Shall We Play a Game?

Craig Caulfield (Corresponding author)
School of Computer Science and Security Science, Edith Cowan University
2 Bradford Street, Mount Lawley, Western Australia 6050, Australia
Tel: 61-8-9370-6295 E-mail: ccaulfie@our.ecu.edu.au

S P Maj
School of Computer Science and Security Science, Edith Cowan University
2 Bradford Street, Mount Lawley, Western Australia 6050, Australia
Tel: 61-8-9370-6277 E-mail: p.maj@ecu.edu.au

Jianhong (Cecilia) Xia
Department of Spatial Sciences, Curtin University
Kent Street, Bentley, Western Australia 6102, Australia
Tel: 61-8-9266-7563 E-mail: c.xia@curtin.edu.au

D Veal
School of Computer Science and Security Science, Edith Cowan University
2 Bradford Street, Mount Lawley, Western Australia 6050, Australia
Tel: 61-8-9370-6295 E-mail: d.veal@ecu.edu.au

Received: October 8, 2011 Accepted: October 19, 2011 Published: January 1, 2012
doi:10.5539/mas.v6n1p2 URL: http://dx.doi.org/10.5539/mas.v6n1p2

Abstract

This paper presents the results of a qualitative research project that used a simple game of a software project to see if and how games could contribute to better software project management education, and, if so, what features would make them most efficacious. The results suggest that while games are useful pedagogical tools and are well-received by players, they are not sufficient in themselves and must be supplemented by other learning devices.

Keywords: Software engineering, Project management education, Serious games

1. Introduction

In the 1983 movie, *War Games*, a young Matthew Broderick plays David Lightman, a hacker who has broken into WOPR– the War Operation Plan Response supercomputer which is programmed to play out different doomsday scenarios and learn from them so it can eventually take full, automated control of the United State's nuclear arsenal. When David is presented with a screen prompt that asks, "Shall we play a game?", he innocently selects "Global Thermonuclear War". As quickly becomes apparent, WOPR is ready to do more than just play games and it starts executing commands in readiness for a real missile strike against the Soviet Union.

The portentous question asked by WOPR– shall we play a game?– has meaning for the research project discussed in this paper too, but without the same dire consequences. This paper reports on a qualitative research project designed to see if and how games could contribute to better software project management education by helping software engineers and project managers explore some of the dynamic complexities of the field in a safe and inexpensive environment. If games could indeed contribute, then what features made them most efficacious? Games have been used to good effect in other similarly dynamic areas and the researchers believed that an opportunity existed to see what contribution they could make to a better software project education. In effect:

shall we– should we– play games in software project management education?

2. Literature Review and Background

2.1 Defining Games

To play a game, "is to engage in activity directed towards bringing about a specific state of affairs, using only means permitted by specific rules, where the means permitted by the rules are more limited in scope than they would be in the absence of the rules and where the sole reason for accepting such limitation is to make possible such activity" (Suits, 1967, p.156).

A game is different from a model or simulation. For example, a model is "a miniature representation of a complex reality. A model reflects certain selected characteristics of the system it stands for. A model is useful to the extent that it portrays accurately those characteristics that happen to be of interest at the moment" (DeMarco, 1982, p.41). Meanwhile, a simulation is a special kind of model that shows how a system, such as a biological, social, physical, or economic system, changes over time (Miller, 1978, p.83).

The games that this research project deals with are a subset of Caillois's (1961) *agôn* classification, those games "that would seem to be competitive like a combat in which equality of chances is artificially created in order that the adversaries should confront each other under ideal conditions" (Caillois, 1961, p.14), and they use an adjective— serious— to show they are not for simple amusement and they are designed to educate, train, or inform their players (Abt, 1970; Michael & Chen, 2005; Schrage & Peters, 1999).

2.2 The Nature of Software

Software development is an inherently complex endeavour because of both the ephemeral qualities of software itself and the dynamic socio-technical system in which it is developed. For example, software has no fundamental theory (Osterweil, 1987, p.3), like the law of physics, with which we can reason about its behaviour. This makes it difficult to thoroughly test software without actually building it and running it in a live environment (Kruchten, 2005) with all the attendant risks this involves.

Software must also conform to the arbitrary design of the human institutions and processes in which it is deployed and accept change because in a system of software, hardware, and humans, it is the most malleable (Brooks, 1987, p.12). These are naturally properties that organisations want to take advantage of, but constant change, if not managed, can erode the integrity of the original design, and when combined with relatively low manufacturing costs, can lead to shortcuts:

Program implementation is more like preparing a cast in mechanical engineering. The real "manufacturing" of software entails almost no cost; a CD-ROM, for example, costs less than a dollar, and delivery over the Internet only a few cents. Often it doesn't matter if the design is a bit wrong; we can just fix it and manufacture it again. You can't do that with a bridge or a car engine because the cost would be huge, and that forces engineers involved in building these things to get them right the first time (Kruchten, 2004).

