Creating game-like activities in agile software engineering education

David Parsons
Massey University
Auckland, New Zealand

***Abstract*—Game-like activities are often seen as valuable teaching tools, because they foster engagement and can also encourage teamwork and self-directed learning. The agile software engineering community was an early adopter of game-like activities in its mission to educate software developers about various aspects of agile methods. In addition to the potential benefits of participating in game-like learning activities, the creative and analytical process of designing and facilitating such activities can also be a powerful learning tool. In this article we describe how the design and testing of agile learning games has been used as a means to increase students' understanding of agile methods in a higher education context, as well as giving them an opportunity to develop their creative, organizational and analytical skills.**

Keywords—gamification; game-like activity; agile methods; creativity; learning

I. Introduction – Games and Game-Like Activities

Game-like activities are a well-established approach to teaching and learning in many domains, including software engineering. In recent years, the concept of 'gamification' (in its broadest sense) has spread across many aspects of software engineering, not just education. The definition of a game, and the process of gamification, is somewhat subjective, though there must be some set of rules, and goals, and players (individual or team), and possibly some physical or virtual artifacts (cards, dice, whiteboards etc.) to support the activity. Games are intended to be entertaining and are frequently competitive, but may also be cooperative team games. Some game-like activities might only include some of these characteristics, blurring the boundaries of what we might choose to call a game. The 'games' discussed in this article are really just game-like activities, with a serious purpose. Serious games are intended to teach some knowledge or skill, but hopefully remain entertaining in order to engage the participants.

II. Game-like activities for agile software developers

Game-like activities for software developers employing agile methods fall into three general categories. First, some game-like aspects have been introduced into the actual processes of agile software engineering. Their role is to take advantage of certain perceived benefits of gaming and apply them to some activities in the agile development process. Second, there are game-like activities based on creating software that are not targeted at developing a product but on developing software craftsmanship. Third, there are activities that are much closer to games, and do not include the creation of software, but are nonetheless intended to teach certain aspects of agile development. In the following sections, some examples of each type of activity are outlined.

A. Game-like activities in the software development process

Using game-like activities to assist the processes of software development is not a new idea. For example the Class Responsibility Collaboration exercise, which pre-dates agile methods but is often used in conjunction with them, has a number of game-like characteristics, including role play and physical interaction [1]. Another activity grounded in practical software development is the Planning Game, which is part of XP [2] and focuses on high level release planning. The goal is not to create accurate estimates but rather to determine the overall project scope. Approaching this task as a game-like activity is intended to take away some of the emotional stress of project planning. The 'players' are the business stakeholders and the developers, and the 'pieces' in the game are the story cards. There are three main 'moves' in the game; Write story, estimate story and make commitment. The game-like characteristics are primarily based on the terminology used (players, pieces and moves) rather than the actual activity.

Like the Planning Game, Planning Poker [3] is a way of using game-like activities to perform some of the tasks of agile planning. One significant difference is that in Planning Poker there are additional 'pieces', the 'cards' used to estimate stories. The aim of Planning Poker is to create estimates in a short time and involve the whole team. Planning Poker can be made more effective by the use of pre-printed cards. Not only does it save the time of manually writing the estimates, but these cards only have a subset of possible estimated days. The actual values on the cards vary from deck to deck. Cohn [4] suggests either 1, 2, 3, 5, and 8 (Fibonacci sequence) or 1, 2, 4, and 8 days, but there are various other options that can be used. Other game-like artifacts that can be used for this activity include poker chips [5]. Another variation is to use on an on-line version for distributed teams, making the activity something akin to an on-line role playing game [6].

B. Game-like activities for software craftsmanship

The game-like activities described above are incorporated into the normal business process of developing software. A second category of game-like activity is that of software craftsmanship, where code is written not as part of a production process but as a separate activity. To make this activity interesting and challenging, game-like features are often incorporated.

Software developers wishing to develop their craft often use Code Katas [7], short programming exercises that can be coded in many different ways to help the developer practice their skills. From these, Coding Dojos have developed, where groups of people work together on katas [8]. This has some features of a game-like activity in that a Coding Dojo is organized like a spectator sport, where members of a group observe and comment on a pair of developers working on a problem.

In a further development of these ideas, a coderetreat explores a single kata with the full involvement of all participants and within a game-like structure where there are various challenges, and time limits, and team 'play' [9].

