question unit id to add in repository.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    public long getNextQuestionUnitId() throws RepositoryAccessException {
        return nextQuestionUnitId;
    }

    /**
     * Enable data compression.
     */
    public void enableCompress() {
        compressEnabled = true;
    }
```

```
    /**
     * Disable data compression.
     */
    public void disableCompress() {
        compressEnabled = false;
    }

    /**
     * Return the next test unit id to add in repository.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    public long getNextTestUnitId() throws RepositoryAccessException {
        return nextTestUnitId;
    }
}
```

DBRepositoryReader.java

DBRepositoryReader.java

```
package repository;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.TreeSet;

import org.apache.log4j.Logger;
import logging.CustomLogger;

import utils.Range;
import parsedquestion.QuestionUnit;
import parsedquestion.TestUnit;
import parsedquestion.Constants;

import thrift.QuestionSerialization;

/**
 * Question repository reader based on database. Note: Multi-threaded unsafe.
 * Multi-threads need multiple reader objects, one object per thread.
 */

public class DBRepositoryReader implements IRepositoryReader {
    private static final Logger logger = CustomLogger.getLogger("RepositoryReader");

    /**
     * Current question ID.
     */
    private long currentQuestionId = Constants.INVALID_QUESTION_ID;
    private long currentTestId = Constants.INVALID_TEST_ID;

    /**
     * Set the pointer to the first question unit.
     */
    public DBRepositoryReader() throws RepositoryAccessException {
```

```
    resetQuestionUnit();
    resetTestUnit();
  }

  /**
   * Close the reader.
   *
   * @throws RepositoryAccessException if repository access error occurs.
   */
  public void close() {
    // do nothing
  }

  /**
   * Return the question unit at current position and move the position to the next
   * question unit.
   *
   * @return the question unit at current position if there is more question;
   *         otherwise return null.
   * @throws RepositoryAccessException
   *             if repository access error occurs.
   */
  public QuestionUnit nextQuestionUnit() throws RepositoryAccessException {
    QuestionUnit question = readQuestionUnit();
    currentQuestionId++;
    return question;
  }

  /**
   * Reset the pointer right at the first question unit. Call nextQuestionUnit()
   * after this method in order to get the first question.
   * @throws RepositoryAccessException if repository access error occurs.
   */
  public void resetQuestionUnit() throws RepositoryAccessException {
    currentQuestionId = QuestionDataDbAccess.getMinQuestionId();
  }

  /**
   * Seek to the specified question unit.
   *
   * @param questionId
   *            question ID.
   * @throws RepositoryAccessException if questionId is invalid.
   */
  public void seekToQuestionUnit(long questionId) throws RepositoryAccessException {
    if (questionId < 0) {
      throw new RepositoryAccessException("Invalid question ID");
    }
    currentQuestionId = questionId;
  }

  /**
   * @return total number of questions in repository.
   * @throws RepositoryAccessException if repository access error occurs.
   */
  public int numQuestionUnits() throws RepositoryAccessException {
    return QuestionDataDbAccess.getNumQuestions();
  }
```

```
/**
 * @return maximum question ID.
 * @throws RepositoryAccessException if repository access error occurs.
 */
public long getMaxQuestionUnitId() throws RepositoryAccessException {
   return QuestionDataDbAccess.getMaxQuestionId();
}

/**
 * @return question unit at the current position; return null if there is no more question.
 * @throws RepositoryAccessException if repository access error occurs.
 */
private QuestionUnit readQuestionUnit() throws RepositoryAccessException {
   QuestionUnit question = QuestionDataDbAccess.getQuestionUnitByQuestionId(currentQuestionId);
   return question;
}

/**
 * Return the test unit at current position and move the position to the next
 * test unit.
 *
 * @return the test unit at current position if there is more test;
 *         otherwise return null.
 * @throws RepositoryAccessException
 *             if repository access error occurs.
 */
public TestUnit nextTestUnit() throws RepositoryAccessException {
   TestUnit test = readTestUnit();
   currentTestId++;
   return test;
}

/**
 * Reset the pointer to the first test unit. Call nextTestUnit() after this
 * method in order to get the first test.
 * @throws RepositoryAccessException if repository access error occurs.
 */
public void resetTestUnit() throws RepositoryAccessException {
   currentTestId = QuestionDataDbAccess.getMinTestId();
}

/**
 * Seek to the specified test unit.
 *
 * @throws RepositoryAccessException if repository access error occurs.
 */
public void seekToTestUnit(long testId) throws RepositoryAccessException {
   if (testId < 0) {
      throw new RepositoryAccessException("Invalid test ID");
   }
   currentTestId = testId;
}

/**
 * @return total number of tests in repository.
 * @throws RepositoryAccessException if repository access error occurs.
 */
```

```
    public int numTestUnits() throws RepositoryAccessException {
        return QuestionDataDbAccess.getNumTests();
    }

    /**
     * @return maximum test ID in repository.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    public long getMaxTestUnitId() throws RepositoryAccessException {
        return QuestionDataDbAccess.getMaxTestId();
    }

    /**
     * @return test unit at the current position; return null if there is no more test.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    private TestUnit readTestUnit() throws RepositoryAccessException {
        TestUnit test = QuestionDataDbAccess.getTestUnitById(currentTestId);
        return test;
    }
}
```

DBRepositoryWriterSingleton.java

DBRepositoryWriterSingleton.java

```
package repository;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.apache.log4j.Logger;

import parsedquestion.QuestionUnit;
import parsedquestion.TestUnit;
import parsedquestion.QuestionStructUtility;

import logging.CustomLogger;

/**
 * Question repository writer based on database. One writer and single thread at one
 * time only. Multiple-threaded writing is not supported yet.
 *
 * TODO: implement locks to guarantee one instance at one time (even if there
 * are multiple JVMs).
 *
 */
public class DBRepositoryWriterSingleton extends AbstractRepositoryWriter {
    private static final Logger logger = CustomLogger.getLogger("RepositoryWriter");

    private static DBRepositoryWriterSingleton instance = null;

    /**
     * Singleton constructor.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    private DBRepositoryWriterSingleton() throws RepositoryAccessException {
```

```
        nextQuestionUnitId = QuestionDataDbAccess.getMaxQuestionId() + 1;
        nextTestUnitId = QuestionDataDbAccess.getMaxTestId() + 1;
    }

    /**
     * Get a singleton instance of the writer.
     *
     * @return instance of the writer.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    public static synchronized DBRepositoryWriterSingleton getInstance() throws RepositoryAccessException
{
        if (instance == null) {
            instance = new DBRepositoryWriterSingleton();
        }
        return instance;
    }

    /**
     * Reset repository writer.
     */
    public void reset() {
        nextQuestionUnitId = QuestionDataDbAccess.getMaxQuestionId() + 1;
        nextTestUnitId = QuestionDataDbAccess.getMaxTestId() + 1;
    }

    /**
     * Close the writer.
     */
    public void close() {
        /*
         * All attribute/field names have already been flushed to database
         * because writeQuestion() always flushes all attribute/field names when it
         * writes a new question, so no need to do anything here.
         */
    }

    /**
     * Write a question to repository.
     * @param question the given parsed question unit that is going to write.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    public synchronized void writeQuestion(QuestionUnit question) throws RepositoryAccessException {
        validateQuestionID(question);
        if (QuestionDataDbAccess.addQuestion(question, compressEnabled) == 0) {
            throw new RepositoryAccessException("Failed to add a question into database");
        }
        nextQuestionUnitId++;
    }

    /**
     * Remove questions from repository.
     * @param fromId
     *            from which question ID (inclusive).
     * @param toId
     *            to which question ID (inclusive).
     */
    public void removeQuestions(long fromId, long toId) {
```

```
        QuestionDataDbAccess.removeQuestions(fromId, toId);
    }

    /**
     * Write a test to repository.
     * @param test the given parsed test unit that is going to write.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    public synchronized void writeTest(TestUnit test) throws RepositoryAccessException {
        validateTestID(test);
        if (QuestionDataDbAccess.addTest(test, compressEnabled) == 0) {
            throw new RepositoryAccessException("Failed to add a test into database");
        }
        nextTestUnitId++;
    }

    /**
     * Remove tests from repository.
     * @param fromId
     *          from which test ID (inclusive).
     * @param toId
     *          to which test ID (inclusive).
     */
    public void removeTests(long fromId, long toId) {
        QuestionDataDbAccess.removeTests(fromId, toId);
    }

    /**
     * Validate the question ID is set correctly.
     * @param question the question to be written.
     * @throws RepositoryAccessException if question ID is not set or is not set correctly.
     */
    private void validateQuestionID(QuestionUnit question) throws RepositoryAccessException {
        long questionID = QuestionStructUtility.getQuestionID(question);
        if (questionID != nextQuestionUnitId) {
            throw new RepositoryAccessException("Question ID is not set correctly.");
        }
    }

    /**
     * Validate the test ID is set correctly.
     * @param test the test to be written.
     * @throws RepositoryAccessException if test ID is not set or is not set correctly.
     */
    private void validateTestID(TestUnit test) throws RepositoryAccessException {
        long testID = QuestionStructUtility.getTestID(test);
        if (testID != nextTestUnitId) {
            throw new RepositoryAccessException("Test ID is not set correctly.");
        }
    }
}
```

IRepositoryReader.java

IRepositoryReader.java

```
package repository;

import java.util.List;
import java.util.Map;
```

```
import utils.Range;
import parsedquestion.QuestionUnit;
import parsedquestion.TestUnit;

/**
 * Interface for reading questions from question repository.
 *
 */
public interface IRepositoryReader {
    /**
     * Close the reader.
     *
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void close() throws RepositoryAccessException;

    /**
     * Return the question unit at current position and move the position to the next
     * question unit.
     *
     * @return the question unit at current position if there is more question;
     *         otherwise return null.
     * @throws RepositoryAccessException
     *             if repository access error occurs.
     */
    QuestionUnit nextQuestionUnit() throws RepositoryAccessException;

    /**
     * Reset the pointer to the first question unit. Call nextQuestionUnit() after this
     * method in order to get the first question.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void resetQuestionUnit() throws RepositoryAccessException;

    /**
     * Seek to the specified question unit.
     *
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void seekToQuestionUnit(long questionId) throws RepositoryAccessException;

    /**
     * @return total number of questions in repository.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    int numQuestionUnits() throws RepositoryAccessException;

    /**
     * @return maximum question ID in repository.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    long getMaxQuestionUnitId() throws RepositoryAccessException;

    /**
     * Return the test unit at current position and move the position to the next
     * test unit.
     *
     * @return the test unit at current position if there is more test;
```

```
     *         otherwise return null.
     * @throws RepositoryAccessException
     *             if repository access error occurs.
     */
    TestUnit nextTestUnit() throws RepositoryAccessException;

    /**
     * Reset the pointer to the first test unit. Call nextTestUnit() after this
     * method in order to get the first test.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void resetTestUnit() throws RepositoryAccessException;

    /**
     * Seek to the specified test unit.
     *
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void seekToTestUnit(long testId) throws RepositoryAccessException;

    /**
     * @return total number of tests in repository.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    int numTestUnits() throws RepositoryAccessException;

    /**
     * @return maximum test ID in repository.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    long getMaxTestUnitId() throws RepositoryAccessException;
}
```

IRepositoryWriter.java

IRepositoryWriter.java

```
package repository;

import parsedquestion.QuestionUnit;
import parsedquestion.TestUnit;

/**
 * Interface for writing questions to question repository.
 */
public interface IRepositoryWriter {
    /**
     * Return the next question unit id to add in repository.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    long getNextQuestionUnitId() throws RepositoryAccessException;

    /**
     * Enable data compression.
     */
    void enableCompress();

    /**
     * Disable data compression.
     */
    void disableCompress();
```

```
    /**
     * Close the writer.
     *
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void close() throws RepositoryAccessException;

    /**
     * Write a new whole question to repository.
     *
     * @param question
     *            question with full content.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void writeQuestion(QuestionUnit question) throws RepositoryAccessException;

    /**
     * Remove questions from repository.
     * @param fromId from which question ID (inclusive).
     * @param toId to which question ID (inclusive).
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void removeQuestions(long fromId, long toId) throws RepositoryAccessException;

    /**
     * Return the next test unit id to add in repository.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    long getNextTestUnitId() throws RepositoryAccessException;

    /**
     * Write a new test suite to repository.
     *
     * @param test
     *            test with full content.
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void writeTest(TestUnit test) throws RepositoryAccessException;

    /**
     * Remove tests from repository.
     * @param fromId from which test ID (inclusive).
     * @param toId to which test ID (inclusive).
     * @throws RepositoryAccessException if repository access error occurs.
     */
    void removeTests(long fromId, long toId) throws RepositoryAccessException;
}
```

QuestionDataAccess.java

QuestionDataDbAccess.java

```
package repository;

import java.sql.Blob;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
```

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.LinkedList;
import java.util.Map;
import java.util.Set;
import java.util.SortedMap;
import java.util.TreeMap;

import java.io.IOException;
import java.io.ByteArrayOutputStream;

import org.apache.thrift.TBase;

import utils.Range;
import utils.CATDbAccess;
import utils.CATDatabaseManager;
import utils.Database;
import utils.DatabaseException;
import utils.ByteArrayWrapper;
import utils.Compressor;

import thrift.QuestionSerialization;
import thrift.StreamThriftSerializer;

import parsedquestion.QuestionUnit;
import parsedquestion.TestUnit;
import parsedquestion.QuestionStructUtility;

/**
 * Access interface to question data database.
 * <p>
 * Description of database schema (refer to cat_dbschema.sql for detailed
 * schema definition):
 * <p>
 * A question contains question_id, test_id, question data (which is
 * stored in database as a blob in binary format, and compressed indicating
 * whether question data is compressed. Each question is a row in database
 * `question_data`.
 * <p>
 * Additionally, table `test_data` records the mapping between test suite
 * and question. It also records several attributes of test suite (`name`,
 * `subject`, `topic` and `targetlength`).
 */

// TODO: data insertion is not in transaction yet
public class QuestionDataDbAccess extends CATDbAccess {

    /**
     * Get the maximum question ID.
     *
     * @return maximum question ID.
     */
    public static long getMaxQuestionId() {
        String sql = "SELECT MAX(question_id) AS maxid FROM question_data";
        ResultSetProcessor<Long> rsp = new ResultSetProcessor<Long>() {
            long maxid = 0;
            public void process(final ResultSet rs) throws SQLException {
```

```
            if (rs.next()) {
                maxid = rs.getLong("maxid");
            }
        }
        public Long result() {
            return maxid;
        }
    };
    executeQuery(sql, rsp);
    return rsp.result();
}

/**
 * Get the minimum question ID.
 *
 * @return minimum question ID.
 */
public static long getMinQuestionId() {
    String sql = "SELECT MIN(question_id) AS minid FROM question_data";
    ResultSetProcessor<Long> rsp = new ResultSetProcessor<Long>() {
        long minid = 0;
        public void process(final ResultSet rs) throws SQLException {
            if (rs.next()) {
                minid = rs.getLong("minid");
            }
        }
        public Long result() {
            return minid;
        }
    };
    executeQuery(sql, rsp);
    return rsp.result();
}

/**
 * Get the total number of questions.
 *
 * @return number of questions.
 */
public static int getNumQuestions() {
    String sql = "SELECT COUNT(DISTINCT question_id) AS num FROM question_data";
    ResultSetProcessor<Integer> rsp = new ResultSetProcessor<Integer>() {
        int num = 0;
        public void process(final ResultSet rs) throws SQLException {
            if (rs.next()) {
                num = rs.getInt("num");
            }
        }
        public Integer result() {
            return num;
        }
    };
    executeQuery(sql, rsp);
    return rsp.result();
}

/**
 * Add a question to the database.
```

```
     * @return the number of rows inserted. 1 if insertion succeeds; 0 if fails.
     */
    public static int addQuestion(final QuestionUnit question, boolean compressed) {
        long testID = QuestionStructUtility.getTestID(question);
        long questionID = QuestionStructUtility.getQuestionID(question);
        byte[] questionData;
        try {
            questionData = QuestionSerialization.serializeThriftObject(question);
        } catch (IOException e) {
            logger.error("Error while serializing question: ", e);
            return 0;
        }
        if (compressed) {
            questionData = Compressor.compress(questionData);
            if (questionData == null) {
                logger.error("Failed to compress question data");
                return 0;
            }
        }
        Database db = null;
        int updated = 0;
        try {
            db = CATDatabaseManager.getFreeDatabase();
            Connection conn = db.getConnection();
            // insert to question_data
            PreparedStatement stmt = conn
                    .prepareStatement(
                    "INSERT into question_data(question_id,test_id,compressed,data) values (?, ?, ?, ?)");
            stmt.setLong(1, questionID);
            stmt.setLong(2, testID);
            stmt.setBoolean(3, compressed);
            stmt.setBytes(4, questionData);
            updated = stmt.executeUpdate();
            stmt.close();
        } catch (SQLException e) {
            logger.error("SQLException while writing question data: ", e);
        } catch (DatabaseException e) {
            logger.error("DatabaseException while writing question data: ", e);
        } finally {
            if (db != null) {
                CATDatabaseManager.releaseBusyDatabase(db);
            }
        }
        return updated;
    }

    /**
     * Remove question data from database given question ID range.
     *
     * @param fromQuestionId
     *            from which question ID (inclusive).
     * @param toQuestionId
     *            to which question ID (inclusive).
     */
    public static void removeQuestions(long fromQuestionId, long toQuestionId) {
        // remove question data from question_data table
        String sql = String.format("DELETE FROM question_data WHERE question_id>=%d AND
question_id<=%d",
```

```
                fromQuestionId, toQuestionId);
    executeUpdate(sql);
  }

 /**
  * Get question data by question ID.
  *
  * @param questionId
  *           ID of the specific question.
  * @return question data (uncompressed); return null if there is no such
  *       a question in database.
  */
 public static QuestionUnit getQuestionUnitByQuestionId(final long questionId) {
    if (questionId < 0) {
       return null;
    }
    String sql = "SELECT compressed,data FROM question_data WHERE question_id=" + questionId;
    ResultSetProcessor<ByteArrayWrapper> rsp = new ResultSetProcessor<ByteArrayWrapper>() {
       ByteArrayWrapper questiondata = null;
       public void process(final ResultSet rs) throws SQLException {
          if (rs.next()) {
             boolean compressed = rs.getBoolean("compressed");
             Blob blob = rs.getBlob("data");
             byte[] blobdata = blob.getBytes(1, (int) blob.length());  // position is 1-based
             if (compressed) {
                questiondata = new ByteArrayWrapper(Compressor.decompress(blobdata));
             } else {
                questiondata = new ByteArrayWrapper(blobdata);
             }
          }
       }
       public ByteArrayWrapper result() {
          return questiondata;
       }
    };
    executeQuery(sql, rsp);
    ByteArrayWrapper bytes = rsp.result();
    if (bytes == null) {
       return null;
    }

    try {
       QuestionUnit question = QuestionSerialization.deserializeQuestionUnit(bytes.data);
       return question;
    } catch (IOException e) {
       logger.error("IO error occured when reading question: " + e.getMessage(), e.getCause());
       return null;
    }
 }

 /**
  * Get question data by test ID.
  *
  * @param testId
  *           ID of the specific test suite.
  * @return question data (uncompressed); return null if there is no such
  *       a question in database.
  */
```

```
public static List<QuestionUnit> getQuestionUnitByTestId(final long testId) {
    if (testId < 0) {
        return null;
    }
    String sql = "SELECT compressed,data FROM question_data WHERE test_id=" + testId;
    ResultSetProcessor<List<QuestionUnit>> rsp = new ResultSetProcessor<List<QuestionUnit>>() {
        List<QuestionUnit> questions = new LinkedList<QuestionUnit>();
        public void process(final ResultSet rs) throws SQLException {
            while (rs.next()) {
                boolean compressed = rs.getBoolean("compressed");
                Blob blob = rs.getBlob("data");
                byte[] blobdata = blob.getBytes(1, (int) blob.length());  // position is 1-based
                ByteArrayWrapper questiondata = null;
                if (compressed) {
                    questiondata = new ByteArrayWrapper(Compressor.decompress(blobdata));
                } else {
                    questiondata = new ByteArrayWrapper(blobdata);
                }
                QuestionUnit question = null;
                try {
                    question = QuestionSerialization.deserializeQuestionUnit(questiondata.data);
                    questions.add(question);
                } catch (IOException e) {
                    logger.error("IO error occured when reading question: " + e.getMessage(), e.getCause());
                }
            }
        }
        public List<QuestionUnit> result() {
            return questions;
        }
    };
    executeQuery(sql, rsp);
    List<QuestionUnit> ret = rsp.result();
    if (ret.size() == 0) {
        return null;
    }
    return ret;
}

/**
 * Get the maximum test ID.
 *
 * @return maximum test ID.
 */
public static long getMaxTestId() {
    String sql = "SELECT MAX(test_id) AS maxid FROM test_data";
    ResultSetProcessor<Long> rsp = new ResultSetProcessor<Long>() {
        long maxid = 0;
        public void process(final ResultSet rs) throws SQLException {
            if (rs.next()) {
                maxid = rs.getLong("maxid");
            }
        }
        public Long result() {
            return maxid;
        }
    };
    executeQuery(sql, rsp);
```

```
        return rsp.result();
    }

    /**
     * Get the minimum test ID.
     *
     * @return minimum test ID.
     */
    public static long getMinTestId() {
        String sql = "SELECT MIN(test_id) AS minid FROM test_data";
        ResultSetProcessor<Long> rsp = new ResultSetProcessor<Long>() {
            long minid = 0;
            public void process(final ResultSet rs) throws SQLException {
                if (rs.next()) {
                    minid = rs.getLong("minid");
                }
            }
            public Long result() {
                return minid;
            }
        };
        executeQuery(sql, rsp);
        return rsp.result();
    }

    /**
     * Get the total number of tests.
     *
     * @return number of tests.
     */
    public static int getNumTests() {
        String sql = "SELECT COUNT(DISTINCT test_id) AS num FROM test_data";
        ResultSetProcessor<Integer> rsp = new ResultSetProcessor<Integer>() {
            int num = 0;
            public void process(final ResultSet rs) throws SQLException {
                if (rs.next()) {
                    num = rs.getInt("num");
                }
            }
            public Integer result() {
                return num;
            }
        };
        executeQuery(sql, rsp);
        return rsp.result();
    }

    /**
     * Add a test suite to the database.
     */
    public static int addTest(final TestUnit test, boolean compressed) {
        long testId = QuestionStructUtility.getTestID(test);
        String name = test.getName();
        String subject = QuestionStructUtility.getTestSubject(test);
        String topic = QuestionStructUtility.getTestTopic(test);
        int targetlength = test.getTargetlength();
        Database db = null;
        int updated = 0;
```

```
        try {
            db = CATDatabaseManager.getFreeDatabase();
            Connection conn = db.getConnection();
            // insert to test_data
            PreparedStatement stmt = conn.prepareStatement(
                    "INSERT into test_data(test_id,name,subject,topic,targetlength) values (?, ?, ?, ?, ?)");
            stmt.setLong(1, testId);
            stmt.setString(2, name);
            stmt.setString(3, subject);
            stmt.setString(4, topic);
            stmt.setInt(5, targetlength);
            updated = stmt.executeUpdate();
            stmt.close();
        } catch (SQLException e) {
            logger.error("SQLException while writing test data: ", e);
        } catch (DatabaseException e) {
            logger.error("DatabaseException while writing test data: ", e);
        } finally {
            if (db != null) {
                CATDatabaseManager.releaseBusyDatabase(db);
            }
        }
        if (updated == 0) {
            return 0;
        }
        for (QuestionUnit question : test.getQuestions()) {
            updated = addQuestion(question, compressed);
            if (updated == 0) {
                return 0;
            }
        }
        return updated;
    }

    /**
     * Remove test data from database given test ID range.
     *
     * @param fromTestId
     *            from which test ID (inclusive).
     * @param toTestId
     *            to which test ID (inclusive).
     */
    public static void removeTests(long fromTestId, long toTestId) {
        // remove test data from test_data table
        String sql = String.format("DELETE FROM test_data WHERE test_id>=%d AND test_id<=%d",
                fromTestId, toTestId);
        executeUpdate(sql);
    }

    /**
     * Get test data by test ID.
     *
     * @param testId
     *            ID of the specific test.
     * @return test unit; return null if there is no such
     *       a test in database.
     */
    public static TestUnit getTestUnitById(final long testId) {
```

```
        if (testId < 0) {
            return null;
        }
        String sql = "SELECT test_id,name,subject,topic,targetlength FROM test_data WHERE test_id=" +
testId;
        ResultSetProcessor<TestUnit> rsp = new ResultSetProcessor<TestUnit>() {
            TestUnit test = null;
            public void process(final ResultSet rs) throws SQLException {
                if (rs.next()) {
                    test = new TestUnit();
                    Map<String, String> attributes = new HashMap<String, String> ();
                    test.setAttributes(attributes);
                    test.setName(rs.getString("name"));
                    QuestionStructUtility.setTestSubject(test, rs.getString("subject"));
                    QuestionStructUtility.setTestTopic(test, rs.getString("topic"));
                    QuestionStructUtility.setTestID(test, rs.getLong("test_id"));
                    test.setTargetlength(rs.getInt("targetlength"));
                }
            }
            public TestUnit result() {
                return test;
            }
        };

        executeQuery(sql, rsp);
        TestUnit retTest = rsp.result();
        if (retTest != null) {
            List<QuestionUnit> questions = getQuestionUnitByTestId(testId);
            if (questions != null) {
                retTest.setQuestions(questions);
            }
        }
        return retTest;
    }
}
```

