AUTOMATED ASSESSMENT AND GAMEPLAY ADAPTATION IN SERIOUS EDUCATIONAL GAMES

by

Bharathi Balasubramaniam

APPROVED BY SUPERVISORY COMMITTEE:

Dedicated to my friends and family, who have been with me in every step of my life.

AUTOMATED ASSESSMENT AND GAMEPLAY ADAPTATION IN SERIOUS EDUCATIONAL GAMES

by

BHARATHI BALASUBRAMANIAM, B.E., M.S.

THESIS

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2012

ACKNOWLEDGEMENTS

My advisor, Dr. Kendra Cooper, has been a constant source of support, suggestions, and encouragement throughout my thesis, and this thesis would not have been possible without her. The members of my supervisory committee, Dr. Lawrence Chung, Dr. Xiaohu Guo, have been extremely helpful, and their comments and suggestions have made my research stronger. The SimSYS research team has been instrumental in helping me understand the problem statement better. My thanks to Rockwell Collins Inc., for tuition fee reimbursement of my thesis credits. My parents, P. Balasubramaniam and Kandammal Balu, have been supporting my studies since my school days. I wouldn't be who I am today without their love and affection! I wouldn’t forget their struggle to make me an MS. I am happy that I am blessed with the best of the people as my friends, who have believed in me more than I did in myself! Without their motivation and encouragement, I wouldn’t be doing an MS or research.

March 2012

AUTOMATED ASSESSMENT AND GAMEPLAY ADAPATION IN SERIOUS

EDUCATIONAL GAMES

Publication No. ___________________

Bharathi Balasubramaniam, M.S.
The University of Texas at Dallas, 2012

Supervising Professor: Dr. Kendra Cooper

Serious educational games (SEGs) are used to teach a particular subject area, for example software engineering, chemistry, mathematics, etc. SEGs can provide an in-depth learning environment and keep the students engaged. Although this is an active research area, little has been reported in the literature on student assessment and game adaptation approaches. This research identifies key goals, proposes an assessment and adaptation approach to address many of the goals, and validates the approach with a sensitivity analysis. The key goals addressed in this work are rigorousness, specificity, automatability, accuracy, applicability and performance cost. In the future additional research is needed to improve the ability to configure the assessment and adaptation approach in order to improve the applicability and provide more flexible dynamic adaptation options.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Games have a reputation for being fun and engaging; more importantly immersive, requiring deep thinking and complex problem solving [1]. Serious Educational games (SEGs) are games which are used as Educational medium. Current research in SEGs is concentrated on effectiveness of games in converting theoretical data into practical games [2], using interactive graphics for computer adaptive testing [3], assessment of the game's usability [4], game adaptation to better suit the users [5], [6].

In software engineering (SE), SEGs are viewed as valuable tools to help students better understand SE from multiple perspectives, as well as reinforcing the effects of good and bad decision-making [7]. Serious educational games have recently received attention in software engineering education literature. Individual games teach specific software engineering topics (e.g., project management, design patterns, and so on) [8]. The current research in SE is focused on application of software engineering principles to make the game development process better, use of simulation games in teaching software engineering [9], etc. Research is also carried out on assessment of game's performance in teaching software engineering and making changes in the games to be particularly suitable for the student learning software engineering, etc.

Consider a scenario that involves the student player, instructor, game developers, administration, and accreditation auditors using a game (For E.g.: SE based Serious Educational Game) that has been integrated into a curriculum. As the student player progresses in the game, player's performance is captured and stored. Later in time, the instructor can acquire the data and assess the student's performance and provide feedback [10], [11], [12], [13], both to the student and to the game developers. The game developer can adapt a game, usually for a collection of players [14], not specifically for an individual player. Administrators and auditors requiring information about learning objectives covered in the game and the students' evaluations need to acquire this information from the game developers and instructors. Currently, the assessment and adaptation of SEGs is a manual process, which is time consuming, prone to errors, and requires substantial expertise. It is also limited with respect to tailoring games specifically for an individual student. This assessment and adaptation process for serious educational games has received little attention in the literature.

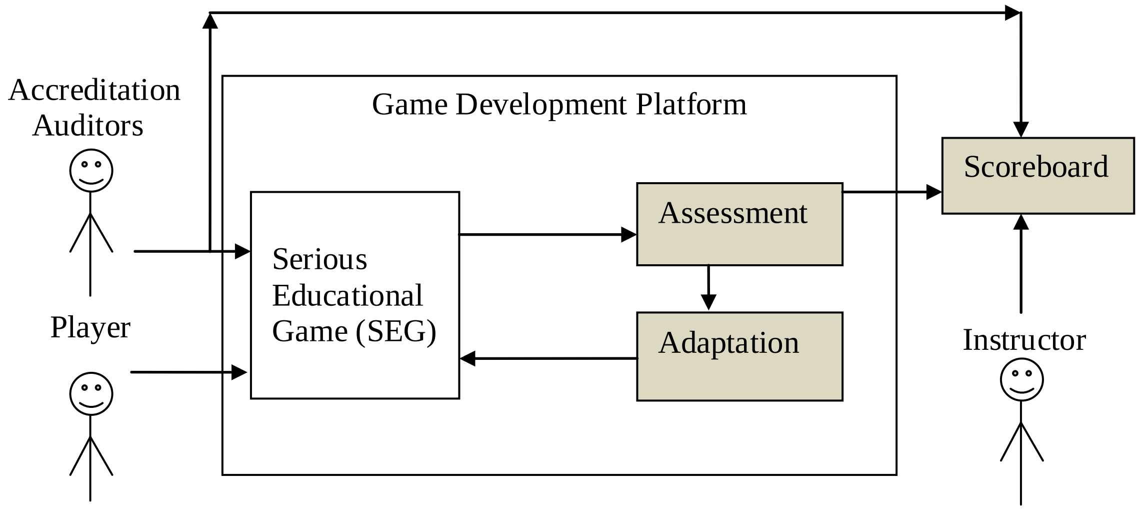

Figure 1.1. Automating the Assessment and Adaptation of Serious Educational Games.

The solution proposed here is to automatically assess a game's usability and the student's performance by collecting and analyzing game play data (Figure 1.1). The proposed assessment and adaptation approach needs to have the following characteristics:

1) Assessment Overview:

   a. Rigorousness: Support concept level and goal level assessment. In order to thoroughly assess the student's subject knowledge, any student playing a serious game has to be assessed by two types of evaluations, Concept level evaluation and Goal level evaluation [15]. Concept level evaluation is evaluating a student's knowledge in a particular concept and Goal level evaluation is evaluating the student's overall performance in the course. This goal based score results in ranking of the player in comparison to the other players [16] and also in presenting the entire set of player's performance.

b. **Specificity:** Assessment should be done based on individual student's current performance and not based on the previous performance of the class. That is, the student should not be assessed only on parameters in which the class's performance was bad in the past. By assessing every single student on all the parameters makes assessment specific.

c. **Automatability:** The assessment system should be automated, so that it does not depend on the feedback of instructor/subject matter expert to assess a student. When the system is automated, the student's performance need not be stored in the system for future assessment.

d. **Accuracy:** The system should assess the student's performance accurately. That is, the grading system should be good enough to allow the students fall in all three categories:- good, average and bad. By considering assessment parameters which actually reflect the student's performance makes sure not all students fall under a single category**.**

2) **Adaptation Overview:**

a. **Specificity:** The game should be adapted based on result from the individual student's assessment. This is only dependent on the particular student's subject knowledge and not based on the entire set of students' performance.

b. **Accuracy:** In order to accurately adapt the game, both difficulty adaptation [5], [6] and level adaptation should be done. Adaptation of the difficulty of questions is called difficulty adaptation. Level adaptation is adapting the level of game based on whether the student passed the current level or not.

3) **Performance cost:** The system needs to have a good response time performance. The choice for assessment of the game's difficulty and the player's performance is statistical analysis [4] The alternative approaches available in the literature are discussed in detail in the related work chapter. Statistical analysis has very less performance cost [17] when compared to artificial intelligence based algorithms with performance cost of O(n^2) or O(nlogn) and stochastic algorithm(probability and statistics based algorithm) with worst case complexity of O(n^2). Assessment based on the collected statistics has a complexity of 'n' where 'n' is the total number of questions (across all concepts and levels) since assessment is done after every question is

answered. If a concept or a level is also completed after a question is answered, concept based and level based assessment is done in addition. Since the number of concepts and levels are very small in number compared to the number of questions, the worst case complexity would be O(n).

4) Application: The solution should be straightforward to integrate into a variety of serious educational games, which may already be developed. To accomplish this, an aspect-oriented approach is selected; both the assessment and adaptation components are proposed as aspects; the prototype is developed using AspectJ.

5) Validation: The system developed should be rigorously validated by integrating with different games. Strong validation results should be provided.

The Assessment-Adaptation solution provided is for single player, quiz based games. When a game has graphic based objects to guide the player through the game along with quiz based exercise, as long as the player does not take any action on those objects (like moving the objects, etc.), the game is considered to be quiz based and not graphics based. Since these games are not animation/graphic based digital game, there are not many parameters (e.g., angle of rotation of the object, degree of inclination of the object, etc.) contributing to the assessment and adaptation.

The remainder of this thesis is structured as follows. The related work survey is in Chapter 2, which presents a comparison of various assessment and adaptation approaches in e-learning, electronic board games and electronic character based games using performance cost, accuracy, application and validation approaches as comparison criteria. An overview of the Assessment-Adaptation approach is presented in Chapter 3, followed by more detailed presentations of the Assessment Component and the Adaptation Component in Chapters 4 and 5. The validation of the approach is presented in Chapter 6. Conclusions and Future Work in Chapter 7.

CHAPTER 2

RELATED WORK

2.1 Introduction

The related work section covers the evaluation method followed in e-learning, different digital board games and e-games in order to assess the effectiveness of education/games. Since we are interested in games with learning objectives we are dealing only with serious games. This section also covers the adaptation techniques used in all these various fields based on the evaluation results. Adaptation is the process of changing the game or educational system in the next level depending on the areas of improvement for the player/student and also to keep his/her involvement and interest level high. Since the current literature does not clearly specify the performance cost of assessment and adaptation, performance cost is approximated from the application of the algorithms in other systems.

2.2 E-Learning Assessment and Adaptation

There are four principles that any assessment and adaptation system or approach must address to be useful in e-learning settings, according to the BEAR (Berkeley Evaluation & Assessment Research Center) Assessment Principles [18]:

(a) Assessments should be based on a developmental perspective of student learning.

(b) Assessments in e-learning should be clearly aligned with the goals of instruction.

(c) Assessments must produce valid and reliable evidence of what students know and can do.

(d) Assessment data should provide information that is useful to teachers and students to improve learning outcomes.

2.2.1 Probability and Statistics

2.2.1.1 *<u>Statistical Analysis Approach:</u>* Statistical analysis of different factors to assess the effectiveness of e-learning in providing the learning objective is a common method.

1) **Assessment Overview:**
   a. **Rigorousness:** [19] proposes a method where the examination was conducted in two steps. The learners were tested before and after study by e-Learning and applied statistical analysis to experimental data. Categorizing [20] the different parameters for statistical analysis provides more easy analysis and assessment. Assessment is rigorous by collecting different parameters.
   b. **Specificity:** The work of building the proposal is based on benchmarking activities and statistical analysis of frequent observable elements in e-learning in systems of indicators. So, the assessment is not specific to the student's performance. The intention is to identify relevant works in measuring e-learning, and also to characterize them in terms of categories, variables and indicators.
   c. **Automatability:** Not discussed in the paper.
   d. **Accuracy:** Not discussed in the paper.
2) **Adaptation Overview:**
   a. **Specificity:** A personalized e-learning service provides learning content to fit learners' individual differences.
   b. **Accuracy:** [21] The one-way Analysis of Covariance (ANCOVA) results indicate that the experimental group scored significantly higher than the control group in the post-test.
3) **Performance cost:** The complexity of statistical analysis is close to, stochastic algorithm's performance complexity, but not quite the same [17]. Las Vegas algorithm is a Stochastic/Randomized Algorithm. Las Vegas algorithm has average case complexity of O(nlogn) and worst case complexity of O(n^2).
4) **Application:** Apart from being used in fields like business statistics, data mining, engineering, the rapid and sustained increases in computing power have had a substantial impact on the practice of statistical science.
5) **Validation:** Questionnaires were applied to evaluate the learners' degree of satisfaction in the experimental group. The five-point Likert scale was applied to evaluate the degree of satisfaction with the proposed system. Experimental result shows that the personalized service of the proposed system is quite acceptable for learners.

2.2.1.2 *Statistical approach in factor analysis algorithm:* A personalized intelligent question answering algorithm can be put up, which takes learners' personality (background, age, etc) into account based on learners' information.

1) **Assessment Overview:** Not discussed in the paper.

2) **Adaptation Overview:**

   a. **Specificity:** [22] analyzed how to integrate the learner's characteristic and his/her question using harmony theory to dynamically produce a question model. Using the question model, this algorithm obtained the parameters of answer depth and presentation pattern adapted to the learner's personality.

   b. **Accuracy:** With this, the learners felt that the answers were matched with their questions well. According to the learners' feedback data, it indicated that the personalized intelligent question answering algorithm [22] had more efferent than others because the personalized answer made the learner assimilate answer information in his cognitive structure and had really explained the puzzle.

3) **Performance cost:** Not discussed in the paper.

4) **Application:** Domestic people working in education still confine the understanding about online question answering [22] to the interactivity model of traditional teaching.

5) **Validation:** In the test-use of [22] algorithm, a questionnaire investigation was designed on answering website. Its purpose was to collect the data.

2.2.2 Artificial Intelligence

2.2.2.1 *Artificial Neural Network approach:* Neural network can be used as a method to explore a student's performance, who is using e-learning system. Such a system would consider various student characteristics.

1) **Assessment Overview:** [23] explores the learning performance of various students using 4-step approach based on an artificial neural network (ANN) core data mining technique. Three different levels of teaching content were set for adaptive learning.

   a. **Rigorousness:** Not discussed in the paper.

   b. **Specificity:** Not discussed in the paper.

   c. **Automatability:** Not discussed in the paper.

   d. **Accuracy:** Not discussed in the paper.

2) **Adaptation overview:**
   a. **Specificity:** Based on students' gender, personality types, and anxiety levels, the proposed system sets different levels of teaching content for students with different characteristic combinations. This allows students to learn with a personalized adaptive learning.
   b. **Accuracy:** Statistical results show that the proposed e-learning system was better than a regular online course in improving student learning performance.

3) **Performance cost:** [24] There are two types of neural sorting networks with O(1) time complexity.

4) **Application:** The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations and also to use it. Unsupervised neural networks can also be used to learn representations of the input that capture the salient characteristics of the input distribution.

5) **Validation:** To confirm that the proposed system is feasible and effective, [23] conducted a pilot study on 70 freshmen.

2.2.2.2 *Fuzzy logic approach*

1) **Assessment Overview:** [25] presents modeling of diagnostic systems of taxonomies using fuzzy logic.
   a. **Rigorousness:** Not discussed in the paper.
   b. **Specificity:** Not discussed in the paper.
   c. **Automatability:** The intelligent system that is developed based on the modeling can make easier the use of diagnostic systems in education since the test correction is extremely hard and demands experts that are not always available.
   d. **Accuracy:** Not discussed in the paper.

2) **Adaptation Overview:**
   a. **Specificity:** [26] presents a case-based reasoning approach to Adaptive Web-based Educational Systems using fuzzy logic to adapt e-learning contents and contexts according to the student learning style and individual needs.

b. **Accuracy:** [26] The integration of agents with case-based reasoning methodology permits a more effective learning experience, adapting and personalizing the contents and contexts to the students' needs.

3) **Performance cost:** [27] The worst case time complexity of the fuzzy logic control is O(1).

The rate of the extraction of results is a reason for using and distributing such tools (diagnostic systems) in the educational process. In general, the computational complexity of the method grows linearly with the size of the environment, somehow limiting its scalability over very large environments.

4) **Application:** The intelligent tutoring systems are typically used in computer-based training (CBT) and don't support the collaboration and cooperation like groupware and cooperative work technologies. [26] proposes the adoption of a CBR agent in Adaptive Web based Educational Systems. The diagnostic systems of taxonomies can later be used in a variety of diagnostic systems.

5) **Validation:** In order to prove the effectiveness of the assessment tool [25], the development system was applied to 100 different high school and senior high school students, and it was tested in mathematics.

2.2.2.3 *Knowledge Algorithm based approach:* Major providers of e-learning mostly design their products to aim at corporative organizations such as the universities. They are usually large systems with support only on very standard operations, lacking subsequently knowledge-based components to intelligently optimize the online delivery.

1) **Assessment Overview:**

a. **Rigorousness:** There are a number of concrete knowledge-based algorithms [28] that would enable subject designers to effectively and efficiently select drill or assessment questions.

b. **Specificity:** Selecting assessment questions target specific learning outcome, and at the same time automatically enrich the characterization of such questions through their usage.

c. **Automatability:** Selecting assessment questions target specific learning outcome, and at the same time automatically enrich the characterization of such questions through their usage.

d. **Accuracy:** Not discussed in the paper.

2) **Adaptation overview:** Not discussed in the paper.

3) **Performance cost:** Not discussed in the paper.

4) **Application:** Not discussed in the paper.

5) **Validation:** Not discussed in the paper.

2.2.2.4 *Ontologies based approach:* Under conditions of adaptive learner-oriented e-learning an algorithm of navigation through content learning objects can be composed.

1) **Assessment Overview:** Not discussed in the paper.

