I.J. Modern Education and Computer Science, **2015, 10, 1-11**
Published Online October 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2015.10.01

Development of a Novel Computer Application to Teach Counting in K-2 Classroom using the Unified Process Method

Jiang Li
Department of Computer Science and Information Technology Austin Peay State University, Clarksville, TN 37044, USA
E-mail: lij@apsu.edu

Ling Wang
Department of Teaching and Learning Austin Peay State University, Clarksville, TN 37044, USA
E-mail: wangl@apsu.edu

Lacey L. Williams and Christina A. Allan
Department of Computer Science and Information Technology Austin Peay State University, Clarksville, TN 37044, USA
E-mail: {lwilliams82, callan}@my.apsu.edu

***Abstract*—**Classroom teachers of Kindergarten, 1st Grade, or 2nd Grade often use the Hundreds Chart puzzles to teach children the basic mathematical knowledge and skills like counting. Computer programs have been developed to help them to create puzzle sheets by cutting out the numbers, but students still need to solve the puzzles on the printed paper. This study designs and implements a computer application named Hundred Acorn Forest as an instructional method to teach basic counting skills with the Hundreds Chart puzzles using the Unified Process method. The application not only offers an easy way for classroom teachers to create and edit puzzles based on patterns, but also engages students in a unique interactive game play environment, which helps to maintain their interests of learning. In addition, the video playback that demonstrates the process of students solving the puzzle provides classroom teachers valuable information to set up levels of control and consistency that have not been available before.

***Index Terms*—**Math Education, Teaching Counting, Computer Application, Unified Process.

I. Introduction

Counting is a very important math skill taught in Kindergarten through 2nd Grade. The Hundreds Chart, which presents a continuous sequence of numbers from 1 to 100 with some numbers left out for the students to fill in, is one of the most widely used puzzle tools for classroom teachers to teach children counting [1]. Educators have dedicated their time into researching and developing the most useful way for teachers to teach mathematical methods to young children using the Hundreds Chart [2]. While there are computer programs that may help teachers to create sheets by cutting out the numbers, students still have to solve the puzzles on the printed paper, and it is inconvenient for the teachers to observe each individual student and identify the difficulties the student may have experienced during the puzzle solving problem [3]. The goal of this study is to develop a computer application named Hundred Acorn Forest for classroom teachers to teach children in Kindergarten, 1st grade, or 2nd grade mathematics using the Hundreds Chart. The teachers will be able to create puzzles freely or based on certain patterns, edit existing puzzles, and save puzzles into a database that can be utilized as a supplemental tool later. Meanwhile, students will be able to practice lessons learned while maneuvering through an interactive computer game that has a storyline to keep them interested. In addition, this new software will allow for classroom teachers to have instant feedback to identify students who are struggling with counting concepts.

As the Hundreds Chart is consistently used in K-2 classrooms, research have revealed that the need to view how students complete the Hundreds Chart puzzles - the process, rather than the final puzzle - the product, has become more pertinent [4]. Therefore, the design of the application should provide teachers with just that feedback while presenting a fun interactive game for students that reinforces the math lessons previously taught. Hundred Acorn Forest allows teachers to create different levels of Hundreds Chart puzzles. After a student completes a puzzle by playing the game, the teacher can view a video playback of how the student completed the puzzle through a unique automatic screen recording function. The playback will give teachers insight on what parts of the lesson students are grasping and what parts he or she may need to reinforce.

The development of such as an application requires

Copyright © 2015 MECS

joint efforts from experts in both education and computer science. For instance, the requirements of classroom teachers who create and edit the puzzles can be collected from a professor in math education who has had experience in practical classroom activities using the paper-based Hundreds Chart. On the other hand, the user interface of the application, such as the onscreen instructions, layout of buttons and pictures, user actions, and animations, must consider the reading and comprehension ability of children who are at the early stage of their literacy development [5]. As a result, the input from a professor in content reading can provide useful insight of how students apply literacy knowledge in content area - mathematics. In addition, the design and implementation of the application demands experts with strong programming knowledge as well as those who know the principles of software engineering [6] and have applied the software development models in real-world projects.

The rest of the paper is organized as follows. Previous work related to this study is listed in Section II. Section III introduces the four phases of the software development method used to design and implement the functionalities of application. The experiments and results are presented in section IV. Section V discusses the contributions and implications of this research, and section VI concludes with proposals for future work.

II. Related Work

Computer applications have been widely used in modern education, such as instructional method design, course content delivery, and e-learning. A meta-analysis of comparative studies on computer-managed instruction (CMI) and interactive computer-assisted instruction (CAI) showed that computer-based education has generally had positive effects on the achievement of elementary school students [7]. Studies have revealed the positive impact of computer-based learning environments in different mathematical domains, including arithmetic, algebra, geometry, statistics, and calculus as the students are more engaged in committed learning through computer-aided instructions [8]. In addition, web-based computer application has offered functionalities and tools to bring students, teachers, and enormous Internet resources together in online learning environments [9].

Although computer-based educational games as an approach to enhancing student learning experience in a variety of disciplines have drawn attention of educational researchers and classroom teachers, no consensus has been reached on the effects of computer games on the achievements and outcomes of student learning due to the lack of empirical research on differential effects of diverse learners. The use of educational computer games to facilitate elementary students' cognitive math achievement and attitudes toward math education have been examined by case studies [10]. The results indicated that students developed more positive attitudes toward learning math through computer math gaming, which highlighted the value of situating learning activities within the game story and making games enjoyably challenging in elementary math education.

Focusing on gender and language minority groups, empirical studies investigating the effects of playing computer games on math achievement of elementary students have indicated that male language minority students who daily played computer games in math demonstrated higher math performance scores compared with their male English-speaking counterparts who never played [11]. Furthermore, studies examining the effects of computer games on students' math achievement and motivation illustrated that students who played the games in their classrooms and school labs reported greater motivation compared to those who played the games only in the school labs [12].

III. Unified Process Method

Commonly used software development models fall into two categories: traditional models, such as Waterfall and Evolutionary Prototyping for more static projects whose requirements do not change much during the development, and contemporary models like Scrum, Adaptive Software Development and Unified Process that may handle changing requirements and project goals better [13]. In this study, we adopted the Unified Process, a model based on the Unified Modeling Language (UML) that has been widely accepted in industry [14]. The programming language is Visual C# integrated with the Visual Studio .NET platform that has a large collections of tools and libraries [15, 16]. The development cycle of the Unified Process model includes four major phases: Inception, Elaboration, Construction, and Transition, while each phase contains five workflows: Requirements, Analysis, Design, Implementation, and Test [17]. The following subsections discuss each phase in details with emphasis on different workflows.