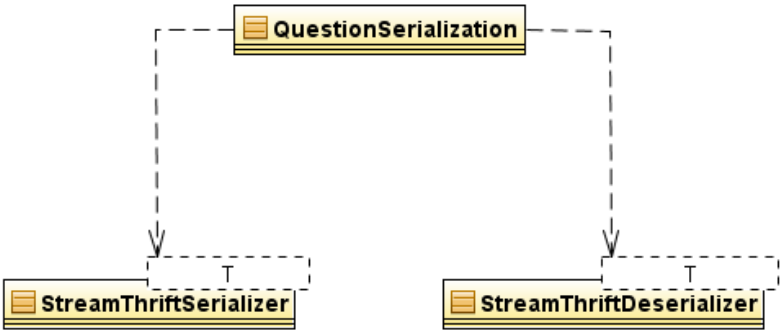
RepositoryAccessException.java

RepositoryAccessException.java

```
package repository;

public class RepositoryAccessException extends Exception {
    private static final long serialVersionUID = 1L;

    /**
     * Constructs a new repository access exception with the specified detail
     * message.
     *
     * @param message
     *            the detail message.
     */
    public RepositoryAccessException(String message) {
        super(message);
    }

    /**
     * Constructs a new repository access exception with the specified detail
     * message and cause.
     *
     * @param message
```

```
     *            the detail message.
     * @param cause
     *            the cause. A null value is permitted, and indicates that the
     *            cause is nonexistent or unknown.
     */
    public RepositoryAccessException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

The thrift package



QuestionSerialization.java

QuestionSerialization.java

```
package thrift;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Arrays;

import org.apache.thrift.TBase;

import parsedquestion.QuestionUnit;
import parsedquestion.TestUnit;

/**
 * Utility class to serialize/deserialize question data.
 */
public final class QuestionSerialization {
    /** Utility class. */
    private QuestionSerialization() {
    }
```

```
  /**
   * Serialize thrift object to a byte array.
   * @param t thrift object.
   * @return serialized byte array that represents the given thrift object.
   * @throws IOException if deserialization fails.
   */
  public static byte[] serializeThriftObject(TBase t) throws IOException {
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    StreamThriftSerializer<TBase> serializer = new StreamThriftSerializer<TBase>(baos);
    serializer.serialize(t);
    byte[] bytes = baos.toByteArray();
    baos.close();
    return bytes;
  }

  /**
   * Deserialize question from a byte array.
   *
   * @param bytes input byte array.
   * @return question unit
   * @throws IOException if IO error occurs or deserialization fails.
   */
  public static QuestionUnit deserializeQuestionUnit(byte[] bytes) throws IOException {
    ByteArrayInputStream bais = new ByteArrayInputStream(bytes);
    StreamThriftDeserializer.TBaseConstructor<QuestionUnit> constructor =
      new StreamThriftDeserializer.TBaseConstructor<QuestionUnit>() {
      public QuestionUnit construct() {
        return new QuestionUnit();
      }
    };
    StreamThriftDeserializer<QuestionUnit> deserializer =
      new StreamThriftDeserializer<QuestionUnit>(bais, constructor);
    QuestionUnit question = deserializer.next();
    if (question == null) {
      throw new IOException("Error while deserializing question unit");
    }
    return question;
  }
}
```

StreamThriftDeserializer.java

StreamThriftDeserializer.java

```
package thrift;

import java.io.IOException;
import java.io.InputStream;

import org.apache.thrift.TBase;
import org.apache.thrift.TException;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.transport.TIOStreamTransport;

/**
 * <p>
 * An abstract class for deserializing thrift objects from streams
 * </p>
 * <p>
```

```
 * It allows to easily deserialize series of objects of the same type (T) from a
 * stream by providing an Iterator inteface.
 * </p>
 *
 * @param <T> The type of the thrift generated object that we want to serialize
 */
public class StreamThriftDeserializer<T extends TBase> {

    /**
     * Initialize thrift object so that they can be populated by reading from
     * the input stream
     */
    protected final TBaseConstructor<T> tConstructor;

    protected InputStream in;

    protected TBinaryProtocol thriftProtocol;

    /**
     * Construct the deserializer
     *
     * @param in The stream to read from and deserialize objects. <b>NOTE:</b>
     *        The performance of the deserializer depends on the performance of
     *        <code>read()</code> of <b>in</b>, make sure that an efficient
     *        implementation is passed (e.g. a BufferedInputStream).
     * @param tConstructor The implementation of {@link TBaseConstructor} that
     *        can construct empty <b>T</b> classes. This is necessary because
     *        the <b>AbstractThriftSerializer</b> does not know the type of
     *        <b>T</b> statically
     */
    public StreamThriftDeserializer(InputStream in, TBaseConstructor<T> tConstructor) {
        this.in = in;
        this.tConstructor = tConstructor;
        this.thriftProtocol = new TBinaryProtocol(new TIOStreamTransport(in));
    }

    /**
     * Check to see if there are more objects left to be deserialized. It does
     * so by trying to read one more byte and see if it is available
     *
     * @return true if there are more bytes to be read (therefore more objects
     *         to be deserialized), false otherwise
     * @throws IOException If reading from the input stream fails
     */
    public boolean hasMoreBytes() throws IOException {
        in.mark(1);
        int nextByte = in.read();
        in.reset();
        return nextByte != -1;
    }

    /**
     * Deserialize the next object from the input stream
     *
     * @return An instance of T or null if there are no more objects to read
     */
    public T next() throws IOException {
        T t = tConstructor.construct();
```

```
        try {
            t.read(thriftProtocol);
        } catch (TException e) {
            return null;
        }
        return t;
    }

    /**
     * Close the input stream
     *
     * @throws IOException If <code>in.close()</code> throws an IOException
     */
    public void close() throws IOException {
        in.close();
    }

    /**
     * An interface that defines only one method for constructing objects of
     * type <b>T</b> It should be specialized for the different types of
     * <b>T</T> so that empty instances of <b>T</b> can be constructed
     *
     * @param <T> The type of the thrift generated object that we want to
     *        construct
     */
    public static interface TBaseConstructor<T extends TBase> {
        T construct();
    }
}
```

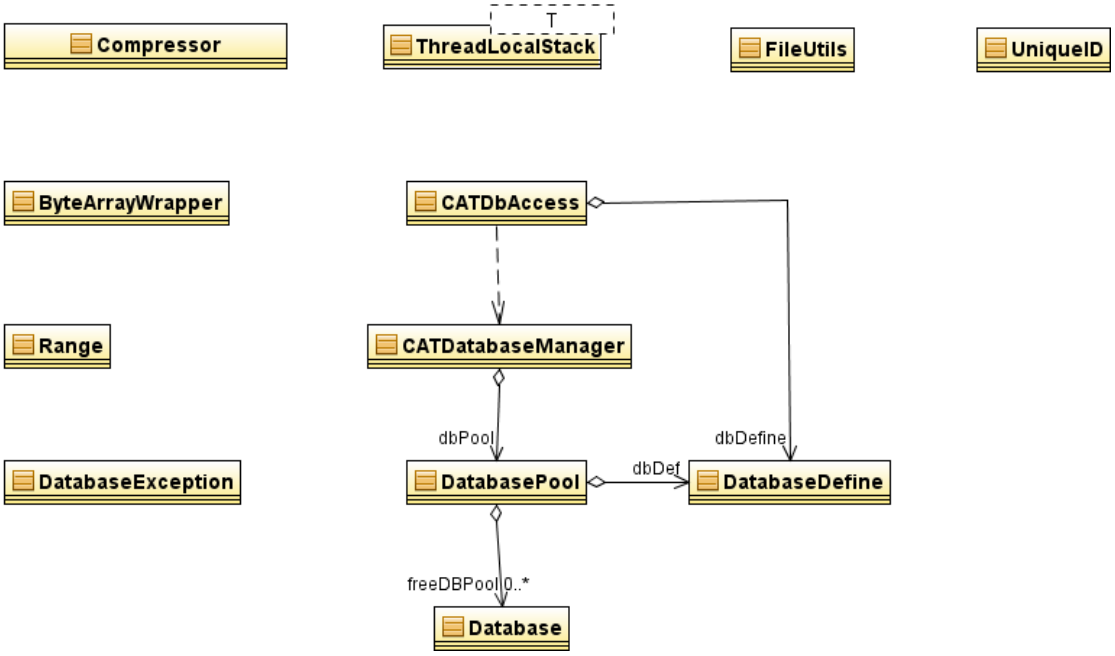
StreamThriftSerializer.java

StreamThriftSerializer.java

```
package thrift;

import java.io.IOException;
import java.io.OutputStream;

import org.apache.thrift.TBase;
import org.apache.thrift.TException;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.transport.TIOStreamTransport;

/**
 * An abstract class for serializing thrift objects to streams
 *
 * @param <T> The type of the thrift generated object that we want to serialize
 */
public class StreamThriftSerializer<T extends TBase> {

    protected OutputStream out;
    protected TBinaryProtocol thriftProtocol;

    /**
     * Construct the serializer
     *
     * @param out The OutputStream where the given objects will be serialized to
     */
    public StreamThriftSerializer(OutputStream out) {
```

```
        this.out = out;
        thriftProtocol = new TBinaryProtocol(new TIOStreamTransport(out));
    }

    /**
     * Serialize an object and write it to the output stream
     */
    public void serialize(T t) throws IOException {
        try {
            t.write(thriftProtocol);
            out.flush();
        } catch (TException e) {
            throw new IOException(e);
        }
    }

    /**
     * Close the output stream
     *
     * @throws IOException If <code>out.close()</code> throws an IOException
     */
    public void close() throws IOException {
        out.close();
    }
}
```

The utils package



ByteArrayWrapper.java

ByteArrayWrapper.java

```
package utils;

import java.io.Serializable;

/**
 * A wrapper class for holding a byte array.
```

```
 */
public final class ByteArrayWrapper implements Serializable {
    private static final long serialVersionUID = 1L;
    public byte[] data;

    /**
     * Construct with byte array.
     */
    public ByteArrayWrapper(byte[] data) {
        this.data = data;
    }
}
```

CATDataManager.java

CATDatabaseManager.java

```
package utils;

public class CATDatabaseManager {
            public static DatabasePool dbPool = null;

    /** load pool and initialize connection
     *  use this function only if there is just one database to be connected
     */
    public static void init() throws DatabaseException {
        if (dbPool != null)
            throw new DatabaseException(DatabaseException.ERR_DB_OPEN, "can't use this function after init
has been called");

        DatabasePool dbPool = new DatabasePool();
        try {
            dbPool.initDatabasePool();
        } catch (Exception e) {
            dbPool = null;
            throw new DatabaseException(DatabaseException.ERR_DB_OPEN, e.getMessage());
        }
    }

    /**
     * initialize one connection pool for a particular database
     * @param dbDef
     * @throws DatabaseException
     */
    public static void init(DatabaseDefine dbDef) throws DatabaseException {
        if (dbPool != null) {
            throw new DatabaseException(DatabaseException.ERR_DB_OPEN, "Database has been
initialized");
        }
        dbPool = new DatabasePool(dbDef);
        try {
            dbPool.initDatabasePool();
        } catch (Exception e) {
            dbPool = null;
            throw new DatabaseException(DatabaseException.ERR_DB_OPEN, e.getMessage());
        }
    }

    /**
     * get a free connection from pool
     * @return database connection instance
```

```
     * @throws DatabaseException
     */
    public static Database getFreeDatabase()
        throws DatabaseException
    {
        if (dbPool != null) {
            return dbPool.getFreeDatabase();
        }
        else  {
            throw new DatabaseException(DatabaseException.ERR_DB_NOTFOUND, "please ensure that
database is connected.");
        }
    }

    /**
     * release an unused connection into pool
     * @param db database connection instance
     */
    public static void releaseBusyDatabase(Database db)
    {
        if (db== null) {
            return;
        }
        if( dbPool != null ) {
            dbPool.releaseBusyDatabase(db);
        }
    }

}
```

CATDbAccess.java

CATDbAccess.java

```
package utils;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.apache.log4j.Logger;
import org.jconfig.Configuration;
import org.jconfig.ConfigurationManager;
import org.jconfig.handler.XMLFileHandler;

import com.mysql.jdbc.exceptions.jdbc4.CommunicationsException;
import environment.IInstallationNames;
import environment.FileManager;

/**
 * Interface to access cat database.
 */
public class CATDbAccess {
    protected static final Logger logger = Logger.getLogger("CATDBAccess");
```

```
    protected static final String DEFAULT_DBHOST = "localhost";
    protected static final String DEFAULT_DBPORT = "3306";
    protected static final String DEFAULT_DATABASE = "logdb";
    protected static final String DEFAULT_DBDRIVER = "com.mysql.jdbc.Driver";
    protected static final String DEFAULT_DBUSER = "dbuser";
    protected static final String DEFAULT_DBPASSWD = "dbuser";

    protected static DatabaseDefine dbDefine = null;
    protected static boolean dbInited = false;

    // obsolete!
    // this connection and statement are for static usage only, not threaded safe.  We will remove them soon.
    private static Connection staticConnection = null;
    private static Statement staticStmt = null;
    private static Statement staticStreamStmt = null;

    static {
            init();
    }


    protected static DatabaseDefine readConfiguration() {
       DatabaseDefine dbDef = new DatabaseDefine();
       Configuration configuration = null;

           try {
              String configFile =
                 FileManager.getLocationInInstallation(IInstallationNames.DATABASE_CONFIGURATION);
                        XMLFileHandler fileHandler = new XMLFileHandler(configFile);
          configuration = fileHandler.load("loginsight");
          logger.info("Read configuration from " + configFile);
       } catch (Exception e) {
          logger.error(e);
          logger.info("Read configuration from default file instead");
          configuration = ConfigurationManager.getConfiguration();
       }

       if (configuration.isNew()) {
            logger.warn("new configuration");
       } else {
            String dbhost = configuration.getProperty("HOST", DEFAULT_DBHOST, "JDBC");
            String dbport = configuration.getProperty("PORT", DEFAULT_DBPORT, "JDBC");
              String dbname = configuration.getProperty("DATABASE", DEFAULT_DATABASE, "JDBC");
            dbDef.setName(dbname);
            dbDef.setUrl("jdbc:mysql://" + dbhost + ":" + dbport + "/" + dbname + "?autoReconnect=true");
                        dbDef.setDriver(configuration.getProperty("DRIVER", DEFAULT_DBDRIVER,
"JDBC"));
                        dbDef.setUser(configuration.getProperty("USER", DEFAULT_DBUSER, "JDBC"));
                        dbDef.setPassword(configuration.getProperty("PWD", DEFAULT_DBPASSWD,
"JDBC"));
       }
       dbDef.setInitNum(1);
       dbDef.setMaxNum(10);

       return dbDef;
    }
```

```
    /**
     * Obsolete!
     * Check weather the database connection has been established.
     *
     * @return whether the database connection has been established
     */
    private static boolean isConnected() {
        try {
            return (staticConnection != null && !staticConnection.isClosed());
        } catch (SQLException e) {
            logger.warn("Database access error while testing connection");
            return false;
        }
    }

    /**
     * Obsolete!
     * connect to database.
     * @return true if connection succeeds; false otherwise.
     */
    private static boolean connect() throws DatabaseException {
        if (!dbInited) {
            init();
        }

        // Get a connection from database pool
        try {
             if (staticStmt != null) {
             staticStmt.close();
                                                staticStmt = null;
             }
             if (staticStreamStmt != null) {
             staticStreamStmt.close();
                                                staticStreamStmt = null;
             }
             Database staticDb = CATDatabaseManager.getFreeDatabase();
            staticConnection = staticDb.getConnection();
            if (!isConnected()) {
                logger.error("Failed to get a static connection to database.");
                return false;
            }
             staticStmt = staticConnection.createStatement();
             staticStreamStmt = staticConnection.createStatement(ResultSet.TYPE_FORWARD_ONLY,
ResultSet.CONCUR_READ_ONLY);
             staticStreamStmt.setFetchSize(Integer.MIN_VALUE);
            return true;
        } catch (Exception e) {
            logger.error("Unknown error occured when connect to database: ", e);
            return false;
        }
    }

    public static void init() {
             if (dbInited) {
            logger.warn("Cannot initialize database connection more than once");
            return;
              }
             dbDefine = readConfiguration();
```

```
        try{
            CATDatabaseManager.init(dbDefine);
            dbInited = true;
        } catch(Exception e){
            logger.fatal("Cannot initialize database connection");
        }
    }

    protected interface ResultSetProcessor<T> {
        public void process(ResultSet rs) throws SQLException;
        public T result();
    }

    /**
     * Run a SQL query on the log database and process the result set
     *
     * @param sql SQL query
     * @param rsProcessor the processor to handle the result set
     */
    public static void executeQuery(String sql, ResultSetProcessor<?> rsProcessor) {
        Database db = null;
        ResultSet rs = null;
        try {
            if (!dbInited) {
                init();
            }
            try {
                db = CATDatabaseManager.getFreeDatabase();
            } catch (DatabaseException e) {
             logger.error("Fail to get free database for query " + sql);
                return;
            }
            logger.debug("Executing Query: "+ sql);
            long start = System.currentTimeMillis();
            try {
             rs = db.executeQuery(sql);
            } catch (DatabaseException de) {
             logger.warn("" + de);
             logger.warn("retry to connect db again");
             db.reConnect();
             rs = db.executeQuery(sql);
            }
            logger.debug("Time: " + (System.currentTimeMillis() - start) + " milliseconds");
            try {
                rsProcessor.process(rs);
            } catch (SQLException e) {
                logger.error("SQLException while executing: " + sql);
                logger.info(e.getMessage());
            }
        } catch (DatabaseException e) {
            logger.fatal("DatabaseException", e);
        } finally {
            if (rs != null) {
                try {
                    rs.close();
                } catch (SQLException e) {
                    logger.error("SQLException while closing result set : " + rs.toString() +
                            " for " + sql);
```

```
                logger.info(e.getMessage());
              }
          }
          CATDatabaseManager.releaseBusyDatabase(db);
      }
  }

  /**
   * Run a SQL update query on the log database
   *
   * @param query update query
   * @return results number of rows that were affected by the update query
   */
  public static int executeUpdate(String sql) {
      Database db = null;
      int updated = 0;
      try {
          if (!dbInited) {
              init();
          }
          try{
              db = CATDatabaseManager.getFreeDatabase();
          } catch (DatabaseException e) {
           logger.error("Fail to get free database for query " + sql);
              return 0;
          }
          logger.debug("Executing Query: "+ sql);
          long start = System.currentTimeMillis();
          try {
           updated = db.executeUpdate(sql);
          } catch (DatabaseException de) {
           logger.error("DatabaseException, retry to connect:" + de);
           db.connect(dbDefine.getDriver(), dbDefine.getUrl(), dbDefine.getUser(), dbDefine.getPassword());
           updated = db.executeUpdate(sql);
          }
          logger.debug("Time: " + (System.currentTimeMillis() - start) + " milliseconds");
      } catch (DatabaseException e) {
          logger.fatal("DatabaseException", e);
      } finally {
           CATDatabaseManager.releaseBusyDatabase(db);
      }
      return updated;
  }


  /**
   * Obsolete!
   * Run a SQL query on the log database and return result set in non-streaming mode.
   *
   * @param query SQL query
   * @return results of the query.
   */
  public static ResultSet runSQLQuery(String query) {
          ResultSet rs = null;
      try {
          // if connection is not establish yet, connect it now
          if (!isConnected()) {
                          if (!connect()) {
```

```
                                        return null;
                                }
                        }
                        logger.debug("Executing query in obsolete mode: "+ query);
                                rs = staticStmt.executeQuery(query);
                        } catch (CommunicationsException ce) {
                                logger.error("SQL CommunicationsException, retry to connect");
                                try {
                                        if (!connect()) {
                                        return null;
                                }
                                rs = staticStmt.executeQuery(query);
                        } catch( Exception e ) {
                                logger.error("Unknown error occured when run query: ", e);
                        }
                        } catch( Exception e ) {
                                logger.error("Unknown error occured when run query: ", e);
                        }
                        return rs;
                }

                /**
                 * Obsolete!
                 * Run a SQL query on the log database and return result set in streaming mode.
                 *
                 * @param query SQL query
                 * @return results of the query. It is forward only and read only result set. The caller must consume all the
rows in the result set and close it.
                 */
                public static ResultSet runSQLQueryStreaming(String query) {
                        ResultSet rs = null;
                        try {
                                // if connection is not establish yet, connect it now
                                if (!isConnected()) {
                                        if (!connect()) {
                                                return null;
                                        }
                                }
                                logger.debug("Executing query in obsolete streaming mode: "+ query);
                                        rs = staticStreamStmt.executeQuery(query);
                        } catch (CommunicationsException ce) {
                                logger.error("SQL CommunicationsException, retry to connect");
                                try {
                                        if (!connect()) {
                                        return null;
                                }
                                        // logger.info("SQL query (streaming): " + query);
                                        rs = staticStreamStmt.executeQuery(query);
                                } catch( Exception e ) {
                                        logger.error("Unknown error occured when run query: ", e);
                                }
                        } catch( Exception e ) {
                                logger.error("Unknown error occured when run query: ", e);
                        }
                        return rs;
                }
        }
```

Compressor.java

Compressor.java

```
package utils;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.zip.DataFormatException;
import java.util.zip.Deflater;
import java.util.zip.Inflater;

import org.apache.log4j.Logger;

import logging.CustomLogger;

/**
 * Data compressor and decompressor.
 *
 * (1) Compress data: given a plain text, call compress() to get compressed
 * data using ZLIB compression library.
 *
 * (2) Decompress data: given compressed data, call decompress() to get the
 * original plain text.
 *
 */

public final class Compressor {
    private static final Logger logger = CustomLogger.getLogger("Compressor");

    /**
     * It is utility class, so no need constructor.
     */
    private Compressor() {
        // do nothing
    }

    /**
     * Deflater buffer size (bytes)
     */
    private static final int DEFLATER_BUFFER_SIZE = 1024;

    /**
     * Inflater buffer size (bytes)
     */
    private static final int INFLATER_BUFFER_SIZE = 1024;

    /**
     * Compress plain text using ZLIB compression library.
     *
     * @param text
     *            input data in plain text.
     * @return compressed data; null if failed.
     */
    public static byte[] compress(final String text) {
        return compress(text.getBytes());
    }

    /**
     * Compress data using ZLIB compression library.
     *
```

```
     * @param bytes
     *            input data.
     * @return compressed data; null if failed.
     */
    public static byte[] compress(final byte[] bytes) {
        try {
            Deflater compressor = new Deflater();