Because software is complex, difficult to reason about and test, and yet cheap and easy to change, it is perhaps understandable that many implementations are not right the first time, if at all. These qualities of software are often the root cause of many quality and productivity issues:

From the complexity comes the difficulty of communication among team means, which leads to product flaws, cost overruns, schedule delays. From the complexity comes the difficulty of enumerating, much less understanding, all the possible states of the program, and from that comes the unreliability. From the complexity of the functions comes the difficulty of invoking those functions, which makes programs hard to use. From complexity of structure comes the difficulty of extending programs to new functions without creating side effects. From complexity of structure comes the unvisualized states that constitute security trapdoors (Brooks, 1987, p.11).

How then do we prepare software engineers to work in an environment that is complex in and of it itself, and which is, in turn, used to create a complex product? To answer this we need to look at the state of current professional practice and the educational programs that produce new software engineers.

2.3 The Education of Software Engineers

In response to some of the productivity and quality problems in the field, steps are being taken to make software engineering more reliable, more predictable, more like its engineering namesake:

A body of knowledge, the SWEBOK (Bourque, Dupuis, Abran, Moore, & Tripp, 1999), has been defined which captures accepted practice in the field and which also forms the basis of curriculum development and

accreditation, licensing and certification programs.

Standards of ethics and conduct have been developed to guide software engineers in responsible behaviour, although these are still optional and unenforceable (Gotterbarn, 1999; McConnell, 2004, p.57).

Professionally-endorsed curriculum guidelines for graduate and post-graduate software engineering education have been developed to meet the latest technical developments and evolving industry demands. These include Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (Joint Task Force on Computing Curriculum, 2004), Curriculum Guidelines for Graduate Degree Programs in Software Engineering (iSSEc Project, 2009), and Curriculum Guidelines for Undergraduate Degree Programs in Information Systems (Joint IS2010 Curriculum Task Force, 2010).

Of interest for this research project was the way software engineers are educated because this directly and significantly affects so many other areas of professional practice. In the curriculum guidelines considered mentioned above, each identifies better software project management skills and better soft, or peopleware (DeMarco & Lister, 1999), skills as critical for all graduating students, but the guidelines are intentionally vague on how institutions should teach these.

This pedagogical gap is exposed most often when students finish their requisite courses and attempt their final, important, and synthesising capstone project. While there are many cases of capstone projects bringing great benefits for the students and their clients (Boehm, et al., 1998; Johns-Boast & Patch, 2010), these are balanced by stories of significant failures in which student/client relationships broke down, there was severe internal team dissension, or the final software was unusable (Brereton, et al., 2000; Cheng & Lin, 2010; Polack-Wahl, 2006). For those capstone projects that failed, there was little opportunity for reflection or remedial action because the project was the final unit of study for the course. Some research has been conducted that recommends guidelines for successful capstone projects (Robillard & Robillard, 1998), such as providing students with basic training in project control, reviewing the design documents, and having an experienced software engineer mentor certain stages, but these are relatively costly or time-consuming course attributes and there is little evidence they have been widely adopted.

2.4 One Possible Solution

One possible way to tackle these problems is to use a serious game— a game designed to teach and educate players about some of the dynamic complexities of the field in a safe and inexpensive environment. Importantly, games are not one-shot opportunities in the way capstone projects are: a game can be played, studied, tweaked, and replayed as many times as needed.

Games have been used in many different business (Michael & Chen, 2005; Schrage & Peters, 1999), military (Perla, 1990; Riddell, 1997; Zyda, 2007), and social environments (Gee, 2007; Prensky, 2006; Salen & Zimmerman, 2005), and have proven to be efficacious. They also draw their intellectual integrity from sound education theory such as problem-based learning. Problem-based learning is a pedagogic method that uses problem scenarios to encourage students to work out solutions for themselves (Barell, 2006; Barrows & Tamblyn, 1976; Savin-Baden, 2003; Savin-Baden & Major, 2004). Usually working in small teams, students explore the problem, bring their personal experience to bear, identify any gaps in their knowledge, and eventually come up with viable solutions. The problems themselves are usually complex, ill-defined, real-world situations for which there may not necessarily be a single right or wrong solution (Maxwell, Mergendoller, & Bellisimo, 2004, p.2). The students build new knowledge through self-directed learning while their tutors act as facilitators or consultants rather than more traditional instructors (Dempsey, Haynes, Lucassen, & Casey, 2002, p.5; McCall, 2011). Games therefore have an authority to be taken seriously as learning and research tools.

This research project is not the first to look at games in software engineering education. A systematic survey of the field (Caulfield, Xia, Veal, & Maj, 2011) discovered over two dozen research projects using mostly single-user computer games to teach various aspects of the software development lifecycle. However, few of these games were developed or repeated beyond their initial implementations, which suggests that their design lacked some essential feature.

An opportunity therefore existed to explore more fully if and how games could contribute to better software engineering management and help fill some of the pedagogical gaps in the current curriculum guidelines; and if they could, then what design features were most valuable.

3. Methodology

3.1 Introducing Simsoft

The primary research tool for this project was a game called Simsoft (Caulfield, Veal, & Maj, 2011a). Physically, Simsoft comes in two pieces. There is an A0-sized printed game board around which the players gather to discuss the current state of their project and to consider their next move. The board shows the flow of the game while plastic counters are used to represent the staff of the project. Poker chips represent the team's budget, with which they can purchase more staff, and from which certain game events may draw or reimburse amounts depending on decisions made during the course of the game. There is also a simple Java-based dashboard (Caulfield, Veal, & Maj, 2011b), through which the players can see the current and historical state of the project in a series of reports and messages; and they can adjust the project's settings. The engine behind Simsoft is a system dynamics model which embodies the fundamental causal relationships of simple software development projects.