A. Inception Phase

The inception phase is to achieve agreement between the development team and the user on the requirements, the expectations for the functionality, and the procedural approach that is to be taken to implement the application [18]. After meeting with classroom teachers and completing an initial requirement analysis, the team decided to divide the Hundred Acorn Forest into two major components: Teacher Module and Student Module. The Teacher Module is where a teacher can manage his or her student roster, create new puzzles or edit existing puzzles, and view student attempts through automatically recorded video playback. The Student Module is where a student plays the puzzle game, i.e., a student can progress through the game by solving puzzles created by the teacher in the Teacher Module.

Requirement keys were captured to scope the application in this phase (See Appendix A). In addition, inception phase includes the use case analysis [19]. The UML defines a use case as an objective users want to achieve with an application. It aims at describing a system from external usage viewpoint, rather than from

Copyright © 2015 MECS

developer's perspective. Fig. 1 is the use case diagram for the Teacher Module and Fig. 2 is the use case diagram for the Student Module.

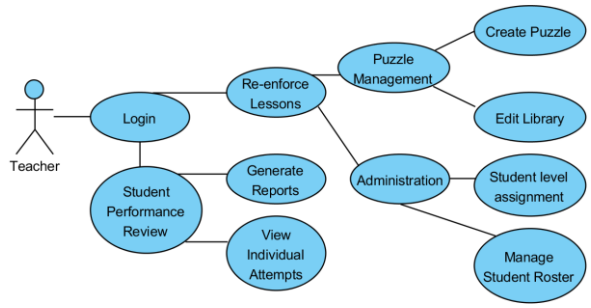

Fig.1. Use Cases of the Teacher Module.

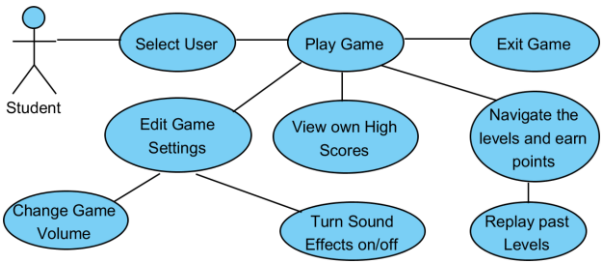

Fig.2. Use Cases of the Student Module.

B. Elaboration Phase

The elaboration phase provides an architectural baseline that implements a working application with limited functionality, and to formulate a project agreement with the user to further pursue the project [20]. The project vision, business case, requirements, and system scope were refined. The analysis workflow describes what the application is supposed to do without defining how it is done given that how to implement the application will be completed in the design workflow of the construction phase [21]. The design task in this phase is to develop a stable architecture using UML. A detailed project schedule was finalized, the initial UML class diagrams were created, and the prototype of the application was implemented.

The main documents developed in this phase were Activity Diagrams and Sequence Diagrams. An activity diagram describes procedural logic, business process, and work flow, which is like a flowchart but supports parallel behavior that allows the user to choose the order in which to do things [22]. Fig. 3 shows the activity diagram for the Teacher Module, which decomposes user actions captured in the use case into detailed activities involving conditional branches and concurrent flows.

A sequence diagram, which describes how groups of objects collaborate in some behavior, is one of the several forms of interaction diagrams defined in the UML [22]. It usually shows a number of example objects and the messages that are passed between these objects within the use case. Fig. 4 indicates the interactions and messages passed among the objects in the scenario that a student plays a game in the Student Module. Similarly, Fig. 5 illustrates a scenario in the Teacher Module that involves the actions of maintaining a student list, creating or editing puzzles, and viewing recorded puzzle solving actions of student attempts, which gives a good picture about which objects are doing which processing.

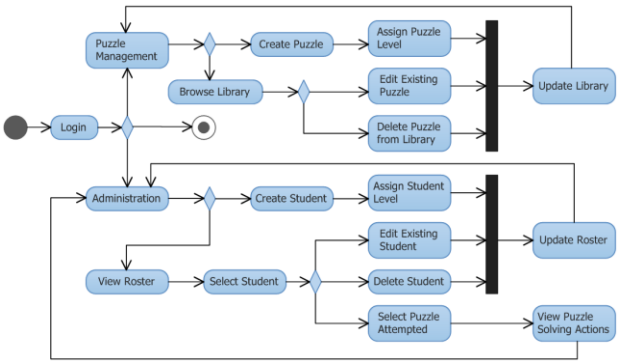

Fig.3. Activity Diagram of the Teacher Module.

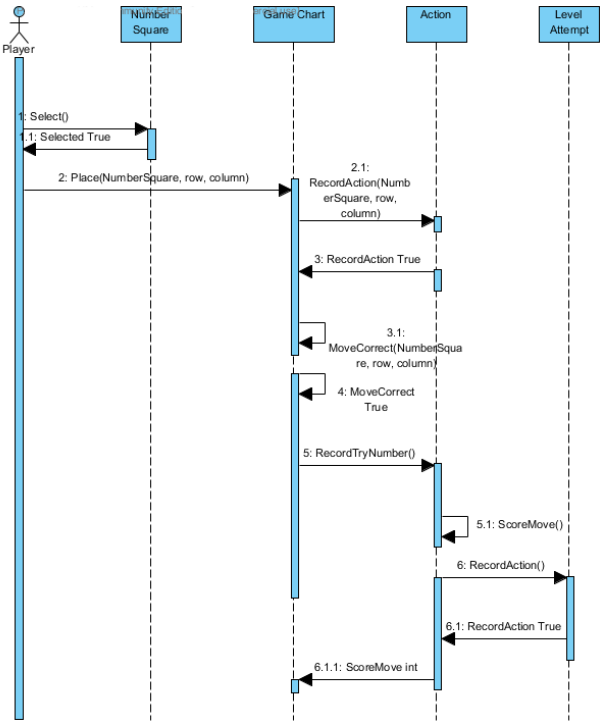

Fig.4. Sequence Diagram of the Student Module.

C. Construction Phase

The UML model with supporting documents, the software product, the test suite, and the user manuals are deliverables for the construction phase [17]. This phase reveals and analyzes any requirement that has been missed, and refines the requirements that have been implemented [21]. The design task is to complete the UML design model for functionalities to be implemented in this phase. Fig. 6 shows the UML class diagram of the Hundred Acorn Forest application, which defines classes, their attributes, operations, and relationships including aggregation, association, and inheritance.

The functionalities, as defined in the use cases and requirement keys, were implemented through four iterations. The first iteration was to implement the

Copyright © 2015 MECS

administration feature in the Teacher Module which allows a teacher to add, edit, or delete students and assign student skill levels. It also let the teacher to view students' attempts of puzzles.

