Int. J. Engng Ed. Vol. 24, No. 4, pp. 681–688, 2008
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Developing and Evaluating a Game-Based Software Engineering Educational System*

WEN-HSIUNG WU
Department of Information Management, National Kaohsiung University of Applied Sciences, 415 Chien-kung Road, Kaohsiung 807, Taiwan, R.O.C.

WEI-FAN CHEN
College of Information Sciences and Technology, The Pennsylvania State University, P.O. Box PSU, Lehman, PA 18627-0217, USA. E-mail: weifan@psu.edu

TSUNG-LI WANG
Institute of Manufacturing Engineering, National Cheng Kung University, 1, University Road, Tainan 701, Taiwan, R.O.C.

CHUNG-HO SU
Department of Software Engineering, D&S Digital Technology Co., Ltd, 16F-2, 110, San-Duo 4th Road, Kaohsiung 802, Taiwan, R.O.C.

Research in software engineering education has, in recent years, attempted to achieve the equilibrium between academia and practice. The software engineering education research community has obtained a number of valuable outcomes in the areas of content curriculum, pedagogy, and technology, respectively. However, very few studies have successfully integrated these three dimensions into a single learning environment. This study developed and evaluated a Game-Based Software Engineering Educational System (GBSEES) for software engineering education. GBSEES adopted a role-playing strategy using a digital game-based learning model. This game-based system was based on the educational theory of Technological Pedagogical Content Knowledge, which integrates pedagogical knowledge, content knowledge, and technological knowledge. In the game-based learning system, students learned about the process of software development in a team-based environment by using a role-playing gaming strategy. The study also investigated the effect of the GBSEES on the students' attitude to learning.

Keywords: software engineering education; technological pedagogical content knowledge; digital game-based learning; system development and evaluation

INTRODUCTION

SOFTWARE ENGINEERING EDUCATION is facing challenges in the development of curriculum and instruction as the advancement of technological innovations accelerates. Blake [1] observed that traditional software engineering education over-emphasized teacher-centered instruction and failed to involve students in real-world applications. He proposed a model of "student-enacted simulation" to improve student learning in software engineering education. By the same token, Hadjerrouit [2] argued that software engineering educators should not only teach practical applications of engineering concepts, but also design a student-centered environment. Saiedian [3] suggested that software engineering curricula should adopt a more realistic approach to produce future skillful software engineers.

Previous studies in software engineering education focused on discussing its curricular content, pedagogy, and technology. For the curricular content, Ohlsson and Johanson [4] proposed a new software engineering curriculum based on practical perspectives for teaching content. For the pedagogy, Blake [1] used a pedagogical strategy of simulation for implementing a large software engineering project. For the technology, Rodriguez et al. [5] applied an e-learning technology platform to the software engineering curriculum; Baker et al. [6] developed an educational card game to assist in teaching a project management unit. However, those studies did not integrate the three dimensions of content, pedagogy, and technology into a single learning environment. Additionally, very few studies provided a theoretical foundation of using advanced technology in software engineering education.

Therefore, the purpose of this study is to develop and evaluate a Game-Based Software Engineering Educational System (GBSEES) for

* Accepted 25 April 2008.

software engineering undergraduate students. GBSEES adopts a role-playing strategy in a digital gaming environment proposed by a digital game-based learning model [7]. It is designed according to the educational theory of Technological Pedagogical Content Knowledge [8–9] that integrates pedagogical knowledge, content knowledge, and technological knowledge. In the game-based learning system, students are able to understand the process of software development in a team-based environment by using a role-playing gaming strategy. The study also investigates the effect of the GBSEES on student learning achievement and attitude.

LITERATURE REVIEW

Software engineering education

The terminology 'software engineering' was first introduced in the late 1960s. Software engineering is currently a core course in several computing disciplines: computer science, computer engineering, software engineering, and information systems [10]. Moore [11] argued that the most important issue was to define an acceptable body of knowledge for software engineering. Hence, researchers in the software engineering community focused on organizing existing knowledge and attempted to find ways to transform this knowledge into a curriculum. These efforts resulted in the Software Engineering Body of Knowledge (SWEBOK) and SE2004 projects [12–14]. Notably, SWEBOK reflected an agreement of what a software engineer with a bachelor degree or four years of practical experience should know. The SE2004 offered curriculum guidelines, such as the knowledge required to develop software in both technical aspects (i.e., analysis and design) and managerial aspects (i.e., quality management), for teaching an undergraduate software engineering degree program. Additionally, the SE2004 project provided sample courses and curriculum patterns.