            // set input data for the compressor
            compressor.setInput(bytes);
            compressor.finish();    // indicate input ends

            ByteArrayOutputStream baos = new ByteArrayOutputStream(bytes.length);

            // Compress the data
            byte[] buf = new byte[DEFLATER_BUFFER_SIZE];
            while (!compressor.finished()) {
                int count = compressor.deflate(buf);
                baos.write(buf, 0, count);
            }
            try {
                baos.close();
            } catch (IOException e) {
                logger.error("Error occured when compressing data: " + e);
            }

            return baos.toByteArray();
        } catch (Exception e) {
            logger.error("Error occured when compressing data: ", e);
        }
        return null;
    }

    /**
     * Decompress data.
     *
     * @param data
     *            compressed data.
     * @return original uncompressed data; return null if the data
     *         are in wrong format.
     */
    public static byte[] decompress(final byte[] data) {
        Inflater decompressor = new Inflater();
        decompressor.setInput(data);

        ByteArrayOutputStream baos = new ByteArrayOutputStream(data.length);

        byte[] buf = new byte[INFLATER_BUFFER_SIZE];
        while (!decompressor.finished()) {
            try {
                int count = decompressor.inflate(buf);
                baos.write(buf, 0, count);
            } catch (DataFormatException e) {
                logger.error("Data format error while uncompressing: " + e);
                return null;
            }
        }
        byte[] bytes = baos.toByteArray();
```

```
        try {
            baos.close(); // no effect
        } catch (IOException e) {
            // no such an exception under any situation
        }
        return bytes;
    }

}
```

Database.java

Database.java

```
package utils;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Types;

import org.apache.log4j.Logger;

/**
 * maintain database connection instance
 */
public class Database
{
    protected static final Logger logger = Logger.getLogger("database");

    /** database name */
    private String name;

    /** database connection */
    private Connection          connection;

    /** statement for query */
    private Statement           queryStatement;

    /** statement for update */
    private PreparedStatement    updateStatement;

    /** statement for prepared SQL */
    private PreparedStatement    preparedStatement;

    /** flag of connection status, false means Not Connected yet */
    private boolean connected = false;

    private String driver;
    private String url;
    private String user;
    private String password;

    public Database(String name)
    {
        this.name = name;
    }
```

```
    public void setName(String name)
    {
       this.name = name;
    }

    public String getName()
    {
       return this.name;
    }

    public void setAutoCommit(boolean autoCommit)
    {
       try {
          connection.setAutoCommit(autoCommit);
       } catch (SQLException e) {
          logger.error("Errors when re-set the 'auto commit' property");
       }
    }
    /**
     * establish connection
     * (here, we don't provide function of closing connection, cause database pool will manage it)
     * @param driver
     * @param url
     * @param user
     * @param passwd
     * @throws DatabaseException
     */
    public void connect(String driver, String url, String user, String passwd)
       throws DatabaseException
    {
       if( this.connected ) return;
       this.driver = driver;
       this.url = url;
       this.user = user;
       this.password = passwd;

       try
       {
          Class.forName( driver );

          this.connection = DriverManager.getConnection( url, user, passwd);
          //this.connection.setAutoCommit(false);
          this.connected = true;

       } catch (SecurityException securityexception) {
          logger.error("", securityexception);
          String s2 = "SecurityException:ErrorMessage=" + securityexception.getMessage();
          throw new DatabaseException(DatabaseException.ERR_DB_OPEN, s2);
       } catch (SQLException sqlexception) {
          //sqlexception.printStackTrace();
          logger.error("", sqlexception);
          String s3 = "SQLException:ErrorMessage=" + sqlexception.getMessage() + "; ErrorCode=" +
sqlexception.getErrorCode() + "; SQLState=" + sqlexception.getSQLState();
          throw new DatabaseException(DatabaseException.ERR_DB_OPEN, s3);
       } catch (Exception exception) {
          logger.error("", exception);
          String s4 = "ErrorMessage=" + exception.getMessage();
          throw new DatabaseException(DatabaseException.ERR_DB_OPEN, s4);
```

```
    }
}

/**
 * re-connect database if there is some problem with db connection
 * @throws DatabaseException
 */
public void reConnect() throws DatabaseException
{
    try {
        if (this.queryStatement != null)
            this.queryStatement.close();
        if (this.updateStatement != null)
            this.updateStatement.close();
        if (this.preparedStatement != null)
            this.preparedStatement.close();
        if (this.connection != null)
            this.connection.close();
    } catch (Exception e) {
        logger.error(e);
    }
    this.connected = false;
    this.queryStatement = null;
    this.updateStatement = null;
    this.preparedStatement = null;
    this.connection = null;
    connect(this.driver, this.url, this.user, this.password);
}

/**
 * begin database transaction
 */
public void beginWork()
{
}

/**
 * commit database transaction
 */
public void commitWork()
    throws DatabaseException
{
    try {
        if (!connection.getAutoCommit()) {
            this.connection.commit();
        }
    } catch (SQLException e) {
        logger.error("", e);
        throw new DatabaseException(DatabaseException.ERR_DB_TX_FAIL, e.getMessage());
    }
}

/**
 * rollback database transaction
 */
public void rollbackWork()
    throws DatabaseException
{
```

```
        try {
            this.connection.rollback();
        } catch (SQLException e) {
            logger.error("", e);
            throw new DatabaseException(DatabaseException.ERR_DB_TX_FAIL, e.getMessage());
        }
    }

    /**
     * prepare the SQL, eg:
     *     prepareSQL("select * from tb_example where col1 = ? and col2 = ?");
     *     setPrepareStatementParaNull( 1 ); //set first argument col2 = null
     *     setPrepareStatementPara( 2, "1234.56" ); //set second argument col2 = 1234.56
     * @param sql
     * @throws PbsDBException
     */
    public void prepareSQL(String sql)
        throws DatabaseException
    {
        try {
            this.preparedStatement = this.connection.prepareStatement(sql);
        } catch (SQLException se) {
            logger.error("", se);
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, se.getMessage());
        }
    }

    /**
     * prepare the SQL, eg:
     *     prepareSQL("select * from tb_example where col1 = ? and col2 = ?");
     *     setPrepareStatementParaNull( 1 ); //set first argument col2 = null
     *     setPrepareStatementPara( 2, "1234.56" ); //set second argument col2 = 1234.56
     * @param sql
     * @throws PbsDBException
     */
    public PreparedStatement prepareSQLStatement(String sql)
        throws DatabaseException
    {
        try {
            return this.connection.prepareStatement(sql);
        } catch (SQLException se) {
            logger.error("", se);
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, se.getMessage());
        }
    }

    /**
     * prepare the Function, eg:
     *     prepareSQL("{call demo_function(arg1, arg2)};");
     *     setPrepareStatementParaNull( 1 ); //set first argument col2 = null
     *     setPrepareStatementPara( 2, "1234.56" ); //set second argument col2 = 1234.56
     * @param function
     * @throws DatabaseException
     */
    public void prepareFunction(String function)
        throws DatabaseException
    {
        try {
```

```
            this.preparedStatement = this.connection.prepareCall(function);
        } catch (SQLException se) {
            logger.error("", se);
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, se.getMessage());
        }
    }

    /**
     * end of prepare statement
     * @throws DatabaseException
     */
    public void unprepareStatement()
    {
        try {
            if ( this.preparedStatement != null)
                this.preparedStatement.close();
        } catch (SQLException se) {
            logger.error("", se);
        }
    }

    /**
     * set prepared SQL or Function argument to null
     * @param index argument index(start from 1)
     * @throws DatabaseException
     */
    public void setPrepareStatementParaNull(int index)
            throws DatabaseException
    {
        if (this.preparedStatement == null)
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, "PreparedStatement");
        try {
            this.preparedStatement.setNull(index, Types.NULL);
        } catch (SQLException se) {
            logger.error("", se);
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, se.getMessage());
        }
    }

    /**
     * set prepared SQL or Function argument to some value
     * @param index argument index(start from 1)
     * @param val value
     * @throws DatabaseException
     */
    public void setPrepareStatementPara(int index, String val)
            throws DatabaseException
    {
        if (this.preparedStatement == null)
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, "PreparedStatement");
        try {
            this.preparedStatement.setString(index, val);
        } catch (SQLException se) {
            logger.error("", se);
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, se.getMessage());
        }
    }
```

```
    /**
     * execute prepared query
     * @return query result
     * @throws DatabaseException
     */
    public ResultSet executeQuery()
            throws DatabaseException
    {
        if (this.preparedStatement == null)
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, "PreparedStatement");
        try {
            this.preparedStatement.executeQuery();
            ResultSet rs = preparedStatement.getResultSet();
            return (rs);
        } catch (SQLException se) {
            logger.error("", se);
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, se.getMessage());
        }
    }

    /**
     * execute prepared update
     * @return number of records been updated
     * @throws DatabaseException
     */
    public int executeUpdate()
            throws DatabaseException
    {
        if (this.preparedStatement == null)
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, "PreparedStatement");
        try {
            int rows = this.preparedStatement.executeUpdate();
            return rows;
        } catch (SQLException se) {
            logger.error("", se);
            throw new DatabaseException(DatabaseException.ERR_DB_PREPARE, se.getMessage());
        }
    }

    /**
     * execute query by specific SQL
     * @param sql
     * @return query result
     * @throws DatabaseException
     */
    public ResultSet executeQuery(String sql)
            throws DatabaseException
    {
        if (connection == null)
            throw new DatabaseException(DatabaseException.ERR_DB_QUERY, "connection not available");
        try {
            if( queryStatement == null )
                queryStatement = connection.createStatement();
            ResultSet resultset = queryStatement.executeQuery(sql);
            //ResultSet resultset = queryStatement.getResultSet();
            return (resultset);
        } catch (SQLException sqlexception) {
            logger.error("SQL:<"+sql+">", sqlexception);
```

```
            String err = "SQLException:ErrorMessage=" + sqlexception.getMessage() + ";ErrorCode=" +
sqlexception.getErrorCode() + ";SQLState=" + sqlexception.getSQLState();
            throw new DatabaseException(DatabaseException.ERR_DB_QUERY, err);
        } catch (Exception exception) {
            logger.error("", exception);
            throw new DatabaseException(DatabaseException.ERR_DB_QUERY, exception.getMessage());
        }
    }

    /**
     * execute update by specific SQL
     * @param sql
     * @return number of records been updated
     * @throws DatabaseException
     */
    public int executeUpdate(String sql)
            throws DatabaseException
    {
        if (connection == null)
            throw new DatabaseException(DatabaseException.ERR_DB_QUERY, "connection not available");
        try {
            updateStatement = connection.prepareStatement(sql);
            int rows = updateStatement.executeUpdate();
            return rows;
        } catch (SQLException sqlexception) {
            logger.error("SQL:<"+sql+">", sqlexception);
            String err = "SQLException:ErrorMessage=" + sqlexception.getMessage() + ";ErrorCode=" +
sqlexception.getErrorCode() + ";SQLState=" + sqlexception.getSQLState();
            throw new DatabaseException(DatabaseException.ERR_DB_QUERY, err);
        } catch (Exception exception) {
            logger.error("", exception);
            throw new DatabaseException(DatabaseException.ERR_DB_QUERY, exception.getMessage());
        } finally {
            if (updateStatement != null) {
                try {
                    updateStatement.close();
                } catch (Exception e) {}
            }
        }
    }

    /**
     * @return Returns the preparedStatement.
     */
    public PreparedStatement getPreparedStatement()
    {
        return preparedStatement;
    }

    /**
     * @param preparedStatement The preparedStatement to set.
     */
    public void setPreparedStatement(PreparedStatement preparedStatement)
    {
        this.preparedStatement = preparedStatement;
    }

    /**
```

```
     * @return Returns the statement.
     */
    public Statement getQueryStatement()
    {
        return queryStatement;
    }

    /**
     * @param statement The statement to set.
     */
    public void setQueryStatement(Statement statement)
    {
        this.queryStatement = statement;
    }

    /**
     * @return Returns the statement.
     */
    public Statement getUpdateStatement()
    {
        return updateStatement;
    }

    /**
     * @param statement The statement to set.
     */
    public void setUpdateStatement(PreparedStatement statement)
    {
        this.updateStatement = statement;
    }

    /**
     * @return Returns the connection.
     */
    public Connection getConnection() {
        return connection;
    }

    /**
     * @param statement The connection to set.
     */
    public void setConnection(Connection connection) {
        this.connection = connection;
    }
}
```

DatabaseDefine.java

DatabaseDefine.java

```
package utils;

/**
 * database definition
 */
public class DatabaseDefine {
    /** default db name is "default_db" */
    public static final String DEFAULT_DB_NAME = "default_db";

    /** database type */
    public static final String TYPE_SQLITE     = "sqlite";
```

```
    public static final String TYPE_MYSQL      = "mysql";
    public static final String TYPE_POSTGRESQL = "postgresql";

    /** database driver */
    public static final String DRIVER_SQLITE     = "org.sqlite.JDBC";
    public static final String DRIVER_MYSQL      = "com.mysql.jdbc.Driver";
    public static final String DRIVER_POSTGRESQL = "org.postgresql.Driver";

    /** database url prefix */
    public static final String URL_PREFIX_SQLITE     = "jdbc:sqlite:";
    public static final String URL_PREFIX_MYSQL      = "jdbc:mysql://";
    public static final String URL_PREFIX_POSTGRESQL = "jdbc:postgresql://";

    /** database url suffix */
    public static final String URL_SUFFIX_MYSQL      = "?autoReconnect=true";

    /** database host and port by default */
    public static final String DEFAULT_HOST           = "localhost";
    public static final String DEFAULT_PORT_MYSQL     = "3306";
    public static final String DEFAULT_PORT_POSTGRESQL = "";

    /** database user by default */
    public static final String DEFAULT_USER_SQLITE     = "";
    public static final String DEFAULT_USER_MYSQL      = "root";
    public static final String DEFAULT_USER_POSTGRESQL = "";

    /** database password by default */
    public static final String DEFAULT_PWD_SQLITE     = "";
    public static final String DEFAULT_PWD_MYSQL      = "";
    public static final String DEFAULT_PWD_POSTGRESQL = "";

    /** database name */
    private String name;

    /** database jdbc driver */
    private String driver;

    /** database ip:port, setting it will automatically generate the url */
    private String host;

    /** database connection url */
    private String url;

    /** database user name */
    private String user;

    /** database user password */
    private String password;

    /** number of initialized database connection, default 1 */
    private int initNum = 1;

    /** number of maximun database connection, default 50 */
    private int maxNum = 50;

    /** database type, default SQLite */
    private String type = TYPE_SQLITE;
```

```
    public String getDriver() {
        return driver;
    }

    /**
     * set db driver, and set db type automatically
     * @param driver
     */
    public void setDriver(String driver) {
        if (driver == null)
            return;

        this.driver = driver;
        if (driver.equals(DRIVER_SQLITE))
            this.type = TYPE_SQLITE;
        else if (driver.equals(DRIVER_MYSQL))
            this.type = TYPE_MYSQL;
        else if (driver.equals(DRIVER_POSTGRESQL))
            this.type = TYPE_POSTGRESQL;
    }

    public int getInitNum() {
        return initNum;
    }

    public void setInitNum(int initNum) {
        if( initNum <= 0 ) return;
        this.initNum = initNum;
    }

    public int getMaxNum() {
        return maxNum;
    }

    public void setMaxNum(int maxNum) {
        if( maxNum <= 0 ) return;
        this.maxNum = maxNum;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
        makeUrlConsistent();
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getHost() {
        return host;
```

```
    }

    /**
     * set db host, and set url automatcially
     * @param host
     */
    public void setHost(String host) {
        if (host == null)
            return;
        this.host = host;
        makeUrlConsistent();
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        if (url != null)
            this.url = url;
    }

    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getType() {
        return type;
    }

    /**
     * set db type, and set default driver, user, and password automatically
     * @param type
     */
    public void setType(String type) {
        if (type == null)
            return;

        this.type = type;
        if (type.equalsIgnoreCase(TYPE_MYSQL)) {
            if (this.driver == null)
                this.driver = DRIVER_MYSQL;
            if (this.user == null)
                this.user = DEFAULT_USER_MYSQL;
            if (this.password == null)
                this.password = DEFAULT_PWD_MYSQL;
        } else if (type.equalsIgnoreCase(TYPE_POSTGRESQL)) {
                if (this.driver == null)
                    this.driver = DRIVER_POSTGRESQL;
                if (this.user == null)
                    this.user = DEFAULT_USER_POSTGRESQL;
                if (this.password == null)
                    this.password = DEFAULT_PWD_POSTGRESQL;
        } else { // TYPE_SQLITE
```

```
            if (this.driver == null)
                this.driver = DRIVER_SQLITE;
            if (this.user == null)
                this.user = DEFAULT_USER_SQLITE;
            if (this.password == null)
                this.password = DEFAULT_PWD_SQLITE;
        }
        makeUrlConsistent();
    }

    private void makeUrlConsistent() {
        if (this.url != null || this.host == null
         || this.type == null || this.name == null)
            return;
        if (this.type.equalsIgnoreCase(TYPE_MYSQL)) {
            this.url = URL_PREFIX_MYSQL + host + "/" + name + URL_SUFFIX_MYSQL;
        } else if (this.type.equalsIgnoreCase(TYPE_POSTGRESQL)) {
            this.url = URL_PREFIX_POSTGRESQL + host + "/" + name;
        }
    }
}
```

DatabaseException.java

DatabaseException.java

```
package utils;

/**
 * database exception, define some exception type
 */
public class DatabaseException extends Exception {

    private static final long serialVersionUID = 1L;

    /** exceed maximum number of connection */
    public final static String ERR_DB_MAX_CONN = "ERR_DB_MAX_CONN";

    /** open database error */
    public final static String ERR_DB_OPEN     = "ERR_DB_OPEN";

    /** query database error */
    public final static String ERR_DB_QUERY    = "ERR_DB_QUERY";

    /** insert, delete or update database error */
    public final static String ERR_DB_UPDATE   = "ERR_DB_UPDATE";

    /** no record for query */
    public final static String ERR_DB_NOTFOUND = "ERR_DB_NOTFOUND";

    /** transaction error */
    public final static String ERR_DB_TX_FAIL  = "ERR_DB_TX_FAIL";

    /** PrepareStatement error */
    public final static String ERR_DB_PREPARE  = "ERR_DB_PREPARE";

    /** database cursors error */
    public final static String ERR_DB_FETCH    = "ERR_DB_FETCH";

    private String exceptionID;
```

```
    public DatabaseException(String id)
    {
        this.exceptionID = id;
    }

    public DatabaseException(String id,String msg)
    {
        super(msg);
        this.exceptionID = id;
    }
}
```

DatabasePool.java

DatabasePool.java

```
package utils;

import java.util.Stack;

/**
 * Cache database connections.
 */
public class DatabasePool {

    /**
     * specific database configuration xml file,
     * eg: -Ddb_config_file=/cpminer/Database.xml
     * default file is $CPMiner_HOME/configuration/Database.xml
     */
    public static final String DATABASE_CONF = "db_config_file";

    /** database definition */
    private DatabaseDefine dbDef;

    /** free database pool */
    private Stack<Database> freeDBPool;

    /** number of using database connection */
    private int busyNum = 0;

    public DatabasePool()
    {
       loadDatabasePoolDefine();
    }

    public DatabasePool(DatabaseDefine dbDef)
    {
       this.dbDef = dbDef;
    }

    /**
     * load Database.xml into DatabaseDefine instance
     */
    private void loadDatabasePoolDefine()
    {
       if( dbDef != null ) return;

    /* modified by Cheng Xiao on 12/20/2007 *
       String xmlFilename = System.getProperty(DATABASE_CONF);
       if( xmlFilename == null )
```

```
            xmlFilename = "Database.xml";
        dbDef = (DatabaseDefine)XmlLoader.loadObject(xmlFilename);
    */

        dbDef = new DatabaseDefine();
        dbDef.setDriver("org.sqlite.JDBC");
        dbDef.setUrl("jdbc:sqlite:" + TOMCAT_LOC + DB_LOC);
        dbDef.setUser("");
        dbDef.setPassword("");
        dbDef.setInitNum(5);
        dbDef.setMaxNum(10);

        /* added by Cheng Xiao on 12/25/2007 */
        dbDef.setName(DatabaseDefine.DEFAULT_DB_NAME);
        /* end of addition */
    }

    private static final String FILE_SEPARATOR = System.getProperty("file.separator");
    private static final String TOMCAT_LOC = System.getenv("SYSMINER_DATA");
    private static final String DB_LOC = FILE_SEPARATOR + "db" + FILE_SEPARATOR +
"CPMiner.db";
    /* end of modification */

    /**
     * initialize database connection pool
     */
    public void initDatabasePool()
        throws DatabaseException
    {
        if( dbDef == null )
            throw new DatabaseException(DatabaseException.ERR_DB_OPEN, "database definition never
loaded");

        freeDBPool  = new Stack<Database>();

        for(int i = 0; i < dbDef.getInitNum(); i++ )
        {
            Database db = this.createDatabase();
            freeDBPool.push( db );
        }
    }

    /**
     * release database connection pool
     */

    public void releasePool( )
    {
        int num = this.freeDBPool.size();
        for(int i = 0; i < num; i++){
            Database db = this.freeDBPool.get(i);
            try{
                db.getConnection().close();
            }catch(Exception e){
                e.printStackTrace();
            }
        }
        this.freeDBPool.clear();
```

```
        this.freeDBPool = null;
    }

    /**
     * get a free connection from pool
     */
    public Database getFreeDatabase()
        throws DatabaseException
    {
        if( getFreeNum() > 0 ){
            this.busyNum++;
            return this.freeDBPool.pop();
        }else{
            if( this.busyNum >= this.dbDef.getMaxNum() )
                throw new DatabaseException(DatabaseException.ERR_DB_MAX_CONN, "reach maximum
connections of database");
            else{
                this.busyNum++;
                return this.createDatabase();
            }
        }
    }

    /**
     * release an unused connection into pool
     * @param db database connection instance
     */
    public void releaseBusyDatabase(Database db)
    {
        if(db!=null)
        {
            this.freeDBPool.push(db);
            this.busyNum--;
        }
    }

    /**
     * get number of current used connection
     */
    public int getBusyNum()
    {
        return this.busyNum;
    }

    /**
     * get number of connection free to use
     */
    public int getFreeNum()
    {
        return this.freeDBPool.size();
    }

    /**
     * establish a database connection
     * @return a database instance
     * @throws DatabaseException
     */
    private Database createDatabase()
```

```
        throws DatabaseException
    {
        Database db = new Database(dbDef.getName());
        db.connect(this.dbDef.getDriver(), this.dbDef.getUrl(), this.dbDef.getUser(), this.dbDef.getPassword());
        return db;
    }
}
```

FileUtils.java

FileUtils.java

```
package utils;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Date;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;

/**
 * A collection of methods useful for working with files
 */
public class FileUtils {

    /**
     * Reads in the entire file and places it into a string with
     * each line delimited by a \n character.  <b>Note:</b> that the default
     * enconding is used, this might create problems with files that contain
     * strange characters. In that case can call getFileAsByteArray and use
     * the appropriate encoding to create a String from the bytes.
     *
     * @param  file The file to read in
     * @return A String with the contents of the file.
     * @throws FileNotFoundException If the file does not exist
     * @throws IOException If there is an error while reading the file
     */
    public static String getFileAsString(File file)
        throws IOException, FileNotFoundException {
        return new String(geFileAsByteArray(file));
    }

    /**
     * Reads the contents of the given file into a byte array.
     *
     * @param file The file to be read
     *
     * @return A byte array with the contents of the file
     * @throws FileNotFoundException If the file does not exist
     * @throws IOException If there is an error while reading the file
     */
    public static byte[] geFileAsByteArray(File file) throws FileNotFoundException, IOException {
        long length = file.length();
        MappedByteBuffer inBuffer = new FileInputStream(file).getChannel().map(
```

```
            FileChannel.MapMode.READ_ONLY, 0, length);
        byte[] destination = new byte[(int) length];
        inBuffer.get(destination);
        return destination;
    }