3.2 Game Sessions

Simsoft game sessions were held between May and September 2010 in which teams of students, and practicing project managers and software engineers managed a hypothetical software development project with the aim of completing the project on time and within budget (with poker chips left over). Based on the starting scenario of the game, information provided during the game, and their own real-world experience, the players made decisions about how to proceed— whether to hire more staff or reduce the number, what hours should be worked, and so on. After each decision set had been entered, the game was run for another next time period, (a week, a month, or a quarter). The game was now in a new state which the players had to interpret from the game board and decide how to proceed.

3.3 Participants

Purposive sampling (Lincoln & Guba, 1984; Patton, 2002) was used to select the participants (n=59) of the study from the following pools:

Post-graduate project management students from two Perth, Western Australia Universities.

Software engineers, project managers, and account managers from a Perth-based software consulting company.

A call for participation was distributed by email and the participants replied if they wished to take part. Snowball sampling (Marshall, 1996) was allowed, whereby those reading the email were encouraged to refer others in the same field they thought would be interested in taking part.

Although the participants each had an information technology or project management background, they exhibited notable variances in experience (from recent graduates to 25-year industry veterans); skills (from those still studying to highly-certified professionals); and cultural diversity (the participants came from Australia, Europe, the Middle East, Asia, and South Africa).

3.4 Data Collection

Before the game session, the players completed an online survey designed to test their knowledge of general software engineering and project management principles. The survey questions were based on those in examination preparation guides for the IEEE's Certified Software Development professional certification (Naveda & Seidman, 2006) and the Project Management Professional certification (Heldman, 2007).

After the game session, the players completed another online survey. Post-game surveys are a common feature of game research (Eldredge & Watson, 1996; Faria, 1987, 1998; Faria & Wellington, 2004; Faria & Wellington, 2005; McKenna, 1991) and in problem-based learning (Tang, et al., 1997), the key foundation of Simsoft's design. Based on these exemplars, a survey was designed to capture their experience of playing the game, whether they found it useful, how it might compare to other forms of instruction such as lectures or case studies, and what may have been learned through the game.

Therefore, this research project had multiple data sources: the Simsoft game database, the pre- and post-game surveys, interviews with the players, researcher memos (Maxwell, 2004), and field notes.

3.5 Data Analysis

The analysis and interpretation of the data for this project followed a path used many times before in qualitative research: collect the data, analyse it for themes or perspectives, and then report on four or five of those themes (Bloomberg & Volpe, 2008; Creswell, 2009; Lincoln & Guba, 1984). In more detail, the following steps were taken:

Organized and prepared the data for analysis by transcribing the interviews, and writing up the field notes and memos.

Re-read, examined, and explored the data to get a high-level sense of what had been collected.

Started to analyse the data by first coding it— breaking it into named chunks or categories that can then be used to make comparisons between things in the same category and to help develop theoretical concepts (Rossman & Rallis, 1998; Strauss, 1987). The software package NVivo (http://www.qsrinternational.com/) was used for this task (Richards, 2009).

Used the coding to identify and describe themes and patterns in the data, which then became candidates for more detailed analysis and, potentially, major findings of the project (Maxwell, 2004).

Formulated the finding statements and supported these will specific data instances and then summarized the key findings.

Sought meaning in the findings by linking it to experience, insight, or the literature. The most commonly asked question was: "What were the lessons learned?" (Lincoln & Guba, 1984).

The above step-by-step list might give the impression that the analysis and interpretation of the data proceeded in a linear fashion once all the data had been collected. In reality, the process was highly iterative and started as soon as the first data was available— a feature common to this type of research (Lincoln & Guba, 1984, pp.241 -242).

3.6 Reliability, Validity, and Applicability of the Findings

Compared to objective, deductive quantitative research, qualitative research has often been called undisciplined, sloppy, merely subjective, and indiscriminately responsive to the loudest bangs and brightest lights (Lincoln & Guba, 1984, p.289). Add to this a researcher intimately involved in the data collection and carrying certain biases, and it is natural to question the trustworthiness of the results: "The basic issue in relation to trustworthiness is simple: How can an inquirer persuade his or her audiences (including self) that the findings of an inquiry are worth paying attention to, worth taking account of? What arguments can be mounted, what criteria invoked, what questions asked, that would be persuasive on this issue?" (Guba & Lincoln, 2005, p.290). That is, how can we demonstrate the findings are reliable, valid, and applicable?

For this project the following means were used:

Reliability, or "the extent to which a measurement procedure yields the same answer however and whenever it is carried out" (Kirk & Miller, 1986, p.19): the coding scheme peer reviewed by an independent party.

Validity, or the "degree to which the finding is interpreted in a correct way" (Kirk & Miller, 1986, p.20): more than data source was used so that the results could be triangulated; and the findings were reviewed by a random sample of four players from the games sessions.