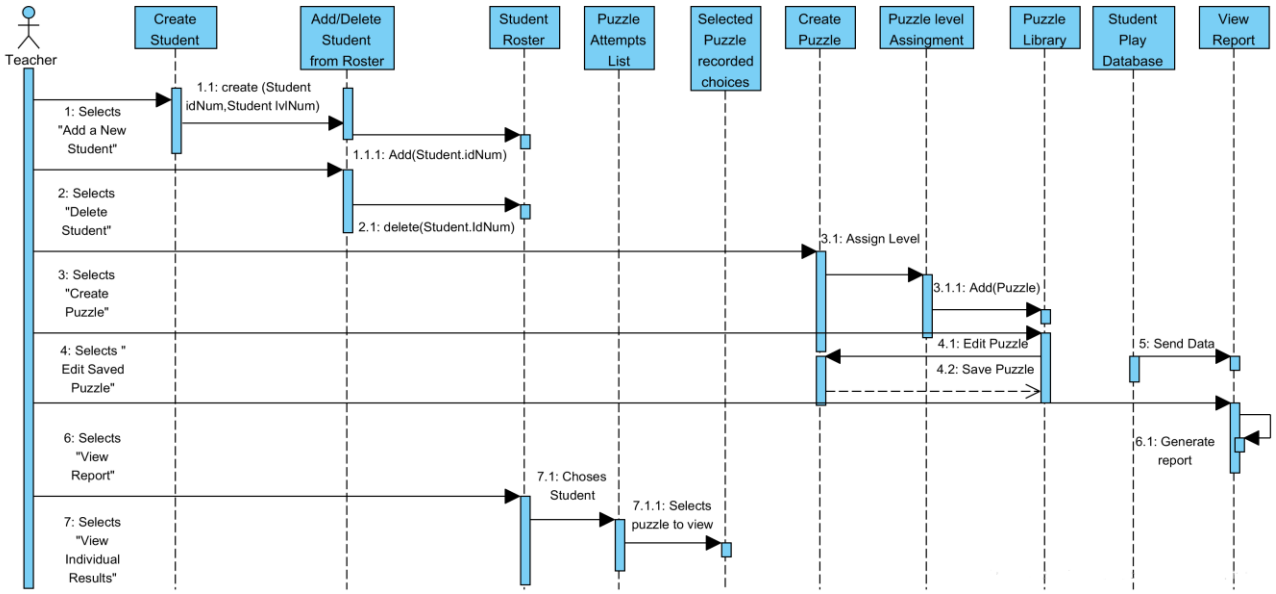

Fig.5. Sequence Diagram of the Teacher Module.

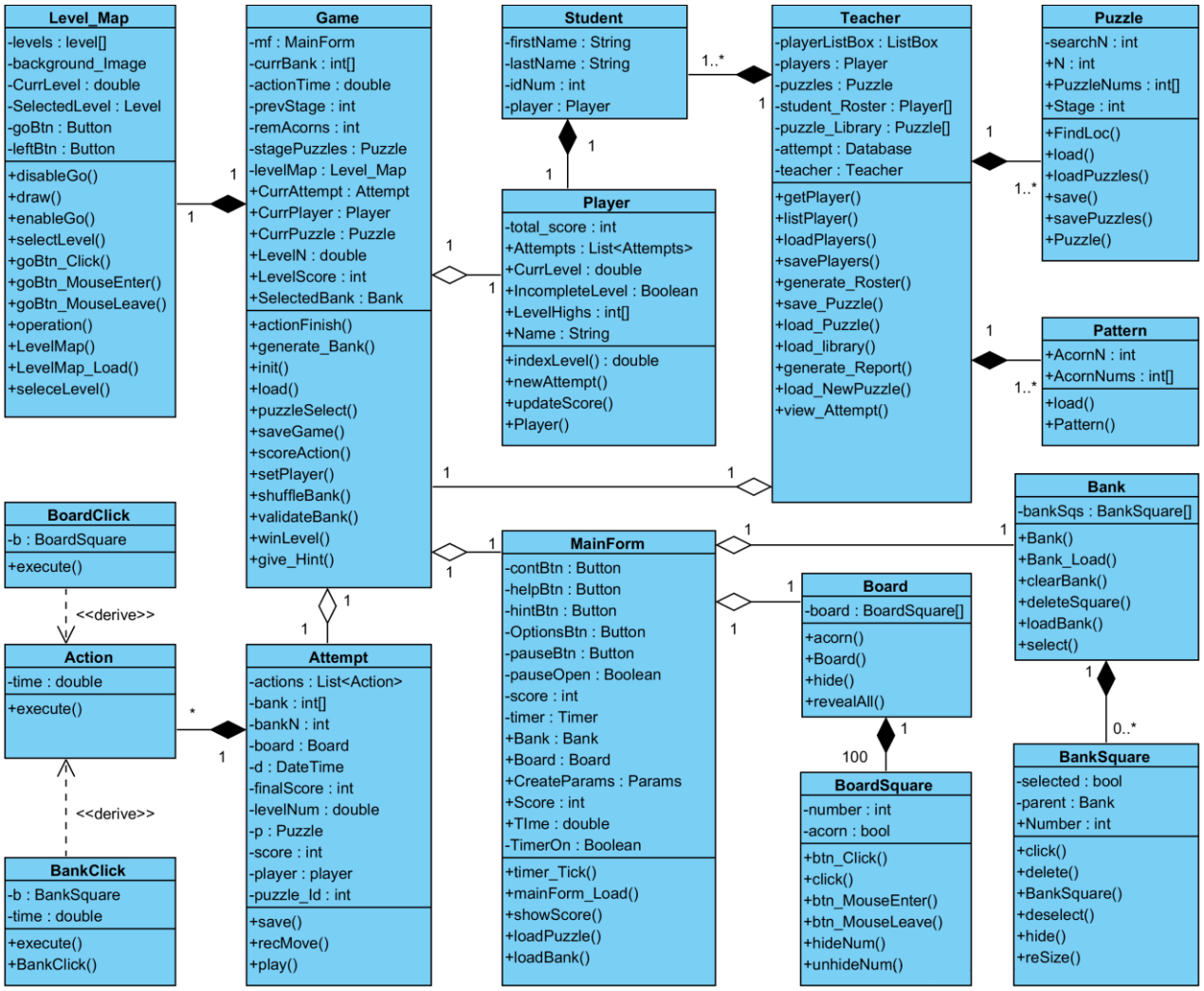

Fig.6. The UML Class Diagram of the Hundred Acorn Forest Application.

