An Agile Software Engineering Process Improvement Game

Bruce R. Maxim, Raspinder Kaur, Christopher Apzynski, David Edwards, Ethan Evans
University of Michigan-Dearborn
Department of Computer and Information Science
Dearborn, Michigan, USA
bmaxim, raspinde, capczyns, dkedward, ewevans @umich.edu

***Abstract*—Many computing students do not receive adequate training in software quality management. Some students do not have the opportunity to practice software process improvement activities even if they do see the topics covered in their course lectures and textbooks. Serious games are gaining popularity as a means of instruction in higher education. Some excellent prescriptive software process simulation games have been created, as well as a few software engineering drill and practice games. In general, these games do not allow students to create agile process models or experiment with process improvement strategies. We are creating a serious game that will serve as a virtual learning environment to allow students to explore agile process improvement practices. Our game is designed as a single-player game where the player takes the role of software team leader and plays against an AI (artificial intelligence) opponent representing the customer's interests and needs. Players are rewarded for developing project strategies that allow for completion of projects on time, within budget, and meet the necessary software quality requirements. It is our intention to create a game with sufficiently detailed instructions to allow instructors to introduce hands on practice with agile process improvement activities without requiring additional class time.**

Keywords—agile; scrum; process improvement; gaming

I. INTRODUCTION

Experiential learning has proven to be an effective means of teaching engineering topics. Serious games can provide an effective and enjoyable way to improve student learning in engineering courses [1]. Serious games are gaining popularity as a means of instruction in higher education [2]. Serious games make use of the artistic medium of games to deliver a message, teach a lesson, or provide an experience. Serious games may be entertaining, but that is not their primary purpose.

Computer games can teach hand-eye coordination, spatial relationships, and encourage exploratory experiences. Immersion in simulated environments has increased learning speed and retention for some tasks. Some students lose interest in classroom activities in as little as fifteen minutes, yet in some cases these same students may be engaged playing a computer game for two or more hours [3].

Computer games can provide opportunities for students to experience success in new areas. This often encourages them to try more things and attempt more difficult tasks. Trying to achieve the visible evidence of accomplishment or "leveling up" that appears on a player's in-game profile often is part of this motivation. Games can provide environments where failure is not catastrophic, but instead provides learning opportunities. Games that require critical thinking skills may induce creative participation in students' gameplay if they provide abundant problem solving opportunities. It may be the case that computer games make it easier to transfer in-game project experiences to real-life project experiences [4].

In one study, Taran used a computer game to teach the principles of risk analysis in project management course. Taran argues that practicing concepts in a game makes them less likely to be forgotten. Games provide a means for students to try out and experiment with strategies in safe environments. [5]. This type of game play can help teach players the kind of critical thinking that is important to STEM coursework [6].

Some excellent computer-based prescriptive software process simulation games such as SimSE [7] have been created. SimSE provides a very good immersive game experience and allows students to experience the role of project manager. SimSE players are rewarded for showing good judgement and the ability to manage development risks wisely. Students are encouraged to replay each simulation and attain a perfect score of 100 for their work.

A few drill and practice computer games have been created to help reinforce student knowledge of various parts of the software development process [8]. In general, these drill and practice programs focus on lower level skills (e.g. recall and recognition) and use entertainment to motivate game replay.

Most of the existing software engineering process games run on laptops or are implemented as web applications, rather than as mobile apps. There are also some non-computer games which allow students to experience elements of the software engineering [9]. At least one non-computer game focuses on teaching software process improvement, but does so using paper and scissors to complete a non-software production task [10]. These games often make use of the waterfall model as their process model.

This project was partially funded by the Association of Computing Machinery Special Interest Group on Computer Science Education .

There are several non-computer agile games and exercises such as the XP-Game that are used in agile training programs [11]. Most of these games are intended as team building exercises and require several participants and a great deal of class time to use. The time required to set up and manage these games can be substantial. Many of these games involve the creation of paper artifacts (e.g. pizzas) rather than virtual software products.

A critical analysis of several software engineering games appears in von Wangenheim and Shull [12]. According to von Wangenheim and Shull, software engineering games may be better at reinforcing knowledge learned previously than teaching new knowledge. They also found that software engineering process simulation games had more impact on student learning than other types of software engineering games. This was especially true when games allowed students to develop competence through discovery learning and learning by failure. It is important for games to provide instruction and guidance inside the game itself. It is important to provide students with sufficient feedback or explanation of their game scores and results, including instructor debriefing sessions. Subjective feedback from the students using these games indicates that they prefer game-based methods to traditional instruction for some topics.