Dieste et al. [15] indicated that software engineering education faced a series of challenges in recent years. For example, real-world applications of software engineering concepts did not match with current software engineering curriculum in some cases as also mentioned by Blake [1]. Hence, researchers provided different approaches, such as practice-driven simulation of a large project with group work, developing Web-based instruction, or applying an e-learning platform, to bridge the gap between academia and practice. Ohlsson and Johanson [4] advocated applying a practice-driven approach rather than a theoretical approach to software engineering education. They observed that students were enthusiastic about the role-playing method used in project courses. Brereton et al. [16] indicated that group work had a crucial function in the software engineering curriculum. Computer science students at three U.K. universities conducted group-work using low-cost communication tools (i.e., desktop video conferencing) to facilitate collaboration among universities. Blake [1] indicated that most software engineering courses focused on a 'theory-push,' that is, teachers lecture on the basic concepts and methodology; yet students did not have an opportunity to work on practical team projects. Hence, he further proposed a 'student-enacted simulation' approach to simulate a real-world project team in improving software engineering instruction. In addition, Hadjerrout [2] proposed simulating real-world applications by integrating real-world applications into academic teaching, and by using Web technology to implement this innovation. Rodriguez et al. [5] applied an e-learning system (i.e., project simulator) to assist students in learning project management. Experimental results indicated that the group using the e-learning system had a significantly better understanding of the topic of project management than the group that did not use it.

Technological Pedagogical Content Knowledge

Based on the theoretical foundation of Pedagogical Content Knowledge created by Schulman [17–18], Koehler et al. [19] proposed a new educational theory—Technological Pedagogical Content Knowledge (TPCK). This framework consisted of three areas of knowledge: content, pedagogy, and technology that stressed the connections, interactions, affordances, and constraints among these three areas of knowledge. Content is the subject matter taught and learned; technology encompasses standard technologies, such as a blackboard, and advanced technology such as the Internet or games; pedagogy comprises the process and practice or methods of teaching and learning such as methods used to teach and strategies for evaluating student learning.

Regarding the connections and interactions among those three areas, Koehler et al. [20] depicted three types of knowledge: Pedagogical Content Knowledge (PCK), Technological Content Knowledge (TCK), and Technological Pedagogical Knowledge (TPK). Notably, PCK, proposed by Schulman, encompassed representation and formulation of concepts, pedagogical approaches, knowledge of what makes concepts difficult or easy to learn, knowledge of student prior knowledge, and epistemological theories. TCK involved understanding the manner in which technology and content were reciprocally related, to each other. TPK stressed the existence, components, and capabilities of various technologies teachers used in the teaching and learning context.

By considering all three areas, Koehler et al. [21] proposed the Technological Pedagogical Content Knowledge (TPCK). TPCK requires understanding the representation and formulation of concepts using technologies, pedagogical methods that utilize technologies to teach content, knowledge of what makes concepts difficult or easy to learn,

how technology can help address these issues, and knowledge of student prior knowledge and epistemological theories.

A variety of instructional techniques, such as workshops, tutorials and technical support groups, has been developed and implemented by higher education institutions to assist teachers in developing content and pedagogy. However, Koehler et al. [19] indicated that these approaches had weakness: they were simplistic and ignored the complexities inherent in technology based on pedagogy of specific content areas. Hence, they proposed a design team approach, called 'learning by design,' to overcome this weakness. This approach depended on the process used to design and develop the necessary skills and relationships for understanding the nuances inherent in integrating technology, pedagogy, and content. Additionally, this approach attempted to make faculty adept at negotiating the interactions among pedagogy, content, and technology by enhancing their competencies in using technology, and by sharing their experiences in these three areas.

According to Koehler et al. [19], the learning-by-design approach had three stages: (1) getting started, (2) solidifying roles and grappling with issues, and (3) bringing it all together. The first stage defined roles in a design team and determined what an online course should be. The second stage established the roles and responsibilities of group members. The third stage was concerned with time management issues for group members.