The second iteration implemented the features for creating and editing puzzles in the Teacher Module. The teacher may create two types of puzzles: Basic Puzzle or Patterned Puzzle. A Basic Puzzle is one that includes all 100 number boxes, and the teacher may select which ones are hidden freely. Creating a Patterned Puzzle is a two-step process. The teacher needs to create a new pattern or select an existing pattern with a certain sequence of numbers, and then choose what numbers to hide. Meanwhile, the teacher may assign a difficulty stage ranging from 1 to 6 to each puzzle that will be loaded into the game of the Student Module.

The third iteration implemented the game play of the Student Module. As the student progresses through the levels, puzzles will increase in difficulty according to the stages teacher has chosen during the puzzle design.

The screen recording feature was completed in the fourth and final iteration. Fig. 7 shows the code map of the classes related to video streaming and recording. The video file is in the standard AVI format which can be played back on any computer system.

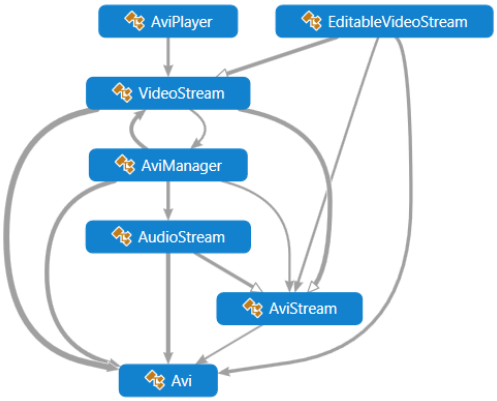

Fig.7. The Code Map of the Screen Recording Function.

During the entire construction phase, UnitTest, a software testing framework integrated into Visual Studio .NET, was used to verify that all the required functionalities are actually implemented and that the various components are working in isolation [23]. The UnitTest is seamlessly integrated with Visual Studio to take advantage of many .NET language features such as custom attributes and other reflection related capabilities. For example, UnitTest verified that the puzzles created and saved in the Teacher Module can be successfully loaded into the game in the Student Module. These tests were further extended into the transition phase.

D. Transition Phase

This phase starts after the initial application testing has been performed and the application is ready to be deployed [21]. Bugs discovered during testing are fixed or deferred to the next version, and the application is prepared for release. An integration test [23] was performed to verify each component and module work correctly together, such as the interactions between the Teacher Module and the Student Module. The defects on the remaining components were analyzed and all the known errors were fixed. A defect tracking sheet were used to record this testing and fixing process.

Transition phase also requires a way of distributing the developed software. Installation packages, which include executable binary code of the Hundred Acorn Forest application together with supporting libraries and files, were created using the Visual Studio .NET built-in setup utilities and deployment tools. The application was successfully deployed on computers in a lab.

In addition, the team manager collected and archived all the documents including requirement keys, use case diagrams, activity diagrams, sequence diagrams, UML class diagram, and defect tracking sheet, etc. The manager also exported the source code documentation from the Visual Studio .NET and composed the help file for the released software product. Archiving these development documents is very important for code reuse, software maintenance, new release, and future training.

The user manual was created to give users detailed instructions of how to use the application. Since both teachers and students will be using the application, the instructions came in two versions, one for classroom teacher's use, and the other for student's use. The teacher's version includes not only how to design and set up the puzzles, but how to track students' performance through video playbacks.

The student's version was developed with a more children-friendly tone and the complexity of vocabulary and sentence structure in the instructions were carefully controlled to make sure they are at the appropriate instructional reading level of the children. It is suggested that when classroom teachers start to teach their students how to use this application, they should gradually release teacher's responsibility by following these four steps [24]: (1) teacher models how to use it; (2) teacher invites students to join him or her to use it; (3) teacher encourages students to use it on their own but is prepared to offer timely support; (4) teacher observes students to use it independently.

IV. Experiments and Results

The design of this application adopted a document-view architecture with traditional looking form-based interface [16]. Although the implementation of the user input and data storage is quite different for Teacher Module and Student Module, the application shows the characteristics of usability, robustness, and efficiency based on the results of the acceptance test [23] that verifies the user can complete all the tasks and actions according to the requirements without problems.

A. Teacher Module

The Teacher Module has two major functionalities: Puzzle Management and Administration. Puzzle Management let the teacher to create, edit, and save puzzles. As aforementioned, a puzzle can be a Basic Puzzle created freely, or a Patterned Puzzle based on a certain pattern that can be reused.

On the Basic Puzzle design form shown in Fig. 8, the

Copyright © 2015 MECS

teacher may click on any number and it will be hidden by an Acorn. Clicking an Acorn will remove it, and the number will appear again. In the bottom left the teacher may select a stage for this puzzle based upon the level of difficulty, with 1 being the easiest and 6 being the hardest. The default stage is set to 1. Once the teacher has set the stage and hidden all the numbers he/she wishes to hide, the puzzle can be saved and added to the game. The teacher can continue to create new basic puzzles or return to the previous menu.

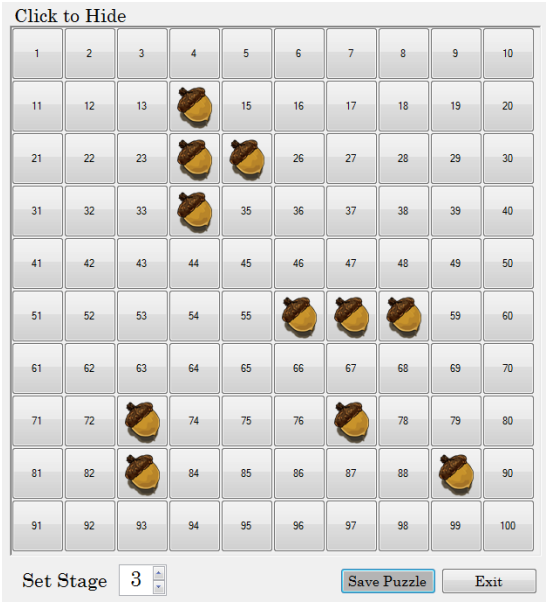

Fig.8. The Basic Puzzle design in the Teacher Module.

With the option of Patterned Puzzle, the teacher should first create a pattern, for example, the orange boxes in Fig. 9, which contain only a subset of numbers that the teacher want their students to work on. As the teacher clicks a number, the box will turn orange, indicating it belongs to a pattern. Clicking it again will return it back to a number. Once the teacher has the desired pattern, he/she may click Set Pattern at the bottom of the screen so that only the pattern will be displayed.