    /**
     * Count the number of lines in a file. It depends on BufferedReader.readLine()
     * to break down in lines the contents of the given file
     *
     * @param file The file to count the lines for
     *
     * @return The number of lines the file contains. -1 if the file does not exist
     * or an error happens while reading it.
     */
    public static int getTotalLines(File file) {
        BufferedReader in = null;
        int totalLines = 0;
        try {
            try {
                in = new BufferedReader(new FileReader(file));
                while ((in.readLine()) != null) {
                    totalLines++;
                }
            } finally {
                if (in != null) {
                    in.close();
                }
            }
        } catch (IOException e) {
            totalLines = -1;
        }
        return totalLines;
    }

    /**
     * Write a string to the given file. Creates the file and directory
     * structure if it does not exist
     *
     * @param s The string to write.
     * @param file The File to write the string to.
     * @throws IOException If an error occurs when creating the file or
     *         writing to it
     */
    public static void writeStringToFile(String s, File file) throws IOException {
        if (!file.exists()) {
            createFileStructure(file);
        }
        BufferedWriter out = new BufferedWriter(new FileWriter(file));
        try {
            out.write(s);
            out.flush();
        } finally {
            out.close();
        }
    }

    /**
```

```
     * Copy a file from one absolute path to another. If the source does not exist
     * the method returns false. If the destination does not exist it is created
     * along with all it's parent directories as needed. If the source or the
     * destination is a directory the method returns false without copying anything
     *
     * @param source A Sting representing the path of a file on disk
     * @param dest A String representing the path of a file
     *
     * @throws IOException If there is a problem when creating the destination
     *         File path or while copying the data
     */
    public static boolean copyFile(String source, String destination) throws IOException {
        return copyFile(new File(source), new File(destination));
    }

    /**
     * Copy a file from one absolute path to another. If the source does not exist
     * the method returns false. If the destination does not exist it is created
     * along with all it's parent directories as needed. If the source is a
     * directory the method returns false without copying anything.If the destination
     * is a directory just copy the source into the destination directory
     *
     * @param source A File object representing a file on disk
     * @param dest A file object representing an abstract file path
     *
     * @throws IOException If there is a problem when creating the destination
     *         File path or while copying the data
     */
    public static boolean copyFile(File source, File destination) throws IOException {
        if (source == null || destination == null) {
            return false;
        }

        if (!source.exists()) {
            return false;
        }
        if (source.isDirectory()) {
            return false;
        }
        //if destination is directory copy the source into the directory
        if (destination.isDirectory()) {
            File newDestination = new File(destination, source.getName());
            return copyFile(source, newDestination);
        }

        FileChannel in = new FileInputStream(source).getChannel();
        if(!destination.exists()) {
            createFileStructure(destination);
        }
        FileChannel out = new FileOutputStream(destination).getChannel();

        try {
            in.transferTo(0, in.size(), out);
        } finally {
            if (null != in) {
                in.close();
            }
            if (null != out) {
```

```
            out.close();
        }
    }
    return true;
}

/**
 * Reads in and returns the range of lines from the file, the result
 * is placed in a string and delimited by <code>System.getProperty("line.separator");</code>
 * characters. If the range exists partially in the file, e.g. if the file has 100 lines
 * and the range asks for [90, 110], only the overlapping lines are returned -- in
 * this case [90, 100]
 *
 * @param  file   The file to read in
 * @param  range  The line number range to read in, both ends are inclusive
 *                Line numbers start at 1, and include the start and end lines
 * @return the string containing the file range
 * @throws FileNotFoundException If the file does not exist
 * @throws IOException If there is a problem while reading the file
 */
public static String getFileRangeAsString(File file, Range range)
    throws FileNotFoundException, IOException{
    final int READ_BUFFER_SIZE = 4096;

    StringBuffer fileContents = new StringBuffer();
    FileReader fileReader = new FileReader(file);
    BufferedReader in = new BufferedReader(fileReader, READ_BUFFER_SIZE);
    String lineSeparator = System.getProperty("line.separator");
    try {
        String line;
        int lineNumber = 1;
        while ((line = in.readLine()) != null && lineNumber <= range.end) {
            if (lineNumber >= range.start) {
                fileContents.append(line);
                if (lineNumber < range.end) {
                    fileContents.append(lineSeparator);
                }
            }
            lineNumber++;
        }
    } finally {
        in.close();
        fileReader.close();
    }
    return fileContents.toString();
}

/**
 * Attempts to determine if the file is binary by reading through the file
 * looking for any 0 bytes, if it finds one then it returns True
 *
 * @param file The file to check
 * @returns True if binary, false otherwise
 *
 * @throws IOException If there is an error while reading the file
 */
public static boolean isBinary(File file) throws IOException {
    return !isTextContent(file);
```

```
    }

    /**
     * Attempts to determine if the content of a file is text. This method is probabilistic
     * as it reads a limited number of characters from the file trying to see if there
     * any non text characters in it. It is <b>not guaranteed</b> that the response is
     * accurate.
     *
     * @param file The file to test
     *
     * @return true if the contents are textual, false otherwise
     * @throws IOException If the file does not exist or there is a problem while reading it
     */
    public static boolean isTextContent(File file) throws IOException {
        boolean isText = true;
        final int MAX_NUMBER_OF_CHARACTERS_TO_TEST = 100;
        BufferedReader bufferedReader = new BufferedReader(new FileReader(file));
        try {
            int singleChar;
            int count = 0;
            while ((singleChar = bufferedReader.read()) != -1 && count <
MAX_NUMBER_OF_CHARACTERS_TO_TEST) {
                count++;
                // Reads through the file looking for any bytes with value less than 0
                // or less than 31 with the exception of 9:tab, 11:vertical tab,1 0:\n,
                // 12:page, 13:enter. If it finds one, it determines that the file is
                // binary
                if (singleChar < 0 ||
                    (singleChar < 31 && singleChar != 9 && singleChar != 10
                            && singleChar != 11 && singleChar != 12 && singleChar != 13)) {
                    isText = false;
                    break;
                }
            }
        } finally {
            try {
                bufferedReader.close();
            } catch (IOException e) {
                // Ignore Exception as it does not prevent the method from
                // deciding on whether the file is text or not
            }
        }

        return isText;
    }

    /**
     * Copy the source abstract path to the destination. It uses the 'cp'
     * semantics:
     * <ul>
     * <li> If the source or destination are null it returns false</li>
     * <li> If the source does not exist it returns false</li>
     * <li> If the destination is a directory, it copies the source
     * (regardless of whether it's a file or a directory) <b>into</b> the
     * destination and return true</li>
     * <li> If the source is a directory and the destination is a file
     * the method does not copy anything and returns false</li>
     * <li> If the source is a file and the destination is a file
```

```
     * the method <b>overwrites</b> the destination and returns true</li>
     * <li> If the source is a directory and the destination does
     * not exist, it creates the target and copies recursively all it's
     * contents</li>
     * <li> If the source is a file and the destination does
     * not exist, it creates the target and copies it's contents</li>
     *</ul>
     * Also, if the source is a directory and the copy is partially successful,
     * i.e. only when some of the files in the directory are copied successfully,
     * false is returned.
     *
     * @param source A File object that represents a file or directory on the disk
     * @param destination A File object that represents an abstract path to a file or directory
     * @return
     *         true when it is successful, false otherwise as described above
     * @throws IOException When cannot read/read to/from the source/destination
     */
    public static boolean copyFiles(File source, File destination) throws IOException {
        if ( source == null || destination == null ) {
            return false;
        }

        if(!source.exists()) {
            return false;
        }

        if (source.isFile()) {
            return copyFile(source, destination);
        } else {
            if( destination.exists() && destination.isFile()) {
                // we could not copy a directory into a file
                return false;
            } else {
                if (!destination.exists()) {
                    destination.mkdir();
                }
                String destPath = destination.getAbsolutePath();
                boolean wasSuccesfull = true;
                for (File child : source.listFiles()) {
                    if (child.isFile()) {
                        wasSuccesfull &= copyFile(child, new File(destPath + File.separator + child.getName()));
                    }
                    if (child.isDirectory()) {
                        wasSuccesfull &= copyFiles(child, new File(destPath + File.separator + child.getName()));
                    }
                }
                return wasSuccesfull;
            }
        }
    }

    /**
     * Delete a file. <b>CAUTION:</b> If file represents a directory,
     * it is deleted recursively (i.e. first deletes all the files/dirs it
     * contains), i.e. behaves like <i>rm -rf</i>. The operation is not
     * atomic, if false is returned, part of the directory structure might
     * have still been removed successfully.
     *
```

```
     * @param path A The file to be deleted.
     * @return It returns true if the file (and all it's children in case
     *         of a directory) have been successfully removed, false otherwise.
     *         Note: if a file or a directory does not exist, it returns false.
     */
    public static boolean deleteFile(File path) {
        if (!path.exists()) {
            return false;
        }
        if (path.isFile()) {
            return path.delete();
        }
        File[] files = path.listFiles();
        boolean succesfullyDeletedChildren = true;
        for (int i = 0; i < files.length; i++) {
            // All the deletions need to be successful to declare victory
            succesfullyDeletedChildren = succesfullyDeletedChildren && deleteFile(files[i]);
        }
        return (path.delete() && succesfullyDeletedChildren);
    }

    /**
    *
    * convert ../ and ./ to t/
    *
    * @param dir
    * @return
    */
    public static String convertPath(String dir){
        if ( dir != null){
            dir = dir.replace("../", "t/");
            dir = dir.replace("./", "t/");
            return dir;
        }
        return null;
    }

    /**
     * A python-like path join method. It joins a list of path segments into
     * a complete path in the way the python method os.path.join() works.
     * The first path in the list, or the last path that start with '/' (File.separator)
     * is placed first in the resulting file:<br/>
     * FileUtils.join("foo", "bar") produces foo/bar <br/>
     * FileUtils.join("foo", "/bar", "baz") produces /bar/baz
     *
     * @param dirs A list of path segments.
     * @return A String representing the join path. The file might not exist on disk.
     */
    public static String join(String ... paths) {
        //Use String instead of StringBuilder, since StringBuilder doesn't provide
        //endsWith(). Moreover, this operation is very light-weight anyway,
        //so String won't be worse than StringBuilder.
        String finalDir = "";
        String fileSep = File.separator;
        for (String dir : paths) {
            if (dir.startsWith(fileSep)) {
                finalDir = dir;
                continue;
```

```
            } else if (finalDir.length() > 0 && !finalDir.endsWith(fileSep)) {
                finalDir += fileSep;
            }
            finalDir += dir;
        }
        return finalDir;
    }

    /**
     * Similar method to join(), the difference is that joinSafely() allow path
     * item begin with "/". E.g.: FileUtils.join("foo", "/bar", "baz") produces
     * foo/bar/baz
     *
     * @param dirs A list of path segments.
     * @return A String representing the join path. The file might not exist on disk.
     * @see <code>FileUtils.join()</code>
     */
    public static String joinSafely(String ... paths) {
        String finalDir = "";
        String fileSep = File.separator;
        for (String dir : paths) {
            if (finalDir.length() > 0 && !finalDir.endsWith(fileSep) && !dir.startsWith(fileSep)) {
                finalDir += fileSep;
            }
            finalDir += dir;
        }
        return finalDir;
    }

    /**
     * Touch a file. If the given file doesn't exist, create an empty file, otherwise,
     * change its timestamp to the current time. Works just like the unix command "touch".
     *
     * Note: if the file doesn't exsit, it will create the file and all neccessary parent
     * directories.
     *
     * @param filename The file you want to create.
     * @return A boolean that indicates if the file has been touched successfully.
     */
    public static boolean touch(String filename) {
        File file = new File(filename);
        try {
            if (file.exists()) {
                Date date = new Date();
                file.setLastModified(date.getTime());
            } else {
                createFileStructure(new File(filename));
            }
        } catch (Exception e) {
            return false;
        }
        return true;
    }

    /**
     * Creates an empty directory in the default temporary-file directory.
     * It works just like File.createTempFile(), but it creates a directory instead of
     * a file.
```

```
     *
     * By default, it looks for the system temp directory, say /tmp in Linux or something
     * I'm not sure in Windows. You can also specify the temp directory when you start
     * the JVM by:
     *
     * <tt>java -Djava.io.tmpdir=the-directory-name</tt>
     *
     * Note: Remember to remove this directory after you use it. Unlike createTempFile(),
     *       this temp directory won't be able to be removed automatically when the JVM
     *       is done, if you have files/subdirectories in it.
     *
     * @param dirName The name of the temp directory.
     * @return An abstract pathname denoting a newly-created empty directory. Return
     *         null if fails.
     * @throw IOException, when it fails to create the temp directory.
     */
    public static File createTempDirectory(String dirName) throws IOException {
       File tmpDir = null;
       tmpDir = File.createTempFile(dirName, "");
       tmpDir.delete();
       tmpDir.mkdir();
       return tmpDir;
    }

    /**
     * Get the parent directory name of the given directory/file.
     *
     * Note: the difference between this method and <code>java.io.File.getParent()</code> is
     * that (1) you can directly get parent path from a <code>String</code> and (2) it returns
     * an empty string instead of a null when there is no parent directory in the given path.
     *
     * @param path The name of a directory of a path.
     * @return The name of the parent directory of it. Return an empty string, "",
     *         if the path do not have a parent directory.
     */
    public static String getParentPath(String path) {
       String parent = (new File(path)).getParent();
       if (parent == null) {
          parent = "";
       }
       return parent;
    }

    /**
     * Removes all instances of a/foo/../foo/b, yielding a/foo/b .
     * In other words we remove instances where we would backout one directory
     * level and immediately re-enter that directory.
     *
     * @param path The original path
     * @return The normalized path
     */
    public static String normalizePath(String path) {
       // (\w+)/\.\./\1/ - matches (foo)/../foo/
       String dupDirRegex = "([^/]+)/\\.\\./\\1/";
       return path.replaceAll("/+", "/").replaceAll("/\\./", "/").replaceAll(dupDirRegex, "$1\\/");
    }

    /**
```

```
     * Get the extension of a file name. When a filename contains a revision number
     * at the end, e.g. Perforce does that, the revision number will be removed.
     * For example, "foo.c#12" has extension "c".
     *
     * @param path The name of the file.
     * @return The extension of the given file name. Return an empty string when the
     *         filename does not have extension.
     */
    public static String getExtension(String filename) {
        if (filename.lastIndexOf(File.separator) >= filename.lastIndexOf(".")) {
            return "";
        } else {
            String ext = filename.substring(filename.lastIndexOf(".") + 1, filename.length());
            int hashIndex = ext.indexOf("#");
            if (hashIndex != -1) {
                ext = ext.substring(0, hashIndex);
            }
            return ext;
        }
    }

    private static void createFileStructure(File file) throws IOException {
        File parent = file.getParentFile();
        if (parent != null) {
            if(!parent.exists()) {
                parent.mkdirs();
            }
            if(!file.createNewFile()) {
                throw new IOException(file.getAbsolutePath() + " cannot be created");
            }
        }
    }

}
```

Range.java

Range.java

```
package utils;

import java.io.Serializable;

public class Range implements Serializable {
    private static final long serialVersionUID = 1L;

    public int start;
    public int end;

    public Range() {
        start = 0;
        end = 0;
    }

    public Range(int s, int e) {
        start = s;
        end = e;
    }

    public int getStart() {
        return start;
```

```
    }

    public void setStart(int start) {
        this.start = start;
    }

    public int getEnd() {
        return end;
    }

    public void setEnd(int end) {
        this.end = end;
    }

    public int middle() {
        return this.start + (this.end - this.start) / 2;
    }

    public int range(){
        return end - start + 1;
    }

    public boolean intersects(Range other) {
        return other.start <= end && other.end >= start;
    }

    public String toString() {
        return "Range ("+start+","+end+")";
    }

    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (!(obj instanceof Range)) return false;
        Range r = (Range)obj;
        return (start == r.start && end == r.end);
    }

    public int hashCode() {
        return toString().hashCode();
    }
}
```

ThreadLocalStack.java

ThreadLocalStack.java

```
package utils;

import java.util.Deque;
import java.util.LinkedList;
import java.util.NoSuchElementException;

/**
  * A thread-local stack of values, inherited across threads.  This class is
  * used to implement a dynamic variable binding-like pattern.  A child
  * thread's stack is an independent copy of its parent thread's stack
  * (however, the same exact <i>objects</i> appear on both).
  */
public class ThreadLocalStack<T> {
    /**
      * Represents the capability to remove an element from a
```

```
     * ThreadLocalStack.
     *
     * @see ThreadLocalStack#push
     * @see ThreadLocalStack#pop
     */
    public class Key {
        final T value;
        public Key(T value) {
            this.value = value;
        }
    }

    /**
     * Implementation of the thread-local stack extends
     * InheritableThreadLocal to allow new threads to copy their parents'
     * stacks.
     */
    private class ThreadLocalStackImpl extends InheritableThreadLocal<Deque<Key>> {
        ThreadLocalStackImpl() {
            super();
            set(new LinkedList<Key>());
            get().push(new Key(null));
        }
        /**
         * Called internally by InheritableThreadLocal to create a new
         * thread's local stack from its parent's stack.
         */
        protected Deque<Key> childValue(Deque<Key> parentValue) {
            if (parentValue != null) {
                return new LinkedList<Key>(parentValue);
            } else {
                return new LinkedList<Key>();
            }
        }
        public Deque<Key> get() {
            Deque<Key> stack = super.get();
            if (stack == null) {
                stack = new LinkedList<Key>();
                set(stack);
            }
            return stack;
        }
    }

    private final ThreadLocalStackImpl threadLocal;

    public ThreadLocalStack() {
        this.threadLocal = new ThreadLocalStackImpl();
    }

    /**
     * Returns the value at the top of the stack.
     */
    public T peek() {
        Deque<Key> threadLocalStack = threadLocal.get();
        if (threadLocalStack.size() > 0) {
            return threadLocalStack.peek().value;
        } else {
```

```
            return null;
        }
    }

    /**
     * Pushes a new value onto the stack, and returns the Key that allows
     * you to later pop it from the stack.
     */
    public Key push(T value) {
        Key key = new Key(value);
        threadLocal.get().push(key);
        return key;
    }

    /**
     * Pops the value off the top of the stack <i>if</i> the correct Key is
     * supplied.  The Key is necessary in order to somewhat avoid
     * accidental stack corruption due to programming errors.
     */
    public void pop(Key key) throws NoSuchElementException {
        Deque<Key> threadLocalStack = threadLocal.get();
        if (threadLocalStack.peek() == key) {
            threadLocalStack.pop();
        } else {
            throw new NoSuchElementException();
        }
    }
}
```

UniqueID.java

UniqueID.java

```
package utils;

import java.io.Serializable;
import java.util.Random;

public class UniqueID implements Serializable {
    private static final long serialVersionUID = 1L;

    private static final Random randomGen = new Random();

    /** Maintains a thread-local stack of UniqueIDs to simulate dynamic binding. */
    public static final ThreadLocalStack<UniqueID> threadLocalStack = new
ThreadLocalStack<UniqueID>();

    /** First component of a unique id: creation time. */
    public final long time;
    /** Second component of a unique id: a random number. */
    public final long random;

    public UniqueID() {
        this.time   = System.currentTimeMillis();
        this.random = randomGen.nextLong();
    }

    public boolean equals(UniqueID other) {
        return (other != null) && (time == other.time) && random == other.random;
    }
```

```
    public int hashCode() {
        return toString().hashCode();
    }

    public String toString() {
        return time + ":" + random;
    }

    /**
     * Returns the String value of the UniqueID currently at the top of the
     * thread-local stack.
     */
    public static String currentString() {
        UniqueID uniqueID = threadLocalStack.peek();
        if (uniqueID != null) {
            return uniqueID.toString();
        } else {
            return "null";
        }
    }
}
```

The parsedquestion package

Constants.java

Constants.java

```
/**
 * Autogenerated by Thrift
 *
 * DO NOT EDIT UNLESS YOU ARE SURE THAT YOU KNOW WHAT YOU ARE DOING
 */
package parsedquestion;

import org.apache.commons.lang.builder.HashCodeBuilder;
import java.util.List;
import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;
import java.util.Set;
import java.util.HashSet;
import java.util.Collections;

public class Constants {

  public static final String FIELD_PREFIX = "__CAT_";

  public static final String QUESTION_ID = "__CAT_QUESTIONID";

  public static final String MODEL_NAME = "__CAT_MODELNAME";

  public static final String TEST_ID = "__CAT_TESTID";

  public static final String TEST_SUBJECT = "__CAT_TESTSUBJECT";

  public static final String TEST_TOPIC = "__CAT_TESTTOPIC";

  public static final int INVALID_QUESTION_ID = -1;

  public static final int INVALID_TEST_ID = -1;
```

```
  public static final String THREE_PL_MODE = "__CAT_3PL";

}
```

QuestionStructUtility.java

QuestionStructUtility.java

```
package parsedquestion;

import java.util.Arrays;
import java.util.List;
import java.io.IOException;
import java.util.Collections;
import java.util.Comparator;

import org.apache.thrift.TBase;

import thrift.QuestionSerialization;

/**
 * Utility class to get information from questions and messages.
 */
public final class QuestionStructUtility {
    /** Utility class. */
    private QuestionStructUtility() {
    }

    /**
     * Compare two question units, return true if they have the same cotents,
     * return false otherwise.
     */
    public static boolean compareQuestionUnits(QuestionUnit q1, QuestionUnit q2) {
        try {
            if (q1.getDescription().compareTo(q2.getDescription()) != 0) {
                return false;
            }
            if (q1.getOptions().equals(q2.getOptions()) == false) {
                return false;
            }
            if (q1.getAnswers().equals(q2.getAnswers()) == false) {
                return false;
            }
            if (q1.getAttributes().equals(q2.getAttributes()) == false) {
                return false;
            }
            if (q1.getParameters().equals(q2.getParameters()) == false) {
                return false;
            }
            if (q1.getModel().compareTo(q2.getModel()) != 0) {
                return false;
            }
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    /**
     * Compare two test units, return true if they have the same cotents,
     * return false otherwise.
```

```
 */
public static boolean compareTestUnits(TestUnit t1, TestUnit t2) {
    try {
        if (t1.getName().compareTo(t2.getName()) != 0) {
            return false;
        }
        if (t1.getTargetlength() != t2.getTargetlength()) {
            return false;
        }
        if (t1.getAttributes().equals(t2.getAttributes()) == false) {
            return false;
        }
        List<QuestionUnit> questions1 = t1.getQuestions();
        List<QuestionUnit> questions2 = t2.getQuestions();
        if (questions1.size() != questions2.size()) {
            return false;
        }
        Collections.sort(questions1, new QUComparator());
        Collections.sort(questions2, new QUComparator());

        for (int i = 0; i < questions1.size(); i++) {
            QuestionUnit q1 = t1.getQuestions().get(i);
            QuestionUnit q2 = t2.getQuestions().get(i);
            if (!compareQuestionUnits(q1, q2)) {
                return false;
            }
        }
        return true;
    } catch (Exception e) {
        System.out.println("4");

        return false;
    }
}

/** Sort question units on question id .*/
private static class QUComparator implements Comparator {
    public int compare(final Object o1, final Object o2) {
        QuestionUnit q1 = (QuestionUnit) o1;
        QuestionUnit q2 = (QuestionUnit) o2;
        long id1 = getQuestionID(q1);
        long id2 = getQuestionID(q2);

        if (id1 <= id2) {
            return -1;
        } else {
            return 1;
        }
    }
}

/**
 * Return test suite ID of the given question.
 * @return test ID of the given question, or null if the field cannot be found or
 *         in wrong format.
 */
public static long getTestID(QuestionUnit question) {
    String idStr = question.getAttributes().get(Constants.TEST_ID);
```