Applicability, or "the extent to which the findings of a particular enquiry have applicability in other contexts or with other subjects" (Guba & Lincoln, 2005, p.290): rich, detailed descriptions are provided that will allow subsequent researchers to determine if the findings are relevant to their particular setting.

4. Findings

Six major findings emerged from the research.

4.1 Finding 1— There was evidence the participants were learning by doing.

A key tenet of problem-based based learning (Savin-Baden & Major, 2004), one of the theoretical foundations of Simsoft, is that when people work through problems for themselves, the knowledge they build 'sticks' and they are more able to apply what they have learned in new situations. The following comments indicate that playing Simsoft indeed helped the participants figure things out for themselves:

"Aha!"

"Our team figured out we could move more counters [work units] by investing in a couple of expensive, experienced developers, more middies, and some quality control people. Makes sense really"

"We spent our poker chips on lots of cheap newbies and before long had most of our counters [work units] in rework. We should have bought some old timers for guidance"

"Now I see why"

"I hadn't appreciated the level of productivity variability between developers before"

In addition, all participants completed pre- and post-game surveys that included a number of questions designed to test their general level of knowledge about project management and software engineering concepts. Table 1 shows the results of the tests broken by the participants' role and years of experience, and shows each group performed better after playing the game. Two non-parametric statistical tests were run over the pre- and post-game results to determine if this improved performance was significant.

A Mann-Whitney *U* test (Z = -1.091, p = 0.275 > 0.05) indicated that there was no significant differences between the pre- and post-game results when considering the broad groups of project managers, software developers, and students. A Wilcoxon signed ranks test (Z = -1.604, p = 0.109 > 0.05) also showed there was no significant difference between the pre- and post-game results of the three groups.

The same statistical tests were then run at a finer level of detail: against the years-of-experience sub-groups with the three main groupings of project managers, software developers, and students. Both the Mann-Whitney *U* test (Z = -2.951, p = 0.003 < 0.05) and the Wilcoxon signed ranks test (Z = -2.552, p = 0.011 < 0.05) showed there was a significant improvement between the pre- and post-game tests.

Together, these results indicate that while playing the game helped, none of the three main groups performed significantly better than the others. However, the years of experience a person has may affect how much they take from the game.

4.2 Finding 2— Games such as Simsoft are not sufficient learning vehicles by themselves and need to be supplemented by other methods.

While most players (40 out of 59) said that Simsoft helped put project management and software engineering theories into a practical context, the mean score was 2.64 out 5 (SD = 0.760) when they were asked if games were a better way of learning and understanding technical material than through more conventional methods such as books, lectures, case studies.

From an experienced software developer:

"I saw in the game aspects of theory covered at uni, but without knowing the theory first I probably wouldn't have recognised the significance."

And these comments from two students:

"I was out of my depth"

"I could see the logic behind my team's decision, but I wouldn't have known enough to make the decision by myself."

One project manager expressed an interest in using Simsoft as part of an under-graduate computer science he teaches part-time, but:

"It would have to be used on the final weeks of the course when the students have some theory under their belt. Plus, there is little momentum behind problem-based learning at [my university] so the resources aren't available to design a proper PBL based curriculum"

Table 1 also shows that the greatest improvement between the pre- and post-game tests was in those groups with the greatest work experience, so that relatively inexperienced participants took less from the game. This suggests that some level of *a priori* knowledge is needed for games like Simsoft to be truly effective.

However, when asked if games were a better way of more thoroughly learning a topic than through more conventional methods such as books, lectures, case studies, a significant minority (21 out of 59 participants) agreed or strongly agreed (mean = 3.00 out of 5, SD = 0.964). Self discovery seems to be the motive:

"I like to figure things out for myself"

On six occasions over the seven game sessions, the researcher overheard players saying they wished they could set Simsoft to match their work environment so they could game through some current issues.

4.3 Finding 3— Simsoft is a suitable pedagogical device for participants of different skills and backgrounds.

When asked if the game was easy or hard to play (1 = too easy, 3 = about right, and 5 too hard), the majority of the participants (47 out of 59) felt that the game was a pitched at about the right level of difficulty (see Table 2).

This comment was from a student:

"Even though I'm still studying and don't have much [practical work] experience, I was able to understand the game's project and contribute to the decisions"

And, from a project manager with 10 to 15 years experience:

"[The] game was not too easy so that it was boring, but not too hard that newbies couldn't undetstad (sic) it."

Across the seven game sessions there were no teams composed entirely of one group only, so each had a mixture of skills and experience. This was viewed positively:

"Our team had a mixture of abilities and life experience. I think this helped us make good choices"

"[One of our team] had read about Brooks' model and could let us know if we were on the right track"

4.4 Finding 4— The majority (49 out of 59) of participants said they would be prepared to invest greater time and effort in games such as Simsoft if the reward was deeper understanding of a problem domain.

Many players said they reached the end of the game before they had time to fully explore the dynamics of the scenario, or they wanted to take more discussing their options before committing to a decision. For example:

"The game was too short to discover what I wanted to know"

"I wanted to know more"

"We wanted more time to talk about our options"

The database of Simsoft game transactions showed that games lasted an average of 35 minutes (SD = 7.082) and that 80% of games finished within 40 minutes. The players were encouraged to stay behind after the game sessions to discuss and compare their results with other teams. Often, these after-game sessions lasted longer than the games themselves.