Gamification is the use of game elements and game-like thinking in non-gaming environments. Gamification of coursework is often times used when creating a computer game is too costly or time-consuming. One of the difficulties encountered when using gamification in a training activity is inadequate technological support for visualizing the outcomes selected game mechanics or updating scoring quickly in real-life settings. Many of today's classroom management systems offer limited support for gamifying courses [13].

Experience points, scoreboards, and awards are the most frequently used game elements in course gamification. It is important to provide students with a goal and set of rules or game mechanisms that they may use to attain this goal. The infrastructure set up for the gamified activity can support elements of both competition and collaboration among the participants [14]. Creating a non-computer game can be very time-consuming and may require the cooperation of several participants simultaneously. This makes it hard for students to use the game to study by themselves.

We decided to create a serious computer game that allows students to manage virtual software projects using agile software engineering practices. Students will be able to experiment with various agile process improvement practices while managing a virtual development team. As a result of participating in the learning activities embodied in the game, we expect that students will discover the importance of using software process improvement activities while managing these virtual projects. We are working on a means of providing players with opportunities to create dynamic software process models in the game environment. We are making use of entertainment elements to encourage game replayability.

II. SYSTEM DESIGN

The team examined the relevant literature on games for educational purposes, gamification strategies, and agile software process improvement practices. The literature findings, along with our previous experiences in building serious games, influenced the design of our game and the player interactions. The team decided to create a single-player 2D card game with an artificially intelligent (AI) opponent. We choose to use SCRUM (an agile process framework) as the basis of our gameplay because it is popular agile project management strategy.

The reason we choose to create a card game rather than a 3D simulation with animated characters was to reduce the system requirements for players' mobile devices. We wanted to emphasize player decision making and focus on the quality of the game AI instead of using high resolution graphics. Multi-player gaming was rejected to allow students practice opportunities that were dependent only on individual player's available time.

The card game Mille Bornes [15] was selected by the design team as a model for game play. Mille Bornes is a multi-player game with the goal of completing a trip by playing distance cards until the trip is complete. Opponents can delay your progress by playing cards that halt progress (e.g. flat tire). When progress is interrupted the player needs to either find a remedy (e.g. deploy a spare tire) or have taken previous measures to anticipate that problem (e.g. own a puncture proof tire). This seemed like a good model for proactive risk management as we began to develop our software project management game.

The object of the Process Improvement Game (PIG) is to complete software artifact construction tasks on-time and within budget. Players will need to ensure that reasonable software quality practices are employed to ensure that the quality targets set for the software products are met. PIG will encourage players to adopt an agile approach to managing requirements and customer change requests by the design of its reward structure.

Fig. 1 contains a state transition diagram that summarizes the gameplay in the Process Improvement Game. The game states are connected by the activities typically included in a variation of the SCRUM framework used for project management [16]. The game will contain some random event behavior so that players will be encouraged to look for ways to vary their daily management decisions to maintain project velocity.

The final process improvement game prototype will contain a virtual tutorial software construction challenge (project) and two additional virtual software construction challenges. Players start a new game by selecting one of the three virtual projects to complete. The tutorial project must be completed to unlock access to the other two projects. All three project levels may be repeated as players seek to improve their scores.

The three construction challenges will be similar to one another to allow players to work on improving their software process management decisions. There will also be an element

of chance to encourage students to replay the game more than once.

Each software construction challenge will have its own backstory. This backstory provides the players with motivation for building artifacts. The backstory may also provide players with insight into selection of the tasks to be included in each new sprint. In agile software development, a sprint is a set period of time during which specific work has to be completed and made ready for customer review. The project story line may provide opportunities to inspire the player to perform process experiments in the virtual environment.

Each virtual project will require the completion of one or more sprints. The player selects the tasks to be completed in the current sprint from the product backlog list. At the beginning of each sprint, the player allocates his group of virtual developers into one of three roles (coder, tester, or debugger). Roles may be reassigned during one of the daily SCRUM meetings, if the player has the right card in hand.

At the daily SCRUM meeting, the player allocates the activities assigned to the team, based on the cards in his or her hand. The game AI analyzes dynamic playing behavior and watches for opportunities to exploit a weakness in the player's strategy. For example, the player writing code over several days without testing will trigger a defect card being played by the AI. Failing to communicate with the Product Owner may trigger a change request card played by the AI.

When the sprint is complete, the player receives a summary of the effectiveness of team during the time allocated for the sprint. Uncompleted tasks or rejected products are returned to the product backlog. If the project is completed the player statistics are updated and displayed on the in-game dashboards. If the project is not complete the player makes plans to begin the next sprint.

As players become more experienced in playing the game, defects, accidents, schedule slippage, and change requests are introduced into the game. These events are intended to help players to become attuned to watching for problems and planning for them. This also adds to the challenge and enjoyment of the game, as long as the interruptions are not seen as too frequent or too arbitrary.