C. Game-like actvities that teach about agile methods

The examples we have described so far are activities for software developers that are focused on the craft of software development but incorporate some game-like features. The third category we will investigate is game-like activities where no software is created at all. Rather, games are played that help to explain key features of agile software development. In these games, the players typically act as developers, stakeholders, testers or some other kind of role relevant to their tasks, thus they tend to be live action role playing games. In line with the usual structure of software development organizations or groups, the players are often put into teams rather than working alone. The rules are derived from the software processes being addressed in the game, whatever they may be. These games are often used as a *process miniature*, a way of understanding how a software development process works, condensed into a much shorter timescale [10]. In order to create a miniature version of a process in a game, activities other than real software development are usually used, such as drawing pictures or building things out of Lego^{TM}, so they can be performed in short time scales. In recent years, the number of games being used in the agile development community has grown enormously, to the extent that there is now a conference series dedicated to agile games [11]. In this section we provide a brief introduction to some of the many game-like activities that have been used to teach aspects of agile software development.

1) The XP Game

The XP Game [12] is a development of the Planning Game, and is primarily about story estimation and release planning. Unlike the Planning Game, it is intended to allow developers and business people to switch from their normal roles, participating in teams guided by a coach. In the game there are three phases. In the first phase ('estimation') all the team members play the role of developers. In the second ('make a plan') everyone is a customer. In the third phase ('implementation') everyone returns to the developer role.

The practical activities performed in the game have nothing to do with software development, but the tasks help to develop relevant skills in estimation. They include doing things with playing cards, balloons, mental arithmetic etc.

2) The Extreme Hour

Like the XP game, the Extreme Hour involves performing some activities in various roles, and estimating and prioritizing stories [13]. However instead of comparing many different tasks it focuses on a single project deliverable; building a better mousetrap. User stories are not pre-supplied but written by stakeholders, and deliverables are drawn by developers, in pairs, on a whiteboard. A tracker is responsible for copying the features as they are drawn, so when they are erased from the whiteboard there is a 'repository' of completed features. In addition to the developer and stakeholder roles, there is a quality assurance role for writing acceptance tests. 'Unit tests' are performed by developers, and are failed if they cannot understand what other developers have drawn. Overall, the approach of the Extreme Hour is to try to model more of the overall process of XP than the XP Game does, rather than focusing specifically on estimation and planning. Thus they can be seen to be complementary.

3) The Lego Games

Using Lego for agile games has been a well-developed theme, including 'XP: the LEGO brick road' [14] 'XP Lego Game' [15], 'Agile (Lego) Hour' [16] and the 'Lego lean game' [17], amongst others. In these games, Lego is used to construct vehicles, animals, buildings etc. within some kind of agile process miniature. The fact that the use of Lego has remained popular as these games have evolved suggests that it offers some compelling features for these kinds of activities. Lubke and Schneider [16] outline a number of reasons why they started to use Lego (instead of drawings) in their variation of the XP Hour, including the fact that bricks can more easily be reorganized and combined into modules than drawings, and can also better mimic the reuse of pre-existing components.

4) The Agile technique Hour

The Agile Technique Hour was designed to focus on how specific techniques integrate together in agile development. The main task in the activity is to design a human powered vehicle. Teams are allocated a set of user stories describing required features of such a vehicle. The overall design is created by overlaying features drawn on A4 transparencies, with each transparency being used to depict exactly one feature. Teams develop these features concurrently, and new user stories are introduced within each iteration. The teams consist of; stakeholders, developers and acceptance testers. The various techniques are introduced in a controlled way in three 20-minute iterations. At the end of the game the winning team is the one that has the most complete vehicle.

III. CREATING AN AGILE GAME AS A LEARNING ACTIVITY

The main issue addressed in this paper is how we might best utilize the concept of game-like activities in teaching agile software engineering. In an academic course we may find it difficult to provide opportunities for students to engage in game-like activities while working in real world software projects, but special events such as coderetreats or role playing games like those described here can be useful teaching aids. However, although playing agile games is a useful learning activity, higher level skills are better developed by creative actions [18]. Thus the class activity described in this paper was based on the idea that creating an agile game, rather than just playing one, is a challenging and insightful process that can benefit the learner in many ways. To exercise this idea a group of students learned about agile games as part of a post graduate course in agile software engineering, and experienced both the XP Game and the Agile Technique Hour. They were then asked to create and demonstrate their own agile games.

For this task, the students were asked to develop a game-like activity that could help to teach a group of 'players' about one aspect of agile methods. An important constraint in the task was that they were not to replicate the style of the games they had experienced, which covered many aspects of agile development, but instead they had to confine themselves to one specific aspect.

They were given the following definition of 'game-like':

"*To be 'game-like', an activity should be fun to do, include some level of competition (individual or team) have clear goals and some way of checking if those goals have been reached*"

They were required to create a suitable 'user manual' that would enable someone to run the activity. If any materials were required, they had to specify what these were (e.g. pack of cards, pencil and paper, Lego bricks etc.) They also had to provide these materials as part of their practical tests of the activities.