Considering the amount of time they had spent playing Simsoft, a majority of the players (49 out of 59) said would be prepared to invest greater time and effort in games like Simsoft if the reward was greater understanding of the problem domain:

"What about running the game in real time, like the stock market game. That would give us time to make really considered judgements, people could be assigned research topics during the week"

"I hope that future versions will let me set up specific scenario and play them out. That would really help me in my work"

Outside of this research project, 10 players had previously participated in a long-running online stock market game in which notional shares were bought and sold based on actual prices published in a daily newspaper. Buy and sell decisions were submitted weekly and the team with the largest portfolio after three months was declared the winner.

4.5 Finding 5— The majority (44 out of 59) of the participants found that working in groups was a positive experience

An important component of many of the pedagogical theories behind Simsoft is the aspect of working in groups or teams, so it was important to assess how this was received by the players. A majority of players (44 out of 59) said they found it useful or very useful to work as a team and that this reflected how things often happened in the workplace:

"It was like [the agile] stand up meeting we have every morning"

"We organised ourselves into roles we felt comfortable with or that fitted our day-job: someone on the calculator, someone moving the developer pieces, someone moving the units of work"

However, one student found something new in the practice:

"I thought software development was a solitary experience but it's not really"

Others liked the opportunity to share opinions and learn from more experienced peers:

"Everyone had a chance to offer an opinion"

"I have little real-world project experience so it was good to get the advice of others and see how they approached problems"

But, as in any group activity, the game facilitator needs to be aware of cultural differences that may make some less inclined to contribute and of players who are dominating in their groups:

"Generally, everyone had their say in final decision but a couple of times we were overridden"

4.6 Finding 6— The majority (44 out of 59) of participants preferred playing a board game rather than a fully computerized game

The players' responses to different features of the game were generally positive (Table 3). Notable in Table 3 is that a majority of players (44 out of 59) preferred playing with a game board rather than a fully computerized version. Some typical comments were:

"The board game [was] simple and I could easily see the state of the game"

"When a group plays the game on a PC, someone controls the mouse and keyboard and they tend to dominate"

"Compared to computer-based games, the design was simple and we started playing without too much wasted time"

"Sometimes technology gets in the way"

"Everyone plays board games so we all knew what to do"

Outside of this research project, seven players had played The Beer Game, four-point distribution chain, originally developed at MIT and now used widely as a management educational tool in a variety of academic and commercial settings (Caulfield, Kohli, & Maj, 2004; Goodwin & Franklin, 1994; Senge, Kleiner, Roberts, Ross, & Smith, 1994; Sterman, 1989). In The Beer Game all calculations are performed by hand on simple worksheets. This found favour:

"Doing the calculations by hand means we have to understand"

"The calculator half of the game hides details. Just give us a calculator and we can work it out"

Although the players' reception of the game was generally positive, clear written instructions are essential to make sure best use is made of the game session time. This comment was made by a player in the very first game session:

"Wasn't sure of what we were supposed to do"

Initially, instructions for playing the game were delivered by the researcher after the players had completed the pre-game survey and just before they started the game. For the second game session onwards, a one-page instruction sheet was emailed to each player a couple of days beforehand so they could be prepared.

The database of Simsoft game transactions showed that only three games had to be abandoned and restarted. It was observed that once teams had made the first couple of decisions, they were able to continue with too much trouble.

5. Discussion

The purpose of this research project was to see if and how games could contribute to better software project management education by helping software engineers and project managers explore some of the dynamic complexities of the field in a safe and inexpensive environment.

The major finding was that participants *were* learning as they played the game. However, the findings also suggested that games alone are not more effective than more traditional pedagogical means such as lectures, case studies, and readings. It also seems that simple games, and games in which the participants are able to relate game play to an external context, such as their real-world roles, are the most efficacious.

This section analyses and discusses the findings in more detail along the following broad analytic categories:

Games and learning in Simsoft.

Games in context.

The relative complexity of games.

5.1 Learning in Simsoft

The results showed that each group of participants (students, project managers, and software developers) improved their performance between the pre- and post-game tests. This suggests that the participants were constructing knowledge for themselves based on what they had experienced in the game. Comments from the participants supported this:

"Aha!"

"Now I see why"

When each group was further classified by years of experience in the field, the same improvement between the

pre- and post-game tests was seen, with the greatest improvement being in those with more experience. For example, students gained relatively less from the game than more experienced software developers and project managers.

Together these results suggest that learning *is* happening, but for some participants at least some level of *a priori* knowledge is necessary to make more sense of what is happening in the game. So, participants can learn some, but not all, of what they need to know from a game.

5.2 Games in Context

A common comment during the after-game gatherings, and something that was reflected in the post-game survey, was that most participants were prepared to invest greater time and effort in games such as Simsoft if the reward was deeper understanding of the problem domain.