2) **Adaptation Overview:** The algorithm [29] includes dynamic calculation of possible routes of knowledge acquiring. A method and algorithm of using ontologies in e-learning sessions should be used in this case.

a. **Specificity:** Not discussed in the paper

b. **Accuracy:** The present web technologies don't allow effective semantic learning objects search and supply which is done using the adaptive navigation algorithm [29].

3) **Performance cost:** Not discussed in the paper.

4) **Application:** Ontologies are used for semantic searching of learning objects, conceptual navigation through learning object set based on relations between classes of these ontologies.

5) **Validation:** Not discussed in the paper.

2.2.2.5 *Immune approach based Genetic algorithm:* A novel approach Immune Algorithm (IA) **[30]**can be applied to improve the efficiency of composing near optimal test sheet from item banks to meet multiple assessment criteria.

1) **Assessment Overview:** From the experimental results, the IA approach is desirable in composing near optimal test-sheet from large item banks than GA. And objective conceptual vector and objective test sheet item numbers can be effectually achieved.

a. **Rigorousness:** It can support the needs of precisely evaluating student's learning status. The algorithm is not completely rigorous.

   b. **Specificity:** Not discussed in the paper.
   c. **Automatability:** Not discussed in the paper.
   d. **Accuracy:** GA is defeated by the low rate of convergence during the evaluation process and the above-mentioned methods usually may not achieve the anticipated objectives. So, assessment is not accurate.

2) **Adaptation Overview:** Not discussed in the paper.

3) **Performance cost:** (1 + 1) EAs with mutation probability pm = 1/n, where n is the number of bits in a binary string (i.e., individual) and pm is the mutation probability, converges in average time O(n log n) for the ONE-MAX problem [31]

4) **Application:** Based on the quality of the results, the TSCIA [30] could be suitably applied to evaluate the students learning, and to practice in actual evaluation system.

5) **Validation:** To efficiently determine the performance of the proposed TSCIA [30], experiments were conducted to compare to GA with different objective conceptual vector (OCV). The item bank consisted of 1000 Chinese test items on the subject of ”Introduction to Computers”.

2.2.3 Psychology based approaches

2.2.3.1 <u>*Emotion estimation algorithm*</u>

1) **Assessment Overview:**
   a. **Rigorousness:** Not discussed in the paper.
   b. **Specificity:** Not discussed in the paper.
   c. **Automatability:** [32] proposes an emotion estimation algorithm from e-Learning user's facial image. The criteria used to relate an e-Learning user's emotion to a representative emotion were obtained from the time sequential analysis of user's facial expressions. So, this approach is semi-automated.
   d. **Accuracy:** Not discussed in the paper.

2) **Adaptation overview:**
   a. **Specificity:** Not discussed in the paper.

b. **Accuracy:** By examining the emotions of the e-Learning users and the positional change of the facial expressions from the experiment results, various procedures are introduced [32] to improve the estimation reliability.

3) Performance cost: In general, [33] the emotion-learning agent with the emotion learning algorithm used for finding one shortest path from the initial state to the goal state in an arbitrary deterministic environment in time which does not depend on the number of states, but only on the length of the shortest path (it is task dependent), in the worst case, depending on the position of the goal state, it can be at most O (n). The only assumption is that there is a path from the initial state to the goal state, and the maximal number of actions in each state is the constant.

4) Application: Not discussed in the paper.

5) Validation: Not discussed in the paper.

A summary of the different e-learning approaches are given in Table 2.1. The different approaches are compared with each other using parameters like performance cost, accuracy, applicability and validation.

Table 2.1. Summary table for e-learning approaches.

Algorithm Group	**Specific Algorithm**	**Performance cost**	**Accuracy**	**Application**	**Validation**
Probability and Statistics	Statistical analysis	Average case complexity is O(nlogn), worst case complexity is $O(n^2)$	Significantly better results	Business statistics, data mining, engineering, nonlinear models.	Questionnaire
Probability and Statistics	Factor analysis algorithm		Is able to make learners understand the answer better.	Online question answering [40] similar to the interactivity model of traditional teaching	Questionnaire

Table 2.1 continued

Artificial Intelligence	Artificial Neural Network	O(1)	Enhance student's learning performance	To infer a function from observations, to learn representations of the input that capture the salient characteristics of the input distribution.	A pilot study on 70 freshmen
Artificial Intelligence	Fuzzy logic	Computational complexity grows linearly with the size of the environment	A more effective learning experience.	Used in computer-based training (CBT), Adaptive Web-based Educational Systems.	Was applied to 100 different high school and senior high school students.
Artificial Intelligence	Ontologies based approach		Effective semantic learning objects search and supply.	Semantic searching of learning objects, conceptual navigation through learning object set	Not discussed in the paper
Artificial Intelligence	Immune approach based genetic algorithm	Average time complexity O(n log n) for the ONE-MAX problem	Immune Algorithm is more approximate than GA.	Could be applied to evaluate the students learning, and to practice in actual evaluation system.	Experiments were conducted to compare GA with IA
Psychology based approach	Emotion estimation algorithm	Worst case complexity O(n).	Improved the estimation reliability.		Not discussed in the paper

2.3 Electronic Game Assessment and Adaptation

[34] mentions that the tracking and assessment features of some modern LMS would give the instructor more power to control aspects of the learning experience such as when the students accessed the game, the time each student spent playing it and the outcomes of the game session.

2.3.1 Probability and Statistics

2.3.1.1 *Statistical analysis:* Statistical analysis on a game is performed to analyze the player's performance during the game and to assess whether the game served in learning the objectives.

1) Assessment Overview: [4] Simulated the game and during simulation, the data like the position of the ball, the scored goals with associated timestamps were logged every second. This will be used for statistical analysis to see the improvement in the player's performance.

- **a. Rigorousness:** Not discussed in the paper.
- **b. Specificity:** Not discussed in the paper.
- **c. Automatability:** Not discussed in the paper.
- **d. Accuracy:** The behavior described in [4] was evaluated in simulations and has shown to yield significantly better results than the old behavior, regarding goal difference and goal occlusion. The results at RoboCup 2009 show that also in real matches with opponents, the improved supporter behavior is efficient and contributes to the overall team success.

2) Adaptation Overview: Not discussed in the paper.

3) Performance cost: The complexity of statistical analysis is close to, stochastic algorithm's performance complexity, but not quite the same [17]. Las Vegas algorithm is a Stochastic/Randomized Algorithm. Las Vegas algorithm has average case complexity of O(nlogn) and worst case complexity of $O(n^2)$.

4) Application: Apart from being used in fields like business statistics, data mining, engineering, the rapid and sustained increases in computing power have had a substantial impact on the practice of statistical science.

5) Validation: The developed behavior [4] has been evaluated in a simulation based statistical analysis. The videos of the relevant matches of RoboCup 2009 have been analyzed manually.

2.3.1.2 *Stochastic algorithm based approach*

1) Assessment Overview: Each player observes only the outcome vector, namely, the payoff

[35] Obtained at every stage and the payoff that would have resulted in a different move had been played. Players build statistics of their past performance and infer what the outcome would have been if a different strategy had been played.

a. **Rigorousness:** Not discussed in the paper.
b. **Specificity:** Not discussed in the paper.
c. **Automatability:** Not discussed in the paper.
d. **Accuracy:** Not discussed in the paper.

2) Adaptation Overview: Leads to configurations where no player regrets the choices he makes. This analysis is based on techniques borrowed from the theory of stochastic algorithms and proceeds by studying an associated continuous dynamical system which represents the evolution of the players' evaluations.

a. **Specificity:** Not discussed in the paper.
b. **Accuracy:** The updating rule [35] uses instead a time average criteria that induces a specific dynamics on perceptions and strategies which appears to be structurally different from the previously studied ones, while preserving the qualitative features of probabilistic choice and sluggish adaptation.

3) Performance cost: Las Vegas algorithm which is a Stochastic/Randomized Algorithm has average case complexity of O(nlogn) and worst case complexity of O(n^2).

4) Application: [35] method is applied to the network topology which is very simple with only a set of parallel routes, and a natural extension is to consider more general networks. Parallel link structure allows modeling more complex decision problems such as the simultaneous choice of route and departure time, by using the standard trick of replacing a physical route by a set of parallel links that represent the route at different time windows.

5) Validation: In this case [35], through extensive simulation, observation is that always converges so conjectured that it is a global attractor.

2.3.2 Artificial Intelligence

2.3.2.1 *Artificial intelligence based approach:* Artificial intelligence techniques have been developed for computer games in different aspects. Online learning algorithm, which is a machine learning type of algorithm is not only applied for solving the problems of path finding, controlling the non-player characters with a variety of reactions to players in intelligent and challenging ways, but also learning the behaviors of players **[36]** .

1) **Assessment Overview:** Not discussed in the paper.

2) **Adaptation Overview:** Some techniques may be involved with dynamically adjusting the game difficulty for game balancing [5], [6] or even changing game parameters via online learning algorithms [37].

   a. **Specificity:** Not discussed in the paper.

   b. **Accuracy:** With small sample sizes, the CLT is less accurate and standard deviation more likely to fluctuate.

3) **Performance cost:** There is also a performance tradeoff between accuracy and generality/speed. If the goal was to do a correct simulation, several trials per user would need to be performed. For a best fit [5] this is not necessary. Earliest Deadline First (EDF) is an appropriate algorithm to use for online scheduling on uniform multiprocessors with a time complexity of ($n\log_2 n$).

4) **Application:** This approach [5] is best used in games where predicting events is possible in not too- distant future.

5) **Validation:** This approach is not yet validated. [5] plans to evaluate a broad sample of users interacting with the Hamlet system. A simple heart-rate monitor will be attached to a variety of players, have them play the game on both settings, and then map out the results. If the adjusted game is able to keep test subjects equally aroused over time, while dynamically adjusting to keep them alive longer, then the work will be considered a success.

2.3.2.2 *Genetic algorithm:* Genetic algorithms are used for adapting the game, so that it is entertaining and at the same time not too tough.

1) **Assessment Overview:** Not discussed in the paper.

2) **Adaptation Overview:** [38] applied both a single-objective and a multi-objective real-coded genetic algorithm (GA) to evolve tracks involving both a wide variety of turns and straights and also a large range of driving speeds.

   a. **Specificity:** Not discussed in the paper.

b. **Accuracy:** Successfully evolves tracks with a high degree of diversity both in terms of shape and achievable speeds. There is a statistically significant agreement [38] between the subjects' visual preferences and the fitness definitions. The agreement is stronger for simpler tracks and when more experienced players are considered. The results of the second survey, suggest that the tracks with a higher degree of diversity are more challenging.

3) Performance cost: For building blocks rather than single genes, both population size and time to convergence grow linearly [39]with the problem length, giving an overall quadratic time complexity in terms of fitness function evaluations.

4) Application: [38]

5) Procedural content generation (PCG) is widely applied by the game industry and its importance for the development has dramatically increased. PCG is used in games which do not reproduce a specific event but take place in a fictional game universe and thus focus on the driving experience itself.

6) Validation: A preliminary validation of the approach [38] was performed on human subjects. Two surveys were designed: one focused on the visual properties of evolved tracks, and one focused on the actual playing experience.

A summary of the different electronic character based games are given in Table 2.3. The different approaches are compared with each other using parameters like performance cost, accuracy, applicability and validation.

Table 2.2. Summary table for electronic games.

Algorithm Group	**Specific Algorithm**	**Performance cost**	**Accuracy**	**Application**	**Validation**
Probability and Statistics	Statistical analysis	Average case complexity of O(nlogn) and worst case	Significantly better results than the old behavior.	Business statistics, data mining, engineering,	Evaluated in a simulation based statistical
		complexity of O(n^2)		nonlinear models.	analysis and analyzed relevant matches videos manually
Probability and Statistics	Stochastic algorithm	Average case complexity of O(nlogn) and worst case complexity of O(n^2)	Preserves the qualitative features of probabilistic choice and sluggish adaptation.	network topology, more general networks, more complex decision problems	Extensive simulation
Artificial Intelligence	Artificial intelligence algorithm	$O(n \log_2 n)$.	Less accurate with small sample sizes	Games where predicting events is possible in the not too-distant future.	A simple heart-rate monitor will be attached to players
Artificial Intelligence	Genetic algorithm	Quadratic time complexity	Better accuracy for simpler tracks and experienced players. Others are more challenging.	In games which do not reproduce a specific event but take place in a fictional game universe.	Two surveys, one focused on the visual properties and other on the actual playing experience.

CHAPTER 3

OVERVIEW OF THE ASSESSMENT AND ADAPTATION APPROACH FOR SERIOUS EDUCATIONAL GAMES

Assessment and Adaptation are two most important aspects of serious educational games in order to assess the student's knowledge and to gradually increase the student's understanding of the course while keeping the student engaged in the game. Traditionally, the student's performance is captured and stored for assessment and adaptation before the next game play. Hence it is not automated. And most methods do assessments based on the overall scores but not at the individual concept level. The proposed Assessment-Adaptation system addresses these issues. A reusable solution is proposed which can be used in all quiz based single user games. This system does both concept-level evaluation and goal-level evaluation by doing assessment at three different degrees, quiz assessment, concept assessment, and level assessment. Traditionally, adaptation is done based on certain factors like overall class's performance. This might not help the individual student as adaptation is based on a factor derived from the overall class performance. Most of the adaptation methods either adapt the game very frequently resulting in distracting the user or adapt only during the next game play which is adaptation at a very later stage. The frequency of adaptation is either too fast or too slow. In the proposed solution, the game is adapted at the end of every level based on the concept-level and goal level evaluation results. And adaptation is based on individual student's performance and not based on any other students. In order to achieve this, adaptation is done at two degrees, difficulty adaptation and level adaptation. After every level is completed, goal level scores are updated in a scoreboard maintained on the server side which helps in ranking the students and providing feedback.

To summarize, the proposed solution provides the following key aspects which is not addressed in the traditional assessment-adaptation techniques in SEGs:

1) The Assessment-Adaptation system is re-usable in any quiz based single player SEG.

2) The Assessment-Adaptation system is highly automated.

3) Assessment is based on both the overall performance and individual concept level performance of the student.

4) Adaptation of the game is based on individual student's performance and not based on the entire set of students' performance.

5) Adaptation of the game is done at the end of every level. So, the frequency of adaptation is not too fast or slow.

3.1 Static View of the system

The components in figure 3.1 are Assessment-Adaptation system, Server, Challenge Template for quizzes and Test harness. Challenge Template for quizzes provides the questions for the quiz. Gameplay input provider sends information about the game play of the current player to the Assessment-Adaptation system for the current set of questions. The Assessment-Adaptation system has components for Assessment and Adaptation. After assessment, the scores are updated in the scoreboard in the server. Adaptation is done based on the assessment results. After adaptation, the adaptation decision for the next level is sent to challenge template for quizzes.

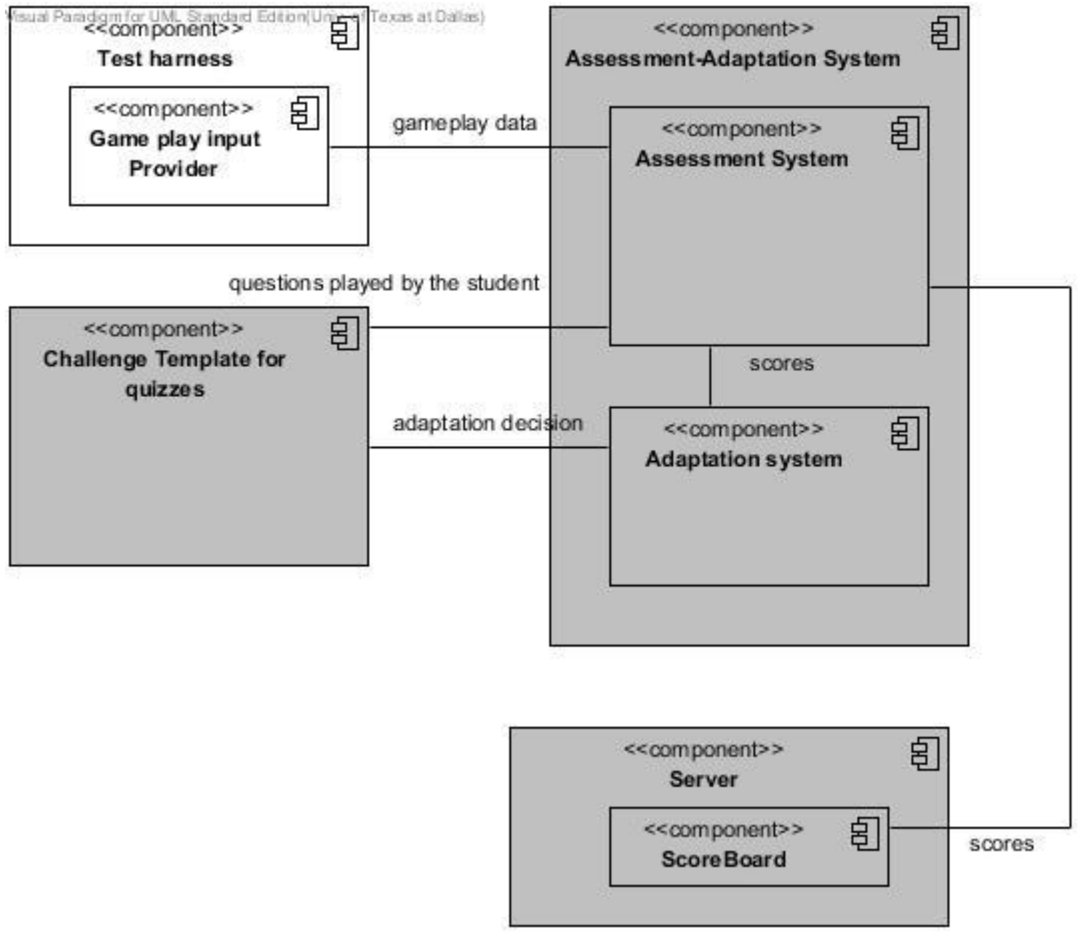

Figure 3.1. Architecture of the Assessment-Adaptation Approach.