To create puzzles based on this pattern, the teacher can click on which numbers to hide, similar to the process of creating a Basic Puzzle. Clicking on a number will cause an Acorn to appear and hide the number while clicking an Acorn will remove it and the number will appear again. In the bottom left, the teacher may select a stage for this puzzle based upon the level of difficulty, with 1 being the easiest and 6 being the hardest. The default stage is set to 1 like in the Basic Puzzle design. Once the teacher has set the stage and all the numbers he/she wishes to hide, the puzzle can be saved and added to the game. The teacher can continue to create a new Patterned Puzzle or return to the previous menu to create a new pattern.

One of the advantages of using a Patterned Puzzle is that the teacher can quickly create similar puzzles without repeating the same process as that in the Basic Puzzle design. Additionally, in the situation when students have trouble solving puzzles that contain a certain type of sequences, the teacher may want the students to practice more on similar sequences but with different numbers and difficulty levels. Patterned Puzzle design offers the teacher a more flexible way to group or arrange puzzles of different difficulty level in each stage of a game to reinforce the students' ability to count similar sequences of numbers.

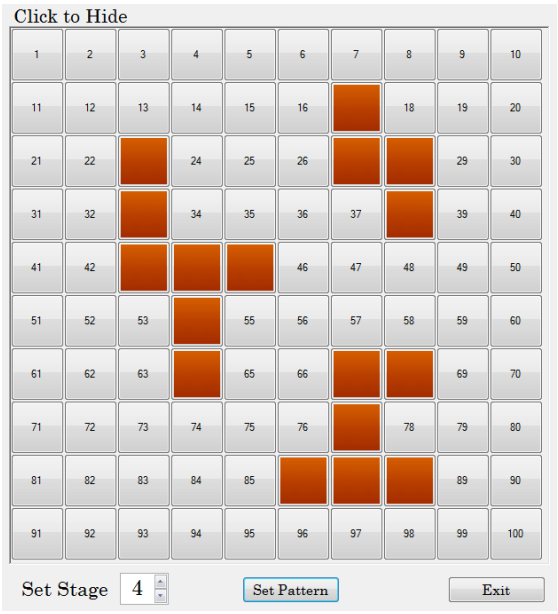

Fig.9. (a) The Patterned Puzzle Design in the Teacher Module

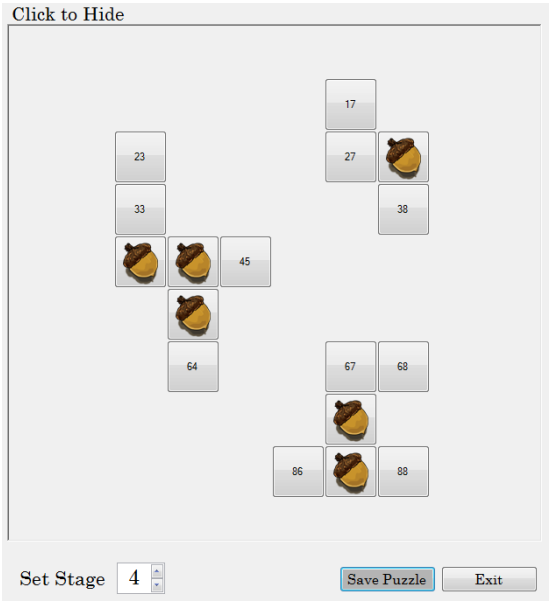

Fig.9. (b) The Puzzle Created based on the Pattern.

The Administration function in the Teacher Module loads the Student List form as shown in Fig. 10 (a), with which the teacher can add or delete students and view student attempts of solved puzzles. The Add command displays a new Student information form where the teacher may fill out student information, such as Student ID and Student Name, and meanwhile choose a Student Level (See Fig. 10 (b)). Student Levels are set at 1 to 6 corresponding to the difficulty of the puzzles appropriate for this student. The Ok command dismisses the Student

Copyright © 2015 MECS

information form and puts the student in the Student List. The teacher may select a student in the list to view the information. When deleting a student, a warning message will pop up to ask the teacher to confirm the deletion operation if the student has attempted any puzzle.

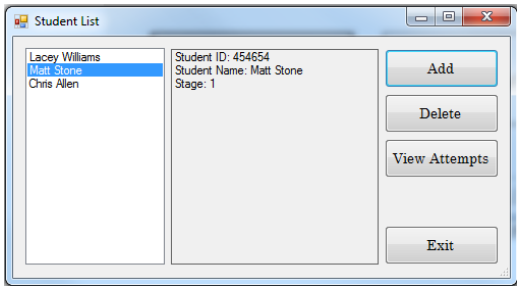

Fig.10. (a) Student List Form

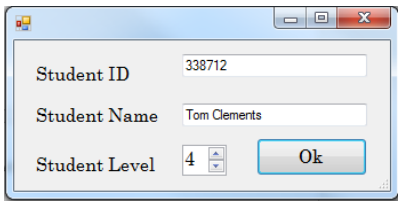

Fig.10. (b) Student Information Form

With the View Attempts command, the teacher may view a list of attempts in order of completion by the selected student (See Fig. 11). This list will be empty if the student has yet to complete a puzzle. The teacher can select an attempt labeled by a sequence number, date, time, and score. The Play command starts the recorded video showing how the student was completing the level. When finished playing, the teacher may select another attempt to view or exit this screen and return to the previous menu.

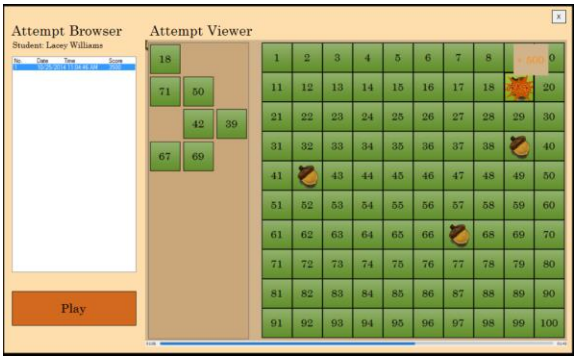

Fig.11. The Attempt Browser and Viewer in the Teacher Module.

B. Student Module

Selecting Student from the application's starting screen will take the student to the Player menu where the student will select their name from the Player list as shown in Fig. 12 (a). Once their name is highlighted the user may click Play to access the game menu. However, if this is the first time the student plays the game, he/she will be asked to set up a nickname as shown (See Fig. 12 (b)) before continuing to the Game menu. Nickname will be saved and will appear in the Player List the next time he/she returns to play. If the nickname the student selected is already being used, he/she will be asked to enter a different name.