```
        if (idStr == null) {
            return Constants.INVALID_TEST_ID;
        }
        try {
            long id = Long.parseLong(idStr);
            return id;
        } catch (NumberFormatException nfe) {
            return Constants.INVALID_TEST_ID;
        }
    }

    public static void setTestID(QuestionUnit question, long testId) {
        question.getAttributes().put(Constants.TEST_ID, Long.toString(testId));
    }

    public static void setTestID(TestUnit test, long testId) {
        test.getAttributes().put(Constants.TEST_ID, Long.toString(testId));
    }

    /**
     * Return question ID of the given question.
     * @return question ID of the given question, or null if the field cannot be found or
     *         in wrong format.
     */
    public static long getQuestionID(QuestionUnit question) {
        String idStr = question.getAttributes().get(Constants.QUESTION_ID);
        if (idStr == null) {
            return Constants.INVALID_QUESTION_ID;
        }
        try {
            long id = Long.parseLong(idStr);
            return id;
        } catch (NumberFormatException nfe) {
            return Constants.INVALID_QUESTION_ID;
        }
    }

    public static void setQuestionID(QuestionUnit question, long questionId) {
        question.getAttributes().put(Constants.QUESTION_ID, Long.toString(questionId));
    }

    /**
     * Return model name of the given question.
     * @return model name of the given question, or null if the field cannot be found or
     *         in wrong format.
     */
    public static String getModelName(QuestionUnit question) {
        String modelName = question.getAttributes().get(Constants.MODEL_NAME);
        if (modelName == null) {
            return null;
        }
        return modelName;
    }

    /**
     * Check whether an attribute name is an internal
     * predefined name that is starting with Constants.FIELD_PREFIX.
     *
```

```
     * @param name
     *            attribute name.
     * @return true if it is an internal predefined name.
     */
    public static boolean isInternalName(final String name) {
        return name.regionMatches(0, Constants.FIELD_PREFIX, 0, Constants.FIELD_PREFIX.length());
    }

    /**
     * Return test suite ID of the given question.
     * @return test ID of the given question, or null if the field cannot be found or
     *         in wrong format.
     */
    public static long getTestID(TestUnit test) {
        String idStr = test.getAttributes().get(Constants.TEST_ID);
        if (idStr == null) {
            return Constants.INVALID_TEST_ID;
        }
        try {
            long id = Long.parseLong(idStr);
            return id;
        } catch (NumberFormatException nfe) {
            return Constants.INVALID_TEST_ID;
        }
    }

    /**
     * Return subject of the given test.
     * @return subject of the given test, or null if the field cannot be found or
     *         in wrong format.
     */
    public static String getTestSubject(TestUnit test) {
        String subject = test.getAttributes().get(Constants.TEST_SUBJECT);
        if (subject == null) {
            return null;
        }
        return subject;
    }

    public static void setTestSubject(TestUnit test, String subject) {
        test.getAttributes().put(Constants.TEST_SUBJECT, subject);
    }

    /**
     * Return topic of the given test.
     * @return topic of the given test, or null if the field cannot be found or
     *         in wrong format.
     */
    public static String getTestTopic(TestUnit test) {
        String topic = test.getAttributes().get(Constants.TEST_TOPIC);
        if (topic == null) {
            return null;
        }
        return topic;
    }

    public static void setTestTopic(TestUnit test, String topic) {
        test.getAttributes().put(Constants.TEST_TOPIC, topic);
```

```
    }
}
```

QuestionUnit.java

QuestionUnit.java

```
/**
 * Autogenerated by Thrift
 *
 * DO NOT EDIT UNLESS YOU ARE SURE THAT YOU KNOW WHAT YOU ARE DOING
 */
package parsedquestion;

import org.apache.commons.lang.builder.HashCodeBuilder;
import java.util.List;
import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;
import java.util.Set;
import java.util.HashSet;
import java.util.Collections;

import org.apache.thrift.*;
import org.apache.thrift.meta_data.*;
import org.apache.thrift.protocol.*;

/**
 * Question unit.
 */
public class QuestionUnit implements TBase, java.io.Serializable, Cloneable {
  private static final TStruct STRUCT_DESC = new TStruct("QuestionUnit");
  private static final TField DESCRIPTION_FIELD_DESC = new TField("description", TType.STRING,
(short)1);
  private static final TField OPTIONS_FIELD_DESC = new TField("options", TType.LIST, (short)2);
  private static final TField ANSWERS_FIELD_DESC = new TField("answers", TType.SET, (short)3);
  private static final TField ATTRIBUTES_FIELD_DESC = new TField("attributes", TType.MAP, (short)4);
  private static final TField PARAMETERS_FIELD_DESC = new TField("parameters", TType.LIST,
(short)5);
  private static final TField MODEL_FIELD_DESC = new TField("model", TType.STRING, (short)6);

  /**
   * Question Description
   */
  public String description;
  public static final int DESCRIPTION = 1;
  /**
   * Question Options
   */
  public List<String> options;
  public static final int OPTIONS = 2;
  /**
   * Question Answers
   */
  public Set<Integer> answers;
  public static final int ANSWERS = 3;
  /**
   * Other Attributes
   */
  public Map<String,String> attributes;
  public static final int ATTRIBUTES = 4;
```

```
  /**
   * Calibrated Parameters
   */
  public List<Double> parameters;
  public static final int PARAMETERS = 5;
  /**
   * Model
   */
  public String model;
  public static final int MODEL = 6;

  private final Isset __isset = new Isset();
  private static final class Isset implements java.io.Serializable {
  }

  public static final Map<Integer, FieldMetaData> metaDataMap = Collections.unmodifiableMap(new
HashMap<Integer, FieldMetaData>() {{
    put(DESCRIPTION, new FieldMetaData("description", TFieldRequirementType.DEFAULT,
        new FieldValueMetaData(TType.STRING)));
    put(OPTIONS, new FieldMetaData("options", TFieldRequirementType.DEFAULT,
        new ListMetaData(TType.LIST,
            new FieldValueMetaData(TType.STRING))));
    put(ANSWERS, new FieldMetaData("answers", TFieldRequirementType.DEFAULT,
        new SetMetaData(TType.SET,
            new FieldValueMetaData(TType.I32))));
    put(ATTRIBUTES, new FieldMetaData("attributes", TFieldRequirementType.DEFAULT,
        new MapMetaData(TType.MAP,
            new FieldValueMetaData(TType.STRING),
            new FieldValueMetaData(TType.STRING))));
    put(PARAMETERS, new FieldMetaData("parameters", TFieldRequirementType.DEFAULT,
        new ListMetaData(TType.LIST,
            new FieldValueMetaData(TType.DOUBLE))));
    put(MODEL, new FieldMetaData("model", TFieldRequirementType.DEFAULT,
        new FieldValueMetaData(TType.STRING)));
  }});

  static {
    FieldMetaData.addStructMetaDataMap(QuestionUnit.class, metaDataMap);
  }

  public QuestionUnit() {
  }

  public QuestionUnit(
    String description,
    List<String> options,
    Set<Integer> answers,
    Map<String,String> attributes,
    List<Double> parameters,
    String model)
  {
    this();
    this.description = description;
    this.options = options;
    this.answers = answers;
    this.attributes = attributes;
    this.parameters = parameters;
    this.model = model;
```

```
  }

  /**
   * Performs a deep copy on <i>other</i>.
   */
  public QuestionUnit(QuestionUnit other) {
    if (other.isSetDescription()) {
      this.description = other.description;
    }
    if (other.isSetOptions()) {
      List<String> __this__options = new ArrayList<String>();
      for (String other_element : other.options) {
        __this__options.add(other_element);
      }
      this.options = __this__options;
    }
    if (other.isSetAnswers()) {
      Set<Integer> __this__answers = new HashSet<Integer>();
      for (Integer other_element : other.answers) {
        __this__answers.add(other_element);
      }
      this.answers = __this__answers;
    }
    if (other.isSetAttributes()) {
      Map<String,String> __this__attributes = new HashMap<String,String>();
      for (Map.Entry<String, String> other_element : other.attributes.entrySet()) {

        String other_element_key = other_element.getKey();
        String other_element_value = other_element.getValue();

        String __this__attributes_copy_key = other_element_key;

        String __this__attributes_copy_value = other_element_value;

        __this__attributes.put(__this__attributes_copy_key, __this__attributes_copy_value);
      }
      this.attributes = __this__attributes;
    }
    if (other.isSetParameters()) {
      List<Double> __this__parameters = new ArrayList<Double>();
      for (Double other_element : other.parameters) {
        __this__parameters.add(other_element);
      }
      this.parameters = __this__parameters;
    }
    if (other.isSetModel()) {
      this.model = other.model;
    }
  }

  @Override
  public QuestionUnit clone() {
    return new QuestionUnit(this);
  }

  /**
   * Question Description
   */
```

```
public String getDescription() {
  return this.description;
}

/**
 * Question Description
 */
public void setDescription(String description) {
  this.description = description;
}

public void unsetDescription() {
  this.description = null;
}

// Returns true if field description is set (has been asigned a value) and false otherwise
public boolean isSetDescription() {
  return this.description != null;
}

public void setDescriptionIsSet(boolean value) {
  if (!value) {
    this.description = null;
  }
}

public int getOptionsSize() {
  return (this.options == null) ? 0 : this.options.size();
}

public java.util.Iterator<String> getOptionsIterator() {
  return (this.options == null) ? null : this.options.iterator();
}

public void addToOptions(String elem) {
  if (this.options == null) {
    this.options = new ArrayList<String>();
  }
  this.options.add(elem);
}

/**
 * Question Options
 */
public List<String> getOptions() {
  return this.options;
}

/**
 * Question Options
 */
public void setOptions(List<String> options) {
  this.options = options;
}

public void unsetOptions() {
  this.options = null;
}
```

```
// Returns true if field options is set (has been asigned a value) and false otherwise
public boolean isSetOptions() {
  return this.options != null;
}

public void setOptionsIsSet(boolean value) {
  if (!value) {
    this.options = null;
  }
}

public int getAnswersSize() {
  return (this.answers == null) ? 0 : this.answers.size();
}

public java.util.Iterator<Integer> getAnswersIterator() {
  return (this.answers == null) ? null : this.answers.iterator();
}

public void addToAnswers(int elem) {
  if (this.answers == null) {
    this.answers = new HashSet<Integer>();
  }
  this.answers.add(elem);
}

/**
 * Question Answers
 */
public Set<Integer> getAnswers() {
  return this.answers;
}

/**
 * Question Answers
 */
public void setAnswers(Set<Integer> answers) {
  this.answers = answers;
}

public void unsetAnswers() {
  this.answers = null;
}

// Returns true if field answers is set (has been asigned a value) and false otherwise
public boolean isSetAnswers() {
  return this.answers != null;
}

public void setAnswersIsSet(boolean value) {
  if (!value) {
    this.answers = null;
  }
}

public int getAttributesSize() {
  return (this.attributes == null) ? 0 : this.attributes.size();
```

```
}

public void putToAttributes(String key, String val) {
  if (this.attributes == null) {
    this.attributes = new HashMap<String,String>();
  }
  this.attributes.put(key, val);
}

/**
 * Other Attributes
 */
public Map<String,String> getAttributes() {
  return this.attributes;
}

/**
 * Other Attributes
 */
public void setAttributes(Map<String,String> attributes) {
  this.attributes = attributes;
}

public void unsetAttributes() {
  this.attributes = null;
}

// Returns true if field attributes is set (has been asigned a value) and false otherwise
public boolean isSetAttributes() {
  return this.attributes != null;
}

public void setAttributesIsSet(boolean value) {
  if (!value) {
    this.attributes = null;
  }
}

public int getParametersSize() {
  return (this.parameters == null) ? 0 : this.parameters.size();
}

public java.util.Iterator<Double> getParametersIterator() {
  return (this.parameters == null) ? null : this.parameters.iterator();
}

public void addToParameters(double elem) {
  if (this.parameters == null) {
    this.parameters = new ArrayList<Double>();
  }
  this.parameters.add(elem);
}

/**
 * Calibrated Parameters
 */
public List<Double> getParameters() {
  return this.parameters;
```

```
}

/**
 * Calibrated Parameters
 */
public void setParameters(List<Double> parameters) {
  this.parameters = parameters;
}

public void unsetParameters() {
  this.parameters = null;
}

// Returns true if field parameters is set (has been asigned a value) and false otherwise
public boolean isSetParameters() {
  return this.parameters != null;
}

public void setParametersIsSet(boolean value) {
  if (!value) {
    this.parameters = null;
  }
}

/**
 * Model
 */
public String getModel() {
  return this.model;
}

/**
 * Model
 */
public void setModel(String model) {
  this.model = model;
}

public void unsetModel() {
  this.model = null;
}

// Returns true if field model is set (has been asigned a value) and false otherwise
public boolean isSetModel() {
  return this.model != null;
}

public void setModelIsSet(boolean value) {
  if (!value) {
    this.model = null;
  }
}

public void setFieldValue(int fieldID, Object value) {
  switch (fieldID) {
  case DESCRIPTION:
    if (value == null) {
      unsetDescription();
```

```
      } else {
        setDescription((String)value);
      }
      break;

    case OPTIONS:
      if (value == null) {
        unsetOptions();
      } else {
        setOptions((List<String>)value);
      }
      break;

    case ANSWERS:
      if (value == null) {
        unsetAnswers();
      } else {
        setAnswers((Set<Integer>)value);
      }
      break;

    case ATTRIBUTES:
      if (value == null) {
        unsetAttributes();
      } else {
        setAttributes((Map<String,String>)value);
      }
      break;

    case PARAMETERS:
      if (value == null) {
        unsetParameters();
      } else {
        setParameters((List<Double>)value);
      }
      break;

    case MODEL:
      if (value == null) {
        unsetModel();
      } else {
        setModel((String)value);
      }
      break;

    default:
      throw new IllegalArgumentException("Field " + fieldID + " doesn't exist!");
    }
  }

  public Object getFieldValue(int fieldID) {
    switch (fieldID) {
    case DESCRIPTION:
      return getDescription();

    case OPTIONS:
      return getOptions();
```

```
    case ANSWERS:
      return getAnswers();

    case ATTRIBUTES:
      return getAttributes();

    case PARAMETERS:
      return getParameters();

    case MODEL:
      return getModel();

    default:
      throw new IllegalArgumentException("Field " + fieldID + " doesn't exist!");
    }
  }

  // Returns true if field corresponding to fieldID is set (has been asigned a value) and false otherwise
  public boolean isSet(int fieldID) {
    switch (fieldID) {
    case DESCRIPTION:
      return isSetDescription();
    case OPTIONS:
      return isSetOptions();
    case ANSWERS:
      return isSetAnswers();
    case ATTRIBUTES:
      return isSetAttributes();
    case PARAMETERS:
      return isSetParameters();
    case MODEL:
      return isSetModel();
    default:
      throw new IllegalArgumentException("Field " + fieldID + " doesn't exist!");
    }
  }

  @Override
  public boolean equals(Object that) {
    if (that == null)
      return false;
    if (that instanceof QuestionUnit)
      return this.equals((QuestionUnit)that);
    return false;
  }

  public boolean equals(QuestionUnit that) {
    if (that == null)
      return false;

    boolean this_present_description = true && this.isSetDescription();
    boolean that_present_description = true && that.isSetDescription();
    if (this_present_description || that_present_description) {
      if (!(this_present_description && that_present_description))
        return false;
      if (!this.description.equals(that.description))
        return false;
    }
```

```
  boolean this_present_options = true && this.isSetOptions();
  boolean that_present_options = true && that.isSetOptions();
  if (this_present_options || that_present_options) {
    if (!(this_present_options && that_present_options))
      return false;
    if (!this.options.equals(that.options))
      return false;
  }

  boolean this_present_answers = true && this.isSetAnswers();
  boolean that_present_answers = true && that.isSetAnswers();
  if (this_present_answers || that_present_answers) {
    if (!(this_present_answers && that_present_answers))
      return false;
    if (!this.answers.equals(that.answers))
      return false;
  }

  boolean this_present_attributes = true && this.isSetAttributes();
  boolean that_present_attributes = true && that.isSetAttributes();
  if (this_present_attributes || that_present_attributes) {
    if (!(this_present_attributes && that_present_attributes))
      return false;
    if (!this.attributes.equals(that.attributes))
      return false;
  }

  boolean this_present_parameters = true && this.isSetParameters();
  boolean that_present_parameters = true && that.isSetParameters();
  if (this_present_parameters || that_present_parameters) {
    if (!(this_present_parameters && that_present_parameters))
      return false;
    if (!this.parameters.equals(that.parameters))
      return false;
  }

  boolean this_present_model = true && this.isSetModel();
  boolean that_present_model = true && that.isSetModel();
  if (this_present_model || that_present_model) {
    if (!(this_present_model && that_present_model))
      return false;
    if (!this.model.equals(that.model))
      return false;
  }

  return true;
}

@Override
public int hashCode() {
  HashCodeBuilder builder = new HashCodeBuilder();

  boolean present_description = true && (isSetDescription());
  builder.append(present_description);
  if (present_description)
    builder.append(description);
```

```
    boolean present_options = true && (isSetOptions());
    builder.append(present_options);
    if (present_options)
      builder.append(options);

    boolean present_answers = true && (isSetAnswers());
    builder.append(present_answers);
    if (present_answers)
      builder.append(answers);

    boolean present_attributes = true && (isSetAttributes());
    builder.append(present_attributes);
    if (present_attributes)
      builder.append(attributes);

    boolean present_parameters = true && (isSetParameters());
    builder.append(present_parameters);
    if (present_parameters)
      builder.append(parameters);

    boolean present_model = true && (isSetModel());
    builder.append(present_model);
    if (present_model)
      builder.append(model);

    return builder.toHashCode();
  }

  public void read(TProtocol iprot) throws TException {
    TField field;
    iprot.readStructBegin();
    while (true)
    {
      field = iprot.readFieldBegin();
      if (field.type == TType.STOP) {
        break;
      }
      switch (field.id)
      {
        case DESCRIPTION:
          if (field.type == TType.STRING) {
            this.description = iprot.readString();
          } else {
            TProtocolUtil.skip(iprot, field.type);
          }
          break;
        case OPTIONS:
          if (field.type == TType.LIST) {
            {
              TList _list0 = iprot.readListBegin();
              this.options = new ArrayList<String>(_list0.size);
              for (int _i1 = 0; _i1 < _list0.size; ++_i1)
              {
                String _elem2;
                _elem2 = iprot.readString();
                this.options.add(_elem2);
              }
              iprot.readListEnd();
```

```
      }
     } else {
      TProtocolUtil.skip(iprot, field.type);
     }
     break;
    case ANSWERS:
     if (field.type == TType.SET) {
      {
       TSet _set3 = iprot.readSetBegin();
       this.answers = new HashSet<Integer>(2*_set3.size);
       for (int _i4 = 0; _i4 < _set3.size; ++_i4)
       {
        int _elem5;
        _elem5 = iprot.readI32();
        this.answers.add(_elem5);
       }
       iprot.readSetEnd();
      }
     } else {
      TProtocolUtil.skip(iprot, field.type);
     }
     break;
    case ATTRIBUTES:
     if (field.type == TType.MAP) {
      {
       TMap _map6 = iprot.readMapBegin();
       this.attributes = new HashMap<String,String>(2*_map6.size);
       for (int _i7 = 0; _i7 < _map6.size; ++_i7)
       {
        String _key8;
        String _val9;
        _key8 = iprot.readString();
        _val9 = iprot.readString();
        this.attributes.put(_key8, _val9);
       }
       iprot.readMapEnd();
      }
     } else {
      TProtocolUtil.skip(iprot, field.type);
     }
     break;
    case PARAMETERS:
     if (field.type == TType.LIST) {
      {
       TList _list10 = iprot.readListBegin();
       this.parameters = new ArrayList<Double>(_list10.size);
       for (int _i11 = 0; _i11 < _list10.size; ++_i11)
       {
        double _elem12;
        _elem12 = iprot.readDouble();
        this.parameters.add(_elem12);
       }
       iprot.readListEnd();
      }
     } else {
      TProtocolUtil.skip(iprot, field.type);
     }
     break;
```

```
          case MODEL:
            if (field.type == TType.STRING) {
              this.model = iprot.readString();
            } else {
              TProtocolUtil.skip(iprot, field.type);
            }
            break;
          default:
            TProtocolUtil.skip(iprot, field.type);
            break;
        }
        iprot.readFieldEnd();
      }
      iprot.readStructEnd();


      // check for required fields of primitive type, which can't be checked in the validate method
      validate();
    }

    public void write(TProtocol oprot) throws TException {
      validate();

      oprot.writeStructBegin(STRUCT_DESC);
      if (this.description != null) {
        oprot.writeFieldBegin(DESCRIPTION_FIELD_DESC);
        oprot.writeString(this.description);
        oprot.writeFieldEnd();
      }
      if (this.options != null) {
        oprot.writeFieldBegin(OPTIONS_FIELD_DESC);
        {
          oprot.writeListBegin(new TList(TType.STRING, this.options.size()));
          for (String _iter13 : this.options)        {
            oprot.writeString(_iter13);
          }
          oprot.writeListEnd();
        }
        oprot.writeFieldEnd();
      }
      if (this.answers != null) {
        oprot.writeFieldBegin(ANSWERS_FIELD_DESC);
        {
          oprot.writeSetBegin(new TSet(TType.I32, this.answers.size()));
          for (int _iter14 : this.answers)        {
            oprot.writeI32(_iter14);
          }
          oprot.writeSetEnd();
        }
        oprot.writeFieldEnd();
      }
      if (this.attributes != null) {
        oprot.writeFieldBegin(ATTRIBUTES_FIELD_DESC);
        {
          oprot.writeMapBegin(new TMap(TType.STRING, TType.STRING, this.attributes.size()));
          for (Map.Entry<String, String> _iter15 : this.attributes.entrySet())        {
            oprot.writeString(_iter15.getKey());
            oprot.writeString(_iter15.getValue());
```

```
        }
        oprot.writeMapEnd();
      }
      oprot.writeFieldEnd();
    }
    if (this.parameters != null) {
      oprot.writeFieldBegin(PARAMETERS_FIELD_DESC);
      {
        oprot.writeListBegin(new TList(TType.DOUBLE, this.parameters.size()));
        for (double _iter16 : this.parameters)        {
          oprot.writeDouble(_iter16);
        }
        oprot.writeListEnd();
      }
      oprot.writeFieldEnd();
    }
    if (this.model != null) {
      oprot.writeFieldBegin(MODEL_FIELD_DESC);
      oprot.writeString(this.model);
      oprot.writeFieldEnd();
    }
    oprot.writeFieldStop();
    oprot.writeStructEnd();
  }

  @Override
  public String toString() {
    StringBuilder sb = new StringBuilder("QuestionUnit(");
    boolean first = true;

    sb.append("description:");
    if (this.description == null) {
      sb.append("null");
    } else {
      sb.append(this.description);
    }
    first = false;
    if (!first) sb.append(", ");
    sb.append("options:");
    if (this.options == null) {
      sb.append("null");
    } else {
      sb.append(this.options);
    }
    first = false;
    if (!first) sb.append(", ");
    sb.append("answers:");
    if (this.answers == null) {
      sb.append("null");
    } else {
      sb.append(this.answers);
    }
    first = false;
    if (!first) sb.append(", ");
    sb.append("attributes:");
    if (this.attributes == null) {
      sb.append("null");
    } else {
```

```
      sb.append(this.attributes);
    }
    first = false;
    if (!first) sb.append(", ");
    sb.append("parameters:");
    if (this.parameters == null) {
      sb.append("null");
    } else {
      sb.append(this.parameters);
    }
    first = false;
    if (!first) sb.append(", ");
    sb.append("model:");
    if (this.model == null) {
      sb.append("null");
    } else {
      sb.append(this.model);
    }
    first = false;
    sb.append(")");
    return sb.toString();
  }

  public void validate() throws TException {
    // check for required fields
    // check that fields of type enum have valid values
  }

}
```

TestUnit.java

TestUnit.java

```
/**
 * Autogenerated by Thrift
 *
 * DO NOT EDIT UNLESS YOU ARE SURE THAT YOU KNOW WHAT YOU ARE DOING
 */
package parsedquestion;

import org.apache.commons.lang.builder.HashCodeBuilder;
import java.util.List;
import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;
import java.util.Set;
import java.util.HashSet;
import java.util.Collections;

import org.apache.thrift.*;
import org.apache.thrift.meta_data.*;
import org.apache.thrift.protocol.*;