3.2 Dynamic View of the system

The set of activates followed in the overall assessment-adaptation system is given by activity diagram in Figure 3.2 along with swim lanes segregating the activities based on the components in the architecture of the Architecture of Assessment-Adaptation system.

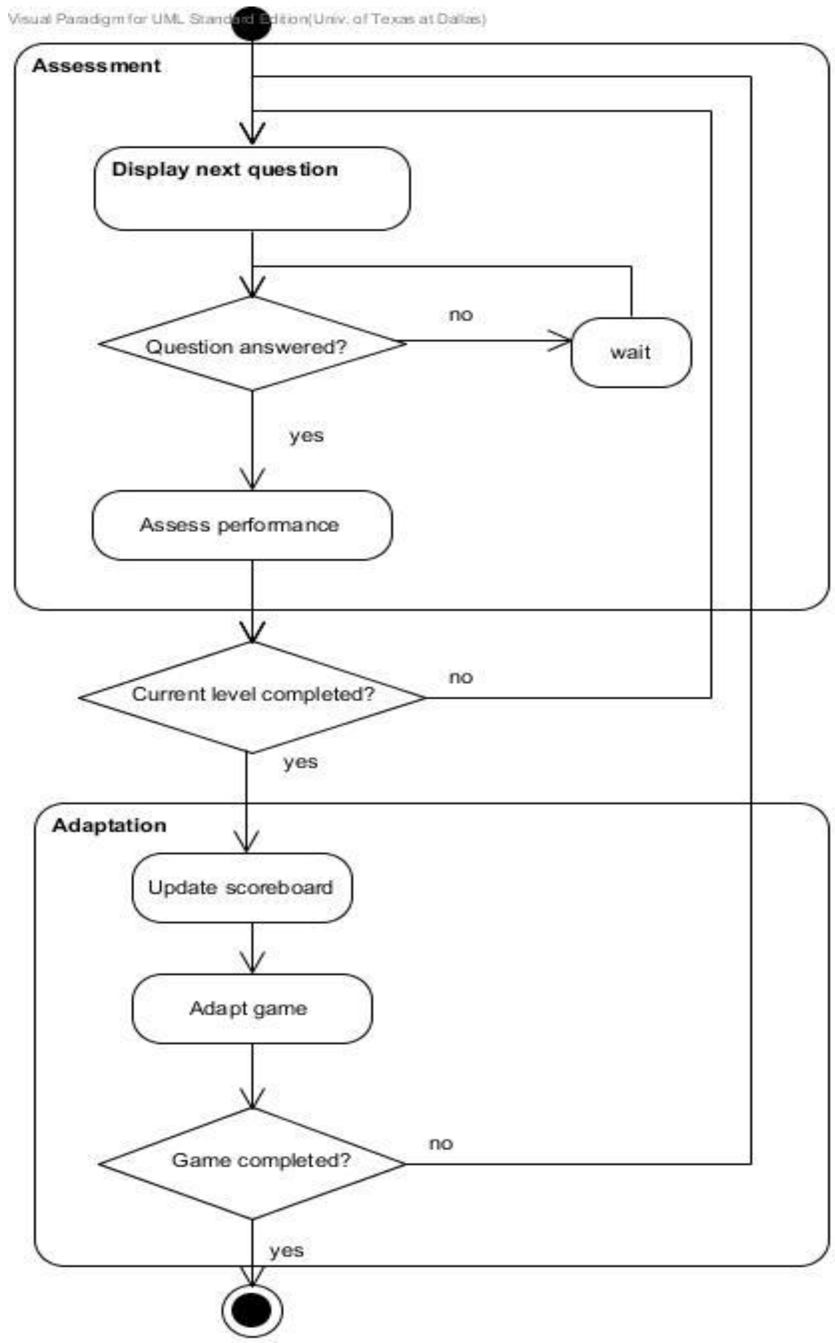

Figure 3.2. Activity diagram of Assessment-Adaptation system.

1) **Assessment Activities**
   a. **Activity 1 (Display Next Question):** The set of questions for a particular level is extracted from *'Challenge Template for quizzes'*.
   b. **Activity 2 (Assess Performance):** When questions are answered, *'Assessment'* is done which includes, quiz assessment, concept assessment and level assessment. The assessment parameters considered are, whether the selected answer is correct or moderate (needs a follow-up question), time taken to answer each question in seconds
   c. [16], [34], numbers of times answers were changed before submitting [40], etc. *'Gameplay Input provider'* in the test harness or *'Gameplay data repository'* in the actual game provides these parameters for the current set of questions. Different weights are assigned to each of these parameters leading to a total score against 100% for each concept and the entire level.
2) **Adaptation Activities**
   d. **Activity 3 (Update Scoreboard):** The results of assessment are submitted *to 'Adaptation'* and 'Score *Board'* in the server.
   e. **Activity 4 (Adapt game):** In *'Adaptation'*, difficulty of each concept and whether the student passed this level or not are decided in difficulty adaptation and level adaptation respectively. The results are passed to *'Challenge Templates for quizzes'* where the specified set of questions is selected for the next level, thus adapting the game.

CHAPTER 4

ASSESSMENT

4.1 Introduction

Assessment is done based on the game play parameters provided either by the Gameplay Input Provider in Test harness or the Game Play Data repository in the actual game.

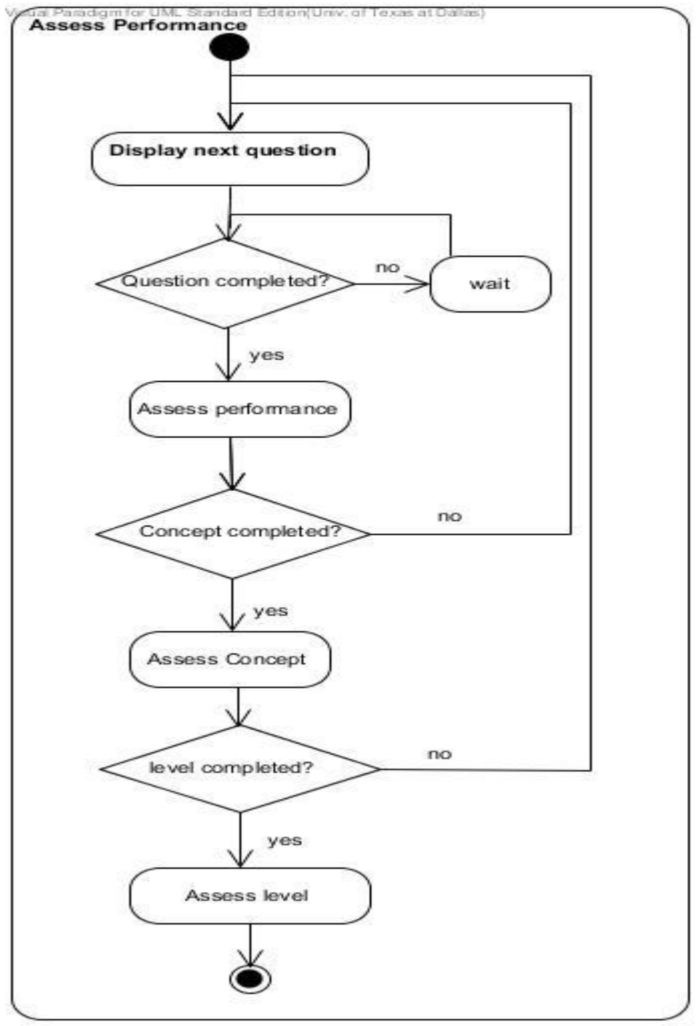

Figure 4.1. Activity diagram of Assessment system.

Figure 4.1 shows the set of activities involved in assessment.

1) **Activity 2.1 (Assess answer):** After every question is answered, *'quiz assessment'* is done where the particular answer is assessed.

2) **Activity 2.2 (Assess concept):** Similarly, when a concept is completed, *'concept assessment'* is done based on individual quiz assessments within each concept and concept level grades are obtained based on the concept level scores.

3) **Activity 2.3 (Assess level):** When an entire level is completed, *'level assessment'* is done based on individual quiz assessment across all concepts within each level. The results of level assessment are goal level grades. Concept level grades and goal level grades together result in final grades.

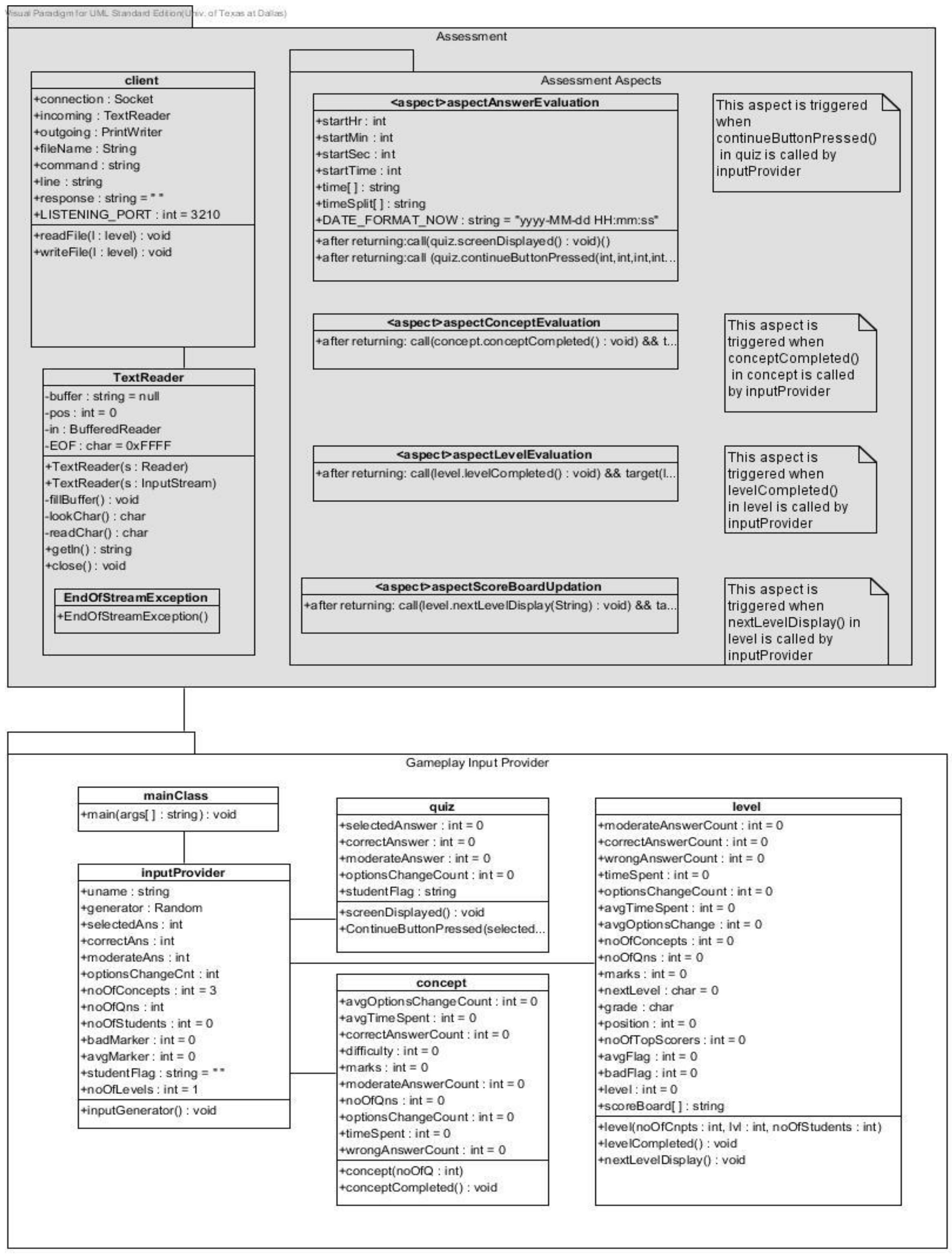
Visual Paradigm for UML Standard Edition(Univ. of Texas at Dallas)
Assessment
client
+connection : Socket
+incoming : TextReader
+outgoing : PrintWriter
+fileName : String
+command : string
+line : string
+response : string = " "
+LISTENING_PORT : int = 3210
+readFile(l : level) : void
+writeFile(l : level) : void
TextReader
-buffer : string = null
-pos : int = 0
-in : BufferedReader
-EOF : char = 0xFFFF
+TextReader(s : Reader)
+TextReader(s : InputStream)
-fillBuffer() : void
-lookChar() : char
-readChar() : char
+getln() : string
+close() : void
EndOfStreamException
+EndOfStreamException()
Assessment Aspects
<aspect>aspectAnswerEvaluation
+startHr : int
+startMin : int
+startSec : int
+startTime : int
+time[ ] : string
+timeSplit[ ] : string
+DATE_FORMAT_NOW : string = "yyyy-MM-dd HH:mm:ss"
+after returning:call(quiz.screenDisplayed() : void)()
+after returning:call (quiz.continueButtonPressed(int,int,int,int...
This aspect is triggered when continueButtonPressed() in quiz is called by inputProvider
<aspect>aspectConceptEvaluation
+after returning: call(concept.conceptCompleted() : void) && t...
This aspect is triggered when conceptCompleted() in concept is called by inputProvider
<aspect>aspectLevelEvaluation
+after returning: call(level.levelCompleted() : void) && target(l...
This aspect is triggered when levelCompleted() in level is called by inputProvider
<aspect>aspectScoreBoardUpdation
+after returning: call(level.nextLevelDisplay(String) : void) && ta...
This aspect is triggered when nextLevelDisplay() in level is called by inputProvider
Gameplay Input Provider
mainClass
+main(args[ ] : string) : void
inputProvider
+uname : string
+generator : Random
+selectedAns : int
+correctAns : int
+moderateAns : int
+optionsChangeCnt : int
+noOfConcepts : int = 3
+noOfQns : int
+noOfStudents : int = 0
+badMarker : int = 0
+avgMarker : int = 0
+studentFlag : string = " "
+noOfLevels : int = 1
+inputGenerator() : void
quiz
+selectedAnswer : int = 0
+correctAnswer : int = 0
+moderateAnswer : int = 0
+optionsChangeCount : int = 0
+studentFlag : string
+screenDisplayed() : void
+ContinueButtonPressed(selected...
concept
+avgOptionsChangeCount : int = 0
+avgTimeSpent : int = 0
+correctAnswerCount : int = 0
+difficulty : int = 0
+marks : int = 0
+moderateAnswerCount : int = 0
+noOfQns : int = 0
+optionsChangeCount : int = 0
+timeSpent : int = 0
+wrongAnswerCount : int = 0
+concept(noOfQ : int)
+conceptCompleted() : void
level
+moderateAnswerCount : int = 0
+correctAnswerCount : int = 0
+wrongAnswerCount : int = 0
+timeSpent : int = 0
+optionsChangeCount : int = 0
+avgTimeSpent : int = 0
+avgOptionsChange : int = 0
+noOfConcepts : int = 0
+noOfQns : int = 0
+marks : int = 0
+nextLevel : char = 0
+grade : char
+position : int = 0
+noOfTopScorers : int = 0
+avgFlag : int = 0
+badFlag : int = 0
+level : int = 0
+scoreBoard[ ] : string
+level(noOfCnpts : int, lvl : int, noOfStudents : int)
+levelCompleted() : void
+nextLevelDisplay() : void

Figure 4.2. Class diagram of Assessment system integrated with Test harness.

The class diagram in Figure 4.2 captures the different classes and aspects contained in the Assessment system and the Gameplay Input provider. Assessment has aspects needed for quiz assessment, concept assessment, level assessment and scoreboard updation. Client is a class which updates the scores on a scoreboard maintained in a remote server. TextReader is a class for reading data stream from the socket used for client-server communication.

Game play input provider is a part of any test harness. It has input provider which provides the data captured from the player's gameplay. And input provider calls methods which are member functions of quiz, concept and level classes. These calls trigger specific aspects. aspectAnswerEvaluation is triggered every time the player answers a single question. aspectConceptEvaluation is triggered after the player completes answering an entire concept. aspectLevelEvaluation is triggered after the player completes answering an entire game level.

The gameplay input provider captures the player's gameplay and provides for assessment. The parameters observed from the game play (Figure 5) are:

1) The option selected by the player.

2) Time taken to answer each question in seconds [16], [34]

3) Numbers of times answers were changed before submitting each answer [40]

4.2 Quiz Assessment

Evaluation of the selected answer by checking if it is correct answer, moderate answer or wrong answer.

1) Quiz Assessment Rules: Table 4.1 gives the Pseudo code for quiz assessment.

Table 4.1. Pseudo code for quiz assessment.

```
IF selectedAnswer == correctAnswer
correctAnswerCount++
ENDIF
IF selectedAnswer == moderateAnswer
moderateAnswerCount++
ENDIF
IF selectedAnswer == wrongAnswer
wrongAnswerCount++
```

Table 4.1 continued

```
ENDIF
```

2) Example: Table 4.2 gives the sample code for quiz assessment.