To evaluate the development of the approach, both quantitative and qualitative methods, such as surveys or case studies, can be utilized. For instance, Koehler and Mishra [21] used a survey to assess students' and teachers' perceptions about their learning context, theoretical and practical knowledge of technology, online course content, and the growth of TPCK. Koehler et al. [20] conducted a quantitative discourse analysis to investigate fifteen-week field notes for two design teams whose participants moved from considering technology, pedagogy, and content as independent constructs to a richer conception that stressed the interconnections among these three knowledge bases. Furthermore, Niess [22] used five case studies in science and mathematics education to identify the difficulties and success using technology in molding into the TPCK.

Digital Game-Based Learning

Applying a game-based design or game-based tool to the teaching and learning context has recently become a trend. Prensky [7] indicated that higher education institutes were where Digital Game-Based Learning (DGBL) was making great headway as an increasing number of teachers had realized the power of games for engaging and instructing students. However, colleges and universities may currently be facing an increasing challenge. For instance, instructors have traditionally been conservative bastions of knowledge that changes slowly; however, knowledge now moves very rapidly. Furthermore, they have to teach 'game generation' students. Hence instructors need to make the effort to use complex DGBL tools or developing systems.

Commercial off-the-shelf games, such as SimCity, are integrated into classrooms. Likewise, classroom teachers have performed resource-intensive work and attempted to integrate a DGBL system into their classrooms. For example, in medical education, Mann et al. [23] implemented an interactive game system to teach surgical management algorithms. This system was developed using Microsoft Visual Basic; interactive 3-D physical examination simulations were created using NewTek lightware version 6. In addition, Roubidoux et al. [24] developed an interactive Web-based breast imaging game using JavaScript. In computer science education, Baker et al. [6] developed a 'Problems and Programmers' system, an educational card game that practically simulated the software engineering development cycle based on the waterfall model, which was not sufficiently highlighted via traditional lectures and projects. In sum, researchers found that instruction incorporating game features led to improved teaching and learning. However, developing an effective DGBL system in higher education still needs further work.

CONCEPTUAL MODEL

Based on the theoretical foundations listed above, this study proposes a conceptual model for developing a Game-Based Software Engineering Educational System (GBSEES) (Fig. 1). The model includes three modules: (1) input resources; (2) developmental process in the GBSEES, and (3) evaluation of the GBSEES from the perspective of input–process–output (IPO).

The input module of the model concerns content, pedagogy, and technology. This module encompasses instructional content of software engineering, pedagogy of software engineering, and gaming technology. The instructional content of software engineering consists of subject matters such as process modeling. For the pedagogy of software engineering, this work adopts DGBL strategies, such as role playing, for teaching and evaluating student learning performance. For the gaming technology, this work selects a game-based development tool to simulate real-world software development tasks. Based on the interactions among the three inputs, TPCK can be created as a foundation for developing processes in the GBSEES.

The process module of the model is concerned with developmental processes in the GBSEES. The developmental processes are modified from the learning-by-design approach proposed by Koehler et al. [20]. The processes mainly depict the system

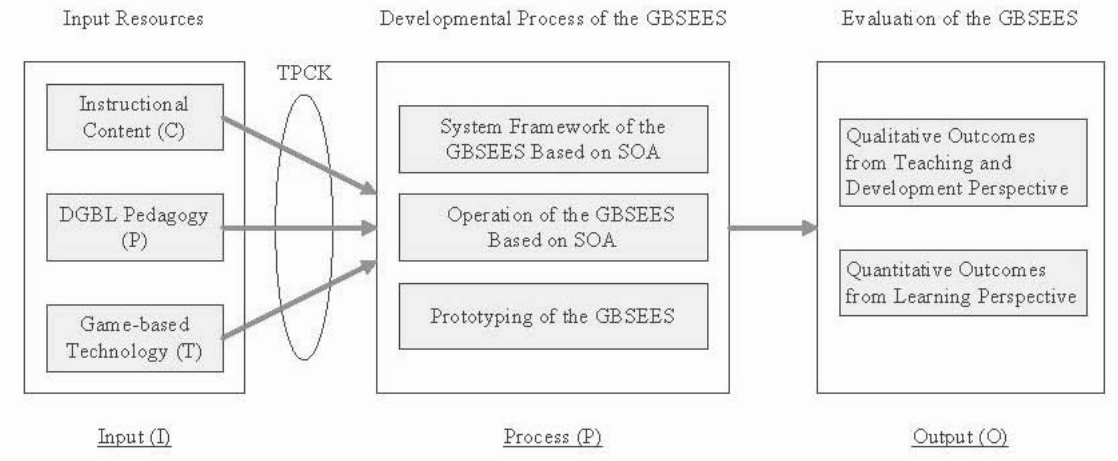

Fig. 1. Conceptual model for developing a Game-Based Software Engineering Educational System (GBSEES).