/**
 * Test unit.
 */
public class TestUnit implements TBase, java.io.Serializable, Cloneable {
  private static final TStruct STRUCT_DESC = new TStruct("TestUnit");
  private static final TField NAME_FIELD_DESC = new TField("name", TType.STRING, (short)1);
```

```
  private static final TField TARGETLENGTH_FIELD_DESC = new TField("targetlength", TType.I32,
(short)2);
  private static final TField ATTRIBUTES_FIELD_DESC = new TField("attributes", TType.MAP, (short)3);
  private static final TField QUESTIONS_FIELD_DESC = new TField("questions", TType.LIST, (short)4);

  /**
   * Test Name
   */
  public String name;
  public static final int NAME = 1;
  /**
   * Target Length
   */
  public int targetlength;
  public static final int TARGETLENGTH = 2;
  /**
   * Other Attributes
   */
  public Map<String,String> attributes;
  public static final int ATTRIBUTES = 3;
  /**
   * Questions
   */
  public List<QuestionUnit> questions;
  public static final int QUESTIONS = 4;

  private final Isset __isset = new Isset();
  private static final class Isset implements java.io.Serializable {
    public boolean targetlength = false;
  }

  public static final Map<Integer, FieldMetaData> metaDataMap = Collections.unmodifiableMap(new
HashMap<Integer, FieldMetaData>() {{
    put(NAME, new FieldMetaData("name", TFieldRequirementType.DEFAULT,
        new FieldValueMetaData(TType.STRING)));
    put(TARGETLENGTH, new FieldMetaData("targetlength", TFieldRequirementType.DEFAULT,
        new FieldValueMetaData(TType.I32)));
    put(ATTRIBUTES, new FieldMetaData("attributes", TFieldRequirementType.DEFAULT,
        new MapMetaData(TType.MAP,
            new FieldValueMetaData(TType.STRING),
            new FieldValueMetaData(TType.STRING))));
    put(QUESTIONS, new FieldMetaData("questions", TFieldRequirementType.DEFAULT,
        new ListMetaData(TType.LIST,
            new StructMetaData(TType.STRUCT, QuestionUnit.class))));
  }});

  static {
    FieldMetaData.addStructMetaDataMap(TestUnit.class, metaDataMap);
  }

  public TestUnit() {
  }

  public TestUnit(
    String name,
    int targetlength,
    Map<String,String> attributes,
    List<QuestionUnit> questions)
```

```
{
  this();
  this.name = name;
  this.targetlength = targetlength;
  this.__isset.targetlength = true;
  this.attributes = attributes;
  this.questions = questions;
}

/**
 * Performs a deep copy on <i>other</i>.
 */
public TestUnit(TestUnit other) {
  if (other.isSetName()) {
    this.name = other.name;
  }
  __isset.targetlength = other.__isset.targetlength;
  this.targetlength = other.targetlength;
  if (other.isSetAttributes()) {
    Map<String,String> __this__attributes = new HashMap<String,String>();
    for (Map.Entry<String, String> other_element : other.attributes.entrySet()) {

      String other_element_key = other_element.getKey();
      String other_element_value = other_element.getValue();

      String __this__attributes_copy_key = other_element_key;

      String __this__attributes_copy_value = other_element_value;

      __this__attributes.put(__this__attributes_copy_key, __this__attributes_copy_value);
    }
    this.attributes = __this__attributes;
  }
  if (other.isSetQuestions()) {
    List<QuestionUnit> __this__questions = new ArrayList<QuestionUnit>();
    for (QuestionUnit other_element : other.questions) {
      __this__questions.add(new QuestionUnit(other_element));
    }
    this.questions = __this__questions;
  }
}

@Override
public TestUnit clone() {
  return new TestUnit(this);
}

/**
 * Test Name
 */
public String getName() {
  return this.name;
}

/**
 * Test Name
 */
public void setName(String name) {
```

```
  this.name = name;
}

public void unsetName() {
  this.name = null;
}

// Returns true if field name is set (has been asigned a value) and false otherwise
public boolean isSetName() {
  return this.name != null;
}

public void setNameIsSet(boolean value) {
  if (!value) {
    this.name = null;
  }
}

/**
 * Target Length
 */
public int getTargetlength() {
  return this.targetlength;
}

/**
 * Target Length
 */
public void setTargetlength(int targetlength) {
  this.targetlength = targetlength;
  this.__isset.targetlength = true;
}

public void unsetTargetlength() {
  this.__isset.targetlength = false;
}

// Returns true if field targetlength is set (has been asigned a value) and false otherwise
public boolean isSetTargetlength() {
  return this.__isset.targetlength;
}

public void setTargetlengthIsSet(boolean value) {
  this.__isset.targetlength = value;
}

public int getAttributesSize() {
  return (this.attributes == null) ? 0 : this.attributes.size();
}

public void putToAttributes(String key, String val) {
  if (this.attributes == null) {
    this.attributes = new HashMap<String,String>();
  }
  this.attributes.put(key, val);
}

/**
```

```
 * Other Attributes
 */
public Map<String,String> getAttributes() {
  return this.attributes;
}

/**
 * Other Attributes
 */
public void setAttributes(Map<String,String> attributes) {
  this.attributes = attributes;
}

public void unsetAttributes() {
  this.attributes = null;
}

// Returns true if field attributes is set (has been asigned a value) and false otherwise
public boolean isSetAttributes() {
  return this.attributes != null;
}

public void setAttributesIsSet(boolean value) {
  if (!value) {
    this.attributes = null;
  }
}

public int getQuestionsSize() {
  return (this.questions == null) ? 0 : this.questions.size();
}

public java.util.Iterator<QuestionUnit> getQuestionsIterator() {
  return (this.questions == null) ? null : this.questions.iterator();
}

public void addToQuestions(QuestionUnit elem) {
  if (this.questions == null) {
    this.questions = new ArrayList<QuestionUnit>();
  }
  this.questions.add(elem);
}

/**
 * Questions
 */
public List<QuestionUnit> getQuestions() {
  return this.questions;
}

/**
 * Questions
 */
public void setQuestions(List<QuestionUnit> questions) {
  this.questions = questions;
}

public void unsetQuestions() {
```

```
  this.questions = null;
}

// Returns true if field questions is set (has been asigned a value) and false otherwise
public boolean isSetQuestions() {
  return this.questions != null;
}

public void setQuestionsIsSet(boolean value) {
  if (!value) {
    this.questions = null;
  }
}

public void setFieldValue(int fieldID, Object value) {
  switch (fieldID) {
  case NAME:
    if (value == null) {
      unsetName();
    } else {
      setName((String)value);
    }
    break;

  case TARGETLENGTH:
    if (value == null) {
      unsetTargetlength();
    } else {
      setTargetlength((Integer)value);
    }
    break;

  case ATTRIBUTES:
    if (value == null) {
      unsetAttributes();
    } else {
      setAttributes((Map<String,String>)value);
    }
    break;

  case QUESTIONS:
    if (value == null) {
      unsetQuestions();
    } else {
      setQuestions((List<QuestionUnit>)value);
    }
    break;

  default:
    throw new IllegalArgumentException("Field " + fieldID + " doesn't exist!");
  }
}

public Object getFieldValue(int fieldID) {
  switch (fieldID) {
  case NAME:
    return getName();
```

```
    case TARGETLENGTH:
      return new Integer(getTargetlength());

    case ATTRIBUTES:
      return getAttributes();

    case QUESTIONS:
      return getQuestions();

    default:
      throw new IllegalArgumentException("Field " + fieldID + " doesn't exist!");
    }
  }

  // Returns true if field corresponding to fieldID is set (has been asigned a value) and false otherwise
  public boolean isSet(int fieldID) {
    switch (fieldID) {
    case NAME:
      return isSetName();
    case TARGETLENGTH:
      return isSetTargetlength();
    case ATTRIBUTES:
      return isSetAttributes();
    case QUESTIONS:
      return isSetQuestions();
    default:
      throw new IllegalArgumentException("Field " + fieldID + " doesn't exist!");
    }
  }

  @Override
  public boolean equals(Object that) {
    if (that == null)
      return false;
    if (that instanceof TestUnit)
      return this.equals((TestUnit)that);
    return false;
  }

  public boolean equals(TestUnit that) {
    if (that == null)
      return false;

    boolean this_present_name = true && this.isSetName();
    boolean that_present_name = true && that.isSetName();
    if (this_present_name || that_present_name) {
      if (!(this_present_name && that_present_name))
        return false;
      if (!this.name.equals(that.name))
        return false;
    }

    boolean this_present_targetlength = true;
    boolean that_present_targetlength = true;
    if (this_present_targetlength || that_present_targetlength) {
      if (!(this_present_targetlength && that_present_targetlength))
        return false;
      if (this.targetlength != that.targetlength)
```

```
      return false;
    }

    boolean this_present_attributes = true && this.isSetAttributes();
    boolean that_present_attributes = true && that.isSetAttributes();
    if (this_present_attributes || that_present_attributes) {
      if (!(this_present_attributes && that_present_attributes))
        return false;
      if (!this.attributes.equals(that.attributes))
        return false;
    }

    boolean this_present_questions = true && this.isSetQuestions();
    boolean that_present_questions = true && that.isSetQuestions();
    if (this_present_questions || that_present_questions) {
      if (!(this_present_questions && that_present_questions))
        return false;
      if (!this.questions.equals(that.questions))
        return false;
    }

    return true;
  }

  @Override
  public int hashCode() {
    HashCodeBuilder builder = new HashCodeBuilder();

    boolean present_name = true && (isSetName());
    builder.append(present_name);
    if (present_name)
      builder.append(name);

    boolean present_targetlength = true;
    builder.append(present_targetlength);
    if (present_targetlength)
      builder.append(targetlength);

    boolean present_attributes = true && (isSetAttributes());
    builder.append(present_attributes);
    if (present_attributes)
      builder.append(attributes);

    boolean present_questions = true && (isSetQuestions());
    builder.append(present_questions);
    if (present_questions)
      builder.append(questions);

    return builder.toHashCode();
  }

  public void read(TProtocol iprot) throws TException {
    TField field;
    iprot.readStructBegin();
    while (true)
    {
      field = iprot.readFieldBegin();
      if (field.type == TType.STOP) {
```

```
      break;
    }
    switch (field.id)
    {
      case NAME:
        if (field.type == TType.STRING) {
          this.name = iprot.readString();
        } else {
          TProtocolUtil.skip(iprot, field.type);
        }
        break;
      case TARGETLENGTH:
        if (field.type == TType.I32) {
          this.targetlength = iprot.readI32();
          this.__isset.targetlength = true;
        } else {
          TProtocolUtil.skip(iprot, field.type);
        }
        break;
      case ATTRIBUTES:
        if (field.type == TType.MAP) {
          {
            TMap _map17 = iprot.readMapBegin();
            this.attributes = new HashMap<String,String>(2*_map17.size);
            for (int _i18 = 0; _i18 < _map17.size; ++_i18)
            {
              String _key19;
              String _val20;
              _key19 = iprot.readString();
              _val20 = iprot.readString();
              this.attributes.put(_key19, _val20);
            }
            iprot.readMapEnd();
          }
        } else {
          TProtocolUtil.skip(iprot, field.type);
        }
        break;
      case QUESTIONS:
        if (field.type == TType.LIST) {
          {
            TList _list21 = iprot.readListBegin();
            this.questions = new ArrayList<QuestionUnit>(_list21.size);
            for (int _i22 = 0; _i22 < _list21.size; ++_i22)
            {
              QuestionUnit _elem23;
              _elem23 = new QuestionUnit();
              _elem23.read(iprot);
              this.questions.add(_elem23);
            }
            iprot.readListEnd();
          }
        } else {
          TProtocolUtil.skip(iprot, field.type);
        }
        break;
      default:
        TProtocolUtil.skip(iprot, field.type);
```

```
          break;
        }
        iprot.readFieldEnd();
      }
      iprot.readStructEnd();


      // check for required fields of primitive type, which can't be checked in the validate method
      validate();
    }

    public void write(TProtocol oprot) throws TException {
      validate();

      oprot.writeStructBegin(STRUCT_DESC);
      if (this.name != null) {
        oprot.writeFieldBegin(NAME_FIELD_DESC);
        oprot.writeString(this.name);
        oprot.writeFieldEnd();
      }
      oprot.writeFieldBegin(TARGETLENGTH_FIELD_DESC);
      oprot.writeI32(this.targetlength);
      oprot.writeFieldEnd();
      if (this.attributes != null) {
        oprot.writeFieldBegin(ATTRIBUTES_FIELD_DESC);
        {
          oprot.writeMapBegin(new TMap(TType.STRING, TType.STRING, this.attributes.size()));
          for (Map.Entry<String, String> _iter24 : this.attributes.entrySet())        {
            oprot.writeString(_iter24.getKey());
            oprot.writeString(_iter24.getValue());
          }
          oprot.writeMapEnd();
        }
        oprot.writeFieldEnd();
      }
      if (this.questions != null) {
        oprot.writeFieldBegin(QUESTIONS_FIELD_DESC);
        {
          oprot.writeListBegin(new TList(TType.STRUCT, this.questions.size()));
          for (QuestionUnit _iter25 : this.questions)        {
            _iter25.write(oprot);
          }
          oprot.writeListEnd();
        }
        oprot.writeFieldEnd();
      }
      oprot.writeFieldStop();
      oprot.writeStructEnd();
    }

    @Override
    public String toString() {
      StringBuilder sb = new StringBuilder("TestUnit(");
      boolean first = true;

      sb.append("name:");
      if (this.name == null) {
        sb.append("null");
```

```
    } else {
      sb.append(this.name);
    }
    first = false;
    if (!first) sb.append(", ");
    sb.append("targetlength:");
    sb.append(this.targetlength);
    first = false;
    if (!first) sb.append(", ");
    sb.append("attributes:");
    if (this.attributes == null) {
      sb.append("null");
    } else {
      sb.append(this.attributes);
    }
    first = false;
    if (!first) sb.append(", ");
    sb.append("questions:");
    if (this.questions == null) {
      sb.append("null");
    } else {
      sb.append(this.questions);
    }
    first = false;
    sb.append(")");
    return sb.toString();
  }

  public void validate() throws TException {
    // check for required fields
    // check that fields of type enum have valid values
  }

}
```

Appendix – A2 Source Code of the Proposed Adaptive e-Assessment Method

This section includes the source code of the proposed adaptive e-Assessment method.

The xiaobin.testing package

AdaptiveTestManager.java

AdaptiveTestManager.java

```
package xiaobin.testing;

import java.util.List;
import java.util.ArrayList;
import java.util.Set;
import java.util.HashSet;

/**
 * Maintains an in memory list of questions in the system and administers
 * adaptive test.
 * @author Ben
 */
public class AdaptiveTestManager {
    /** Sequence number to use for generating IDs. */
    private static int idSequence = 0;

    /** The test information indicator to terminate the adaptive test. */
    private static double MAX_TEST_INFORMATION = 10.0;
    /** Current question. */
    private static QuestionUnit currentQuestion;
    /** Current ability estimate. */
    private static double currentAbility;

    /** Storage for all known questions. */
    private static List<QuestionUnit> questions = new ArrayList<QuestionUnit> ();

    static {
        QuestionUnit question = new QuestionUnit();
        question.setDescription("Q1: Although all of the following are possible, " +
                "what is the primary focus of most home computers? ");
        List<String> options = new ArrayList<String> ();
        options.add("a) database development" );
        options.add("b) Internet and email access" );
        options.add("c) typing practice" );
        options.add("d) programming language instruction" );
        question.setOptions(options);
        Set<Integer> answers = new HashSet<Integer> ();
        answers.add(2);
        question.setAnswers(answers);
        List<Double> parameters = new ArrayList<Double> ();
        double a = 1.0;
        double b = -1.31;
        double c = 0.25;
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);
```

```
question = new QuestionUnit();
question.setDescription("Q4: Which part of the OS, loaded into memory at " +
      "boot time, controls memory allocation?  ");
options = new ArrayList<String> ();
options.add("a) user interface" );
options.add("b) application software" );
options.add("c) kernel" );
options.add("d) file system" );
question.setOptions(options);
answers = new HashSet<Integer> ();
answers.add(3);
question.setAnswers(answers);
parameters = new ArrayList<Double> ();
a = 1.0;
b = -0.3;
c = 0.25;
parameters.add(a);
parameters.add(b);
parameters.add(c);
question.setParameters(parameters);
saveOrUpdateInternal(question);

question = new QuestionUnit();
question.setDescription("Q5: What is a major disadvantage of a GUI?");
options = new ArrayList<String> ();
options.add("a) It requires an understanding of commands." );
options.add("b) It requires more memory and storage space." );
options.add("c) It is more difficult to use." );
options.add("d) It is difficult to install." );
question.setOptions(options);
answers = new HashSet<Integer> ();
answers.add(2);
question.setAnswers(answers);
parameters = new ArrayList<Double> ();
a = 1.0;
b = -3.62;
c = 0.25;
parameters.add(a);
parameters.add(b);
parameters.add(c);
question.setParameters(parameters);
saveOrUpdateInternal(question);

question = new QuestionUnit();
question.setDescription("Q6: What is a major advantage of a CLI?");
options = new ArrayList<String> ();
options.add("a) It is easier for inexperienced individuals to use." );
options.add("b) A mouse or other point-and-click device may be used to input data." );
options.add("c) It takes less processing power to run." );
options.add("d) It can be added later if disk space is limited." );
question.setOptions(options);
answers = new HashSet<Integer> ();
answers.add(3);
question.setAnswers(answers);
parameters = new ArrayList<Double> ();
a = 1.0;
b = 1.0;
```

```
        c = 0.25;
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);

        question = new QuestionUnit();
        question.setDescription("Q7: Which acronyms describe the two general " +
              "categories of OS user interfaces?");
        options = new ArrayList<String> ();
        options.add("a) DOS and UNIX" );
        options.add("b) DOS and GUI" );
        options.add("c) UNIX and GUI" );
        options.add("d) CLI and GUI" );
        question.setOptions(options);
        answers = new HashSet<Integer> ();
        answers.add(4);
        question.setAnswers(answers);
        parameters = new ArrayList<Double> ();
        a = 1.0;
        b = 1.14;
        c = 0.25;
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);

        question = new QuestionUnit();
        question.setDescription("Q8: What is a beneficial feature of " +
              "preemptive multitasking?");
        options = new ArrayList<String> ();
        options.add("a) Applications share the use of the processor by time slicing. " );
        options.add("b) The processor dynamically adjusts the amount of memory needed by the application
programs." );
        options.add("c) Application programs share memory addresses and exchange information." );
        options.add("d) The operating system controls the allocation of processor time." );
        question.setOptions(options);
        answers = new HashSet<Integer> ();
        answers.add(4);
        question.setAnswers(answers);
        parameters = new ArrayList<Double> ();
        a = 1.0;
        b = 2.24;
        c = 0.25;
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);

        question = new QuestionUnit();
        question.setDescription("Q9: Which device is used to input data from a CLI?");
        options = new ArrayList<String> ();
        options.add("a) mouse" );
        options.add("b) keyboard" );
        options.add("c) microphone" );
```

```
options.add("d) monitor" );
question.setOptions(options);
answers = new HashSet<Integer> ();
answers.add(2);
question.setAnswers(answers);
parameters = new ArrayList<Double> ();
a = 1.0;
b = -1.31;
c = 0.25;
parameters.add(a);
parameters.add(b);
parameters.add(c);
question.setParameters(parameters);
saveOrUpdateInternal(question);

question = new QuestionUnit();
question.setDescription("Q10: What is the basic unit of logical storage on a FAT formatted disk?");
options = new ArrayList<String> ();
options.add("a) disk file allocation tables" );
options.add("b) disk directories" );
options.add("c) disk clusters" );
options.add("d) disk folders" );
question.setOptions(options);
answers = new HashSet<Integer> ();
answers.add(3);
question.setAnswers(answers);
parameters = new ArrayList<Double> ();
a = 1.0;
b = 1.95;
c = 0.25;
parameters.add(a);
parameters.add(b);
parameters.add(c);
question.setParameters(parameters);
saveOrUpdateInternal(question);

question = new QuestionUnit();
question.setDescription("Q11: Why was the FAT32 file system developed?");
options = new ArrayList<String> ();
options.add("a) to allow for the use of file names of 32 characters" );
options.add("b) to allow for the creation and storage of 32-bit file sizes" );
options.add("c) to allow for the use of smaller cluster sizes on large disks" );
options.add("d) to allow for a longer format for specifying the path of a file" );
question.setOptions(options);
answers = new HashSet<Integer> ();
answers.add(3);
question.setAnswers(answers);
parameters = new ArrayList<Double> ();
a = 1.0;
b = 2.33;
c = 0.25;
parameters.add(a);
parameters.add(b);
parameters.add(c);
question.setParameters(parameters);
saveOrUpdateInternal(question);

question = new QuestionUnit();
```

```
        question.setDescription("Q12: Caldera, Slackware, Suse, and Debian are all distributions of which
NOS?");
        options = new ArrayList<String> ();
        options.add("a) UNIX" );
        options.add("b) Solaris" );
        options.add("c) Linux" );
        options.add("d) OS X" );
        question.setOptions(options);
        answers = new HashSet<Integer> ();
        answers.add(3);
        question.setAnswers(answers);
        parameters = new ArrayList<Double> ();
        a = 1.0;
        b = 0.06;
        c = 0.25;
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);

        question = new QuestionUnit();
        question.setDescription("Q13: What is meant by the statement, \"We have to support our legacy
system\"?");
        options = new ArrayList<String> ();
        options.add("a) Support must be provided for the newest Linux system." );
        options.add("b) Support must be provided for the system called legacy." );
        options.add("c) Support must be provided for the newest Microsoft system." );
        options.add("d) Support must be provided for outdated software." );
        question.setOptions(options);
        answers = new HashSet<Integer> ();
        answers.add(4);
        question.setAnswers(answers);
        parameters = new ArrayList<Double> ();
        a = 1.0;
        b = 0.76;
        c = 0.25;
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);

        question = new QuestionUnit();
        question.setDescription("Q14: Which of the following is a characteristic of cooperative multitasking?");
        options = new ArrayList<String> ();
        options.add("a) The operating system controls the allocation of processor time." );
        options.add("b) Programs run in their own separate address spaces." );
        options.add("c) One crashed program will not affect other running programs." );
        options.add("d) Applications share the processor using time-slicing." );
        question.setOptions(options);
        answers = new HashSet<Integer> ();
        answers.add(4);
        question.setAnswers(answers);
        parameters = new ArrayList<Double> ();
        a = 1.0;
        b = 1.21;
        c = 0.25;
```

```
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);

        question = new QuestionUnit();
        question.setDescription("Q16: Which component is supported in Windows XP Professional but is
unavailable in the XP Home Edition?");
        options = new ArrayList<String> ();
        options.add("a) remote administration" );
        options.add("b) remote access" );
        options.add("c) multiple user accounts" );
        options.add("d) file security and resource sharing" );
        question.setOptions(options);
        answers = new HashSet<Integer> ();
        answers.add(2);
        question.setAnswers(answers);
        parameters = new ArrayList<Double> ();
        a = 1.0;
        b = 2.22;
        c = 0.25;
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);

        question = new QuestionUnit();
        question.setDescription("Q17: A user needs to know which version of the Windows operating system is
installed. " +
                "Which tab from the System Properties window should they click on to obtain that information?");
        options = new ArrayList<String> ();
        options.add("a) General" );
        options.add("b) Environment" );
        options.add("c) My computer" );
        options.add("d) Settings" );
        question.setOptions(options);
        answers = new HashSet<Integer> ();
        answers.add(1);
        question.setAnswers(answers);
        parameters = new ArrayList<Double> ();
        a = 1.0;
        b = 0.32;
        c = 0.25;
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);

        question = new QuestionUnit();
        question.setDescription("Q18: Which command is entered in the Run dialog box of a Windows XP
system to open a DOS command prompt window?");
        options = new ArrayList<String> ();
        options.add("a) run" );
        options.add("b) start" );
        options.add("c) cmd" );
```