Table 4.2. Sample code for quiz assessment.

```
if(q.selectedAnswer == q.correctAnswer){      //If selected answer is correct
  c.correctAnswerCount ++;
  l.correctAnswerCount ++;
}else if(q.selectedAnswer == q.moderateAnswer){//If selected answer is
                                               //moderate
  c.moderateAnswerCount ++;
  l.moderateAnswerCount ++;
}else{
  c.wrongAnswerCount ++;
  l.wrongAnswerCount ++;
}
```

4.3 Concept Assessment

Calculations are done for a particular concept as mentioned below:

1) Calculated parameters: The following parameters are calculated for concept assessment from the game play data.

a. Total number of correct answers (based on quiz assessment)

b. Total number of moderate answers (based on quiz assessment)

c. Average time taken (based on the gameplay data 1).

d. Average number of times each answer was changed before submitting (based on gameplay data 2)

Based on these calculation results, concept level score is calculated by applying the rules mentioned below:

2) Score calculation rules: Table 4.3 gives the pseudo code for concept assessment.

a. The four calculated parameters above are given weight of a,b,c and d where $a+b+d = 100$ and $c < d$.

b. The weight given for *'Average time taken'* is 'a'

Table 4.3. Pseudo code for concept assessment.

```
IF avgTime < minTime
score = a
ENDIF
IF avgTime >= minTime AND <= maxTime
score = a- a/3
ENDIF
IF avgTime > maxTime
score = a-2a/3
ENDIF
```

c. The weight given for *'Average number of times answer was changed before submitting'* is 'b'

```
IF avgChangeCnt < minChangeCnt
score = b
ENDIF
IF avgChangeCnt >= minChangeCnt and avgChangeCnt <=maxChangeCnt
score = b- b/3
ENDIF
IF avgChangeCnt > maxChangeCnt
score = b-2b/3
ENDIF
```

d. The weightage given for *'Total number of moderate answers'* is 'c'
(moderateAnsCnt/noOfQns)*c

e. The weightage given for *'Total number of correct answers'* is 'd'
(correctAnsCnt/noOfQns)* d

3) Example: Following is a discussion of an example of score calculation. Table 4.4 gives the sample code for concept assessment.

a. Average time taken is given 10% (If the average time spent is less than 20 seconds, the player is awarded 10 points, if the average time spent is [20-40] seconds, player is awarded 6 points, if the average time spent is (40-60] seconds, player is awarded 2 points and if the average time is greater than 60 seconds, the player is not awarded any points)

Table 4.4. Sample code for concept assessment.

```
/*Assign marks based on average time spent on the concept*/
int minTime = 20;
int avgTime = 40;
int maxTime = 60;
int minScore = 2;
int avgScore = 6;
int maxScore = 10;
if(c.avgTimeSpent < minTime){
  c.marks += maxScore;
}else if(c.avgTimeSpent >= minTime && c.avgTimeSpent <= avgTime){
  c.marks += avgScore;
}else if(c.avgTimeSpent > avgTime && c.avgTimeSpent <= maxTime){
  c.marks += minScore;
}
```

b. Average number of times answer was changed before submitting is given 10% (If the average option change count is 0, the player is awarded 10 points, if the average option change count is 1, player is awarded 6 points, if the average option change count is 2, player is awarded 2 points and for any average option change count greater than 2, the player is not awarded any points)

Table 4.4 continued

```
/*Assign marks based on average number of times options are changed on the
        concept*/
int minCount = 0;
int avgCount = 1;
int maxCount = 2;
int minScore = 2;
int avgScore = 6;
int maxScore = 10;
if(c.avgOptionsChangeCount == minCount){
  c.marks += maxScore;
}else if(c.avgOptionsChangeCount == avgCount ){
  c.marks += avgScore;
}else if(c.avgOptionsChangeCount == maxCount){
  c.marks += minScore;
}
```

c. Total number of moderate answers is given 35%

d. Total number of correct answers is given 80%

4.4 Level Assessment

Calculations are done for a particular level as mentioned below:

1) Calculated parameters: The following parameters are calculated for level assessment from game play data:

a. Total number of correct answers (based on quiz assessment)

b. Total number of moderate answers (based on quiz assessment)

c. Average time taken (based on the observed parameter 1).

d. Average number of times each answer was changed before submitting (based on observed parameter 2)

Based on these calculation results, goal level score is calculated by applying the rules mentioned below:

2) **Score calculation rules:** Table 4.5 gives the pseudo code for level assessment.

   a. The four calculated parameters above are given weight of a, b, c and d where a+b+d = 100 and c < d.

   b. The weight given for *'Average time taken'* is 'a'

Table 4.5. Pseudo code for level assessment.

```
IF avgTime < minTime
score = a
ENDIF
IF avgTime >= minTime AND <=maxTime
score = a- a/3
ENDIF
IF avgTime > maxTime
score = a-2a/3
ENDIF
```

   c. The weightage given for *'Average number of times answer was changed before submitting'* is 'b'

```
IF avgChangeCnt < minChangeCnt
score = b
ENDIF
IF avgChangeCnt >= minChangeCnt and avgChangeCnt <=maxChangeCnt
score = b- b/3
ENDIF
IF avgChangeCnt > maxChangeCnt
score = b-2b/3
ENDIF
```

   d. The weightage given for *'Total number of moderate answers'* is 'c'

   (moderateAnsCnt/noOfQns)*c

e. The weightage given for *'Total number of correct answers'* is 'd'
(correctAnsCnt/noOfQns)* d

The parameters considered in goal level score calculation are parameters measured for the entire level. They are average time taken to complete the entire level, average number of times options were changed in the entire level, number of moderate answers in the entire level, total number of correct answers in the entire level.

3) Example: Following is a discussion of an example of score calculation. Table 4.6 shows the sample code for level assessment.

a. Average time taken is given 10% (If the average time spent is less than 20 seconds, the player is awarded 10 points, if the average time spent is [20-40] seconds, player is awarded 6 points, if the average time spent is (40-60] seconds, player is awarded 2 points and if the average time is greater than 60 seconds, the player is not awarded any points)

Table 4.6. Sample code for level assessment.

```
/*Assign marks based on average time spent on the level*/
int minTime = 20;
int avgTime = 40;
int maxTime = 60;
int minScore = 2;
int avgScore = 6;
int maxScore = 10;
if(c.avgTimeSpent < minTime){
  c.marks += maxScore;
}else if(c.avgTimeSpent >= minTime && c.avgTimeSpent <= avgTime){
  c.marks += avgScore;
}else if(c.avgTimeSpent > avgTime && c.avgTimeSpent <= maxTime){
  c.marks += minScore;
}
```

b. Average number of times answer was changed before submitting is given 10% (If the average option change count is 0, the player is awarded 10 points, if the average option change count is 1, player is awarded 6 points, if the average option change count is 2, player is awarded 2 points and for any average option change count greater than 2, the player is not awarded any points)

Table 4.6 continued

```
/*Assign marks based on average number of times options are changed on the
    level*/
int minCount = 0;
int avgCount = 1;
int maxCount = 2;
int minScore = 2;
int avgScore = 6;
int maxScore = 10;
if(c.avgOptionsChangeCount == minCount){
  c.marks += maxScore;
}else if(c.avgOptionsChangeCount == avgCount ){
  c.marks += avgScore;
}else if(c.avgOptionsChangeCount == maxCount){
  c.marks += minScore;
}
```

c. Total number of moderate answers is given 35%

d. Total number of correct answers is given 80%

4.5 Results

After every level is completed by every student, the goal level score obtained by the player is updated in the server at the correct position(in rank order).After all of the students complete all the levels, a bar chart of their performance in every level based on goal level scores can be viewed from the server side. A sample bar chart produced for a sample of 100 students is given in Figure 4.3.

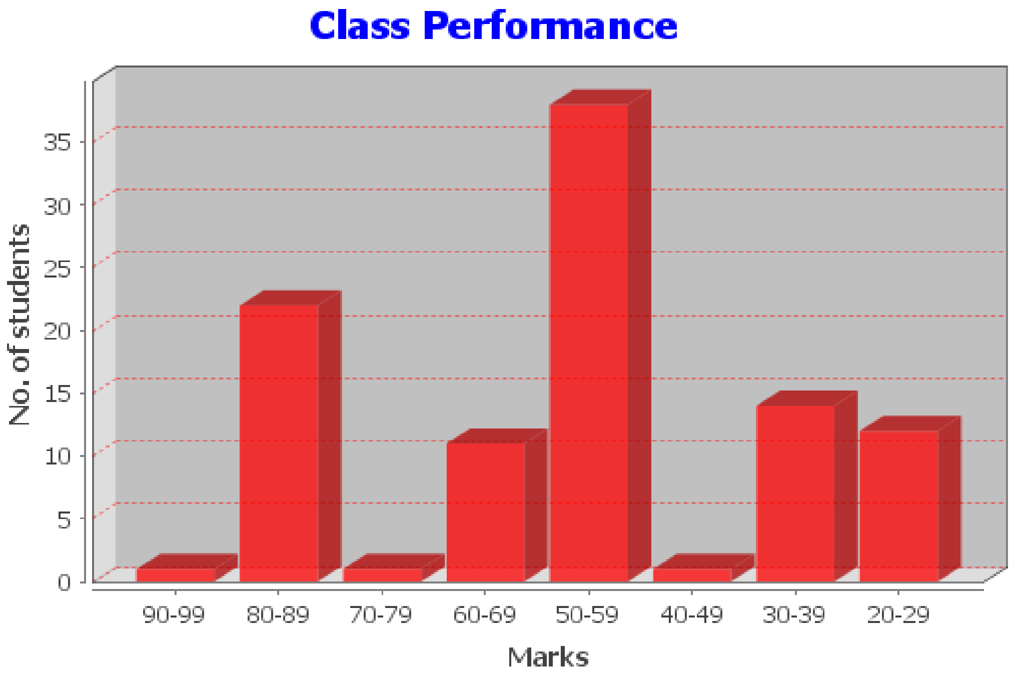

Figure 4.3. Bar chart for the performance of 100 student samples.

4.6 Discussion

Quiz assessment is done for every quiz to assess the correctness of the answer. Concept assessment is assessment of player's performance in a particular concept and it is done for every concept. Similarly, level assessment is assessment of player's performance in a particular level and it is done for every level.

CHAPTER 5

ADAPTATION

5.1 Introduction

Adaptation is done in two stages (Figure 5.1), Difficulty adaptation and Level adaptation.

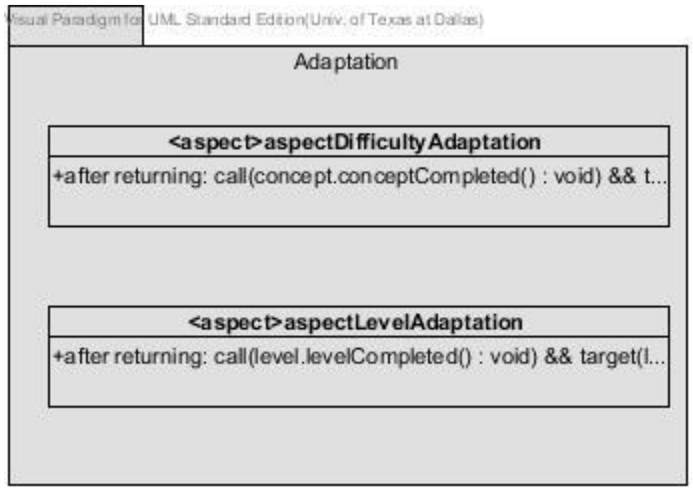
Visual Paradigm for UML Standard Edition(Univ. of Texas at Dallas)
Adaptation
<aspect>aspectDifficultyAdaptation
+after returning: call(concept.conceptCompleted() : void) && t...
<aspect>aspectLevelAdaptation
+after returning: call(level.levelCompleted() : void) && target(l...

Figure 5.1. Class diagram of Adaptation system.

After score calculation in assessment, the student is graded based on the performance in both concept level and goal level. These concept level and goal level grades result in final grades.

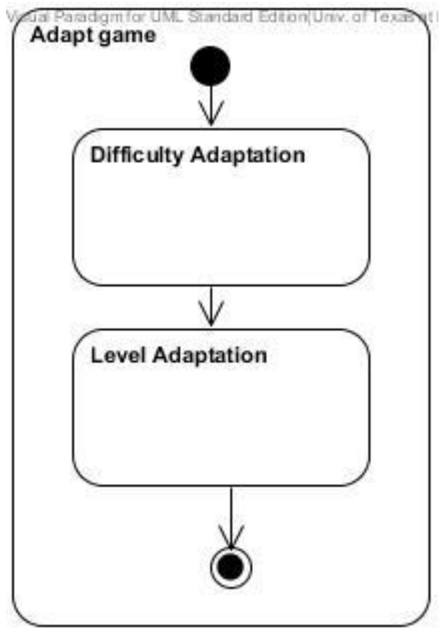

Figure 5.2. Activity diagram of Adaptation system.

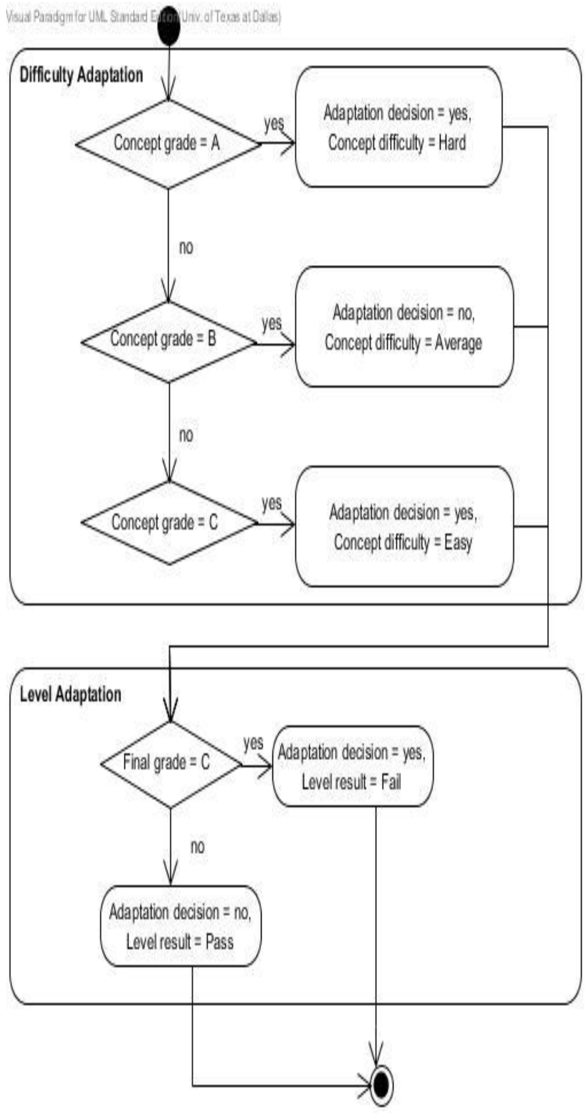
Visual Paradigm for UML Standard Edition(Univ. of Texas at Dallas)
Difficulty Adaptation
Concept grade = A
yes
Adaptation decision = yes, Concept difficulty = Hard
no
Concept grade = B
yes
Adaptation decision = no, Concept difficulty = Average
no
Concept grade = C
yes
Adaptation decision = yes, Concept difficulty = Easy
Level Adaptation
Final grade = C
yes
Adaptation decision = yes, Level result = Fail
no
Adaptation decision = no, Level result = Pass

Figure 5.3. Activity diagram of Difficulty and Level adaptation.

Adaptation consists of two activities, Difficulty adaptation and Level adaptation (Figure 5.2). Difficulty adaptation and Level adaptation are composed of a set of activities (Figure 5.3).

1) **Activity 4.1 (Difficulty Adaptation):** After a level is completed, Concept level grades are used for difficulty adaptation.
2) **Activity 4.2 (Level Adaptation):** After a level is completed, final grades are used for level adaptation.

5.2 Difficulty Adaptation

1) **Concept level grade calculation rules:** MaxThreshold is the score above which the student's performance is considered to be good indicated by a grade 'A'. MinThreshold is the score below which the student's performance is considered to be bad indicated by a grade 'C'. If the student's performance is average, the grade is 'B'. Table 5.1 shows the pseudo code for concept level grade calculation.

Table 5.1. Pseudo code for concept level grade calculation.

```
IF score > maxThreashold
Grade = A
ENDIF
IF score >= minThreashold AND score <= maxThreashold
Grade = B
ENDIF
IF score < maxThreashold
Grade = C
ENDIF
```

2) **Example**: As specified by the grade calculation rules above, concept level grade is calculated based on concept level score as follows:

   a. If concept level score is above 50%, then concept level grade is 'A'
   b. If concept level score is 30%-50%, then concept level grade is 'B'
   c. If concept level score is below 30%, then concept level grade is 'C'

   Table 5.2 shows the sample code for concept level grade calculation.

Table 5.2. Sample code for concept level grade calculation.

```
/*Adaptation decision for the difficulty of the same concept based on the player's performance in the
current level(good or avg or bad)*/
int goodScore = 50;
int badScore = 30;
if(c.marks > goodScore ){
  c.grade = 'A';
}else if(c.marks>= badScore && c.marks <= goodScore){
   c.grade = 'B';
}else{
   c.grade = 'C';
}
```

3) Difficulty adaptation rules: Every concept is provided with average difficulty questions in every level by default unless there is need for adaptation based on the student's performance in the previous level. This 'Average' difficulty level is maintained as long as the student's performance is average. That is, the student gets a grade of 'B' in the concept. If the students gets a grade of 'A' in the concept, difficulty of that concept in the next level is adapted to 'Hard'. If the students gets a grade of 'C' in the concept, the difficulty of that concept in the next level is adapted to 'Easy. Table 5.3 shows the pseudo code for difficulty adaptation.

Table 5.3. Pseudo code for difficulty adaptation.

```
IF grade == goodGrade
Difficulty = moreDiffiuclty
ENDIF
IF grade == avgGrade
Difficulty = avgDifficulty
ENDIF
IF grade == badGrade
Difficulty = lessDifficutly
ENDIF
```

Table 5.4 shows the sample code for difficulty adaptation.