architecture of the GBSEES based on a Service-Oriented Architecture (SOA), operations of the GBSEES, and GBSEES prototype with single-player and multiple-player modes as well as an exam mode.

The output module of the model evaluates the GBSEES. This work collects qualitative data based on seven categories from TPCK such as content-only (C), and three categories—success, difficulties, and solutions—after the prototyping of the GBSEES was completed. Additionally, this work gathers quantitative results about student attitudes when they use the GBSEES prototype. Data are collected via a questionnaire developed by the Flow Theory [25] and the Technology Acceptance Model (TAM) [26].

DEVELOPMENT AND EVALUATION

Phase I—Input resources

The content of software engineering curriculum includes the introduction to software engineering, software process, object-oriented analysis, design, and testing, and project management issues. This study focuses on the content knowledge type of Project Management to illustrate the design of DGBL content. The Project Management content focuses on effective skills in software project management by describing the four P's: People, Product, Process, and Project. The topic of People is concerned with the stakeholders (i.e., project managers) who participate in the software process and the manner (i.e., coordination and communication) in which they are organized to perform effective software engineering skills. The topic of Product depicts the software scope and problem decomposition at the beginning phase of the project. The topic of Process mainly describes the selection of an appropriate process model, the definition of a preliminary plan, and the process decomposition. The topic of Project is concerned with conducting planned and controlled software projects.

The system adopts a role-playing approach for developing game-based learning services. By using this approach, learners play a character that has a 'type' and a set of individual characteristics for a learner [7]. In this learning environment, learners can play different characters, such as a project leader, a system analyst, a system designer, or a programmer. The learners are also able to conduct collaborative activities with other team members during the software development process.

In addition, the system adopts Windows XP and Windows Server 2003 as the developmental platforms. Three software development tools are used to design the game-based system. Shusaku Super 4^{TM} is employed to create shapes and poses, such as sitting or walking, of a virtual role. PhotoImpact 8^{TM} is employed to integrate stage sets, including backgrounds, dialog boxes, frames, and all stage properties. The C# language is applied for interactive activities among characters, such as sending and receiving messages between two characters.

Phase II—Developmental process

The developmental process of the GBSEES modifies the learning-by-design approach proposed by Keohler et al. [20]. The development team includes classroom teachers, a doctoral student teacher, and an expert from a software vendor, thereby reflecting the importance of integrating academic and practical experiences associated with software engineering education.

The operation of the GBSEES first addresses the game-entry procedure and then describes the two major play modes: game operational mode and exam mode. The game operational mode is further divided into single-player mode and multi-player mode. The single-player mode allows learners to play a role, such as project manager, in the GBSEES. The multiple-player mode enables learners with different roles to interact among them-