In-game rewards provide powerful motivation for players to continue to play the game and replay levels. The rewards planned for our game consist of high score lists, virtual awards (badges), title changes, and status level changes. This is consistent with the way successful engineering teams are rewarded.

Players work for experience points as part of the reward system. The experience points will be based on completing the project on time and with-in budget while minimizing the defects in the final product. Players will be given bonus points for improving their performance and achieving the required quality targets each time they play or replay a level.

This game is being implemented using the Unity 3D game engine. The reason for using this game engine is the ease with which games can be exported to multiple platforms (the web, PC, Mac, and mobile – both IOS and Android). Our game scripting is being done using the MonoDevelop language. This makes it easy to interface with external software libraries as needed.

III. Evaluation

A. Usability

The planned evaluation of the game consists of a technical assessment of the game's usability, as well as an assessment of the student learning resulting from game play. Both assessments will be accomplished by having game players complete on-line forms.

As part of the usability assessment, each member of the development team will review the game by completing a usability checklist prior to allowing the release of the PIG prototype for external play. In addition, game players will be asked to complete an on-line usability questionnaire after they have completed each of the game projects for the first time.

B. Learning

Our plan is to evaluate this game with junior level software engineering students. We will introduce the game following their participation in classroom activities which expose them to the nature of prescriptive software process models and the project management elements of SCRUM. The game will be assigned as homework while students are studying software project management activities along with risk monitoring, mitigation and management practices in class.

Student learning of agile process improvement strategies will be assessed by having the students take a short online quiz after completing each of the three game projects (levels) for the first time. The quiz questions will check student understanding of the optimal strategy for managing their project resources and will be based on the rules used to develop the game AI. We may ask students to take each quiz a second time after replaying each game project to see if their understanding improves.

We made a conscious decision to not include the quiz as part of the game, since we want use the entertainment value of the game to inspire players to replay the game and not simply study for the quiz. We plan to ask the students to share their lessons learned from playing the game in a class discussion.

IV. CURRENT STATUS

The design team has created a paper prototype of the game and the user stories for the computer game. The development team created the game infrastructure and exported WebGL, Windows, and Android versions of the initial game. The first playable game prototype containing one level was completed during June 2016. This prototype was deployed on the web and will be play tested by software engineering students during July 2016. Player feedback regarding the game is being used to refine gameplay and create the two additional virtual software projects (including the tutorial level). A revised prototype containing all three projects will be available for public review in August 2016.

V. FUTURE PLANS

It is our hope that the project activities will result in the creation of a prototype for an extensible game that provides a virtual learning environment for teaching software engineering process improvement in agile development environments. We plan to add elements of other agile process models to allow students to experiment with broader process improvement strategies. We also plan to create additional software project challenges.

We plan to make this game and its sequels available as free to play web games hosted on the University of Michigan-Dearborn GAME Lab website. We also plan to make the app version of the games freely available as a download for Android devices.

ACKNOWLEDGMENT

Thanks to the students in the University Michigan-Dearborn GAME Lab for taking the time to play and critique our game.

REFERENCES

[1] Rajab, P.; Raju, P; and Sankar, C. (2003) "Serious Games to Improve Student Learning in Engineering Classes", Computers in Education Journal, Vol. 5 No. 2, pp.68-78

[2] Orszzullok, L. and Knautz, K. (2014) "Orc-based Learning – Evaluating a Game-Based Learning Approach", Proceedings of 2014 iConference, Berlin, German, pp.1009-1012.

[3] Michael, D. and Chen, S. (2006) *Serious Games: Games that Educate, Train, and Inform*, Thomson Course Technology, Indianapolis, IN.

[4] Hsiao, Hui-Chun. (2007) "A Brief Review of Digital Games and Learning," Proceedings of *the First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning, (DIGITEL '07)*, Jhongli City, March 2007, pp.124-129.

[5] Taran, G.. (2007) "Using Games in Software Engineering Education to Teach Risk Management," *Proceedings of 20th Conference on Software Engineering Education & Training*, (CSEET'07) July 2007, IEEE Press, pp.211-220.

[6] Tang, Y.; Shetty, S.; and Chen, X. (2010) "Empowering students with engineering literacy and problem-solving through interactive virtual reality games," Proceedings of 2010 IEEE Consumer Electronics Society's Games Innovations International Conference (ICE-GIC), December 2010, pp.1-6.

[7] Navarro, E. and v. d. Hoek, A. (2004) "SimSE: An Interactive Simulation Game for Software Engineering Education', Proceeding of the Seventh IASTED International Conference on Computers and Advanced Technology in Education, pp. 12-17.