Table 5.4. Sample code for difficulty adaptation.

```
/*Adaptation decision for the difficulty of the same concept based on the player's performance in the
current level(good or avg or bad)*/
int hard = 2;
int avg = 1;
int easy = 0;
if(c.grade == ‘A’){
  c.difficulty = hard;
}else if(c.grade == ‘B’){
  c.difficulty = avg;
}else{
   c.difficulty = easy;
 }
```

5.3 Level Adaptation

1) Goal level grade calculation rules: Table 5.5 shows the pseudo code for goal level grade calculation.

Table 5.5. Pseudo code for goal level grade calculation.

```
IF score > maxThreashold
Grade = A
ENDIF
IF score >= minThreashold AND score <= maxThreashold
Grade = B
ENDIF
IF score < maxThreashold
Grade = C
ENDIF
```

2) Example: As specified by the grade calculation rules above, goal level grade is calculated based on goal level score as follows:

a. If goal level score is above 70%, then goal level grade is ‘A’

b. If goal level score is 50%-70%, then goal level grade is ‘B’

c. If goal level score is below 50%, then goal level grade is ‘C’

Table 5.6 shows the sample code for goal level grade calculation.

Table 5.6. Sample code for goal level grade calculation.

```
//Assign goal level grade based on the player's overall performance
int goodScore = 70;
int badScore = 50;
if(l.marks < badScore){   //If the player's goal level score is bad
  l.grade = 'C';
}else if (l.marks >= badScore && l.marks <= goodScore){ //If the player's goal level
                                                   //score is average
  l.grade = 'B';
}else{ //If the player's goal level score is  good
  l.grade = 'A';
}
```

3) Final grade calculation rules: Minimum of all the concept level and goal level grades. If the student gets concept level grade of ‘A’ in all concepts and also a goal level grade ‘A’, then the final grade is ‘A’. If the student gets concept level grade of ‘B’ in atleast one of the concepts or a goal level grade ‘B’, but not a grade of ‘C’ in any of them, then the final grade is ‘B’. If the student gets concept level grade of ‘C’ in atleast one of the concepts or a goal level grade ‘C’, then the final grade is ‘C’. Table 5.7 shows the pseudo code for final grade calculation.

Table 5.7. Pseudo code for final grade calculation.

```
IF anyConceptGrade == badGrade OR goalGrade == badGrade
finalGrade = badGrade
ENDIF
IF anyConceptGrade == avgGrade OR goalGrade == avgGrade
```

Table 5.7 continued

```
finalGrade = avgGrade
ENDIF
ELSE
finalGrade = goodGrade
ENDIF
```

Table 5.8 shows the pseudo code for final grade calculation.

Table 5.8. Sample code for final grade calculation.

```
/*To calculate final grade and to make adaptation decision whether the player passed the level*/
for(int j=0; j<l[i].noOfConcepts; j++){
  if(c[j].grade == ‘B’){
    l[i].avgFlag = 1;
  }else if (c[j].grade ==’C’){
    l[i].badFlag = 1;
  }
}
/*Assign final grade and whether the player passed this level based on the player's performance in every
concept and goal level*/
if(l.badFlag == 1 || l.grade == 'C'){
  l.finalGrade = 'C';
}else if (l.avgFlag == 1 || l.grade =='B'){
  l.finalGrade = 'B';
}else{
  l.finalGrade = 'A';
}
```

4) Level Adaptation rules: Every student is assumed to have passed the previous level by default unless there is need for adaptation based on the student’s performance in the previous level. If the student gets a final grade of ‘C’, the student has not passed this level to go to the next level. The questions are selected from the same level as the one just completed by the student (i.e.) the student is basically made to repeat the level when the score is ‘bad’. Table 5.9 shows the pseudo code for level adaptation.

Table 5.9. Pseudo code for level adaptation.

```
IF finalGrade == goodGrade
Increment the level
ENDIF
IF grade == avgGrade
Increment the level
ENDIF
IF grade == badGrade
Do not increment the level
ENDIF
```

Table 5.10 shows the sample code for level adaptation.

Table 5.10. Sample code for level adaptation.

```
/*Decide whether the player passed this level based on the player's final grade*/
if (l.finalGrade =='A'){
  l.nextLevel ++;    //passed this level
}else if (l.finalGrade =='B'){
  l.nextLevel ++;    //passed this level
}
```

5.4 Results

In the xml template for questions, the questions are organized into different levels and different concepts within each level and different difficulty within each concept and different questions in each difficulty. A sequential algorithm is used for selecting the questions once the level and difficulty are decided based on adaptation. A sample xml template for the quizzes is as follows:

```
<Level id=1>
  <concept name="Analysis_Phase">
              <difficulty name="hard">
```

```
        <question id=1>ddddd?</question>
        <correct_answer>a</correct_answer>
        <moderate_answer>
                <right_answer>b</right_answer>
                <wrong_answer>c</wrong_answer>
        </moderate_answer>
        <incorrect_answer>d</incorrect_answer>
        <incorrect_answer>e</incorrect_answer>
</difficulty>

<difficulty name="easy">
        <question id=1>eeeeee?</question>
        <correct_answer>a</correct_answer>
        <moderate_answer>
                <right_answer>b</right_answer>
                <wrong_answer>c</wrong_answer>
        </moderate_answer>
        <incorrect_answer>d</incorrect_answer>
        <incorrect_answer>e</incorrect_answer>
</difficulty>
<difficulty name="average">
        <question id=1>aaaaa?</question>
        <correct_answer>a</correct_answer>
        <moderate_answer>
                <right_answer>b</right_answer>
                <wrong_answer>c</wrong_answer>
        </moderate_answer>
        <incorrect_answer>d</incorrect_answer>
        <incorrect_answer>e</incorrect_answer>
```

```
          </difficulty>
  </concept>
</Level>
```

In the above example, let us assume an example where the student has completed level 0. If the student's final grade is A, then the questions in level 1 will be selected for the next level, as the student has passed this level. Also, hard level questions are selected from all concepts as the score in each concept is above average. In the above example, question, 'ddddd?' with the respective options will be displayed in the next level. Suppose the student's scores are average in all concept-level scores and goal-level score, then the adaptation decision is to not adapt, resulting in average level questions selected from the next level. That is, question, 'aaaaa?' with the respective options.

5.5 Discussion

Assessment results in three types of grades: concept level grades, goal level grades and the final grades. Final grade is the assessment results based on individual concept level grades and the goal-level grades. Concept level grades help in adapting each concept's difficulty level. Final grade decides if the player has passed the level or if the player has to repeat the same level again.

CHAPTER 6

VALIDATION

Validation was done by integrating the Assessment-Adaptation system with a test-harness which replicates the properties of a quiz-based game and the gameplay of students.

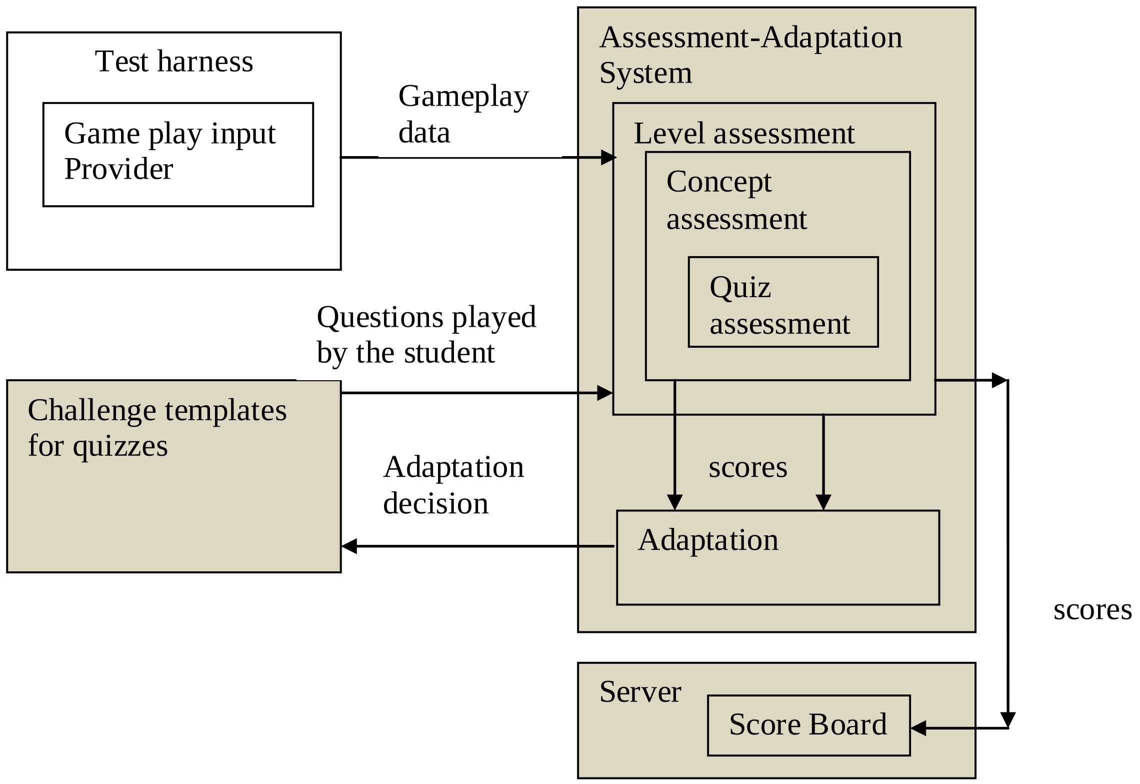

Figure 6.1. Integration of Assessment Adaptation System with Test harness.

As shown in Figure 6.1, Test harness emulates a player’s gameplay. So, various data that will be captured when a player plays a game are generated by game play input provider in test harness. This game play is passed to the Assessment system. Assessment system has components like level assessment, Concept assessment and quiz assessment. Assessment results are updated in the scoreboard maintained on a remote server and also sent to the adaptation system. Adaptation system makes adaptation decision on the difficulty and passes the decision back to the quiz template for selecting the right set of questions for the next level.

6.1 Validation Goal

The goal is to validate the adaptation system. Analyzing how the change in scores affects the adaptation decision, helps in verifying whether adaptation is specific to an individual player's performance or not.

1) Input: In order to supply input to the validation, mutation based input generation method is used. This method of input generation generates gameplay data for different types of students.

2) Method: Sensitivity analysis (SA) is the study of how the variation in the output of a statistical model can be attributed to different variations in the inputs of the model. It can be used to analyze how the change in scores affects the adaptation decision. The scores and adaptation decision are obtained by running the input through assessment-adaptation system.

3) Output: Probability Density Function (pdf) is a good way of representing the validation results. It represents the probability of the game being adapted and the probability that the game is not adapted.

In order to validate the system, both the scores achieved and the adaptation decision taken for each of these scores should be measured. Regular tables are another way of representing the validation results. They map the student's scores with the corresponding adaptation decision. Thus tables show the changes in the adaptation decision with the student's scores.

4) Success criteria: If the adaptation decision changes when the score range crosses a threshold, the validation result is success. Also, neither the probability of adapting the game nor the probability of not adapting the game should be zero, if the validation result should be successful.

6.2 Steps for validation

1) Mutation based input generation: This method of input generation generates gameplay data for different types of students. The input should emulate the performance of a good performance, average performance and bad performance. This can be done by adjusting the different parameters like number of correct answers, number of moderate answers, number of wrong answers, time taken to answer the question, number of times options were changed. A simulation was run to replicate the performance of 100 students, thus generating mutation based input.

2) **Run Assessment and Adaptation system:** Use the mutation based input that is generated in step 1 as input to the assessment adaptation system. Log the scores achieved by each of the sample in every concept and also overall. Also log the adaptation decision taken for each score.

3) **Perform sensitivity analysis:** Sensitivity analysis (SA) is the study of how the variation (uncertainty) in the output of a statistical model can be attributed to different variations in the inputs of the model. In other words, it is a technique for systematically changing variables in a model to determine the effects of such changes.

From the data that is logged, map the output (adaptation decision) to the input (scores) to analyze how much change in input actually makes a change in output. That is to analyze how the change in scores affects the adaptation decision. This proves that the adaptation is actually very specific to an individual player's performance and also very rigorous as it taken into account every concept and overall score. Sensitivity analysis tests the usability of the system.

4) **Calculate Probability Density Function:** In order to graphically represent the adaptation decision, (i.e.) to represent the probability of the game being adapted and the probability that the game is not adapted, probability density function is calculated.

   a) **Calculate probability of score:** For simplicity, consider a range of 10 points in score as a single data point. E.g.: Any score in the range of 20-29 is considered in a single data point 20. So, the data points considered are {0,10,20,30,40,50,60,70,80,90}. The probability at each data point is calculated by the ratio of number of students in that particular score range to the total number of students in the class. P(x) =Number of students at score range x/Total number of students in the class. E.g.: P(20) = 12/100 = 0.12 if number of students in the range 20-29 is 12.

   b) **Calculate mean:** Based on [41], [42], Mean is

   $$\mu = \sum xP(x)$$

   c) **Calculate standard deviation:** Based on [41], [42], Standard deviation is

   $$\sigma = \sqrt{\sum_{i=1}^{N} p_i(x_i - \mu)^2}, \quad \text{where} \quad \mu = \sum_{i=1}^{N} p_i x_i .$$

d) **Calculate Probability Density Function (pdf):** In probability theory, the normal (or Gaussian) distribution is a continuous probability distribution that has a bell-shaped probability density function, known as the Gaussian function or informally the bell curve. Since the performance of students in a class is a normal distribution and also the set of samples observed proves the distribution to be normal, the following formula is used to calculate Probability Density Function.

Using the mean and standard deviation obtained, Probability Mass Function (for discrete sample values) can be calculated using the same formula as Probability Density Function (for continuous sample values) in Mat lab.

Based on [41], [42], Probability Density Function is

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where parameter μ is the mean or expectation (location of the peak) and σ^2 is the variance. σ is known as the standard deviation.

6.3 Validation results

After running the assessment-adaptation system with the mutation input generated, scores achieved are mapped to the adaptation decision taken by the adaptation system in the tables 6.1, 6.2, 6.3, 6.4 given below. These tables are proof that the adaptation decision changes with the student' performance.

6.3.1 Goal level scores

The goal level scores obtained from the simulation, number of students at each score, goal level grades and the adaptation decision are captured in Table 6.1.

Table 6.1. Goal level scores of sample data and the adaptation decision.

Score(Input)	Number of students	Grade	Adaptation decision
22	1	C	Repeat level
23	2	C	Repeat level
24	3	C	Repeat level
26	4	C	Repeat level
29	2	C	Repeat level

Table 6.1. continued

30	1	C	Repeat level
31	2	C	Repeat level
32	2	C	Repeat level
33	2	C	Repeat level
34	3	C	Repeat level
35	1	C	Repeat level
36	1	C	Repeat level
37	1	C	Repeat level
38	1	C	Repeat level
48	1	C	Repeat level
50	2	B	Decision depends on concept level scores
51	1	B	Decision depends on concept level scores
52	4	B	Decision depends on concept level scores
53	1	B	Decision depends on concept level scores
54	1	B	Decision depends on concept level scores
55	7	B	Decision depends on concept level scores
56	4	B	Decision depends on concept level scores
57	5	B	Decision depends on concept level scores
58	6	B	Decision depends on concept level scores
59	7	B	Decision depends on concept level scores
60	3	B	Decision depends on concept level scores
61	2	B	Decision depends on concept level scores
62	2	B	Decision depends on concept level scores
63	1	B	Decision depends on concept level scores
64	1	B	Decision depends on concept level scores
65	1	B	Decision depends on concept level scores
68	1	B	Decision depends on concept level scores
78	1	A	Decision depends on concept level scores

Table 6.1. continued

80	1	A	Decision depends on concept level scores
81	3	A	Decision depends on concept level scores
82	4	A	Decision depends on concept level scores
83	5	A	Decision depends on concept level scores
84	1	A	Decision depends on concept level scores
85	3	A	Decision depends on concept level scores
86	1	A	Decision depends on concept level scores
87	1	A	Decision depends on concept level scores
88	3	A	Decision depends on concept level scores
92	1	A	Decision depends on concept level scores

It is identified that a score of 48, calls for repetition of the same level but a score of 50 doesn't call for repetition as long as the student does not get a 'C' grade in one of the concepts. This shows how the change in input (score) affects the output (adaptation decision).

Applying the above formulae on the set of data in the table, we get the
Mean = 51 and Standard deviation = 19.57

By plotting the Probability Density Function of the goal level scores, Figure 6.2 is obtained.

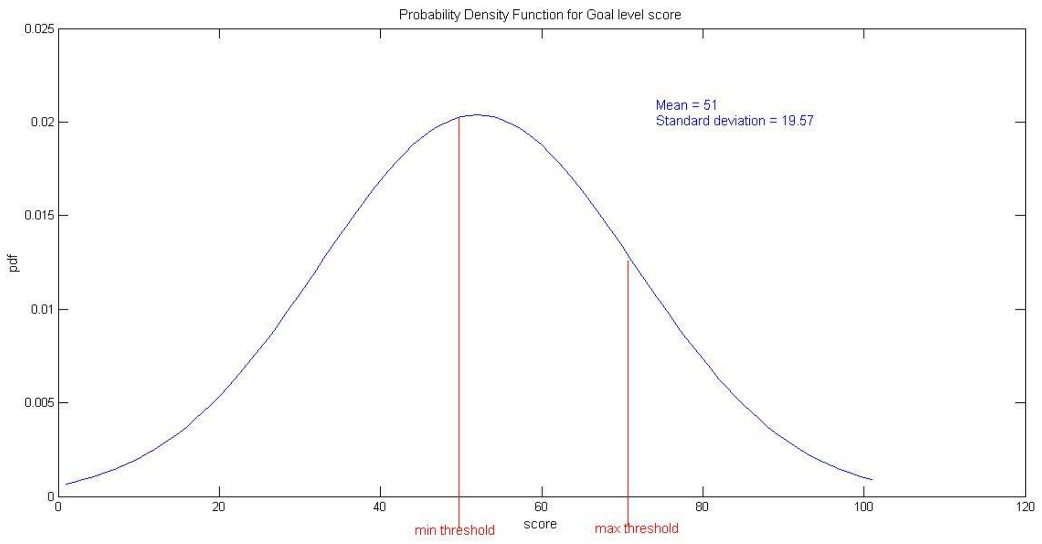

Figure 6.2. Probability Density Function of goal level scores of sample data.