With this in mind, one participant suggested running the game in real time, so that one week of real time equated to one week of project time. During the week, the team members could do research and discuss their options before coming to a carefully considered decision about their next step. This suggestion was influenced by a stock market game a number of participants had played the previous year. Players bought and sold shares on a fantasy stock exchange based on real prices published in the daily newspaper. The winner after three months was the team with the largest portfolio. In the week between submitting buy and sell orders, the players researched likely companies, scanned market reports, and took note of interest rate decisions, the price of oil and gold, and currency fluctuations to see how they might affect the market.

This suggestion represents a desire to put Simsoft more in context, by allowing the participants to step out of the fantasy world of the game, do some study, and then step back into the game with better knowledge. However, Simsoft, and all other software engineering management games discovered during a systematic search of the literature (Caulfield, Xia, Veal, & Maj, 2011), are played in one-off sessions. What players learned, must be learned within the hour or so of the game session. Of course, games can be replayed, but they must have sufficient depth to present alternate, engaging paths through the game in repeat. For even the most sophisticated game in the field, SimSE, players became bored when playing second and subsequent times (Navarro & van der Hoek, 2007).

One way to satisfy this desire for more depth, would be to play the game across multiple sessions over weeks or months as some participants have done with other games. In between, research could be undertaken in order to make the most informed decision.

For some participants, this break is necessary. Evaluation of the pre- and post-game scores showed that students gained relatively less from the game than more experienced project managers and developers. The following comment from a student is illustrative:

"I saw in the game aspects of theory covered at uni, but without knowing the theory first I probably wouldn't have recognised the significance."

That is, students in this research population didn't have the *a priori* knowledge needed to make full sense of the game's dynamics.

Playing the game over multiple, rather than single, sessions would more closely conform to the tenets of problem-based learning where participants begin their project with imperfect knowledge and then have to identify and learn what they needed in order to solve the issue at hand.

5.3 The Relative Complexity of Games

When asked, most Simsoft players said they preferred a board game to a fully computerized version because they could start playing more quickly without having to learn how to navigate a new user interface and without fear of making an unintended move. Apart from the mechanics of playing Simsoft, the simple design meant the state of the game and its underlying causal model were always visible.

The appeal of simplicity over complexity has been noted before. While complex games offer "the richest learning experience available, the game's very formidable appearance probably intimidated a number of players or forced them into a learning situation they were unprepared or unwilling to negotiate" (Wolfe, 1978, p.152). The next most effective game in Wolfe's study was found to be the least complex, supporting similar research that showed relatively simple games can provide essentially the same, if not more, benefits as the more complex (Butler, Pray, & Strang, 1979; Dempsey, et al., 2002; Meadows, 1999; Raia, 1966; Watt, 1977). Therefore, making games only as complex as necessary, or hiding unnecessary detail, could be a way of achieving the best learning outcomes while avoiding the player mortality (boredom and dropout) noted by Wolfe.

A board game also more easily fosters the collaboration needed in any team enterprise such as a software development project. When a computer or online game is played by multiple participants, likely at different physical locations, the basic cues of identity, personality, and body language are hidden. Without these cues, researchers have found that many computer games explicitly designed to be collaborative will degenerate into competitive games at worst or games in which "everyone just kind of does their own thing" (Zagal, Rick, & Hsi, 2006, p.25) at best.

In Simsoft, group play was viewed positively by most participants. It reflected real-world experience and also meant ideas and opinions could be shared:

"It was like [the agile] stand up meeting we have every morning"

"I thought software development was a solitary experience but it's not really"

"Everyone had a chance to offer an opinion"

Notwithstanding these positive aspects, any group activity may devolve into groupthink (Janis, 1971) in which the opinion of a dominant individual or clique prevails, possibly against reasonable evidence. In the Simsoft game sessions, no teams were larger than four participants and many participants were known to each, either professionally or socially, so there was ample opportunity to contribute to the discussions or even dispute the idea of a colleague or friend. There were also no more than four game sessions running at once, which meant the researcher was able to notice any participants standing back and gently prompt them for a contribution.

Few other software development game researchers have looked closely at these same aspects of game design. In (Hainey, Connelly, Stansfield, & Boyle, 2010), players were asked to rate game features such as graphics, realism of the characters, realism of the environment, and sound, but these were evaluations of the verisimilitude of these features, not their appropriateness to the task at hand. On this same rating of game features, collaboration ranked last or second last across all players, but this is to be expected in a single-player game. Similarly, other researchers (Baker, Oh Navarro, & van der Hoek, 2005; Navarro & van der Hoek, 2009; Zapata, 2010) asked their participants if they enjoyed playing the game or whether they found it engaging, but these questions ask the participants to evaluate a particular game's representation of its environment rather than its comparative complexity or its value as a collaborative tool.

5.4 Related Work

A recent systematic survey of games used in software engineering education (C. W. Caulfield, J. Xia, et al., 2011) found that, as a pedagogical device, they are becoming more common, particularly in Europe and the Americas, and students generally enjoyed playing them and felt they gained some value from the experience.

Simsoft differs from these other games in four main areas:

Simsoft is equally, if not more, concerned with *who* does the work in a software development as it is with process of *how* the work is done. This echoes the cover of Boehm's (1981) *Software Engineering Economics* which shows personnel is where the greatest productivity gains are possible.