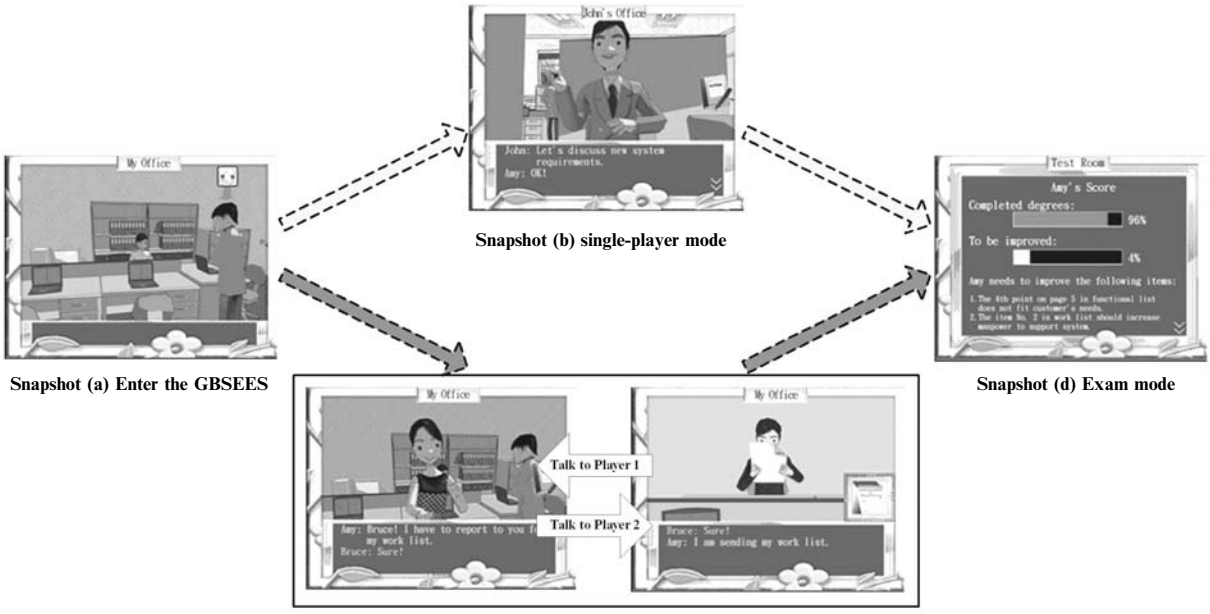

Fig. 2. GBSEES prototype.

selves (e.g., exchanging task assignments and experiences) in the GBSEES. After the learning process is completed, the exam mode evaluates students' learning performance. The game-entry procedure and the two major play modes are explained as follows (Fig. 2).

Figure 2(a) shows the snapshot of the game-entry procedure of the GBSEES. First, (1) the game scene is presented and a user selects a single-player mode or a multi-player mode. Next, learners enter (2) the character selection screen, which allows learners to choose one of four characters with different roles, namely, a project manager, a system analyst, a system designer, or a programmer. The next screen (3), the main topic-selection screen, allows learners to choose from various software engineering topics, such as managing software projects. After choosing one of the main topics, learners enter (4) the subtopic selection screen, where content knowledge for the main topic chosen is listed, such as project management. When the subtopic is chosen, (5) the game operation mode starts. By clicking on the chat icon, learners are able to browse game stages and fulfill the assignments based on the role they are playing, such as manpower allocation for the project manager. Finally, after completing an assignment, learners enter exam mode to evaluate their learning performance.

Figure 2(b) presents a sample game screen shot in which a system analyst, Amy, played by a learner, asks for an interview with a virtual customer, John, in a new system development process. First, (1) John describes new system requirements to Amy. Meanwhile, (2) Amy records customer's requirements using her notebook located in the upper portion of the screen; the interaction via dialog box between John and Amy is shown in the bottom portion of the screen. Finally, when the interview is finished, (3) Amy arranges the requirements as functional items and generate a work list. In short, via this single-player mode usage, a learner can study the content knowledge of project management as described in a textbook, and then learn the professional skills associated with a given role in the real world by interacting with a virtual role.

Figure 2(c) presents a snapshot of the interaction between two learners, the system analyst—Amy and the project manager—Bruce. This interaction starts with Amy presenting her work list to Bruce. First, (1) Amy discusses her work list with Bruce and waits for his response by clicking the chat icon. (2) Bruce receives Amy's request and is willing to start this conversation. In the next steps, (3) Amy asks Bruce to review her work list. (4) Bruce asks Amy to send the work list to him. (5) Amy sends her work list to Bruce. (6) Bruce then receives and reviews her work list. (7) Bruce sends his comments to Amy. (8) Amy receives Bruce's feedback. To summarize, the learners can study the content knowledge of project management as addressed in a textbook, and learn professional skills associated with different roles in a project team through the interaction between (or even among) players in multi-player mode.

Figure 2(d) shows a snapshot of the exam mode after Amy, played by a learner, completes an interview regarding the system requirements. First, (1) learner Amy is requested to complete an exam in which she needs to answer questions related to the assignments. During the exam, (2) learner Amy can refer to information recorded in

her notebook. After the exam, (3) the GBSEES presents the completed degree of learning performance (i.e., 96%) and the degree of requiring improvement (i.e., 4%), along with verbal comments and suggestions to the learner.