For the goal level score, the minimum threshold is 50 and maximum threshold is 70. Any student whose score lies below the minimum threshold will get a goal level grade C and that student is made to repeat the same level again. Any student whose score lies above maximum threshold will get a goal level grade 'A'. The students whose goal level score is in between will get a goal level 'B' grade. The students with goal level grades 'A' or 'B' will not have to repeat the same level as long as they do not get a 'C' grade in one of the concepts.

6.3.2 Concept I scores

The Concept I scores obtained from the simulation, number of students at each score, Concept I grades and Adaptation decision are captured in Table 6.2.

Table 6.2. Concept I scores of sample data and the adaptation decision.

Score(Input)	Number of students	Grade	Adaptation Decision(Output)
17	1	C	Yes- to lower difficulty, Repeat level
18	1	C	Yes- to lower difficulty, Repeat level
19	1	C	Yes- to lower difficulty, Repeat level
20	1	C	Yes- to lower difficulty, Repeat level
21	1	C	Yes- to lower difficulty, Repeat level
23	2	C	Yes- to lower difficulty, Repeat level
24	1	C	Yes- to lower difficulty, Repeat level
26	4	C	Yes- to lower difficulty, Repeat level
27	1	C	Yes- to lower difficulty, Repeat level
28	2	C	Yes- to lower difficulty, Repeat level
30	1	B	No
32	1	B	No
33	1	B	No
34	2	B	No
35	1	B	No
36	1	B	No
38	1	B	No
40	1	B	No

Table 6.2. continued

45	1	B	No
46	1	B	No
47	2	B	No
49	1	B	No
50	3	B	No
51	2	A	Yes- to higher difficulty
52	3	A	Yes- to higher difficulty
53	1	A	Yes- to higher difficulty
54	8	A	Yes- to higher difficulty
55	3	A	Yes- to higher difficulty
57	5	A	Yes- to higher difficulty
58	1	A	Yes- to higher difficulty
60	3	A	Yes- to higher difficulty
61	6	A	Yes- to higher difficulty
62	1	A	Yes- to higher difficulty
63	2	A	Yes- to higher difficulty
65	3	A	Yes- to higher difficulty
66	3	A	Yes- to higher difficulty
69	1	A	Yes- to higher difficulty
70	1	A	Yes- to higher difficulty
75	2	A	Yes- to higher difficulty
77	2	A	Yes- to higher difficulty
80	3	A	Yes- to higher difficulty
82	3	A	Yes- to higher difficulty
83	2	A	Yes- to higher difficulty
84	2	A	Yes- to higher difficulty
86	1	A	Yes- to higher difficulty
87	1	A	Yes- to higher difficulty

Table 6.2. continued

88	1	A	Yes- to higher difficulty
89	2	A	Yes- to higher difficulty
91	4	A	Yes- to higher difficulty
92	1	A	Yes- to higher difficulty
94	1	A	Yes- to higher difficulty

It is identified that a score of 28, calls for adaptation (to a ‘lower’ difficulty level) in the next level but a score of 30 doesn’t. Similarly, a score of 50 doesn’t require adaptation in the next level but a score of 51 needs adaptation (to a ‘higher’ difficulty level). This shows how the change in input (score) affects the output (adaptation decision).

Applying the formulae mentioned before, on the set of data in the table,
Mean = 52.8 and Standard deviation = 21.35.

By plotting the Probability Density Function of the Concept I scores, Figure 6.3 is obtained.

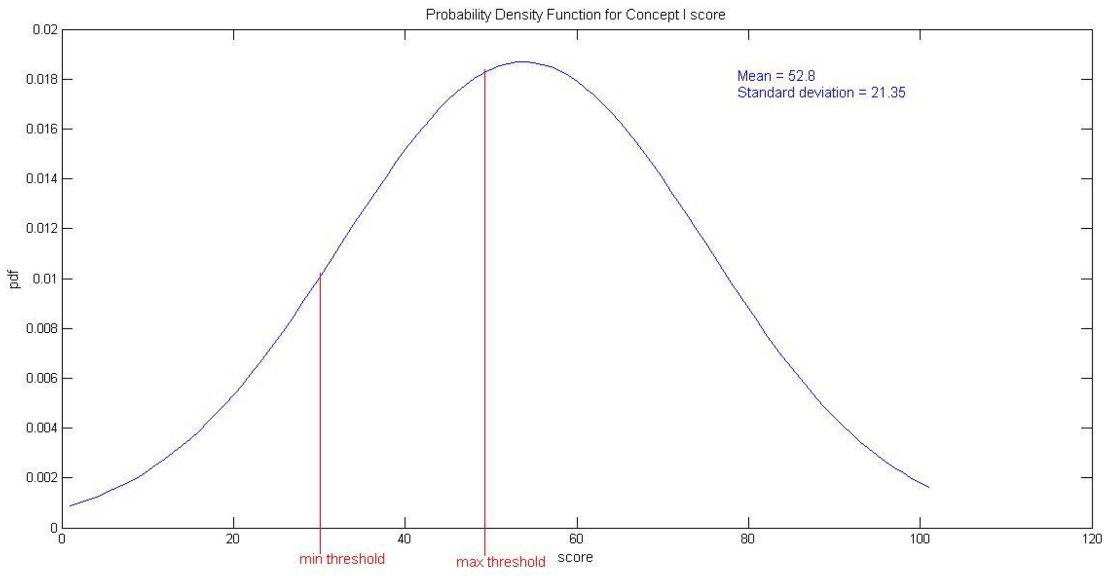

Figure 6.3. Probability Density Function of Concept I scores of sample data.

For the concept wise score, minimum threshold is 30 and maximum threshold is 50. The game will be adapted for Concept I for any student whose score lies below the minimum threshold (adapted to a ‘lower’ difficulty level) or above maximum threshold (adapted to a ‘lower’ difficulty level).

6.3.3 Concept II scores

The Concept II scores obtained from the simulation, number of students at each score, Concept II grades and adaptation decision are captured in Table 6.3.

Table 6.3. Concept II of sample data and the adaptation decision.

Score(Input)	Number of students	Grade	Adaptation Decision(Output)
17	1	C	Yes- to lower difficulty, repeat level
19	1	C	Yes- to lower difficulty, repeat level
21	1	C	Yes- to lower difficulty, repeat level
23	1	C	Yes- to lower difficulty, repeat level
24	1	C	Yes- to lower difficulty, repeat level
25	2	C	Yes- to lower difficulty, repeat level
27	2	C	Yes- to lower difficulty, repeat level
30	2	B	No
31	1	B	No
32	1	B	No
33	1	B	No
34	2	B	No
36	1	B	No
38	1	B	No
39	1	B	No
40	2	B	No
41	1	B	No
42	1	B	No
43	3	B	No
46	1	B	No
47	1	B	No
48	1	B	No

Table 6.3. continued

49	2	B	No
52	2	A	Yes- to higher difficulty
54	4	A	Yes- to higher difficulty
55	4	A	Yes- to higher difficulty
56	4	A	Yes- to higher difficulty
57	7	A	Yes- to higher difficulty
59	2	A	Yes- to higher difficulty
58	3	A	Yes- to higher difficulty
60	2	A	Yes- to higher difficulty
61	4	A	Yes- to higher difficulty
62	5	A	Yes- to higher difficulty
63	1	A	Yes- to higher difficulty
65	3	A	Yes- to higher difficulty
66	3	A	Yes- to higher difficulty
67	1	A	Yes- to higher difficulty
74	1	A	Yes- to higher difficulty
77	1	A	Yes- to higher difficulty
78	4	A	Yes- to higher difficulty
80	2	A	Yes- to higher difficulty
82	5	A	Yes- to higher difficulty
84	1	A	Yes- to higher difficulty
85	2	A	Yes- to higher difficulty
88	1	A	Yes- to higher difficulty
89	2	A	Yes- to higher difficulty
90	1	A	Yes- to higher difficulty
91	2	A	Yes- to higher difficulty
95	2	A	Yes- to higher difficulty

It is identified that a score of 27, calls for adaptation (to a 'lower' difficulty level) in the next level but a score of 30 doesn't. Similarly, a score of 49 doesn't require adaptation in the next level but a score of 52 needs adaptation (to a 'higher' difficulty level). This shows how the change in input (score) affects the output (adaptation decision).

Applying the formulae mentioned before on the set of data in the table, we get the
Mean = 52.9 and Standard deviation = 19.45

By plotting the Probability Density Function of the Concept II scores, figure 6.4 is obtained.

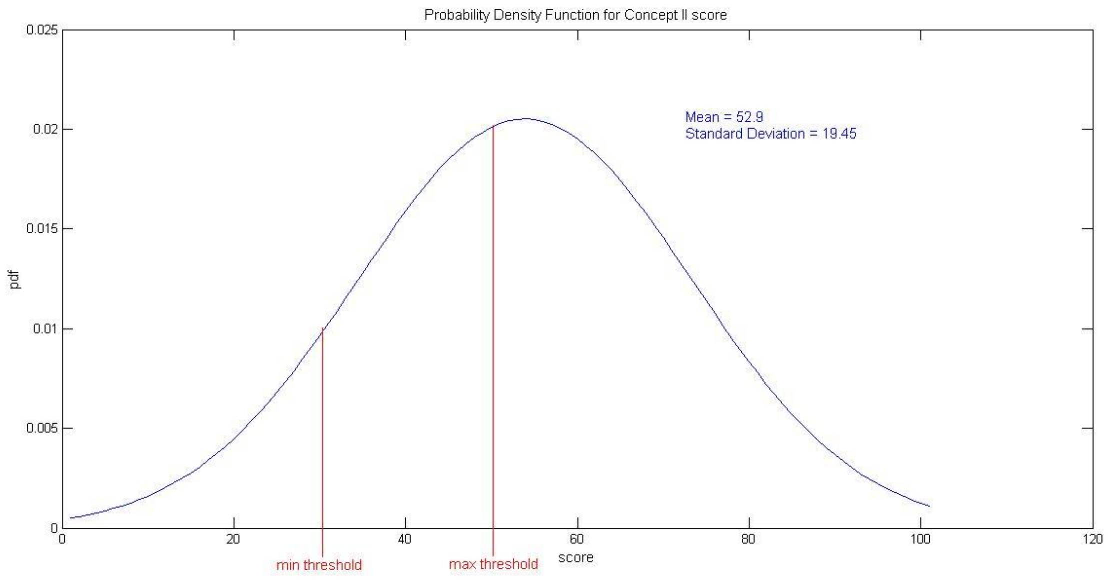

Figure 6.4. Probability Density Function of Concept II scores of sample data.

For the concept wise score, minimum threshold is 30 and maximum threshold is 50. The game will be adapted for Concept II for any student whose score lies below the minimum threshold (adapted to a 'lower' difficulty level) or above maximum threshold (adapted to a 'lower' difficulty level).

6.3.4 Concept III scores

The Concept III scores obtained from the simulation, number of students at each score, Concept III grades and adaptation decision are captured in Table 6.4.

Table 6.4. Concept III of sample data and the adaptation decision.

Score(Input)	Number of students	Grade	Adaptation Decision(Output)
16	2	C	Yes- to lower difficulty, Repeat level
18	1	C	Yes- to lower difficulty, Repeat level
19	1	C	Yes- to lower difficulty, Repeat level
21	1	C	Yes- to lower difficulty, Repeat level
23	1	C	Yes- to lower difficulty, Repeat level
24	2	C	Yes- to lower difficulty, Repeat level
25	3	C	Yes- to lower difficulty, Repeat level
26	2	C	Yes- to lower difficulty, Repeat level
28	3	C	Yes- to lower difficulty, Repeat level
29	1	C	Yes- to lower difficulty, Repeat level
30	1	B	No
32	2	B	No
34	1	B	No
39	1	B	No
41	1	B	No
42	1	B	No
43	2	B	No
44	2	B	No
45	1	B	No
46	3	B	No
49	2	B	No
50	2	B	No
51	2	A	Yes- to higher difficulty
52	3	A	Yes- to higher difficulty
54	5	A	Yes- to higher difficulty
55	3	A	Yes- to higher difficulty
56	3	A	Yes- to higher difficulty

Table 6.4. continued

57	7	A	Yes- to higher difficulty
58	3	A	Yes- to higher difficulty
59	2	A	Yes- to higher difficulty
61	3	A	Yes- to higher difficulty
62	2	A	Yes- to higher difficulty
63	1	A	Yes- to higher difficulty
64	1	A	Yes- to higher difficulty
65	2	A	Yes- to higher difficulty
66	2	A	Yes- to higher difficulty
67	1	A	Yes- to higher difficulty
73	1	A	Yes- to higher difficulty
75	1	A	Yes- to higher difficulty
77	2	A	Yes- to higher difficulty
78	1	A	Yes- to higher difficulty
79	2	A	Yes- to higher difficulty
81	1	A	Yes- to higher difficulty
82	2	A	Yes- to higher difficulty
83	1	A	Yes- to higher difficulty
84	3	A	Yes- to higher difficulty
85	2	A	Yes- to higher difficulty
88	1	A	Yes- to higher difficulty
89	2	A	Yes- to higher difficulty
91	3	A	Yes- to higher difficulty
92	1	A	Yes- to higher difficulty
96	1	A	Yes- to higher difficulty

It is identified that a score of 29, calls for adaptation (to a ‘lower’ difficulty level) in the next level but a score of 30 doesn’t. Similarly, a score of 50 doesn’t require adaptation in the next level but a score of 51 needs adaptation (to a ‘higher’ difficulty level). This shows how the change in input (score) affects the output (adaptation decision). Applying the formulae mentioned before on the set of data in the table, we get the mean = 50.5 and Standard deviation = 20.99. By plotting the Probability Density Function of the Concept III scores, Figure 6.5 is obtained.

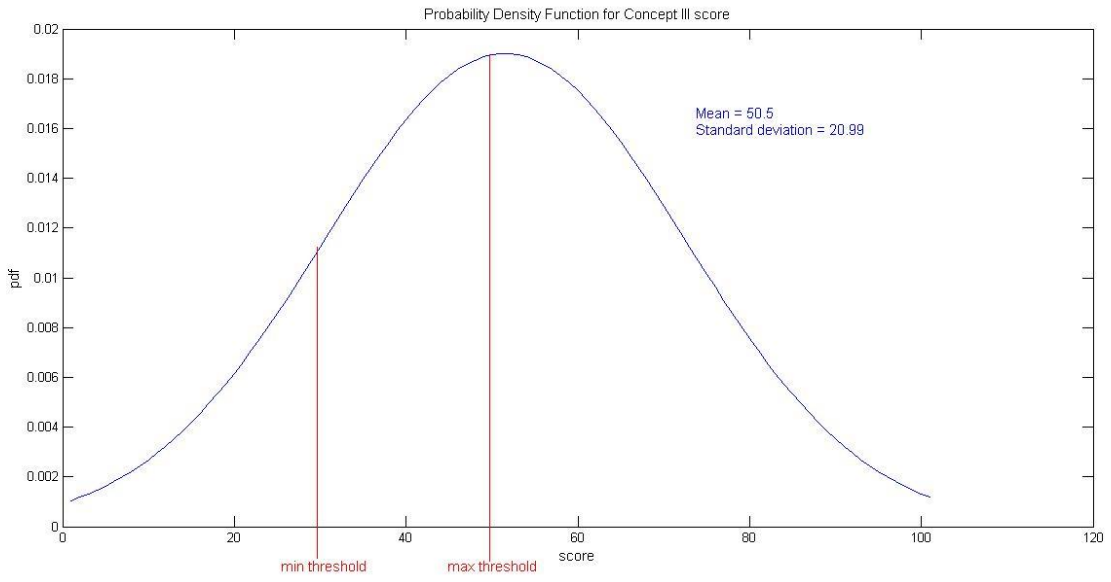

Figure 6.5. Probability Density Function of Concept III scores of sample data.

For the concept wise score, minimum threshold is 30 and maximum threshold is 50. The game will be adapted for Concept III for any student whose score lies below the minimum threshold (adapted to a ‘lower’ difficulty level) or above maximum threshold (adapted to a ‘lower’ difficulty level).

6.4 Conclusion

A student's final grade is the minimum of grades scored individually (all concepts and goal-level). By default, students are provided with 'average' difficulty questions for all the concepts in any given level. If a student scores a final grade of 'C' (grade 'C' in one of the above tables), the difficulty will be adapted to 'easy' for the concepts in which the student scored a ‘C’ grade and ‘hard’ for the concepts in which the student scored ‘A’ grade and all the concepts (concepts in which the student scored ‘B’) will not be adapted (i.e.) ‘average’ difficulty questions will be displayed. And the student will repeat the same level again because of the final grade of ‘C’.