Simsoft is largely a board game (with a small calculator component) in contrast to most other games that use a graphical user interface of varying levels of richness. Often the user interface is simply a conceit of the game for performing housekeeping functions and lends little to the real purpose. Other games that use playing cards or games boards contain an element of chance rather than skill.

Simsoft is cast at a level of detail at which the players can see the movement of individual pieces of work and individuals themselves. Games cast at higher levels can mask some fundamental project dynamics.

The research sample for this project is a mixture of students and experienced professionals rather than wholly students.

6. Conclusions

At the end of *War Games*, as Matthew Broderick and his girlfriend Ally Sheedy reflect on the world's narrow escape, the message is obvious: everyone has learned a lesson and blind reliance on computers is foolish. WOPR is quietly dismantled and won't be able to ask anyone else, "Shall we play a game?"

This paper posed the same question in a different context: shall we– should we– play games in software project management education? The answer, we believe, is a qualified, *yes*. The answer is qualified because our findings show that while games are useful pedagogical tools and are well-received by players, they are not sufficient in themselves and must be supplemented by other learning devices. Also, unless the games are designed with learning aforethought, they will probably miss their mark.

This project also points to some interesting directions for future research. Many participants said they preferred simple, collaborative games to complex games, and they were also prepared to play games in more depth if the reward was greater knowledge of the problem domain. The plan is to develop Simsoft further in these directions.

References

Abt, C. C. (1970). *Serious Games*. New York: The Viking Press.

Baker, A., Navarro, E., & Van der Hoek, A. (2005). An Experimental Card Game for Teaching Software Engineering Processes. *The Journal of Systems and Software,* 75(1 – 2). http://dx.doi.org/10.1016/j.jss.2004.02.033

Barell, J. (2006). *Problem-Based Learning: An Inquiry Approach* (2nd edition ed.). Thousand Oaks: Corwin Press.

Barrows, H. S., & Tamblyn, R. (1976). An Evaluation of Problem-Based Learning in Small Groups Utilizing a Simulated Patient. *Journal of Medical Education,* 51(1), 52 – 54. http://www.eric.ed.gov/ERICWebPortal/detail?accno=EJ131167

Bloomberg, L., Dale, & Volpe, M. (2008). *Completing Your Qualitative Dissertation*. Thousand Oaks: Sage Publications.

Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J., & Madachy, R. (1998). A Stakeholder Win–Win Approach to Software Engineering Education. *Annals of Software Engineering,* 6(1), 295 - 321. http://dx.doi.org/10.1023/A:1018988827405

Boehm, B. W. (1981). *Software Engineering Economics*. Sydney: Prentice-Hall.

Bourque, P., Dupuis, R., Abran, A., Moore, J. W., & Tripp, L. (1999). The Guide to the Software Engineering Body of Knowledge. *IEEE Software,* 16(6), 35 - 44. http://dx.doi.org/10.1109/52.805471

Brereton, O. P., Lees, S., Bedson, R., Boldyreff, C., Drummond, S., Layzell, P. J., et al. (2000). Student Group Work Across Universities: A Case Study in Software Engineering. *IEEE Transactions on Education,* 43(4), 394 – 399. http://dx.doi.org/10.1109/13.883348

Brooks, F. P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer,* 20(4), 10 – 19. http://doi.ieeecomputersociety.org/10.1109/MC.1987.1663532

Butler, R. J., Pray, T. F., & Strang, D. R. (1979). An Extension of Wolfe's Study of Simulation Game Complexity. *Decision Sciences,* 10, 480 – 486. http://dx.doi.org/10.1111/j.1540-5915.1979.tb00038.x

Caillois, R. (1961). *Man, Play and Games* (M. Barash, Trans.). New York: Free Press of Glencoe.

Caulfield, C. W., Veal, D., & Maj, S. P. (2011a). Implementing System Dynamics Models in Java. *International Journal of Computer Science and Network Security,* 11(7), 43 – 49.

Caulfield, C.W., Veal, D., & Maj, S. P. (2011b). Teaching Software Engineering Project Management – A Novel Approach for Software Engineering Programs. *Modern Applied Science,* 5(5), 87 – 104. http://dx.doi.org/10.5539/mas.v5n5p87

Caulfield, C. W., Kohli, G., & Maj, S. P. (2004). Sociology in Software Engineering. *Proceedings of Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition,* Salt Lake City.

Caulfield, C. W., Xia, J., Veal, D., & Maj, S. P. (2011). A Systematic Survey of Games Used for Software Engineering Education. *Modern Applied Science,* in press.

Cheng, Y. P., & Lin, J. M. C. (2010). A Constrained and Guided Apprach for Managing Software Engineering Course Projects. *IEEE Transactions on Education,* 53(3), 430 – 436. http://dx.doi.org/10.1109/TE.2009.2026738

Creswell, J. W. (2009). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (3rd edition ed.). Thousand Oaks: Sage Publications.

DeMarco, T. (1982). *Controlling Software Projects*. New York: Yourdon Press.

DeMarco, T., & Lister, T. (1999). *Peopleware: Productive Projects and Teams* (2nd edition ed.). New York: Dorset House Publishing Co.