```
        options.add("d) command" );
        question.setOptions(options);
        answers = new HashSet<Integer> ();
        answers.add(3);
        question.setAnswers(answers);
        parameters = new ArrayList<Double> ();
        a = 1.0;
        b = -0.24;
        c = 0.25;
        parameters.add(a);
        parameters.add(b);
        parameters.add(c);
        question.setParameters(parameters);
        saveOrUpdateInternal(question);
    }

    /** Initiates adaptive test with initial ability 0.0. */
    private static AdaptiveModel adaptiveTest = new AdaptiveModel (currentAbility, questions);

    /** Gets the question with the corresponding ID, or null if it does not exist. */
    public QuestionUnit getQuestion(int id) {
        return questions.get(id);
    }

    /** Returns a sorted list of all questions in the system. */
    public List<QuestionUnit> getAllQuestions() {
        return questions;
    }

    /** Gets the currently best question or null if it does not exist. */
    public QuestionUnit getCurrentQuestion() {
        int indexOfBestQuestion = adaptiveTest.selectBestQuestion();
        //used up all accessible questions
        if (indexOfBestQuestion == -1) {
            return null;
        }
        currentQuestion = adaptiveTest.getQuestion(indexOfBestQuestion);
        return currentQuestion;
    }

    /** Gets the current ability estimate. */
    public double getCurrentAbility() {
        this.currentAbility = adaptiveTest.getCurrentAbility();
        return currentAbility;
    }

    /** Gets the answers and update the test status. */
    public void getAnswersAndUpdate(Set<Integer> answers) {
        int correctness;
        if (currentQuestion.compareAnswers(answers)) {
            correctness = 1;
        }else {
            correctness = 0;
        }
        adaptiveTest.addQuestionLog(currentQuestion);
        adaptiveTest.addResponseLog(correctness);
        adaptiveTest.updateEstimatedAbility();
    }
```

```
    /** Tests if the test information gathered is greater than the preset MAX_TEST_INOFRMATION */
    public boolean isTestInformationEnough() {
        if (adaptiveTest.getTestInformation() > MAX_TEST_INFORMATION) {
            return true;
        }else {
            return false;
        }
    }

    /** Updates an existing question, or saves a new question if the question is a new one. */
    public void saveOrUpdate(QuestionUnit question) {
        saveOrUpdateInternal(question);
    }

    /** Static helper so that it can be used both by the instance method, and in static init. */
    private static void saveOrUpdateInternal(QuestionUnit question) {
        if (question.getId() == null) {
            question.setId(idSequence++);
        }
        questions.add(question);
    }

}
```

Eval.java

Eval.java

```
package xiaobin.testing;

import java.lang.reflect.Method;
import java.util.Vector;

/**
 * contains mathematical functions for calculating caculus
 * @author Ben
 */
public class Eval implements IEval {

    private boolean shouldContinue = true;
    private Vector stackOpnd = null;
    private Vector stackOptr = null;
    private Vector checkStr = null;
    private int numLength = 0;

    public Eval() {
    }

    private String trimAndCheck(String str) throws Exception {
        String temp = "";
        shouldContinue = true;
        for (int i = 0; i < str.length(); i++) {
            char xChar = str.charAt(i);
            if (xChar >= '0' && xChar <= '9' || xChar == '(' || xChar == ')' || xChar == '+' || xChar == '-' || xChar ==
'*' || xChar == '/' || xChar == '.') {
                temp = temp + xChar;
            } else if (xChar != ' ') {
                temp = "";
                shouldContinue = false;
                throw new Exception("Invalid expression format!(Invalid operator " + xChar + ")");
```

```
            }
        }
        return temp;
    }

    private String getNumber(String str) throws Exception {
        String tempNumber = "";
        numLength = 0;
        int count = 0;
        shouldContinue = true;
        for (int i = 0; i < str.length(); i++) {
            if (str.charAt(i) >= '0' && str.charAt(i) <= '9' || str.charAt(i) == '.') {
                if (str.charAt(i) == '.') {
                    count++;
                }
                if (count <= 1) {
                    tempNumber = tempNumber + str.charAt(i);
                } else {
                    shouldContinue = false;
                    throw new Exception("Invalid expression format!(Numeric error)");
                }
                numLength++;
            } else {
                break;
            }
        }
        return tempNumber;
    }

    /**
     * compute basic arithmetic operations only (brackets, + - * /)
     * @param expression
     * advanced mathematical functions should not be included
     * @return the result of the expression after calculation
     * @throws Exception
     **/
    public double eval(String expression) throws Exception {
        if (shouldContinue) {
            stackOpnd = new Vector();
            stackOptr = new Vector();
            stackOptr.addElement('#');
            checkStr = new Vector();
            String tempStr = scientificExpressionToNormal("(" + expression + ")").replace("-(", "-1*(");
            tempStr = trimAndCheck(tempStr);
            while (tempStr.length() > 0) {
                if (tempStr.charAt(0) == '(' && (tempStr.charAt(1) != '+' && tempStr.charAt(1) != '-') ||
tempStr.charAt(0) == ')' || tempStr.charAt(0) == '+' || tempStr.charAt(0) == '-' || tempStr.charAt(0) == '*' ||
tempStr.charAt(0) == '/') {
                    checkStr.addElement(tempStr.charAt(0));
                    tempStr = tempStr.substring(1, tempStr.length());
                } else if (tempStr.charAt(0) == '(' && (tempStr.charAt(1) == '+' || tempStr.charAt(1) == '-')) {
                    checkStr.addElement(tempStr.charAt(0));
                    tempStr = tempStr.substring(1, tempStr.length());
                    if (tempStr.charAt(0) == '+') {
                        tempStr = tempStr.substring(1, tempStr.length());
                        checkStr.addElement(getNumber(tempStr));
                        tempStr = tempStr.substring(numLength, tempStr.length());
                    } else if (tempStr.charAt(0) == '-') {
```

```
                String tempStrNum = "-";
                tempStr = tempStr.substring(1, tempStr.length());
                tempStrNum = tempStrNum + getNumber(tempStr);
                double tempDouble = Double.parseDouble(tempStrNum);
                checkStr.addElement(tempDouble);
                tempStr = tempStr.substring(numLength, tempStr.length());
             }
          } else if (tempStr.charAt(0) >= '0' && tempStr.charAt(0) <= '9') {
             checkStr.addElement(getNumber(tempStr));
             tempStr = tempStr.substring(numLength, tempStr.length());
          }
       }
       try {
          while (checkStr.size() > 0) {
             if (checkStr.get(0).toString().charAt(0) >= '0' && checkStr.get(0).toString().charAt(0) <= '9' ||
checkStr.get(0).toString().charAt(0) == '-' && checkStr.get(0).toString().length() > 1) {
                stackOpnd.add(0, checkStr.get(0));
                checkStr.removeElementAt(0);
             } else {
                switch (compareOptr(stackOptr.get(0).toString().charAt(
                      0), checkStr.get(0).toString().charAt(0))) {
                   case '<':
                      stackOptr.add(0, checkStr.get(0));
                      checkStr.removeElementAt(0);
                      break;
                   case '=':
                      stackOptr.removeElementAt(0);
                      checkStr.removeElementAt(0);
                      break;
                   case '>':
                      char theta = stackOptr.remove(0).toString().charAt(
                            0);
                      double b = Double.parseDouble(stackOpnd.remove(0).toString());
                      double a = Double.parseDouble(stackOpnd.remove(0).toString());
                      stackOpnd.add(0, operate(a, theta, b));
                      break;
                }
             }
          }
          shouldContinue = true;
          return Double.parseDouble(stackOpnd.remove(0).toString());
       } catch (Exception ex) {
          throw new Exception("Invalid expression!(logic error)");
       }
    } else {
       shouldContinue = true;
       return 0.0;
    }
 }

 private double operate(double a, char x, double b) {
    switch (x) {
       case '+':
          return a + b;
       case '-':
          return a - b;
       case '*':
          return a * b;
```

```
            case '/':
                return a / b;
        }
        return 0.0;
    }

    private char compareOptr(char first, char second) {
        switch (first) {
            case '+':
                switch (second) {
                    case '+':
                        return '>';
                    case '-':
                        return '>';
                    case '*':
                        return '<';
                    case '/':
                        return '<';
                    case '(':
                        return '<';
                    case ')':
                        return '>';
                }
                ;
                break;
            case '-':
                switch (second) {
                    case '+':
                        return '>';
                    case '-':
                        return '>';
                    case '*':
                        return '<';
                    case '/':
                        return '<';
                    case '(':
                        return '<';
                    case ')':
                        return '>';
                }
                ;
                break;
            case '*':
                switch (second) {
                    case '+':
                        return '>';
                    case '-':
                        return '>';
                    case '*':
                        return '>';
                    case '/':
                        return '>';
                    case '(':
                        return '<';
                    case ')':
                        return '>';
                }
                ;
```

```
            break;
        case '/':
            switch (second) {
                case '+':
                    return '>';
                case '-':
                    return '>';
                case '*':
                    return '>';
                case '/':
                    return '>';
                case '(':
                    return '<';
                case ')':
                    return '>';
            }
            ;
            break;
        case '(':
            switch (second) {
                case '+':
                    return '<';
                case '-':
                    return '<';
                case '*':
                    return '<';
                case '/':
                    return '<';
                case '(':
                    return '<';
                case ')':
                    return '=';
            }
            ;
            break;
        case ')':
            switch (second) {
                case '+':
                    return '>';
                case '-':
                    return '>';
                case '*':
                    return '>';
                case '/':
                    return '>';
                case ')':
                    return '>';
            }
            ;
            break;
        case '#':
            return '<';
    }
    return 'x';
}

private String getPartResultWithOneParameter(String expression,
        String functionName) throws Exception {
```

```
        String result = null;

        Method[] m = Math.class.getMethods();
        for (int i = 0; i < m.length; i++) {
            if (m[i].getName().equals(functionName) &&
m[i].getReturnType().getName().toLowerCase().equals(
                    "double")) {
                result = m[i].invoke(null,
                        new Object[]{Double.parseDouble(expression)}).toString();
                break;
            }
        }
        return "(" + result + ")";
    }

    private String getPartResultWithTwoParameter(String expression,
            String functionName) throws Exception {
        String result = null;
        String[] tempParameters = expression.replace(" ", "").split(",");
        double[] parameters = new double[2];
        parameters[0] = Double.parseDouble(tempParameters[0]);
        parameters[1] = Double.parseDouble(tempParameters[1]);

        Method[] m = Math.class.getMethods();
        for (int i = 0; i < m.length; i++) {
            if (m[i].getName().equals(functionName) &&
m[i].getReturnType().getName().toLowerCase().equals(
                    "double")) {
                result = m[i].invoke(null,
                        new Object[]{parameters[0], parameters[1]}).toString();
                break;
            }
        }
        return "(" + result + ")";
    }

    private String[] splitStr(String expression, String functionName)
            throws Exception {
        String[] result = new String[3];
        int tempIntB = expression.indexOf(functionName);
        int beginPosition = -1;
        int endPosition = -1;

        if (tempIntB >= 0) {
            beginPosition = expression.indexOf("(", tempIntB);
            int counter = 0;
            for (int i = beginPosition; i < expression.length(); i++) {
                if (expression.charAt(i) == '(') {
                    counter++;
                } else if (expression.charAt(i) == ')') {
                    counter--;
                }
                if (counter == 0) {
                    endPosition = i;
                    break;
                }
            }
        }
```

```
        result[0] = tempIntB == 0 ? "" : expression.substring(0, tempIntB);
        result[1] = expression.substring(beginPosition + 1, endPosition);
        result[2] = expression.substring(endPosition + 1);

        return result;
    }

    private String getMathFunctionName(String expression) {

        if (expression.indexOf("abs") >= 0) {
            return "abs";
        }
        if (expression.indexOf("acos") >= 0) {
            return "acos";
        }
        if (expression.indexOf("asin") >= 0) {
            return "asin";
        }
        if (expression.indexOf("atan") >= 0) {
            return "atan";
        }
        if (expression.indexOf("cbrt") >= 0) {
            return "cbrt";
        }
        if (expression.indexOf("ceil") >= 0) {
            return "ceil";
        }
        if (expression.indexOf("cos") >= 0) {
            return "cos";
        }
        if (expression.indexOf("cosh") >= 0) {
            return "cosh";
        }
        if (expression.indexOf("exp") >= 0) {
            return "exp";
        }
        if (expression.indexOf("expm1") >= 0) {
            return "expm1";
        }
        if (expression.indexOf("floor") >= 0) {
            return "floor";
        }
        if (expression.indexOf("log") >= 0 || expression.indexOf("ln") >= 0) {
            return "log";
        }
        if (expression.indexOf("log10") >= 0) {
            return "log10";
        }
        if (expression.indexOf("log1p") >= 0) {
            return "log1p";
        }
        if (expression.indexOf("rint") >= 0) {
            return "rint";
        }
        if (expression.indexOf("round") >= 0) {
            return "round";
        }
```

```
        if (expression.indexOf("signum") >= 0) {
            return "signum";
        }
        if (expression.indexOf("sin") >= 0) {
            return "sin";
        }
        if (expression.indexOf("sinh") >= 0) {
            return "sinh";
        }
        if (expression.indexOf("sqrt") >= 0) {
            return "sqrt";
        }
        if (expression.indexOf("tan") >= 0) {
            return "tan";
        }
        if (expression.indexOf("tanh") >= 0) {
            return "tanh";
        }
        if (expression.indexOf("todegrees") >= 0) {
            return "todegrees";
        }
        if (expression.indexOf("toradians") >= 0) {
            return "toradians";
        }
        if (expression.indexOf("ulp") >= 0) {
            return "ulp";
        }

        if (expression.indexOf("atan2") >= 0) {
            return "atan2";
        }
        if (expression.indexOf("hypot") >= 0) {
            return "hypot";
        }
        if (expression.indexOf("ieeeremainder") >= 0) {
            return "ieeeremainder";
        }
        if (expression.indexOf("max") >= 0) {
            return "max";
        }
        if (expression.indexOf("min") >= 0) {
            return "min";
        }
        if (expression.indexOf("pow") >= 0) {
            return "pow";
        }
        return null;
    }

    public double eval2(String expression) throws Exception {
        expression = "(" + expression.toLowerCase().replace(" ", "") + ")";
        String currentExpression = expression;
        String functionName = null;
        String[] temp = null;
        while (getMathFunctionName(currentExpression) != null) {
            functionName = getMathFunctionName(currentExpression);
            temp = splitStr(currentExpression, functionName);
            if (getMathFunctionName(temp[1]) == null) {
```

```
            if (temp[1].indexOf(",") < 0) {
                currentExpression = temp[0] + getPartResultWithOneParameter("" + eval(temp[1]),
                        functionName) + temp[2];
            } else {
                String[] temp2 = temp[1].split(",");
                currentExpression = temp[0] + getPartResultWithTwoParameter(eval(temp2[0]) + "," +
eval(temp2[1]), functionName) + temp[2];
            }
        } else {
            temp[1] = functionName + "(" + eval2(temp[1]) + ")";
            currentExpression = temp[0] + "(" + eval2(temp[1]) + ")" + temp[2];
        }
    }
    return eval(currentExpression);
  }

  public double differentiate(String expression, String value)
      throws Exception {
    double result = 0.0;
    expression = "0+" + expression.toLowerCase();
    expression = expression.replace("expm1", "ewpm1").replace("exp", "ewp").replace("max", "maw");
    String firstExpression = expression.replace("x", "(" + value + ")").replace("ewpm1",
"expm1").replace("ewp", "exp").replace("maw",
        "max");
    String secondExpression = expression.replace("x",
        "(" + value + "+0.00000001)").replace("ewpm1", "expm1").replace("ewp", "exp").replace("maw",
"max");
    String thirdExpression = expression.replace("x",
        "(" + value + "+0.00000002)").replace("ewpm1", "expm1").replace("ewp", "exp").replace("maw",
"max");
    result = (-3 * eval2(firstExpression) + 4 * eval2(secondExpression) - eval2(thirdExpression)) * 1e8 / 2;
    return result;
  }

  public double integral(String expression, String beginValue,
      String endValue, int partNumber) throws Exception {
    double result = 0.0;
    double h = (Double.parseDouble(endValue) - Double.parseDouble(beginValue)) / partNumber;

    expression = expression.toLowerCase();
    expression = expression.replace("expm1", "ewpm1").replace("exp", "ewp").replace("max", "maw");
    partNumber = partNumber % 2 == 1 ? (partNumber + 1) : partNumber;
    int m = partNumber / 2;

    result += eval2(expression.replace("x", "(" + beginValue + ")").replace("ewpm1",
"expm1").replace("ewp", "exp").replace("maw",
        "max")) + eval2(expression.replace("x", "(" + endValue + ")").replace(
        "ewpm1", "expm1").replace("ewp", "exp").replace("maw",
        "max")) + 4 * eval2(expression.replace("x",
        "(" + beginValue + "+" + h + ")").replace("ewpm1",
        "expm1").replace("ewp", "exp").replace("maw", "max"));
    for (int k = 1; k < m; k++) {
      result += 2 * eval2(expression.replace("x",
          "(" + beginValue + "+" + 2 * k * h + ")").replace(
          "ewpm1", "expm1").replace("ewp", "exp").replace(
          "maw", "max")) + 4 * eval2(expression.replace("x",
          "(" + beginValue + "+" + (2 * k + 1) * h + ")").replace("ewpm1", "expm1").replace("ewp",
"exp").replace("maw", "max"));
```

```
        }
        result *= (h / 3);
        return result;
    }

    public String scientificExpressionToNormal(String scientificNumber) {
        String result = scientificNumber.toLowerCase().replace(" ", "");
        int currentLocationOfE = -1;
        int beginLocation = -1;
        int endLocation = -1;
        String[] temp = new String[2];

        while ((currentLocationOfE = result.indexOf("e-")) >= 0) {
            int i = currentLocationOfE - 1;
            char currentChar = result.charAt(i);
            temp[0] = "";
            while (currentChar >= '0' && currentChar <= '9' || currentChar == '.') {
                temp[0] = currentChar + temp[0];
                i--;
                if (i < 0) {
                    beginLocation = i + 1;
                    break;
                }
                currentChar = result.charAt(i);
                if (!(currentChar >= '0' && currentChar <= '9' || currentChar == '.')) {
                    beginLocation = i + 1;
                    break;
                }
            }

            i = currentLocationOfE + 2;
            currentChar = result.charAt(i);
            temp[1] = "";
            while (currentChar >= '0' && currentChar <= '9') {
                temp[1] = temp[1] + currentChar;
                i++;
                if (i >= result.length()) {
                    endLocation = i - 1;
                    break;
                }
                currentChar = result.charAt(i);
                if (!(currentChar >= '0' && currentChar <= '9')) {
                    endLocation = i - 1;
                    break;
                }
            }
            result = result.substring(0, beginLocation) + scientificNumberToNormal(temp[0] + "e-" + temp[1]) +
result.substring(endLocation + 1);
        }

        while ((currentLocationOfE = result.indexOf("e")) >= 0) {
            int i = currentLocationOfE - 1;
            char currentChar = result.charAt(i);
            temp[0] = "";
            while (currentChar >= '0' && currentChar <= '9' || currentChar == '.') {
                temp[0] = currentChar + temp[0];
                i--;
                if (i < 0) {
```

```
                beginLocation = i + 1;
                break;
            }
            currentChar = result.charAt(i);
            if (!(currentChar >= '0' && currentChar <= '9' || currentChar == '.')) {
                beginLocation = i + 1;
                break;
            }
        }

        i = currentLocationOfE + 1;
        currentChar = result.charAt(i);
        temp[1] = "";
        while (currentChar >= '0' && currentChar <= '9') {
            temp[1] = temp[1] + currentChar;
            i++;
            if (i >= result.length()) {
                endLocation = i - 1;
                break;
            }
            currentChar = result.charAt(i);
            if (!(currentChar >= '0' && currentChar <= '9')) {
                endLocation = i - 1;
                break;
            }
        }
        result = result.substring(0, beginLocation) + scientificNumberToNormal(temp[0] + "e" + temp[1]) +
result.substring(endLocation + 1);
    }

    return result;
  }

  private String scientificNumberToNormal(String scientificNumber) {
    String result = scientificNumber;
    String temp[] = result.split("e");
    String[] temp2 = new String[2];
    String sign = temp[0].startsWith("-") ? "-" : "";
    int secondPart = Integer.parseInt(temp[1]);
    while (secondPart > 0) {
        int dot = temp[0].indexOf(".");
        temp2[0] = temp[0].substring(0, dot);
        temp2[1] = temp[0].substring(dot + 1);
        temp[0] = temp2[0] + (temp2[1].length() > 0 && !temp2[1].equals("0") ? (temp2[1].charAt(0) + "." +
temp2[1].substring(1))
                : "0.0");
        secondPart--;
    }
    while (secondPart < 0) {
        int dot = temp[0].indexOf(".");
        temp2[0] = temp[0].substring(0, dot);
        temp2[1] = temp[0].substring(dot + 1);
        temp[0] = "0." + temp2[0] + temp2[1];
        secondPart++;
    }
    result = sign + temp[0];
    return result;
  }
```

```
//The following are examples for the use of Eval
//	public static void main(String[] args) {
//		try {
//			Eval eval = new Eval();
//			System.out.println(eval.eval2("-0.105*(tan(123)+sin(cos(-3*4)))"));
//			System.out.println("-------------------------------------------");
//			System.out.println(eval
//					.eval2("-0.105*abs(tan(123)+sin(cos(-3*4)))"));
//
//			System.out.println("differentiation"
//					+ eval.differentiate("pow(x,2)+pow(x,3)", "2"));
//			System.out.println(eval.eval2("2*2+3*pow(2,2)"));
//			System.out.println(eval.integral("pow(x,2)+sin(x)", "0", "1", 100));
//			System.out.println(eval
//					.scientificExpressionToNormal("-1.23e-
1+4.21312e3"));
//			System.out.println(eval.eval2("pow(1,3)/3-cos(1)")
//					- eval.eval2("pow(0,3)/3-cos(0)"));
//			System.out.println(eval.integral("tan(pow(x,x))", "0", "1", 5000));
//			System.out.println("Result" + eval.eval2("tan(pow(1,2))"));
//			System.out.println(eval.eval2("sin(-1*acos(0))"));
//
//			System.out.println(eval.integral("tan(pow(x,2))", "-3", "3", 3000));
//			System.out.println(eval.differentiate("tan(pow(x,2))", "3.14"));
//			System.out.println(eval.eval2("tan(pow(1,2))"));
//			System.out.println(eval.eval2("tan(pow(-1,2))"));
//			System.out.println(eval.eval2("4*tan(pow(0.75,2))"));
//			System.out.println(eval.eval2("4*tan(pow(0.5,2))"));
//			System.out.println(eval.eval2("4*tan(pow(0.25,2))"));
//			System.out.println(eval.eval2("2*tan(pow(0,2))"));
//
//			System.out.println("--------------------------");
//			System.out
//					.println(eval
//
		.eval("(1.5574077246549023+1.5574077246549023+2.521750695343539+1.021367684884145+0.2
503260302651001+0)/9"));
//			System.out
//					.println(eval
//
		.eval2("tan(pow(1,2))+tan(pow(-
1,2))+4*tan(pow(0.75,2))+4*tan(pow(0.5,2))+4*tan(pow(0.25,2))+2*tan(pow(0,2))"));
//
//			System.out.println(eval.eval2("tan(pow(3,2))"));
//			System.out.println(eval.eval2("tan(pow(-3,2))"));
//			System.out.println(eval.eval2("4*tan(pow(2.25,2))"));
//			System.out.println(eval.eval2("4*tan(pow(1.5,2))"));
//			System.out.println(eval.eval2("4*tan(pow(0.75,2))"));
//			System.out.println(eval.eval2("2*tan(pow(0,2))"));
//			System.out
//					.println(eval
//
		.eval2("(tan(pow(3,2))+tan(pow(-
3,2))+4*tan(pow(2.9,2))+4*tan(pow(2.8,2))+4*tan(pow(2.7,2))+4*tan(pow(2.6,2))+4*tan(pow(2.5,2))+4*tan
(pow(2.4,2))+4*tan(pow(2.3,2))+4*tan(pow(2.2,2))+4*tan(pow(2.1,2))+4*tan(pow(2.0,2))+4*tan(pow(1.9,2)
)+4*tan(pow(1.8,2))+4*tan(pow(1.7,2))+4*tan(pow(1.6,2))+4*tan(pow(1.5,2))+4*tan(pow(1.4,2))+4*tan(po
w(1.3,2))+4*tan(pow(1.2,2))+4*tan(pow(1.1,2))+4*tan(pow(1.0,2))+4*tan(pow(0.9,2))+4*tan(pow(0.8,2))+
4*tan(pow(0.7,2))+4*tan(pow(0.6,2))+4*tan(pow(0.5,2))+4*tan(pow(0.4,2))+4*tan(pow(0.3,2))+4*tan(pow(
```