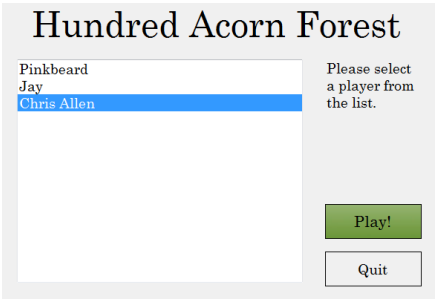

Fig.12. (a) Player Menu

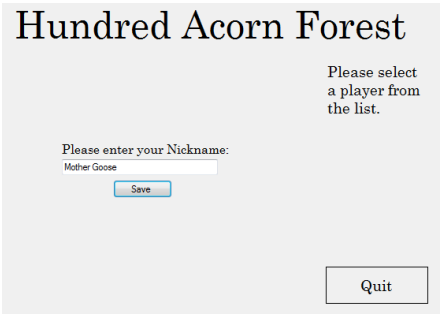

Fig.12. (b) Player's Nickname Creation

The game interface combines a quick access menu and easy-to-identify metaphors as shown in Fig. 13. The buttons to the left allow the student to change options and obtain help if they get stuck on a certain level.

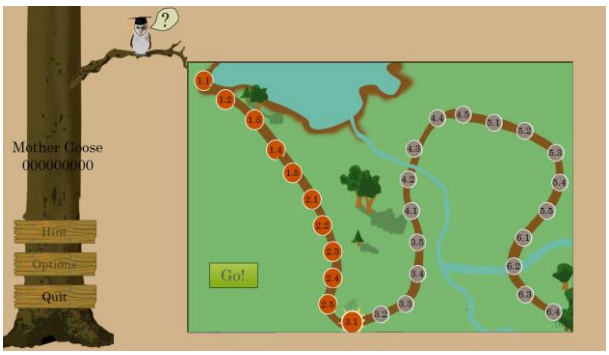

Fig.13. Game Menu and Level Selection in the Student Module.

The game has some notable features. On the Game map, the student can select what level to play. Clicking on any level he/she has attempted already will display the highest score achieved so far for that level. The student may replay any previous level to improve his/her score. However, the student is not allowed to play any levels past the first un-played level.

After choosing the level he/she wishes to play, the student clicks on the Go button to start the puzzle game. The game will randomly load a puzzle from the library of puzzles with the same difficulty created in the Teacher Module based on the level the student is currently at (See Fig. 14).

Copyright © 2015 MECS

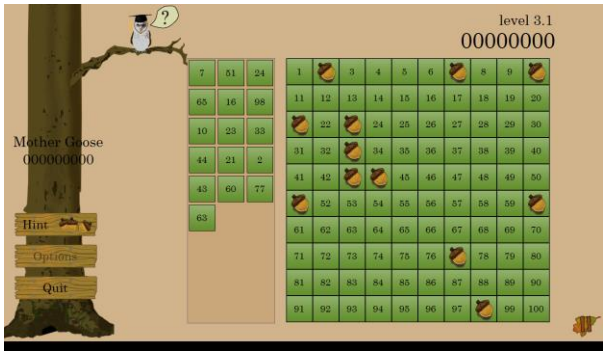

Fig.14. Game play screen in the Student Module.

Fig. 15 illustrates a typical puzzle solving process. To play, the student will click on a number from the number bank to the left and then click on the acorn where the number belongs. If correct, the acorn will crack, the number will be placed, and the score will pops up (see Fig. 15 (a), (c), and (d)). If incorrect, the tile will turn red (see Fig. 15 (b)). Each player is given 3 hints per level. As hints are used, the acorns will disappear from the hint box. When the hint is clicked, the board will highlight a number in the bank and then highlight either the row or the column it belongs in (See Fig. 15 (e)). The faster a student places a number in the correct spot, the more points are awarded. The score for the level is accumulated, and at the end of the game the score and time are saved together with the highest score of each level (see Fig. 15 (f)). The student can replay the level to improve the score or continue on to the next level.

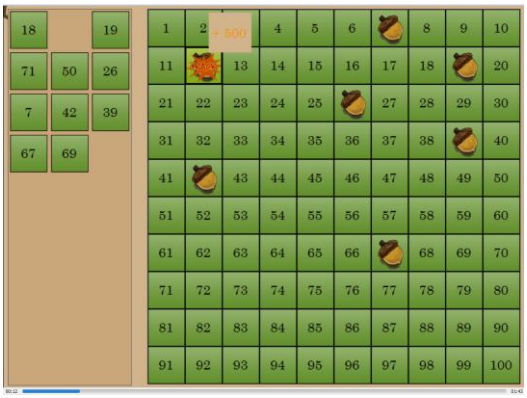

Fig.15. (a) Correct Placement of Number 12.

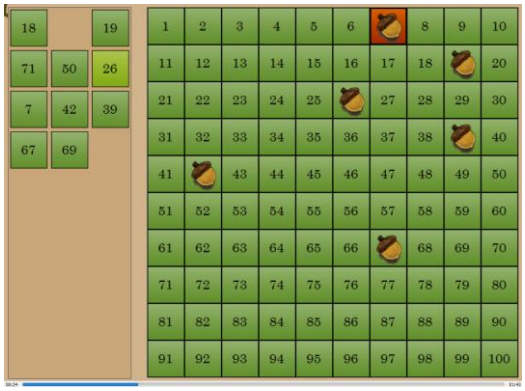

Fig.15. (b) Incorrect Placement of Number 26 on 7.

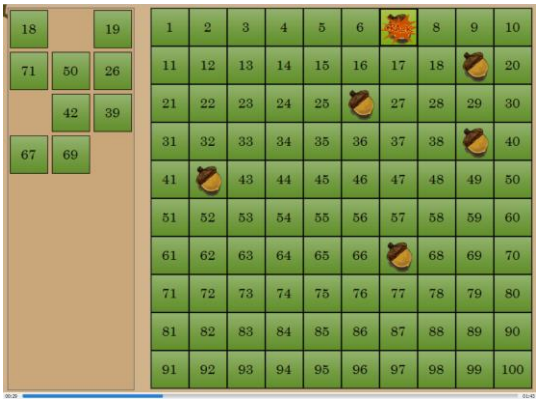

Fig.15. (c) Correct Placement of Number 7.

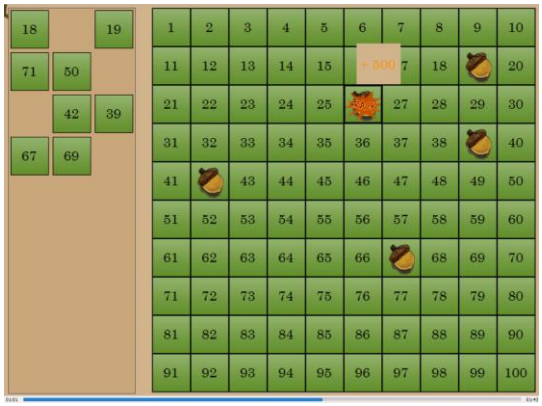

Fig.15. (d) Correct Placement of Number 26.