If the student scores a final grade of ‘B’, ‘hard’ level questions is displayed in the next level for the concepts in which the student scored ‘A’ grade and the other concepts (concepts in which the student scored ‘B’ grade) will not be adapted. If a student scores a final grade of 'A' (grade 'A' in all of the above tables), the difficulty of all the concepts will be adapted to 'hard' in the next level. Thus the change in input (scores) results in a change in output (adaptation decision).

The main limitation of this testing would be not testing for a very large number of sample students. Another limitation would be not testing adaptation for more than 3 ranges (good, average, bad) of scores and grades.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The assessment methods available for serious games mostly measure the parameters and have them recorded to be manually assessed by scholars and professionals at a later point of time [43] and adapt the game for the player during the next run. But the proposed assessment and adaptation approach is automated and it is done as the player is playing the game. Also most of the existing assessment is questionnaire or feedback based [44] This research is very important and unique because statistical analysis based automated assessment and adaptation on a quiz based single player serious game has not been explored so far. And this class of serious games seems to be the most prevalent and the most usable.

Characteristics of the proposed assessment-adaptation system are given below.

1) **Assessment:**
    a. **Rigorousness:** The assessment system is highly rigorous as assessment is done based on both the concept level and goal level performance.
    b. **Specificity:** The assessment system is specific as the assessment is always based on the same set of parameters and not based on parameters in which the class's performance was bad in the previous game.
    c. **Automatability:** Assessment system is automated because the system does not store the student's performance for instructor's feedback.
    d. **Accuracy:** Assessment is accurately done because the assessment results do not categorize the students into a single performance level. But rigorous validation is not done to prove that the assessment is accurate.
2) **Adaptation:**
    a. **Specificity:** Adaptation is done specific to the individual student's performance and not based on the entire class's performance.

b. **Accuracy:** Since both difficulty adaptation and level adaptation are done, adaptation is accurately done.

3) **Performance cost:** The performance cost is O(n) which is less when compared to other existing systems. But it is not the lowest performance cost. This is another limitation.

4) **Application:** The system is applicable to other quiz based serious educational games as the system is developed by Aspect Oriented Design. But the system can be integrated only with games developed in java as the system is developed in AspectJ which is java based. Also, applicability has not yet been validated by testing with a variety of serious educational games.

5) **Validation:** Though the system has been validated, it is not rigorous. Validation is yet to be done with other quiz based serious educational games. It is also vital to measure the increase in the student's knowledge level after playing the game, to actually validate the system.

In the future additional research is needed to improve the ability to configure the assessment and adaptation rules. Validating the system by integrating with games developed in java [45] can be done in future. Extending the assessment adaptation approach for other forms of games like graphic objects based games [46], [47], and multiplayer SEGs can be done in the future.

APPENDIX

Java Implementation for the Assessment_Adaptation_System

level.java

```
/*Level class*/
public class level {
  public
    int moderateAnswerCount =0; //Number of moderate answers in the entire level
    int correctAnswerCount =0;     //Number of correct answers in the entire level
    int wrongAnswerCount =0;       //Number of wrong answers in the entire level
    int timeSpent =0;                       //Time spent on the entire level
    int optionsChangeCount =0;     //Number of times options are changed in the level
    int avgTimeSpent = 0;               //Average Time spent on the entire level
    int avgOptionsChange = 0;       //Average number of times options are changed in the level
    int noOfConcepts =0;                //Number of concepts in the level
    int noOfQns =0;                        //Number of qns in the level
    int marks = 0;                          //Marks in the level
    int nextLevel = 0;                    //Adaptation decision whether the student passed this level or not
                                                  //Next level based on the current level's performance
    char grade;                             //Goal level grade
    char finalGrade;                      //Final Grade in current level
    int position = 0;                     //Position of the player in this level in the score board
    int noOfEntires = 0;              //Number of entries currently in the score board
    int avgFlag = 0;                   //Flag indicating if the player's performance is
                                              //average in any concept of this level
    int badFlag = 0;                //Flag indicating if the player's performance
                                    //is bad in any concept of this level
    int level = 0;                  //current level
    String scoreBoard[];            //Score board of players
    level(int noOfCnpts, int lvl){     //constructor
      noOfConcepts = noOfCnpts;
      level = lvl;
    }
    void levelCompleted(){
    }
    void nextLevelDisplay(String uname){
    }
}
```

concept.java

```
/*Concept class*/
public class concept {
  public
    int moderateAnswerCount =0;   //Number of moderate answers in the entire concept
    int correctAnswerCount =0;      //Number of correct answers in the entire concept
    int wrongAnswerCount =0;      //Number of wrong answers in the entire concept
    int timeSpent =0;                     //Time spent on the entire concept
    int avgTimeSpent =0;             //Average Time spent on the entire concept
    int avgOptionsChangeCount =0; //Average number of times options are changed in concept
    int optionsChangeCount =0;    //Number of times options are changed in the concept
    int noOfQns =0;                       //Number of qns in the concept
    int marks = 0;                        //Marks in the concept
    int difficulty = 0;                  //Adaptation decision for the difficulty of the concept
    char grade;                          //Concept level grade
    concept(int noOfQ){             //Constructor
      noOfQns = noOfQ;
    }
    void conceptCompleted(){
   }
}
```

quiz.java

```
/*Quiz class*/
public class quiz {
  public
    int selectedAnswer = 0;        //answer selected by the player for this qn
    int correctAnswer = 0;          //correct answer for this qn
    int moderateAnswer = 0;      //moderate answer for this qn
    int optionsChangeCount = 0;//number of times the player changes the option for this qn
    String studentFlag;              //Students knowledge level
    void screenDisplayed(){
    }
       void continueButtonPressed(int selectedAns,int correctAns,int moderateAns,int
                        optionsChangeCnt,String stdntFlag,concept c,level l){
     selectedAnswer = selectedAns;
     correctAnswer  = correctAns;
     moderateAnswer = moderateAns;
     optionsChangeCount = optionsChangeCnt;
     studentFlag = stdntFlag;
     System.out.println("selectedAnswer: " + selectedAns + "correctAnswer:" +
                  correctAns + "moderateAnswer:" + moderateAns +
                  "optionsChangeCount:" + optionsChangeCnt);
    }
}
```

inputProvider.java

```
/*Provides input for all the aspects - Will be removed once the main component of the game is
integrated*/
import java.util.Random;
import java.io.*;
import static java.lang.Thread.sleep;
public class inputProvider {
  public
            String uname;          //username of the player
            Random generator = new Random();  //random number generator
            int selectedAns;       //answer selected by the player for a particular qn –
                                  //randomly generated
            int correctAns;        //correct answer for a particular qn – randomly generated
            int moderateAns;   //moderate answer for a particular qn - randomly generated
            int optionsChangeCnt;  //number of times the player changes the option for a
                                       //particular qn - randomly generated
            int noOfConcepts = 3;  //number of concepts in a particular level
           int noOfQns;//number of questions in a particular concept – randomly generated
            int noOfStudents = 0;  //number of students
            int badMarker = 0;     //The number of students in the bad zone
            int avgMarker = 0;     //The number of students in between the bad and average zone
            String studentFlag = "";  //The student's knowledge scale[good/avg/bad]
            int noOfLevels = 5;       //Number of levels in the game
            void inputGenerator(){
              //Get the number of students for whom the input should be generated
              System.out.println("Enter the number of students in class:");
              BufferedReader reader = new BufferedReader(new
                                  InputStreamReader(System.in));
              try{
                        noOfStudents = Integer.parseInt(reader.readLine());
              }catch (IOException e) {
                System.out.println("Error!");
```

```
             System.exit(1);
            }
             //1/4 of total number of students have bad performance
            badMarker = (noOfStudents * 25)/100;
          //Majority(1/2) of students have average performance
            avgMarker = (noOfStudents * 75)/100;
            for(int p = 0;p<noOfStudents;p++){
                      System.out.println("Enter the student's username:");
                      try{
                        uname = reader.readLine();
                      }catch (IOException e) {
                        System.out.println("Error!");
                        System.exit(1);
                      }
       level l[] = new level[noOfLevels];               //level object
       for(int i=0; i<noOfLevels; i++){
                System.out.println("Level:No. of concepts is " + noOfConcepts);
                 //Initialize number of concepts and the level id of a particular level
                  l[i] = new level(noOfConcepts,i);
                  concept c[] = new concept[l[i].noOfConcepts];   //concept object
                           for(int j=0; j<l[i].noOfConcepts; j++){
                //Will generate a random number from 10 to 19.
                             noOfQns = generator.nextInt(10) + 10;
                             System.out.println("Concept:No. of qns is " + noOfQns);
                //Initialize number of qns in a particular concept
                             c[j] = new concept(noOfQns);
                // Total number of qns in a particular level
                             l[i].noOfQns += c[j].noOfQns;
                             System.out.println("Level:no of qns is " + l[i].noOfQns);
                    quiz q[] = new quiz[c[j].noOfQns];    //quiz object

                             for(int k=0; k<c[j].noOfQns; k++){
```

```
                    q[k] = new quiz();
                         q[k].screenDisplayed();   //When a quiz is displayed
                         if(p <= badMarker){       //If the student falls within the bad range
                          /*Generating bad performance*/
                          studentFlag = "bad";
               //Will generate a random number from 1 to 4.
                          selectedAns = generator.nextInt(4) + 1;
               //Will generate a random number from 1 to 4.
                          correctAns = generator.nextInt(4) + 1;
              //Correct and moderate ans cannot be same.Moderate answer will from 1 to 4.
                      moderateAns = 5 - correctAns;
               //Will generate a random number from 0 to 5.
                          optionsChangeCnt = generator.nextInt(6);
              //If the student falls within the avg range
              }else if((p> badMarker) && (p <= avgMarker)){
               /*Generating avg performance*/
                          studentFlag = "avg";
               //Will generate a random number from 1 to 4.
                          selectedAns = generator.nextInt(4) + 1;
               //Try to make selected answer, correct
                          if(selectedAns < 3){
                                correctAns = selectedAns + generator.nextInt(3);
                          }else{
                                correctAns = selectedAns - generator.nextInt(3);
                          }
                          //Make the selected answer either correct or moderate
                          if(selectedAns == correctAns){
                //Correct and moderate ans cannot be same. Moderate answer will from 1 to 4.
                                moderateAns = 5 - correctAns;
               }else{
                                moderateAns = selectedAns;
                          }
                //average options change count
                          optionsChangeCnt = generator.nextInt(4);
               //If the student falls within the good range
                         }else if (p > avgMarker){
                studentFlag = "good";
                          /*Generating good performance*/
                //Will generate a random number from 1 to 4.
                          selectedAns = generator.nextInt(3) + 1;
                //Try to make selected answer, correct
                          if(selectedAns < 3){
                           moderateAns = selectedAns + generator.nextInt(3);
                     }else{
                           moderateAns = selectedAns - generator.nextInt(3);
                          }
               //Make the selected answer either moderate or correct
                          if(selectedAns == moderateAns){
              //Correct and moderate ans cannot be same.Moderate answer will from 1 to 4.
                     correctAns = 5 - moderateAns;
                }else{
                                correctAns = selectedAns;
                          }
                     optionsChangeCnt = generator.nextInt(2);
                         }
                         /*Sleep to make sure some time is spent on each qn*/
                         try
                         {
                          sleep(500);
                         }
                         catch (InterruptedException ex) {}
             //When player has selected answer and pressed continue button for next qns
q[k].continueButtonPressed(selectedAns,correctAns,moderateAns,optionsChangeCnt,
                   studentFlag,c[j],l[i]);
```

```
                    }
                //All qns in a particular concept are answered by the player
                        c[j].conceptCompleted();
/*To calculate final grade and to make adaptation decision whether the player passed
  the level*/
                        if(c[j].marks>= 30 && c[j].marks <= 50 ){
//To assess if the performance was average in any of the concepts in a particular level
                            l[i].avgFlag = 1;
                            System.out.println("avg flag set");
                        }else if (c[j].marks < 30){
//To assess if the performance was bad in any of the concepts in a particular level
                            l[i].badFlag = 1;
                            System.out.println("Bad flag set");
                        }
                      }
               //All concepts in a particular level are answered by the player
                      l[i].levelCompleted();
//This level is completed by the player and the next level is displayed
                   l[i].nextLevelDisplay(uname);
           System.out.println("nextlevel");
         }
             }
    }
}
```

aspectLevelEvaluation.aj

```
/*Evaluate the player's performance in a particular level*/
public aspect aspectLevelEvaluation {
  after(level l) returning: call(void level.levelCompleted()) && target(l){ //after
                                                                           //the level is completed
         l.avgTimeSpent = l.timeSpent / l.noOfQns;  //Average time spent on the entire level
         l.avgOptionsChange = l.optionsChangeCount / l.noOfQns; //Average number of
```

```
                                        //times options are changed in the level
      System.out.println("Level:Avg time spent is " + l.avgTimeSpent);
      System.out.println("Level:Avg option change count is " + l.avgOptionsChange);
      /*Assign marks based on average time spent in the level*/
      if(l.avgTimeSpent <20){
       l.marks += 10;
      }else if(l.avgTimeSpent >=20 && l.avgTimeSpent <=40){
       l.marks += 6;
      }else if(l.avgTimeSpent >40 && l.avgTimeSpent <=60){
       l.marks += 2;
      }
      /*Assign marks based on average number of times options are changed in the
   concept*/
      if(l.avgOptionsChange == 0){
       l.marks += 10;
      }else if(l.avgOptionsChange == 1){
       l.marks += 6;
      }else if(l.avgOptionsChange == 2){
       l.marks += 2;
      }
      System.out.println("Level:Moderate Ans cnt is " + l.moderateAnswerCount);
      System.out.println("Level:Correct Ans cnt is " + l.correctAnswerCount);
      System.out.println("Level:Wrong Ans cnt is " + l.wrongAnswerCount);
      l.marks += (l.moderateAnswerCount * 35 ) /  l.noOfQns;  //convert the moderate
                                                          //answers count to a score on 35
      l.marks += (l.correctAnswerCount * 80 ) /  l.noOfQns;     //convert the
                                                      //correct answers count to a score on 80
      System.out.println("Level:mark is " + l.marks);
      //Assign goal level grade based on the player's overall performance
      if(l.marks < 50){   //If the player's goal level score is bad
       l.grade = 'C';
      }else if (l.marks >= 50 && l.marks <= 70){ //If the player's goal level score is average
```

```
            l.grade = 'B';
           }else{ //If the player's goal level score is  good
            l.grade = 'A';
           }
           /*Assign final grade and whether the player passed this level based on the
        player's performance in every concept and goal level*/
           if(l.badFlag == 1 || l.grade == 'C'){//If the player's performance in a any
                                                 //concept is bad or goal level score is bad
            l.finalGrade = 'C';
           }else if (l.avgFlag == 1 || l.grade =='B'){ //If the player's performance in a
                                                  //any concept is average or goal level score is average
            l.finalGrade = 'B';
            l.nextLevel ++;    //passed this level
           }else{ //If the player's performance in all the concepts are good and goal
                   //level score is  also good
            l.finalGrade = 'A';
            l.nextLevel ++;    //passed this level
           }
           System.out.println("Final Grade: " + l.finalGrade);
  }
}
```

```
aspectConceptEvaluation.aj
```

```
/*Evaluate the player's performance in a particular concept*/
public aspect aspectConceptEvaluation {
  after(concept c) returning: call(void concept.conceptCompleted()) && target(c){
                                                      //after the concept is completed
    c.avgTimeSpent = c.timeSpent / c.noOfQns;  //Average time spent on the entire concept
           c.avgOptionsChangeCount = c.optionsChangeCount / c.noOfQns;  //Average number
                                                      //of times options are changed in the concept
           System.out.println("Concept:Time spent is " + c.timeSpent);
           System.out.println("Concept:Option change count is " + c.optionsChangeCount);
```

```
        System.out.println("Concept:Avg time spent is " + c.avgTimeSpent);
        System.out.println("Concept:Avg option change count is " +
                c.avgOptionsChangeCount);
        /*Assign marks based on average time spent on the concept*/
        if(c.avgTimeSpent <20){
          c.marks += 10;
        }else if(c.avgTimeSpent >=20 && c.avgTimeSpent <=40){
    c.marks += 6;
        }else if(c.avgTimeSpent >40 && c.avgTimeSpent <=60){
          c.marks += 2;
        }
        /*Assign marks based on average number of times options are changed on the
      concept*/
        if(c.avgOptionsChangeCount == 0){
          c.marks += 10;
        }else if(c.avgOptionsChangeCount == 1){
          c.marks += 6;
        }else if(c.avgOptionsChangeCount == 2){
          c.marks += 2;
        }
        System.out.println("Concept:Moderate Ans cnt is " + c.moderateAnswerCount);
        System.out.println("Concept:Correct Ans cnt is " + c.correctAnswerCount);
        System.out.println("Concept:Wrong Ans cnt is " + c.wrongAnswerCount);
        c.marks += (c.moderateAnswerCount * 35 ) / c.noOfQns;  //convert the moderate
                                                               //answers count to a score on 35
        c.marks += (c.correctAnswerCount * 80 ) / c.noOfQns;   //convert the correct
                                                               //answers count to a score on 80
        System.out.println("Concept:mark is " + c.marks);
        /*Adaptation decision for the difficulty of the same concept based on the
      player's performance in the current level(good or avg or bad)*/
   if(c.marks > 50 ){
    c.grade = 'A';
```

```
            c.difficulty = 2;
    }
    else if(c.marks>= 30 && c.marks <= 50 ){
      c.grade = 'B';
      c.difficulty = 1;
    }else{
      c.grade = 'C';
      c.difficulty = 0;
    }
  }
}
```