```
0.2,2))+4*tan(pow(0.1,2)))/60"));
//
//                                      System.out.println("-------------------------");
//                                      System.out
//                                                                      .println(eval
//
            .eval("(1.5574077246549023+1.5574077246549023+2.521750695343539+1.021367684884145+0.2
503260302651001+0)/9"));
//
//                                      System.out.println(eval.scientificExpressionToNormal("1.2345e10"));
//
//                                      System.out.println(eval
//                                                                      .scientificExpressionToNormal("-
5.307179586686775E-6"));
//                                      System.out.println(eval.eval2("sin(sin(3.14159+3.14159))"));
//                                      System.out.println(eval.eval2("sin(-5.307179586686775E-6)"));
//
//                                      System.out.println(eval.eval2("sin(pow(2,2)-2*3.1415926)"));
//                                      System.out.println(eval.eval2("sin(4)"));
//                                      System.out.println(eval
//                                                                      .differentiate("max((sin(x)+x)*10/x,2)", "1"));
//                                      System.out.println(eval.eval("-(1)"));
//                      } catch (Exception e) {
//                                      e.printStackTrace();
//                      }
//          }
}
```

IEval.java

IEval.java

```
package xiaobin.testing;

/**
 * The interface for caculus class
 * @author Ben
 */
public interface IEval {

    /**
     * Convert scientific expression to normal numerical expression
     *
     * @param number scientific number
     * @return normal numerical expression
     * @throws Exception
     **/
    public String scientificExpressionToNormal(String number) throws Exception;

    /**
     * calculate integral
     *
     * @param expression may contain mathematical functions from java.util.Math
     *            independent variable in the expression is denoted as x
     * @param beginValue the begin point of the integral calculation
     * @param endValue the end point of the integral calculation
     * @param partNumber the larger it is, the more precise the result will be,
     *            but slower too
     * @return definite integral
     * @throws Exception
     */
```

```
    public double integral(String expression, String beginValue,
            String endValue, int partNumber) throws Exception;

    /**
     * calculate differentiation
     *
     * @param expression may contain mathematical functions from java.util.Math
     *          independent variable in the expression is denoted as x
     * @param value is the point where the derivative is being calculated
     * @return the derivative of a function at param value point
     * @throws Exception
     */
    public double differentiate(String expression, String value)
            throws Exception;

    /**
     * compute mathematical expressions
     * @param expression may contain mathematical functions from java.util.Math
     *          independent variable in the expression is denoted as x
     * @return the result of the expression after calculation
     * @throws Exception
     */
    public double eval2(String expression) throws Exception;

    /**
     * compute basic arithmetic operations only (brackets, + - * /),
     *          advanced mathematical functions should not be included
     * @return the result of the expression after calculation
     * @throws Exception
     */
    public double eval(String expression) throws Exception;
}
```

IAdaptiveModel.java

IAdaptiveModel.java

```
package xiaobin.testing;

/**
 * The interface for the 3-parameter IRT model class)
 * @author Ben
 */
public interface IAdaptiveModel {


    /**
     * select the best question (i.e., the most informative question)
     * @return the index of the most informative question
     */
    public int selectBestQuestion ();

    /**
     * update currentAbility based on responseHistory and testHistory
     */
    public void updateEstimatedAbility ();

}
```

OutOfBoundException.java

OutOfBoundException.java

```
package xiaobin.testing;

/**
 * defines the possible out of bound exception in computation
 * @author Ben
 */
public class OutOfBoundException extends Exception{

    private String error;

    public OutOfBoundException () {
        super();
        error = "unknown exception";
    }

    public OutOfBoundException (String err) {
        super(err);     // call super class constructor
        error = err;  // save message
    }

    public String getError () {
        return error;
    }

}
}
```

AdaptiveModel.java

AdaptiveModel.java

```
package xiaobin.testing;

import java.util.List;
import java.util.ArrayList;
import java.util.ListIterator;
import java.util.Collections;
import java.util.Comparator;

/**
 * consists of methods used in the computation of the 3-parameter IRT model
 * @author Ben
 */
public class AdaptiveModel implements IAdaptiveModel {

    private static final double D = -1.702; //constant used to approximate the normal ogive function
    private static final double ABILITY_UPPER_BOUNDARY = 3.0; //upper boundary of ability
    private static final double ABILITY_LOWER_BOUNDARY = -3.0; //lower boundary of ability
    private static final double PRECISION = 0.001; //precision for item information

    private double currentAbility;
    private double testInformation;

    private List<QuestionUnit> questionSet; //the pool of candidate questions

    private List<Integer> responseLog; //1 - correct; 0 -incorrect
    private List<QuestionUnit> questionLog; //the questions attempted in the test

    /**
     * Initialize the test
     * @param initAbility is the initial ability for the current Examinee
```

```
     * @param questionSet is the question pool from where the quesiton is selected
     */
    public AdaptiveModel (double initAbility, List<QuestionUnit> questionSet) {
       currentAbility = initAbility;
       testInformation = 0.0;
       this.questionSet = questionSet;
       responseLog = new ArrayList<Integer>();
       questionLog = new ArrayList<QuestionUnit>();
    }

    /**
     * Item Response Function of the three parameter model
     * @param theta - the current examinee's ability,
     * @param a - the discrimination power of the question
     * @param b - the difficulty level of the question
     * @param c - the guessing factor of the question, whose value
     *            must fall in between [0, 0.75]
     * @return the computational result of the item response function
     */
    public double computeIRF (double theta, double a, double b, double c)
           throws OutOfBoundException {
       if (c >= 0.0 && c<=0.75) {
          double p; //the probability of receiving a correct response
          double exp = Math.exp(D*a*(theta-b));
          p = c + (1-c)/(1+exp);
          return p;
       } else {
          throw new OutOfBoundException ("Parameter c must be in [0, 0.75]");
       }
    }

    /**
     * Compute Item Information Function calculation
     * [P'(theta)]^2/P(theta) * (1-P(theta))
     * @param theta - the current examinee's ability,
     * @param a - the discrimination power of the question
     * @param b - the difficulty level of the question
     * @param c - the guessing factor of the question, whose value
     *            must fall in between [0, 0.75]
     * @return the computational result of the item information function
     */
    public double computeIIF (double theta, double a, double b, double c) {
       double p = 0.5;
       try {
          p = computeIRF(theta, a, b, c);
       } catch (OutOfBoundException oe) {
          System.out.println("OutOfBound Exception was" + oe.getError());
       }
       double q = 1-p;
       double dp = computeDerivativeOfIRF (theta, a, b, c); //First derivative of p
       double i = (dp*dp)/(p*q);
       return i;
    }

    /**
     * Computer the Derivative of Item Response Function with regard to theta
     * P'(theta)
     * @param theta - the current examinee's ability,
```

```
     * @param a - the discrimination power of the question
     * @param b - the difficulty level of the question
     * @param c - the guessing factor of the question, whose value
     *           must fall in between [0, 0.75]
     * @return the computational result of derivative
     */
    public double computeDerivativeOfIRF (double theta, double a, double b, double c) {
        Eval eval = new Eval();
        double result = 0.0;
        String irfExpression = "("+c+")"+"+(1-("+c+"))/(1+ewp(("+D+")*("+a+
                ")*(x-("+b+")))";
        //System.out.println("the expression is: "+irfExpression);
        //System.out.println("the theta is: "+theta);
        try{
          result = eval.differentiate(irfExpression, Double.toString(theta));
        }catch (Exception e) {
          e.printStackTrace();
        }
        return result;
    }


    /**
     * update currentAbility based on responseLog and questionLog
     * theta[s+1] = theta[s] + SIGMA{S[i](theta[s]))/SIGMA(I[i](theta[s]))
     * S[i] = (u[i] - p[i])*p'[i]/(p[i](1-p[i]))
     */
    public void updateEstimatedAbility () {
        double sigmaS = 0;
        double sigmaI = 0;
        //traverse the response history to calculate sigmaS and sigmaI
        ListIterator<QuestionUnit> itr = questionLog.listIterator();
        while (itr.hasNext()) {
            //get parameters a,b and c of the current question
            int index = itr.nextIndex();
            QuestionUnit q = itr.next();
            double a = q.getParameters().get(0);
            double b = q.getParameters().get(1);
            double c = q.getParameters().get(2);
            double p = 0.0;
            try {
                p = computeIRF(currentAbility, a, b, c);
            } catch (OutOfBoundException oe) {
                System.out.println("OutOfBound Exception was" + oe.getError());
            }
            double multiplier1 = responseLog.get(index)- p; //(u[i]-p[i])
            double dp = computeDerivativeOfIRF (currentAbility, a, b, c); //First derivative of p
            double s = multiplier1 * dp / (p * (1 - p));
            sigmaS += s;
            double info = computeIIF(currentAbility, a, b, c); //I[i]
            sigmaI += info;
        }

        if (sigmaI < PRECISION) {
            sigmaI = PRECISION;
        }
        testInformation = sigmaI;
        currentAbility += sigmaS/sigmaI;
```

```
        if(currentAbility >= ABILITY_UPPER_BOUNDARY) {
            currentAbility = ABILITY_UPPER_BOUNDARY;
        }else if(currentAbility <= ABILITY_LOWER_BOUNDARY) {
            currentAbility = ABILITY_LOWER_BOUNDARY;
        }
    }

    /**
     * select the best question
     * @return - the index of the most informative question,
     *          -1 is returned if no unused question
     */
    public int selectBestQuestion () {
        //create an array of index
        List<QuestionInfo> questions = new ArrayList<QuestionInfo>();
        //compute question information for every question in questionSet
        ListIterator<QuestionUnit> itr = questionSet.listIterator();
        while (itr.hasNext()) {
            int index = itr.nextIndex();
            QuestionUnit q = itr.next();
            double a = q.getParameters().get(0);
            double b = q.getParameters().get(1);
            double c = q.getParameters().get(2);
            double info = this.computeIIF(currentAbility, a, b, c);
            questions.add(new QuestionInfo(index, info));
        }
        //sort question information
        Comparator comp = new QuestionComparator();
        Collections.sort (questions, comp);
        //find the best question
        int lastIndex = questions.size() - 1;
        //avoid reusing questions
        for (int i=0; i<=lastIndex; i++) {
            int posBestQuestion = questions.get(i).getPosition();
            QuestionUnit bestQuestion = questionSet.get(posBestQuestion);
            if (!questionLog.contains(bestQuestion)) {
                return posBestQuestion;
            }
        }
        return -1;
    }

    /**
     * get a question from questonSet
     * @param index - the index of the question to be retrieved
     * @return - questionUnit
     */
    public QuestionUnit getQuestion (int index) {
        return questionSet.get(index);
    }

    /**
     * add a used question to the question log
     * @param question - the question to be added to the test log
     */
    public void addQuestionLog (QuestionUnit question) {
        questionLog.add(question);
    }
```

```
    /**
     * add a response to the response log
     * @param response - the response to be added to the response log
     */
    public void addResponseLog (int response) {
        responseLog.add(response);
    }

    /**
     * get currentAbility
     */
    public double getCurrentAbility () {
        return currentAbility;
    }

    /**
     * get testInformation
     */
    public double getTestInformation () {
        return testInformation;
    }
}
```

QuestionComparator.java

QuestionComparator.java

```
package xiaobin.testing;

import java.util.Comparator;

/**
 * defines the comparason rules when comparing question information
 * @author Ben
 */
public class QuestionComparator implements Comparator{

    public int compare (Object o1, Object o2) {
        QuestionInfo q1 = (QuestionInfo)o1;
        QuestionInfo q2 = (QuestionInfo)o2;
        double info1 = q1.getInfo();
        double info2 = q2.getInfo();
        if (info1 < info2) {
            return 1;
        }else {
            return 0;
        }
    }
}
```

QuestionInfo.java

QuestionInfo.java

```
package xiaobin.testing;

/** This class is created for sorting purpose. It records the original position
 *  of a question and the information of the question, the latter dynamically
 *  changes as the ability estimate updates.
 **/

public class QuestionInfo {
```

```
    private int position;
    private double information;

    public QuestionInfo(int pos, double info) {
        position = pos;
        information = info;
    }

    public int getPosition() {
        return position;
    }

    public double getInfo() {
        return information;
    }
}
```

QuestionUnit.java

QuestionUnit.java

```
package xiaobin.testing;

import java.util.List;
import java.util.ArrayList;
import java.util.Set;
import java.util.HashSet;
import java.util.Map;
import java.util.HashMap;

/**
 * This class describes the structure of a question.
 * @author Ben
 */

public class QuestionUnit {

    private Integer id; //the Id of the question
    private String description; //the description of the question
    private List<String> options; //options for the question
    private Set<Integer> answers; //answers to the question
    private Map<String, String> attributes; //other attributes
    private List<Double> parameters; //parameters of the question
    private String model; //the applicable adaptive model for the question

    public QuestionUnit() {
        description = "";
        options = new ArrayList<String> ();
        answers = new HashSet<Integer> ();
        attributes = new HashMap<String, String> ();
        parameters = new ArrayList<Double> ();
        model = "";
    }

    /** Sets the description for the question. */
    public void setDescription (String description) {
        this.description = description;
    }

    /** Gets the description of the question. */
```

```
public String getDescription () {
   return description;
}

//** Sets question options. */
public void setOptions (List<String> options) {
   this.options = options;
}

/** Gets the options of the question.*/
public List<String> getOptions () {
   return options;
}

/** Adds a standard answer to the question. */
public void setAnswers (Set<Integer> answers) {
   this.answers = answers;
}

/** Gets the standard answers to the question. */
public Set<Integer> getAnswers () {
   return answers;
}

/** Compares the standard answers with the answers provides. */
public boolean compareAnswers (Set<Integer> answers) {
   return this.answers.equals(answers);
}

/** Sets attributes of the question. */
public void setAttributes (Map<String, String> attributes) {
   this.attributes = attributes;
}

/** Gets the attributes of the question. */
public Map<String, String> getAttributes () {
   return attributes;
}

/** Sets parameters of the question. */
public void setParameters (List<Double> parameters) {
   this.parameters = parameters;
}

//get the indexed parameter of the question
public List<Double> getParameters () {
   return parameters;
}

//set the adaptive model for the question
public void setModel (String model) {
   this.model = model;
}

//get the adaptive model of the question
public String getModel () {
   return model;
}
```

```
    //Set the Id for the question
    public void setId (Integer id) {
        this.id = id;
    }

    //get the Id of the question
    public Integer getId () {
        return id;
    }


}
```

The xiaobin.web package

index.jsp

index.jsp

```
<%--
    Document   : index
    Created on : 11-Jun-2009, 18:52:32
    Author     : Ben
--%>

<%@ include file="/taglibs.jsp" %>

<stripes:layout-render name="/layout/standard.jsp" title="SmartTester">
    <stripes:layout-component name="contents">

        <stripes:form action="/Test.action">
            <stripes:errors/>
            <table width="400" border="0">
                <tr>
                    <p>This demo shows how a computerized adaptive testing works.
                    The test questions were extracted from Cisco IT Essentials II
                    course. The maximum ability estimate is +3 and the minimum
                    ability estimate is -3 (Higher is better).
                    Click the "start test" button to begin.</p>
                </tr>
                <tr>
                    <td colspan="2"> </td>
                </tr>
                <tr>
                    <td colspan="2"><stripes:submit name="StartTest"
                                        value="start test"></stripes:submit></td>
                </tr>
            </table>
        </stripes:form>

    </stripes:layout-component>
</stripes:layout-render>
```

questionpresenter.jsp

questionpresenter.jsp

```
<%--
    Document   : questionpresenter
    Created on : 25-Jun-2009, 15:35:34
    Author     : Ben
--%>
```

```
<%@ include file="/taglibs.jsp" %>

<stripes:layout-render name="/layout/standard.jsp" title="SmartTester">
   <stripes:layout-component name="contents">

      <stripes:form action="/Test.action">
         <stripes:errors/>
         <table width="400" border="0">
            <tr>
               ${actionBean.currentQuestion.description}
            </tr>

            <c:forEach items="${actionBean.currentQuestion.options}" var="option" varStatus="status">
               <tr>
                  <td>
                     <stripes:radio name="userInformation.userChoice" value="${status.count}"
                              id="${status.count}" />
                     <stripes:label for="${status.count}">${option}</stripes:label>
                  </td>
               </tr>
            </c:forEach>

            <tr>
               <td colspan="2"> </td>
            </tr>
            <tr>
               <td colspan="2">The correct answer for this question is:</td>
               <td>${actionBean.currentQuestion.answers}</td>
            </tr>
            <tr>
               <td colspan="2">Your current ability estimate is:</td>
               <td>${actionBean.adaptiveTestManager.currentAbility}</td>
            </tr>
            <tr>
               <td colspan="2"> </td>
            </tr>
            <tr>
               <td colspan="2"><stripes:submit name="NextQuestion"
                                 value="next question"></stripes:submit></td>
            </tr>
         </table>
      </stripes:form>

   </stripes:layout-component>
</stripes:layout-render>
```

taglibs.jsp

taglibs.jsp

```
<%@ taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<%-- Short hand for the context root. --%>
<c:set var="ctx" value="${pageContext.request.contextPath}"/>
```

testabortion.jsp

testabortion.jsp

```
<%--
```

```
    Document   : testabortion
    Created on : 29-Jun-2009, 00:51:47
    Author     : Ben
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Test Abortion</title>
    </head>
    <body>
        <h1>An error occurred during test taking! It might be caused by insufficient questions. </h1>
    </body>
</html>
```

The xiaobin.actionbeans package

TestActionBean.java

TestActionBean.java

```
package xiaobin.actionbeans;

import net.sourceforge.stripes.action.DefaultHandler;
import net.sourceforge.stripes.action.HandlesEvent;
import net.sourceforge.stripes.action.Resolution;
import net.sourceforge.stripes.action.RedirectResolution;
import net.sourceforge.stripes.action.ActionBean;
import net.sourceforge.stripes.action.ActionBeanContext;
import net.sourceforge.stripes.action.UrlBinding;
import net.sourceforge.stripes.action.Wizard;
import net.sourceforge.stripes.validation.ValidationMethod;
import net.sourceforge.stripes.validation.Validate;
import net.sourceforge.stripes.validation.ValidateNestedProperties;
import net.sourceforge.stripes.validation.ValidationErrors;
import net.sourceforge.stripes.validation.SimpleError;
import net.sourceforge.stripes.controller.FlashScope;

import java.util.Set;
import java.util.HashSet;

import xiaobin.testing.QuestionUnit;
import xiaobin.testing.AdaptiveTestManager;
import xiaobin.webui.UserInformation;

/** Action bean that
 *
 * @author Ben
 */
@Wizard(startEvents = "StartTest")
@UrlBinding("/Test.action")
public class TestActionBean implements ActionBean {

    @ValidateNestedProperties({
        @Validate(field = "userChoice", required = true)
    })
    private UserInformation userInformation;
```

```
private ActionBeanContext context;
private static AdaptiveTestManager adaptiveTestManager;
private static QuestionUnit currentQuestion;

public ActionBeanContext getContext() {
   return context;
}

public void setContext(ActionBeanContext context) {
   this.context = context;
}

public QuestionUnit getCurrentQuestion() {
   return currentQuestion;
}

public AdaptiveTestManager getAdaptiveTestManager() {
   return adaptiveTestManager;
}

public UserInformation getUserInformation() {
   return userInformation;
}

public void setUserInformation(UserInformation userInformation) {
   this.userInformation = userInformation;
}

@ValidationMethod(on = "nextQuestion")
public void validateUserChoice(ValidationErrors errors) {
   if (null == userInformation.getUserChoice()) {
      errors.add("userInformation.userChoice", new SimpleError(
            "Please select one answer!"));
   }
   FlashScope.getCurrent(getContext().getRequest(), true).put(this);
}

@HandlesEvent("StartTest")
public Resolution startTest() {
   adaptiveTestManager = new AdaptiveTestManager();
   currentQuestion = adaptiveTestManager.getCurrentQuestion();
   return new RedirectResolution("/questionpresenter.jsp").flash(this);
}

@DefaultHandler
@HandlesEvent("NextQuestion")
public Resolution nextQuestion() {
   //acquires user choice
   Set<Integer> answers = new HashSet<Integer>();
   int userChoice = Integer.parseInt(userInformation.getUserChoice());
   answers.add(userChoice);
   //update test status
   try {
      adaptiveTestManager.getAnswersAndUpdate(answers);
   } catch (NullPointerException e) {
      System.out.println("adaptiveTestManager returned null");
      e.printStackTrace();
   }
```

```
        //Shall test finish?
        if (adaptiveTestManager.isTestInformationEnough()) {
            return new RedirectResolution("/testcompletion.jsp").flash(this);
        } else if ((currentQuestion = adaptiveTestManager.getCurrentQuestion()) == null) {
            return new RedirectResolution("/testabortion.jsp").flash(this);
        }

        return new RedirectResolution("/questionpresenter.jsp").flash(this);
    }
}
        return currentQuestion;
    }

    public AdaptiveTestManager getAdaptiveTestManager() {
        return adaptiveTestManager;
    }

    public UserInformation getUserInformation() {
        return userInformation;
    }

    public void setUserInformation(UserInformation userInformation) {
        this.userInformation = userInformation;
    }

    @ValidationMethod(on = "nextQuestion")
            public void validateUserChoice(ValidationErrors errors) {
                        if (null == userInformation.getUserChoice()) {
                                    errors.add("userInformation.userChoice", new SimpleError(
                                                            "Please select one answer!"));
                        }
        FlashScope.getCurrent(getContext().getRequest(), true).put(this);
            }

    @HandlesEvent("StartTest")
            public Resolution startTest() {
        adaptiveTestManager = new AdaptiveTestManager();
        currentQuestion = adaptiveTestManager.getCurrentQuestion();
                        return new RedirectResolution("/questionpresenter.jsp").flash(this);
            }

    @DefaultHandler
    @HandlesEvent("NextQuestion")
    public Resolution nextQuestion() {
        //acquires user choice
        Set<Integer> answers = new HashSet<Integer>();
        int userChoice = Integer.parseInt(userInformation.getUserChoice());
        answers.add(userChoice);
        //update test status
        try {
            adaptiveTestManager.getAnswersAndUpdate(answers);
        }catch (NullPointerException e) {
            System.out.println("adaptiveTestManager returned null");
            e.printStackTrace();
        }
        //Shall test finish?
        if (adaptiveTestManager.isTestInformationEnough()) {
            return new RedirectResolution("/testcompletion.jsp").flash(this);
```

```
        } else if ((currentQuestion = adaptiveTestManager.getCurrentQuestion())==null){
            return new RedirectResolution("/testabortion.jsp").flash(this);
        }

        return new RedirectResolution("/questionpresenter.jsp").flash(this);
    }
}
```

The xiaobin.webui package

UserInformation.java

UserInformation.java

```
package xiaobin.webui;

/** This class captures user input from webui
 *
 * @author Ben
 */
public class UserInformation {
    private String userChoice;
    public String getUserChoice() {
                return userChoice;
        }

        public void setUserChoice(String userChoice) {
                this.userChoice = userChoice;
        }
}
```