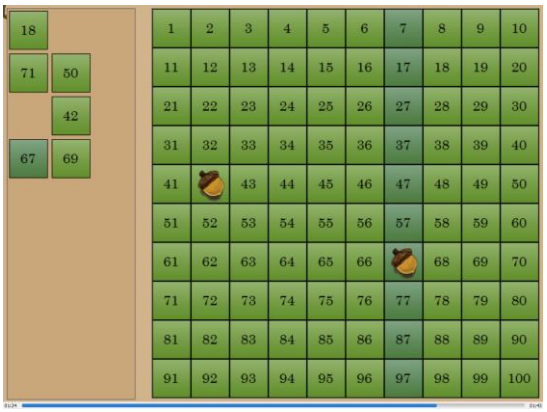

Fig.15. (e) Hint for the Placement of Number 67.

Fig.15. (f) Game Ending Screen.

V. Discussion

The Hundred Acorn Forest application is the first computer game developed to teach counting skills with Hundreds Chart puzzles. Its intelligence lies in the creation of the puzzle as well as the process of how the game is played. Not only has it allowed the teacher to make basic puzzles like those of the traditional paper-based Hundreds Chart, only without the hassle of cutting papers, but provided the teacher a handy tool to create more challenging puzzles based on the same pattern that can be reused. The interactive play with animated presentation following a story line should keep the student interested. The hint feature adds another level of control by giving the student a better chance to solve the puzzle, and therefore helps to avoid the frustration of the student who might otherwise get stuck on a level with difficult puzzles.

Furthermore, the teacher is able to efficiently manage the difficulty level of each puzzle and game as well as the skill stage of individual students. The puzzles with same difficulty levels are randomly selected to generate a sequence of game plays appropriate for a student at a certain stage. The achievement score, which should be a good measurement to reevaluate the skill stage of the student, is awarded according to how quickly a student solves a puzzle as well as whether the student uses hints in the process.

One of the unique feature of the Hundred Acorn Forest application is the video playback for the teacher to review the attempts by the students. Besides showing the score and time it takes a student to solve a certain level of puzzle, the automatically recorded video reveals what missteps the student have taken in the process of game play, whether the student has used any hints, etc., which helps the teacher to identify the specific area of the problems the student may have and therefore to enforce the learning with follow-up instructions.

VI. Conclusion

A complete cycle of using the Unified Process method to develop a computer-based Hundreds Chart puzzle game for teaching K-2 students basic counting skills has been presented. The success of this project demonstrates the effectiveness of the Unified Process method in the design of a math education application which integrates modules for both teachers and students. The teacher will benefit from the application by efficiently creating and editing puzzles, managing the difficulty of the game, evaluating the student skill levels, and reviewing the attempts by the students, while the student will enjoy the game experience to build their counting skills.

Our future work will look into encoding the recorded videos in a compressed file format, such as MP4 [25], which demands much less storage space than the AVI format. Additionally, given the unstructured information in the application, such game maps, puzzles, videos, etc., we will investigate object-oriented databases [26] that offer better indexing and searching functions than the traditional relational databases.

Appendix A Requirement Keys

Requirement Key	Description
Puzzle_Create	A teacher should be able to design his or her own Hundreds Chart puzzle by selecting which numbers will be hidden.
Puzzle_Delete	A teacher should be able to delete any saved user-created puzzles but not the default puzzles supplied in the program.
Puzzle_Load	A teacher should be able to load a puzzle from the database in order to edit it or use it as a template to create more puzzles.
Puzzle_Save	A teacher should be able to save a puzzle into a database once he or she is finished creating it. It should be possible to overwrite existing puzzles.
Pattern_Create	A teacher should be able to design his or her own Hundreds Chart puzzle by first creating a pattern of squares in the puzzle, and then choosing which numbers in the pattern will be hidden.
Pattern_Delete	A teacher should be able to delete any saved user-created patterns but not the default patterns supplied in the program.
Pattern_Load	A teacher should be able to load a pattern from the database in order to edit it or use it as a template to create more patterns.
Pattern_Save	A teacher should be able to save a pattern into a database once he or she is finished creating it. It should be possible to overwrite existing patterns.
Bank_Generate	When a student begins a level, the program must generate a number bank of Hundreds Chart blocks that contains all of the missing number blocks from the puzzle or pattern and additional random number blocks that will not be inserted into the puzzle or pattern.
Level_Edit	A teacher should be able to enable or disable certain features, like hints, for various levels.
Level_Generate	When a student begins a level, the program must load a stored puzzle or pattern of appropriate difficulty.
Level_Load	Upon continuing a game, if the student 's last play ended on a level that was not completed, that level should be loaded into play so that the student may complete the level.
Level_Lobby	Upon continuing a game, if the student's last play did not end on an incomplete level, the student should be taken to a level lobby, from which they may select a level.
Level_Map	The student's level lobby should contain a map of all levels, both complete and incomplete, that should be displayed. The student should be able to scroll through the map, but only to view levels already completed.
Level_Score	Upon selecting a level in the map, the student should be able to see their current highest score and the associated completion time and be given to the option to play the level.

Copyright © 2015 MECS

Level_Load_Saved	When a student user leaves an unfinished level, the level should be saved exactly as is so that the student may continue the level when he or she next plays the game.
Level_Scoring	Upon completing a level, a student must be awarded a score for the level that is the summation of all action scores achieved during the level play.
Level_Select	From the level map, a student may select to play the next incomplete puzzle or a puzzle that he or she has already completed in order to attempt to achieve a better score.
Action_Click	In order to place number blocks into empty blocks on the hundreds chart, a student should be able to click the block from the block bank and click the desired location on the Hundreds Chart.
Action_Hint	For certain difficulties of puzzles, when a student attempts to place a block incorrectly, if the block is in the correct horizontal or vertical row, that row will light up to give the student a hint.
Action_Record	When a student completes an action, that action, as well as any score awarded for completing it, must be recorded into a video file of all the actions completed in the particular attempt at completing the puzzle.
Action_Scoring	Action that is completed during a level by a student must be scored.
Attempt_Database	A database containing all attempts files of all students must be maintained.
Attempt_Save	Once a puzzle is completed, all of the actions completed by the student during the level will be saved as an attempt file in a database so that it may be reviewed by a teacher.
Attempt_View	A teacher should be able to view the actions completed by a student in any one of their attempts.
Student_Add	A teacher should be able to add a student to the student list.
Student_Delete	A teacher should be able to delete a student from the student list.
Student_Edit	A teacher should be able to edit a student's current stage level.
Report_Generate	A teacher should be able to generate a report on a given student that displays their overall score and the student's scores and times for all completed levels.
Player_Create	A student should be able to begin a new game and create a username if they have been added by a teacher into the student list and have not yet begun a game.
Player_Save	Upon leaving or logging out of the game, the player's score (only taking into account the highest score achieved for each level), and current level state must be saved.
Player_Select	Upon opening the game, a student must select his or her username from a list to start playing.
Teacher_Login	A teacher must be able to access the program with a password.