aspectAnswerEvaluation.aj

```
/*Evaluate the player's answer for a particular question*/
import java.util.Calendar;
import java.text.SimpleDateFormat;
public aspect aspectAnswerEvaluation {
  int startHr;       //Hour when the screen was displayed
  int startMin;     //Minute when the screen was displayed
  int startSec;     //Second when the screen was displayed
  int startTime;  //Time when the screen was displayed(in seconds)
  String time[];          //Get time and not the date
  String timeSplit[];  //Split current time in hour,minute and seconds
  String DATE_FORMAT_NOW = "yyyy-MM-dd HH:mm:ss";   //Current date and time
  after() returning:call(void quiz.screenDisplayed()){               //after screen is displayed
    Calendar cal = Calendar.getInstance();
          SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT_NOW); //current date
and time
          time = (sdf.format(cal.getTime())).split(" "); //Get time and not the date
          System.out.println("Starttime is " + time[1]);
          timeSplit = time[1].split(":");                      //Get hour,min and sec from time
          startHr = Integer.parseInt(timeSplit[0]);
          startMin = Integer.parseInt(timeSplit[1]);
          startSec = Integer.parseInt(timeSplit[2]);
 }
 after(quiz q,concept c, level l) returning:call(void
quiz.continueButtonPressed(int,int,int,int,String,concept,level)) && target(q) &&
args(int,int,int,int,String,c,l){ //after continue button is pressed
          int stopHr;    //Hour when the screen was closed
          int stopMin;   //Minute when the screen was closed
          int stopSec;   //Second when the screen was closed
          int stopTime;  //Time when the screen was closed (in seconds)
          int timeSpent; //Time spent by the player on a particular question
   Calendar cal = Calendar.getInstance();
          SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT_NOW); //current
                                                                       //date and time
          time = (sdf.format(cal.getTime())).split(" ");  //Get time and not the date
          System.out.println("Stoptime is " + time[1]);
          timeSplit = time[1].split(":");        //Get hour,min and sec from time
          stopHr = Integer.parseInt(timeSplit[0]);
          stopMin = Integer.parseInt(timeSplit[1]);
          stopSec = Integer.parseInt(timeSplit[2]);
          stopTime = (stopHr*3600) + (stopMin *60) + stopSec ; //convert stop time to seconds
          startTime = (startHr*3600) + (startMin *60) + startSec ;  //convert start time
                                                            //to seconds
          timeSpent = stopTime - startTime;   //Time spent by player in a particular qn
          System.out.println("Timespent is " + timeSpent);
          //since the input is automated, the time spent is manually set based on the student's
         //intelligence
          if (q.studentFlag == "avg"){
            c.timeSpent += timeSpent + 45;
            l.timeSpent += timeSpent + 45;
          }else if(q.studentFlag == "bad"){
            c.timeSpent += timeSpent + 65;
```

```
                l.timeSpent += timeSpent + 65;
              }else if(q.studentFlag == "good"){
                c.timeSpent += timeSpent;
                l.timeSpent += timeSpent;
              }
              if(q.selectedAnswer == q.correctAnswer){      //If selected answer is correct
                c.correctAnswerCount ++;
                l.correctAnswerCount ++;
          }
              else if(q.selectedAnswer == q.moderateAnswer){  //If selected answer is moderate
                c.moderateAnswerCount ++;
                l.moderateAnswerCount ++;
              }else{
                c.wrongAnswerCount ++;
                l.wrongAnswerCount ++;
              }
              c.optionsChangeCount +=q.optionsChangeCount;
              l.optionsChangeCount += q.optionsChangeCount;
    }
  }
```

aspectScoreBoardUpdation.aj

```
/*Evaluate the position of the player in the score board*/
public aspect aspectScoreBoardUpdation {
  before(level l,String uname) : call(void level.nextLevelDisplay(String)) &&
                              target(l) && args(uname){            //After next level is displayed
            int compareScore;              //Score in the score board against which the
                                          //player's score is compared currently
            String scores[] = new String[2];//score string in the score board which is
                                              //currently being examined
            String prevTempString = " ";   //Temporary string from the previous pass
            String tempString;               //Temporary String from current pass
            int i;                                  //'for' loop iterator
            client c = new client();          //client object
     c.readFile(l);                                //read file from server
       if(l.noOfEntires == 0){             //If there are no entries in the score board yet
              l.scoreBoard[0] = Integer.toString(l.marks) + "#" + uname; //Save the
                                                         //player's score and username in the first position
              l.noOfEntires ++;
            }
     else{                              //If there are entries in the score board
       System.out.println(l.noOfEntires - 1);
//ScoreBoard has raw strings which has '#' separating the goal level score and the username of a
//player
       scores = l.scoreBoard[l.noOfEntires - 1].split("#");
              compareScore = Integer.parseInt(scores[0]);
              if(l.marks < compareScore){ //If player's score is less than the last entry
                                                     //in score board
                       l.position = l.noOfEntires;     //Rank of the player in score board
            //Add the player's username in the last position
                       l.scoreBoard[l.noOfEntires] = Integer.toString(l.marks) + "#" + (uname);
            l.noOfEntires ++;
              }else{
         int highScorer =0; //If the player's score is more than a particular entry in score board
                for(i = 0; i <l.noOfEntires; i++){//noOfEntires currently in scoreboard //ScoreBoard is a
raw string which has '#' separating the overall score and the username of a player
            scores = l.scoreBoard[i].split("#");
//goal level score of the current entry in score board
                 compareScore = Integer.parseInt(scores[0]);
//If the player has scored more than any one of the entries in score board
           if(highScorer == 1){
/*Keep moving the entries down in the score board after that entry*/
                         tempString = l.scoreBoard[i];
                         l.scoreBoard[i] = prevTempString;
```

```
                        prevTempString = tempString;
                }
//If the player has scored more than any one of the entries in score board
                else if(l.marks > compareScore){
              l.position = i;          //Position of the player in score board
//set the flag to indicate that the player has scored more than a particular entry in score board
                        highScorer = 1;
//Store the current entry in score board in that position into a temp string
                        prevTempString = l.scoreBoard[i];
//Save the player's score and username in that position
              l.scoreBoard[i] = Integer.toString(l.marks) + "#" + uname ;
//If the player has scored equal to any one of the entries in score board
                }else if(l.marks == compareScore){
                   l.position = i;            //Rank of the player in score board
//Add the player's username in that position (as score would be same)
                        l.scoreBoard[i] = l.scoreBoard[i] + "#" + (uname);
                     break;
                }
              }
              if(highScorer == 1 ){
                l.scoreBoard[i] = prevTempString;
                l.noOfEntires ++;
              }
            }
   }
   c.writeFile(l);     //write file to the server
  }//End of aspect
}//End of aspect_score_board class
```

REFERENCES

[47] "Revenge of the Titans," [Online]. Available: https://en.wikipedia.org/wiki/Revenge_of_the_Titans.

[8] A. Apostolos and S. Ioannis , "Software engineering research for computer games: A systematic review," *Information and Software Technology,* vol. 52, no. 9, pp. 888-901, Sep 2010.

[2] A. Daniel, "Playstations and workstations: identifying and negotiating digital games work," *Information Technology & People,* vol. 24, no. 1, pp. 10-25, 2011.

[20] A. F. Bustamante and J. M. Sanchez-Torres, "Building a system of indicators for measuring e-learning in on-campus higher education," *Information Society (i-Society), 2011 International Conference on,* pp. 464-469, 27-29 June 2011.

[26] A. Paulo, L. Amaral and J. Pires, "Case-Based Reasoning Approach to Adaptive Web-Based Educational Systems," *Advanced Learning Technologies, 2008. ICALT '08. Eighth IEEE International Conference on,* pp. 260-261, 1-5 July 2008.

[37] A. Pfeifer and J. Fürnkranz, *Creating Adaptive Game AI in a Real-time Continuous Environment using Hierarchical Neural Networks,* Darmstadt: TU Darmstadt, Mar 2009.

[29] B. Deliyska and P. Manoilov, "Method and Algorithm of Using Ontologies in E-Learning Sessions," *APPLICATIONS OF MATHEMATICS IN ENGINEERING AND ECONOMICS, AMEE-2009, 35TH INTERNATIONAL CONFERENCE ON,* vol. 1184, pp. 309-315, 7–12 June 2009.

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

[27] C. Basaran, M. H. Suzer, K.-D. Kang and X. Liu, "Model-Free Fuzzy Control of CPU Utilization for Unpredictable Workloads," *in Proceedings of the Fourth International Workshop on Feedback Control Implementation and Design (FeBID'09), in conjunction with IEEE RTAS'09,* 2009.

[46] C. Carter, A. E. Rhalibi and M. Merabti, "Development and Deployment of Cross-Platform 3D Web-based Games," *Developments in E-systems Engineering (DESE), 2010,* pp. 149-154, 6-8 Sep. 2010.

[30] C. L. Lee, C. H. Huang and C. J. Lin, "Test-sheet composition using immune algorithm for E-learning application," *Industrial, engineering, and other applications of applied intelligent systems, IEA/AIE'07 Proceedings of the 20th international conference on,* pp. 823-833, 2007.

[36] C. Thurau, C. Bauckhage and G. Sagerer, "Imitation learning at all levels of game-AI," *Proceedings of the international conference on computer games, artificial intelligence, design and education,* pp. 402-408, Dec 2004.

[9] D. C. Cascini Peixoto, R. M. Possa, R. F. Resende and C. I. P. S. Padua, "An overview of the main design characteristics of simulation games in Software Engineering education," in *24th IEEE-CS Conference on Software Engineering Education and Training, CSEE&T 2011*, Honolulu, May 22-24 2011.

[14] D. Ismailović, D. Pagano and B. Dennis, "WEMAKEWORDS – A COLLABORATIVE SERIOUS GAME FOR LITERACY ACQUISITION," in *IADIS International Conference - Game and Entertainment*, 2011.

[38] D. Loiacono, L. Cardamone and P. L. Lanzi , "Automatic Track Generation for High-End Racing Games Using Evolutionary Computation," *Computational Intelligence and AI in Games, IEEE Transactions on,* vol. 3, no. 3, pp. 245-259, Sept. 2011.

[43] E. M. Raybourn, "Applying simulation experience design methods to creating serious game-based adaptive training systems," *Interacting with Computers*, vol. 19, no. 2, pp. 206-214, 2007.

[44] F. Frapolli, A. Malatras and B. Hirsbrunner, "Exploiting traditional gameplay characteristics to enhance digital board games," *Consumer Electronics Society's Games Innovations Conference, 2nd International IEEE* , 2010.

[39] F. G. Lobo, D. E. Goldberg and M. Pelikan, "Time Complexity of Genetic Algorithms on Exponentially Scaled Problems," *PROCEEDINGS OF THE GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE,* pp. 151-158, 2000.

[16] G. Krzysztof and . N. Krzysztof, "A Novel Architecture for E-Learning Knowledge Assessment Systems," *International Journal of Distance Education Technologies, IJDET,* vol. 7, no. 2, pp. 1-19, 2009.

[3] I. Cheng and A. Basu, "Interactive Graphics for Computer Adaptive Testing," *Computer Graphics Forum,* vol. 28, no. 8, pp. 2033-2045, Dec. 2009.

[15] I. Hatzilygeroudis, C. Koutsojannis and N. Papachristou, "Evaluation of Usability and Assessment Capabilities of an e-Learning System for Nursing Radiation Protection," *Computer-Based Medical Systems, 2007. CBMS '07. Twentieth IEEE International Symposium on*, pp. 301-306, 20-22 June 2007.

[42] J. A. Gubner, in *Probability and Random Processes for Electrical and Computer Engineers, Edition I*, Cambridge University Press, 2006, pp. 63-184.

[19] J. Hayami, T. Noriaki, S. Hiroyuki and U. Hideo, "Construction of e-Learning in Maritime Education," *Proceedings of the Japan Institute of Navigation,* vol. 113, pp. 275-280, 25 Sep. 2009.

[31] J. He and X. Yao, "Drift analysis and average time complexity of evolutionary algorithms," *Artificial Intelligence,* vol. 127, pp. 57-85, 2001.

[1] J. L. Shih, B. J. Shih, H. Y. Shih and C. W. Chuang, "The influence of collaboration styles to children's cognitive performance in digital problem-solving game "William Adventure": A comparative case study," *Computers & Education,* vol. 55, no. 3, pp. 982-993, Nov. 2010.

[13] J. R. Anderson, A. T. Corbett and K. R. Koedinger, "Cognitive Tutors: Lessons Learned," *Journal of the Learning Sciences,* vol. 4, no. 2, pp. 167-207, 1995.

[17] J. Rissanen, in *Stochastic Complexity in Statistical Inquiry*, Singapore, World Scientific Series in Computer Science, 1989.

[48] J. Singh, "An Algorithm to Reduce the Time Complexity of Earliest Deadline First Scheduling Algorithm in Real-Time System," *The Computing Research Repository,* 2011.

[25] J. Verttaros, G. Vouros and A. Drigas, "Development of a Diagnostic System of Taxonomies Using Fuzzy Logic – Case SOLO (useful for e-learning system)," *WSEAS Transactions on Information Science and Applications,* vol. 6, no. 1, Dec. 2004.

[45] K. Cooper, "SimSYS, a serious game for software and systems engineering education," [Online]. Available: http://www.utdallas.edu/~kcooper/SimSYS/.

[4] K. Peterson, G. Stoll and O. v. Stryk, "A supporter behavior for soccer playing humanoid robots," in *Robocup 2010*, Singapore, 2010.

[41] K. S. Trivedi, in *Probability and Statistics with Reliability, Queueing, and Computer Science Applications,* Wiley, 2001, pp. 55-119, 181-211.

[18] K. Scalise, D. J. Bernbaum, M. Timms, V. S. Harrell, K. Burmester, C. A. Kennedy and M. Wilson, "Adaptive technology for e-learning: principles and case studies of an emerging field," *Journal of the American Society for Information Science and Technology,* vol. 58, no. 14, Dec. 2007.

[10] K. Vanlehn, C. Lynch, K. Schulze, J. A. Shapiro, r. Shelby , L. Taylor, D. Treacy, A. Weinstein and M. Wintersgill, "The Andes physics tutoring system: Five years of evaluations," in *Artificial Intelligence in Education, Proceedings of the 12th International Conference on* , Amsterdam, 2005.

[7] M. Daumas, E. Martin-Dorel, A. Truffert and M. Ventou, "A Formal Theory of Cooperative TU-Games," *Modeling Decisions for Artificial Intelligence, MDAI '09 Proceedings of the 6th International Conference on*, pp. 81-91, 2009.

[6] P. Charoenkwan, S.-W. Fang and S.-K. Wong, "A Study on Genetic Algorithm and Neural Network for Implementing Mini-Games," *Technologies and Applications of Artificial Intelligence (TAAI), 2010 International Conference on*, pp. 158 - 165, 18-20 Nov. 2010.

[21] P. T. Yu and M. T. Liu, "Aberrant Learning Achievement Detection Based on Person-Fit Statistics in Personalized e-Learning Systems," *Educational Technology & Society,* vol. 14, no. 1, pp. 107-120, 2011.

[35] R. Cominettia, E. Melob and S. Sorin, "A payoff-based learning procedure and its application to traffic games," *Games and Economic Behavior,* vol. 70, no. 1, pp. 71-83, Sep. 2010.

[40] R. G. Antonio, T. Llanos, R. Salvador, H. Roberto, C. C. Agustin and P. Rafael, "A New e-Learning Architecture for the Automatic Assessment of Network Services," in *Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference on*, 6-8 July 2011.

[5] R. Hunicke and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," in *Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence.*, 2004.

[11] R. Miemiec and H. J. Walberg, "Comparative Effects of Computer-Assisted Instruction: A Synthesis of Reviews," *Journal of Educational Computing Research,* vol. 3, no. 1, pp. 19-37, 1987.

[32] S. Ayuko, K. Takeshi, K. Tomoya and N. Kiyoshi, "Emotion Estimation Algorithm from Facial Image Analyses of e-Learning Users," *IEEJ Transactions on Electronics, Information and Systems,* vol. 129, no. 10, pp. 1881-1888, 2009.

[33] S. Petruseva, "Emotion learning: solving a shortest path problem in an arbitrary deterministic environment in linear time with an emotional agent," *International Journal of Applied Mathematics and Computer Science,* vol. 18, no. 3, pp. 409-421, 2008.

[34] T. Javier, M.-g. Pablo, M.-O. Ivan and F.-M. Baltasar, "Integration and Deployment of Educational Games in e-Learning Environments: The Learning Object Model Meets Educational Gaming," *Educational Technology & Society,* vol. 12, no. 4, pp. 359-371, Oct. 2009.

[24] X. Yao, "A Note on Neural Sorting Networks with O(1) Time Complexity," *Information Processing Letters,* vol. 56, no. 5, pp. 253-254, Dec. 1995.

[23] Y. H. Wang and H. C. Liao, "Datamining for adaptivelearning in aTESL-basede-learningsystem," *Expert Systems with Applications,* vol. 38, no. 6, pp. 6480-6485, June 2011.

[22] Y. W. Wu, Z. H. Wu and Li Jin Ling, "Personalized intelligent question answering algorithm in e-learning," *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 6, pp. 3299-3303, 18-21 Aug. 2005.

[28] Z. Jiang, X. Guo, N. Gangavarapu and K. Khan, "Knowledge-Based Algorithms to Optimise e-Learning Outcome," *FECS*, pp. 247-253, 2009.

VITA

Bharathi Balasubramaniam earned her bachelor's degree in Computer Science and Engineering at PSG College of Technology and her master's degree in computer science at UTD. She entered the Computer Science MS program at UTD in 2010. Prior to joining UTD, her industry experience included Intel Technologies, India and Deloitte Consulting India. While doing her MS program, she worked as a graduate co-op at Rockwell Collins, US.