Phase III—Evaluation

To evaluate the GBSEES prototype from a teaching and development perspective, Appendix A presents qualitative outcomes. The top row in the table of Appendix A has seven categories: Content-only (C); Pedagogical-only (P); Technology-only (T); Joint consideration of Content and Pedagogy (CP); Content and Technology (CT); Pedagogy and Technology (PT); and, Content, Pedagogy, and Technology taken together (CPT). The left-hand column in the table lists three categories: Successes, Difficulties, and Solutions. Those outcomes demonstrate that the concept of TPCK can be applied to a software engineering curriculum for software engineering educators.

From the learners' perspectives, thirty-four undergraduate students in a software engineering program at a university of technology in Taiwan were recruited to evaluate the GBSEES. Of the 34 participants who completed a questionnaire, 19 (55%) were male and 15 (45%) were female. Their age level ranged from 21 to 23.

To evaluate the GBSEES, all the participants first gathered in a computer room. The computer room was equipped with 50 Windows XP desktops with an Internet connection that allowed participants to access the GBSEES. After the participants were randomly assigned to their seats, the researchers spent 20 minutes explaining the purpose of the research study and showing them how to operate the GBSEES. The participants then self-learned the content delivered by the GBSEES for 45 minutes. After the self-learning session ended, the participants completed a 16-item questionnaire within 20 minutes.

The questionnaire was developed based on the Flow Theory and the Technology Acceptance Model (TAM) [25–28]. The 16-item questionnaire was used to assess four major constructs: (1) Flow experience, (2) Interaction and Overall Use, (3) Usefulness, and (4) Intention to Use. The answers of the items were graded using a 5-point Likert scale (1 for 'strongly disagree' and 5 for 'strongly agree'). Appendix B presents the means and standard deviations for all items. Overall, the survey results indicated that students had a positive learning attitude toward the GBSEES usage.

CONCLUSION

This study developed and evaluated the game-based software engineering educational system for software engineering education. This game-based system adopted a role-playing strategy according to a digital game-based learning model. In addition, it was designed based on the educational theory of technological pedagogical content knowledge that integrated pedagogical knowledge, content knowledge, and technological knowledge. In the game-based learning system, students learned about the process of software development in a team-based environment by using a role-playing gaming strategy. The performance of the system prototype was assessed based on different perspectives from the development team, classroom teachers, and students. This result indicated that the system had a positive impact on students' learning process.

The study intends to contribute to the field of software engineering education in two ways. First, the study implements the educational theory, TPCK, into software engineering curricula by developing a game-based educational system, GBSEES. TPCK lays the foundations of understanding effective teaching with technology in a software engineering curriculum.

Second, the study adopts a collaborative model for developing a game-based learning system. This collaborative model involves a development team of classroom teachers, a doctoral student, and an expert from a software vendor. The team-based design approach bridges the gap between academia and practice and therefore improves the learning-by-design approach proposed by Keohler et al. [20]. Additionally, it provides a systematic analytical framework and design procedures for developing such a game-based learning system.

Future research should focus on evaluating students' authentic learning outcomes that would include facts, concepts, comprehensions, problem solving skills and other higher critical thinking skills while they use such a game-based learning system. In designing future research agenda in software engineering education, a quantitative methodology should be used to assess the effectiveness and efficiency of the game-based learning system by comparing with other learning system platforms. In addition, future studies should consider learners' prerequisites and learning styles to further investigate how they interact with a game-based learning system.

REFERENCES

1. M. B. Blake, A student-enacted simulation approach to software engineering, *IEEE Transactions on Education*, **46**(1), 2003, pp. 124–132.
2. S. Hadjerrouit, Learner-centered Web-based instruction in software engineering, *IEEE Transactions on Education*, **48**(1), 2005, pp. 99–104.
3. H. Saiedian, Bridging academics software engineering education and industrial needs, *Computer Sciences Education*, **12**(1–2), 2002, pp. 5–9.