The task appeared to provide a high level of challenge to the creative thinking of the students. A good deal of formative assessment took place where the students would come to the class with their initial ideas for feedback. It challenged their self-reflection and critical thinking skills. They found it easy to think up game-like activities, but much harder to justify in what way these activities would help others learn about a specific aspect of agile methods. It also challenged their planning. A number of students came with ideas that would be extremely difficult to implement in practice. They had to be frequently reminded that the proposed game would have to be tested in class, and that this was not just a theoretical exercise but had the goal of creating a viable product that could be used by others. Thus it seems that the exercise as whole addressed skills essential to agile practice; planning, testing, iterative development, meeting stakeholder requirements and 'doing the simplest thing that could possibly work'

The following four examples give some impression of the range of the more successful games that were developed by members of the class. 'Successful' in this context means that the games turned out to be realistic to deliver in class, met the requirements of focusing on a single aspect of agile development, and gained positive feedback from participants. These four games each aimed to teach a single aspect of agile methods; pair programming, standup meetings, team strategy and refactoring.

A. Assembling Pens

The aim of this game is to explore pair programming by setting a task and experiencing the pairing roles of *driver* and *navigator*. The materials are a set of 10 different ball point pens, of the type that can easily be dismantled and re-assembled. The 'test' is whether the pens have been successfully assembled, and if so, whether they actually work, for example a pen may have been assembled with the wrong spring and not retract properly. The exploration of pair programing is in manipulating the relative roles of driver and navigator, and applying additional rules such as silent pairing. A variation for the game is whether or not to provide a photo of the assembled pens. This might be regarded as an explicit statement of requirements. However, pairing is claimed to have more value on tasks that are not well understood at the beginning [19]. Thus providing a photo is a fall back if the task proves too difficult.

B. Scrumhancer

This game explores stand up meetings as practiced in Scrum. In the game, participants work in teams of three developers plus one Scrum Master. Each developer is given a specific task to do. These tasks involve solving problems such as crosswords, Sudoku puzzles or Boggle word searches. Each team member works for two minutes on their individual task, and then a standup meeting is convened. In a variation on the usual Scrum feedback, each developer reports on; what puzzle they are working on and how much of it they have they solved so far, how much more of the same puzzle they expect to solve in the next cycle, and what difficulties they are facing. In the standup meeting, the team members must come up with a strategy for the next cycle. They can choose to continue with their previous puzzle, or two or three developers can combine their efforts in solving one puzzle, or they can choose to exchange puzzles amongst themselves in order to solve them all by the end of the third cycle. The Scrum Master must ensure that the chosen strategy is being implemented by their team in the subsequent cycle. An observer assesses each team on criteria such as whether meetings were correctly time constrained, focused and constructive. The observer's final scores over three iterations are used to decide the winning team.

C. Agile Poker

This game is about the value of working in pairs and teams. Its purpose is to demonstrate how working with others can produce better decisions than doing things individually, and how team meetings can improve focus and problem solving. The game is based around packs of standard playing cards. One deck is used by each 'team' of four developers, a further deck by the game controller. The task for the team is to make a poker hand by adding 4 cards (1 card each) to the first card drawn by the game controller (each of the four team members will hold 13 cards from their deck). The 5 cards together make a poker hand which is scored increasingly in the order: pair < two pairs < straight < full house < four of a kind < straight flush. Note that not all possible poker hands are valid in the game. The game takes 3 iterations to complete. In the first iteration the team members work individually. They have to think about what cards their team members might play, but have no way of knowing. In the second iteration they work in silent pairs. Each pair can share knowledge of their cards with their partner, but without communicating verbally. Before the final iteration they are allow to speak together in a team meeting to work out a group strategy. Although they cannot swap cards, the ability to plan their strategies with others, first in pairs, and then as a team, should help them appreciate the value of collaboration in problem solving.

D. An Agile Story

This game addresses the issue of incorporating a set of requirements into an iteration, and dealing with changing requirements in subsequent iterations by refactoring. It is based on teams of developers writing stories from a set of requirements based on supplied characters, actions, and locations. In the first iteration, each team must write a story based around two randomly chosen character cards, two randomly chosen action cards and a randomly chosen location. For example they may be given the characters of 'fisherman' and 'lawyer', the actions of 'buying shoes' and 'throwing a Frisbee' at the location of a 'market'. In each subsequent iteration, an additional character, action and location are added to the requirements. This means that the story must be refactored to take account of the new requirements. At the end, the 'customer' judges the best story. The main challenge in the game is to adapt to changing requirements by refactoring the story in each iteration so that its overall design quality is maintained.