ACKNOWLEDGMENT

The authors wish to thank faculty in the Department of Mathematics and Statistics for their support on the requirements analysis of the application. This research was also supported by faculty in the Martha Dickerson Eriksson College of Education at Austin Peay State University.

REFERENCES

[1] R. E. Reys, et al, *Helping Children Learn Mathematics*, Danvers, MA: John Wiley & Sons, 2014.

[2] N. N. Vacc, "Gaining number sense through a restructured Hundreds Chart," *Teaching Exceptional Children*, vol. 28, pp. 50–55, 1995.

[3] D. S. Niederhauser and T. Stoddart. "Teachers' instructional perspectives and use of educational software," *Teaching & Teacher Education*, vol. 17, pp. 15–31, 2001.

[4] P. D. Pearson and J. A. Dole. "Explicit comprehension instruction: A review of research and a new conceptualization of instruction," *The Elementary School J.*, vol. 88, pp. 151–165, 1987.

[5] D. Ogle and J. W. Beers, *Engaging in the Language Arts: Exploring the Power of Language*, Boston, MA: Pearson Publishers, 2012.

[6] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*. NJ: Prentice Hall PTR, 2002.

[7] J. A. Kulik, C. C. Kulik, and R. L. Bangert-Drowns, "Effectiveness of computer-based education in elementary schools," *Computers in Human Behavior*, vol. 1, pp. 59–74, 1985.

[8] N. Balacheff and J. J. Kaput, "Computer-based learning environments in mathematics," *International Handbook of Mathematics Education*, Springer Netherlands, pp. 469–501, 1996.

[9] A. I. Khan, S. Mahaboob, A. M. Ali, and C. V. Bebi. "Study of blended learning process in education context." *Int. J. Modern Edu. & Comp. Sci. (IJMECS)*, vol. 4, pp. 23–29, 2012, DOI: 10.5815/ijmecs.2012.09.03.

[10] S. Kim and M. Chang, "Computer games for the math achievement of diverse students," *J. Educational Technology & Society*, vol. 13, pp. 224–232, 2010.

[11] F. Ke, "A case study of computer gaming for math: Engaged learning from gameplay?" *Computers & Education*, vol. 51, pp. 1609–1620, 2008.

[12] M. Kebritchi, A. Hirumi, and H. Bai, "The effects of modern mathematics computer games on mathematics achievement and class motivation," *Computers & Education*, vol. 55, pp. 427–443, 2010.

[13] L. A. Maciaszek and B. L. Liong, *Practical Software Engineering. A Case Study Approach*, Harlow England: Addison-Wesley, 2005.

[14] G. Lenz and T. Moeller, *.NET - A Complete Development Cycle*, Boston, MA: Addison-Wesley, 2003.

[15] A. Troelsen, *Pro C# 5.0 and the .NET 4.5 Framework*, Berkeley, CA: Apress, 2012.

[16] D. Esposito and A. Saltarello, *Microsoft .NET: Architecting Applications for the Enterprise*, Redmond, WA, Microsoft Press, 2008.

[17] P. Kruchten, *The Rational Unified Process: An Introduction*, Boston, MA: Addison-Wesley, 2003.

[18] D. R. Windle and L. R. Abreo, *Software Requirements*

Using the Unified Process: A Practical Approach, NJ: Prentice Hall PTR, 2002.
[19] F. Armour and G. Miller, *Advanced Use Case Modeling*, Boston, MA: Addison-Wesley, 2000.
[20] J. Li, "Teaching unified process in software design and development courses – a case study," *The Journal of Computing Sciences in Colleges*, vol. 24, pp. 5–11, 2009.
[21] J. Arlow and I. Neustadt, *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*, Boston, MA: Addison-Wesley, 2005.
[22] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Boston, MA: Addison-Wesley Professional, 2003.
[23] L. Copeland, *A Practitioner's Guide to Software Test Design*, Norwood, MA: Artech House Publishers, 2004.
[24] P. D. Pearson and M. C. Gallagher, "The instruction of reading comprehension," *Contemporary Educational Psychology*, vol. 8, pp. 317–344, 1983.
[25] R. Koenen, F. Pereira, and L. Chiariglione, "MPEG-4: Context and objectives," *Signal Processing: Image Communication*, vol. 9, pp. 295–304, 1997.
[26] K. R. Dittrich, "Object-oriented database systems: the next miles of the marathon," *Information Systems*, vol. 15, pp. 161–167, 1990.

Authors' Profiles

Jiang Li received his Ph.D. in Electrical Engineering from the University of Nebraska – Lincoln, USA. He also holds B.S. and M.S. degrees in Electronics Engineering from Beijing Institute of Technology, China.

He is a Professor in Computer Science and Information Technology at Austin Peay State University, USA. His research interest is in image processing, machine learning, data mining, software engineering, and distributed computing. He has published over 20 refereed articles in international journals and conference proceedings. Dr. Li is a member of ACM and he currently serves on the ODBMS.ORG's Panel of Experts.

Ling Wang received her Ph.D. in Literacy Studies from the Middle Tennessee State University, USA. She also holds a M.A. Ed. Degree in Reading from Austin Peay State University, USA and a M.A. degree in Foreign Languages and Applied Linguistics from Shandong University of Finance, China.

She is an Assistant Professor in Education at Austin Peay State University, USA. Her research interest is in educational multimedia, literacy studies, and foreign language acquisition. She has published several refereed journal and conference articles. Dr. Wang is a member of International Literacy Association and she currently serves on the Editorial Boards of Journal of Educational Multimedia and Hypermedia and Journal of Technology and Teacher Education.

Lacey L. Williams holds a B.S. degree in Computer Science and Information Systems from Austin Peay State University, USA.

Christina A. Allan holds a B.S. degree in Computer Science and Information Systems from Austin Peay State University, USA. She is currently a Professional Science Masters student in Data Management and Analysis at Austin Peay State University, USA.