4. L. Ohlsson, and C. Johansson, A practice driven approach to software engineering education, *IEEE Transactions on Education*, **38**(3), 1995, pp. 291–295.
5. D. Rodriguez, M. A. Sicilia, J. J. Guadrado-Gallego, and D. Pfahl, e-Learning in project management using simulation models: A case study based on the replication of an experiment, *IEEE Transactions on Education*, **49**(4), 2006, pp. 451–463.
6. A. Baker, E. O. Navarro, and A. van der Hoek, An experimental card game for teaching software engineering processes, *Journal of Systems and Software,* **75**, 2005, pp. 3–16.
7. M. Prensky, *Digital Game-based Learning*, McGraw Hill: New York, 2001.
8. L. S. Schulman, Knowledge and teaching: Foundations for a new reform, *Harvard Educational Review*, **57**(1), 1987, pp. 1–22.
9. M. J. Koehler, and P. Mishra, What happens when teachers design educational technology? The development of Technological Pedagogical Content Knowledge, *Journal of Educational Computing Research*, **32**(2), 2005, pp. 131–152.
10. M. J. Lutz, and D. Bagert, Software engineering curriculum development, *IEEE Software*, November/December 2006, pp. 16–18.
11. M. M. Moore, Software engineering education, *IEEE Software*, September/ October 2002, p. 103.
12. J. B. Thompson, and K. Reed, Undergraduate software engineering education: The mark of a discipline, *IEEE Software*, November/December 2005, pp. 96–97.
13. T. C. Lethbridge, R. J. LeBlanc Jr., A. E. K. Sobel, T. B. Hilburn, and J. L. Diaz-Herrera, SE2004: Recommendations for undergraduate software engineering curricula, *IEEE Software*, November/ December 2006, pp. 19–25.
14. H. van Vliet, Reflections on software engineering education, *IEEE Software*, May/June 2006, pp. 55–61.
15. O. Dieste, N. Juristo, and A. M. Moreno, How higher-education systems influence software engineering degree programs, *IEEE Software*, July/August 2004, pp. 78–85.
16. O. P. Brereton, S. Lees, R. Bedson, C. Boldyreff, S. Drummond, P. J. Layzell, and M. B. Blake, A student-enacted simulation approach to software engineering education, *IEEE Transactions on Education*, **46**(1), 2003, pp. 124–132.
17. L. S. Schulman, Those who understand: knowledge growth in teaching, *Educational Researchers*, **15**(2), 1986, p. 414.
18. L. S. Schulman, Knowledge and teaching: Foundations for a new reform, *Harvard Educational Reviews*, **57**(1), 1987, pp. 1–22.
19. M. J. Koehler, P. Mishra, K. Hershey, and L. Peruski, With a little from your students: A new model for faculty development and online course design, *Journal of Technology and Teacher Education*, **12**(1), 2004, pp. 25–55.
20. M. J. Koehler, P. Mishra, and K. Yahya, Tracing the development of teacher knowledge in a design seminar: Integrating content, pedagogy and technology, *Computers & Education*, 2005, pp. 1–23.
21. M. J. Koehler, and P. Mishra, What happens when teachers design educational technology? The development of technological pedagogical content knowledge, *Journal of Educational Computing Research*, **32**(2), 2005, pp. 131–152.
22. M. L. Niess, Preparing teachers to teach mathematics with technology: Developing a technology pedagogical content knowledge, *Teaching and Teacher Education*, **21**, 2005, pp. 509–523.
23. B. D. Mann, B. M. Eidelson, S. G. Fukuchi, S. A. Nissman, S. Robertson, and L. Jardines, The development of an interactive game-based tool for learning surgical management algorithms via computer, *The American Journal of Surgery*, **183**, pp. 305–308.
24. M. A. Roubidoux, C. M. Chapman, and M. E. Piontek, Development and evaluation of an interactive Web-based breast imaging game for medical students, *Academic Radiology*, **9**(10), 2002, pp. 1169–1178.
25. M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*, Harper Perennial, New York, (1991).
26. V. Venkatesh, and F. D. Davis, A model of the antecedents of perceived ease of use: Development and test, *Decision Sciences*, **27**(3), 1996, pp. 451–481.
27. C. Evans, N. Gibbons, J. K. Shah, and D. K. Griffin, Virtual learning in the biological sciences: Pitfalls of simply 'putting notes on the Web', *Computers & Education*, **43**, 2004, pp. 49–61.
28. S. C. Kong, and L. F. Kwok, An interactive teaching and learning environment for graph sketching, *Computers & Education*, **32**, 1999, pp. 1–17.