IV. Summary and Conclusions

Table I provides a brief summary of some of the main features of the four games described in the previous section. In each case the learning focus and materials are indicated and, important for a game-like activity, the goals and means of checking them are shown. Perhaps the main point to be taken from this table is that creating an agile game as an assessment activity provides a very broad canvas on which the students can work, enabling them to exercise their creative thinking skills as well as their analytical skills.

TABLE I. Summary of 4 Agile Games

Agile Game	**Game Features**			
	Learning focus	***Material***	***Goals***	***Goal checking***
Assembling Pens	Pair programming	Pens	Cooperate to assemble components	Functional testing
Scrumhancer	Standup meetings	Puzzles	Gain value from meetings	Observer scorecard
Agile Poker	Team strategy	Playing cards	Learn to develop team strategy	Poker scoring
Agile Story	Refactoring	'Story' cards *	Maintain quality while embracing change	'Customer' as judge

* Not the same as the 'story cards' often used in agile development

The activities described in this paper were originally designed for teaching a class, not as a research activity. The aim of this paper is therefore not to report on the outcomes of a research project but to describe some experiences with an approach to teaching aspects of agile methods that also addresses higher level skills. The value of the activity seemed to be at several levels. First, it required the students to apply their analytical skills in identifying the core concepts of one or more aspects of agile software engineering. Second, it enabled them to exercise their creative thinking skills in designing a game-like activity. Third, it required them to apply their organizational skills in running their own games. Further, the students also gained from participating in the tests of the activities, enabling them to reflect on their own game designs and engage in peer review.

Acknowledgment

The author thanks the Agile Software Engineering students whose work is referenced in this article; Gautam Atmakuri, Mridu Gupta, Dylan McLeod and Agus Leonardi Sugianto.

References

[1] K. Beck and W. Cunningham, "A laboratory for teaching object-oriented thinking". SIGPLAN Notices Volume 24, Number 10, October 1989

[2] K. Beck, Extreme Programming Explained: Embracing Change. 1st edition. Boston: Addison-Wesley, 1999.

[3] J. Grenning, Planning Poker or How to avoid analysis paralysis while release planning, [Online]. Available: http://sewiki.iai.uni-bonn.de/_media/teaching/labs/xp/2005a/doc.planningpoker-v1.pdf 2002.

[4] M. Cohn, Agile estimating and planning Addison-Wesley, 2005A. Cockburn, Agile Software Development. Addison Wesley, 2002.

[5] J. Yip. "Hands-on release planning with poker chips". In 14th Conference on Pattern Languages of Programs (PLOP '07). 2007.

[6] Mountain Goat Software, Play. Estimate. Plan. [Online]. Available: http://www.planningpoker.com/ 2013.

[7] Thomas, D. Code kata. [Online]. Available: http://codekata.pragprog.com/2007/01/code_kata_backg.html#more 2013

[8] Emrich, M. (2013). Behaviour Driven Development with JavaScript. Developer Press.

[9] CodeRetreat Community Network. [Online]. Available: http://coderetreat.org/ 2013

[10] A. Cockburn, Agile Software Development. Addison Wesley, 2002.

[11] Agile Games 2012, [Online]. Available: http://www.agilegames2012.com/index.php

[12] V. Peeters and P. Van Cauwenberghe, The XP Game. [Online]. Available: http://www.xp.be/xpgame.html, 2006.

[13] Extreme Hour Wiki. http://c2.com/xp/ExtremeHour.html. 2005

[14] T. Mackinnon, O. Bye and P. Simmons, Extreme Programming: the LEGO brick road, [Online]. Available: http://www.spaconference.org/ot2000/programme/122_Mackinnon_Tim.htm 2000

[15] S. Newman, D. North, amd M. Hill, The Lego XP Game, [Online]. Available: http://www.magpiebrain.com/wp-content/uploads/2006/07/lego_xp_game_submitted.ppt, 2005

[16] D. Lübke and K. Schneider, "Agile Hour: Teaching XP skills to students and IT professionals in product focused software process improvement". Lecture Notes in Computer Science, Volume 3547/2005, 517-529, 2005.

[17] D. Sato and F. Trindade, "The Lego Lean Game" in Agile Processes in Software Engineering and Extreme Programming, Lecture Notes in Business Information Processing, Volume 31, Part 4, 192-193, 2009.

[18] L. Anderson and D. Krathwohl (eds.) A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. New York: Longman, 2001.

[19] K. Lui, "Pair programming productivity: Novice-novice vs. expert-expert". International Journal of Human-Computer Studies 64 (9), 2006