[8] Rusu, A. Et.al. (2010) "Learning Software Engineering Basic Concepts Using a Five-Phase Game", Proceedings of 40th Annual Frontiers in Education Conference, Washington, DC: IEEE Press, pp. S2D1-S2D6.

[9] Baker, A.; Navarro, E.; and v. d. Hoek, A. (2005) "An Experimental Card Game for Teaching Software Engineering Processes", the Journal of Systems and Software, vol. 75, pp. 3-16.

[10] Ramingwong, S. (2012) "CutIT: A Game for Teaching Process Improvement in Software Engineering", Proceeding of the Third International Conference on Information, Communication and Education Application (ICEA 2012),

[11] Agile Games and Exercises List < http://www.agilesparks.com/agile-games-and-exercises-list> (accessed 04.22.16).

[12] von Wangenheim, C. and Shull, F. (2009) "To Game or Not to Game?", IEEE Software, Vol. 28 No. 2, pp.92-94.

[13] Dicheva, D., Dichev, C., Age, G.m and Angelova, G. (2015) "Gamification in Education: A Systematic Mapping Study", Proceedings of E-Learn 2015 World Conference on E-Learning, Kona, HI: ACM Press, pp.1-10.

[14] Sillaots, M. (2015) "Gamification of Higher Education by the Example of Course of Computer Games", Proceedings of eLmL 2015 Seventh International Conference on Mobile, Hybrid, and Online Learning, pp. 106-115.

[15] Mille Bornes <https://en.wikipedia.org/wiki/Mille_Bornes> (accessed 04.20.16).

[16] Pressman, R. S. and Maxim, B. R. (2014) Software Engineering: A Practitioner's Approach, 8th Ed., McGraw-Hill.

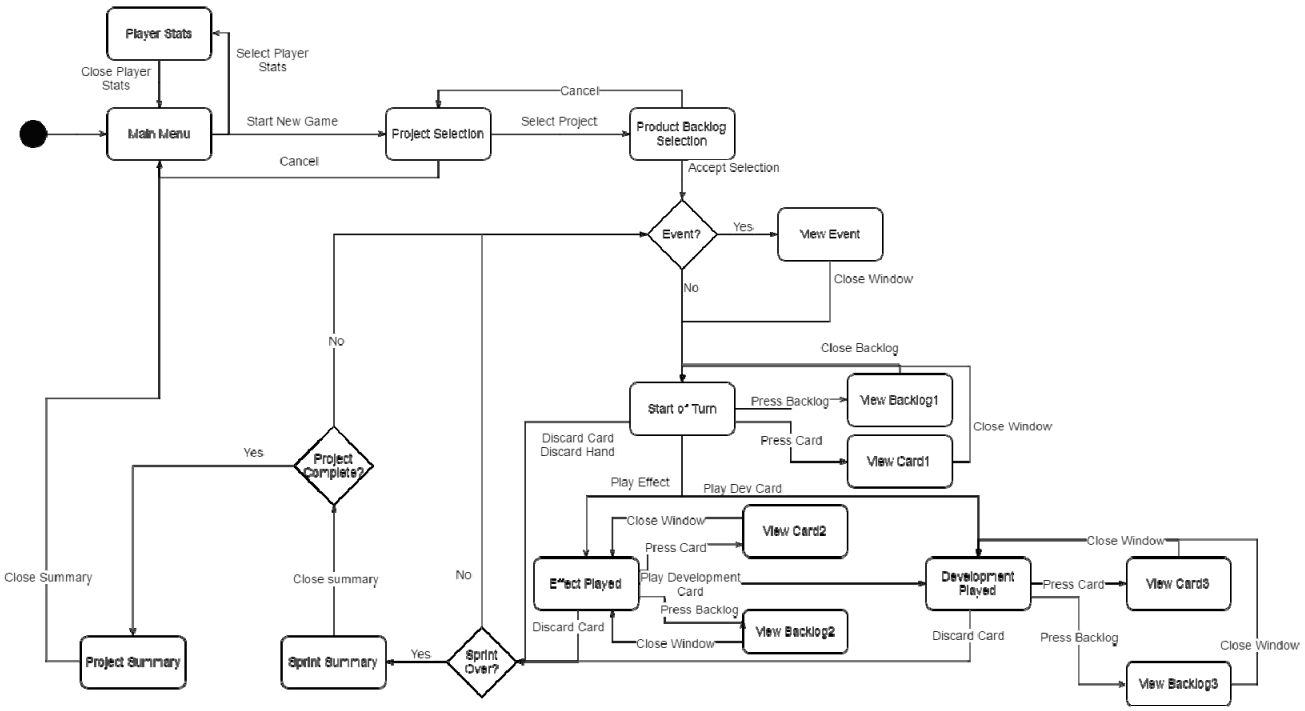

Fig. 1 State Transition Diagram Showing Game Flow