Wen-Hsiung Wu is a Professor of Information Management at National Kaohsiung University of Applied Sciences. He holds a BS in Electronic Engineering from the National Taiwan University of Science and Technology, an MS in Computer Engineering from University of Massachusetts-Lowell, and a Ph.D. in Information Management from National Sun Yat-Sen University. His primary research interests focus on instructional learning system development and assessment, behavior of information system learning, knowledge management, and e-business.

Wei-Fan Chen received his BS in Information and Computer Engineering in 1993, from Chung Yuan Christian University, Taiwan. He received his M.Ed. and Ph.D. in Instructional Systems in 1999 and 2002, respectively, both from The Pennsylvania State University, USA. He is currently an Assistant Professor of Information Sciences and Technology at The Pennsylvania State University. Dr. Chen's research and teaching interests include cognitive and information sciences and technology as related to learning.

APPENDIX A: QUALITATIVE OUTCOMES FROM A TEACHING AND DEVELOPMENT PERSPECTIVE

	C	P	T	CP	CT	PT	CPT
Successes	Illustrate an important content knowledge of project management for demo.	Apply role-play pedagogy for software engineering course.	Use game-based tools to develop the GBSEES system.	Show the content knowledge of project management via role-play pedagogy.	Present the content knowledge of project management via game-based design.	Design two major modes: single-player and multiple-player modes based on role-play approach.	Construct the system framework of GBSEES. Develop a prototype based on content knowledge of project management, role-play approach, and game-based design.
Difficulties	The content knowledge of software engineering curriculum is very comprehensive. The content knowledge has to consider an equilibrium between academia and practice.	Need to clearly describe the professional skill for every character in software project.	The integration of different factors via using game-based tools, such as aesthetic, script design, interactive technique, and effect. Need to spend a lot of time to learn the game-based tools for teachers.	The scenario description of role play in project management content is difficult.	Number of developmental team members, budget, and time for developing GBSEES are limited if we want to complete all content via game-based design.	The design of multiple-player's interaction is complex.	Number of developmental team members, budget, and time for developing GBSEES if we want to complete all modules.
Solutions	Discuss with an expert and then select a suitable content for both academia and practice.	Advice from an expert to understand every role of software project in the real world.	Match between selection of game types and game-based tools. Enhance the experience of using game-based tool via assistance of an expert.	Collaborate planning with an expert and capture real experience of every role's features in the software project	Select an important topic to design first, and then, add another topic.	Adopt an incremental design approach via adding one player each time.	Develop a prototype first and add another module further.

APPENDIX B: QUANTITATIVE OUTCOMES FROM A LEARNING PERSPECTIVE

Dimensions	Question no.	Questions	Mean	Standard deviation
Flow experience	1	Using *GBSEES* added to the fun for my learning.	4.09	0.32
	2	Using *GBSEES* kept me pleasure for my learning.	4.15	0.55
	3	Using *GBSEES* stimulated my curiosity.	4.24	0.49
	4	Using *GBSEES* aroused my imagination.	4.12	0.52
Interaction and overall use	5	I find that *GBSEES* allowed flexible interactions.	3.94	0.34
	6	I interacted with *GBSEES* in a clear and comprehensible manner.	3.94	0.39
	7	My interactions with *GBSEES* did not require much effort on my part.	3.91	0.49
	8	I find *GBSEES* easy to use.	3.71	0.67
	9	I find it easy to access the content knowledge I needed from *GBSEES*.	3.85	0.57
Usefulness	10	Using *GBSEES* gave me more incentive to learn.	4.12	0.57
	11	Using *GBSEES* improved my learning experience.	4.00	0.53
	12	Using *GBSEES* enhanced my knowledge and skills.	4.00	0.53
	13	Using *GBSEES* enhanced the effectiveness in learning.	3.91	0.72
	14	I find the *GBSEES* useful for learning the course.	4.06	0.55
Intention to Use	15	If I have access to *GBSEES*, I have the intention to use it.	4.32	0.48
	16	When I have access to *GBSEES*, I expect myself to make use of it.	4.29	0.50