Dempsey, J. V., Haynes, L. L., Lucassen, B. A., & Casey, M. S. (2002). Forty Simple Computer Games and What They Could Mean to Educators. *Simulation & Gaming,* 33(2), 157 – 168. http://dx.doi.org/10.1177/1046878102332003

Eldredge, D. L., & Watson, H. J. (1996). An Ongoing Study of the Practice of Simulation in Industry. *Simulation & Gaming,* 27(3), 375 – 386. http://dx.doi.ord/10.1177/1046878196273008

Faria, A. J. (1987). A Survey of the Use of Business Games in Academia and Business. *Simulation & Games,* 18(2), 207 – 224. http://dx.doi.org/10.1177/104687818701800204

Faria, A. J. (1998). Business Simulation Games: Current Usage Levels—An Update. *Simulation & Gaming,* 29(3), 295 – 308. http://dx.doi.org/10.1177/1046878198293002

Faria, A. J., & Wellington, W. J. (2004). A Survey of Simulation Game Users, Former-Users, and Never-Users. *Simulation & Gaming,* 35(2), 178 – 207. http://dx.doi.org/10.1177/1046878104263543

Faria, A. J., & Wellington, W. J. (2005). Validating Business Gaming: Business Game Conformity with PIMS Findings. *Simulation & Gaming,* 36(2), 259 – 273. http://dx.doi.org/10.1177/1046878105275454

Gee, J. P. (2007). *Good Video Games and Good Learning: Collected Essays on Video Games, Learning and Literacy*. New York: Peter Lang Publishing.

Goodwin, J. S., & Franklin, S. G. (1994). The Beer Distribution Game: Using Simulation to Teach Systems Thinking. *Journal of Management Development,* 13(8), 7 – 15. http://dx.doi.org/10.1108/02621719410071937

Gotterbarn, D. (1999). How the New Software Engineering Code of Ethics Affects You. *IEEE Software,* 16(6), 58 – 64. http://doi.ieeecomputersociety.org/10.1109/52.805474

Guba, E. G., & Lincoln, Y. S. (2005). Paradigmatic Controversies, Contradictions, and Emerging Confluences. In N. K. Denzin & Y. S. Lincoln (Eds.), *The Sage Handbook of Qualitative Research* (3rd edition ed., pp.191 – 215). Thousand Oaks: Sage Publications.

Hainey, T., Connelly, T. J., Stansfield, M., & Boyle, E. A. (2010). Evaluation of a Game to Teach Requirements Collection and Analysis in Software Engineering at Tertiary Education Level. *Computers & Education,* 56(1), 21 – 35. http://dx.doi.org/10.1016/j.compedu.2010.09.008

Heldman, K. (2007). *PMP: Project Management Professional Exam Study Guide* (4th edition ed.). San Francisco: Sybex.

iSSEc Project. (2009). *Graduate Software Engineering 2009 (GSwE2009): Curriculum Guideline for Graduate Degree Programs in Software Engineering*.

Janis, I. L. (1971). Groupthink. *Psychology Today,* 5(5), 43 - 46, 74 – 76.

Johns-Boast, L., & Patch, G. (2010). A Win-Win Situation: Benefits of Industry-Based Group Projects. *Proceedings of Australasian Association for Engineering Education Conference (AaeE 2010).*

Joint IS2010 Curriculum Task Force. (2010). *Curriculum Guideline for Undergraduate Degree Programs in Information Systems*: Association for Computing Machinery and Association for Information Systems.

Joint Task Force on Computing Curriculum. (2004). *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*: IEEE Computer Society/Association for Computing Machinery.

Kirk, J., & Miller, M. L. (1986). *Reliability and Validity in Qualitative Research*. London: Sage Publications.

Kruchten, P. (2005). Editor's Introduction: Software Design in a Postmodern Era. *IEEE Software,* 22(2), 16-18. http://doi.ieeecomputersociety.org/10.1109/MS.2005.38

Kruchten, P. B. (2004). The Nature of Software: What's So Special About Software Enginering? [Online] Available: www.ibm.com/developerworks/rational/library/4700.html

Lincoln, Y. S., & Guba, E. G. (1984). *Naturalistic Inquiry*. London: Sage Publications.

Marshall, M. N. (1996). Sampling for Qualitative Research. *Family Practice,* 13(6), 522 – 525.

Maxwell, J. A. (2004). *Qualitative Research Design: An Interactive Approach* (2nd edition ed.). Thhousand Oaks: Sage Publications.

Maxwell, N. L., Mergendoller, J. R., & Bellisimo, Y. (2004). Developing a Problem-Based Learning Simulation. *Simulation & Gaming,* 35(4), 488 - 498. http://dx.doi.org/10.1177/1046878104264789

McCall, J. (2011). *Gaming the Past: Using Video Games to Teach Secondary History* London: Routledge.

McConnell, S. (2004). *Professional Software Development*. Boston: Addison-Wesley.

McKenna, R. J. (1991). Business Computerized Simulation: The Australian Experience. *Simulation & Gaming,*

22(1), 36 – 62. http://dx.doi.org/10.1177/1046878191221003

Meadows, D. L. (1999). Learning to Be Simple: My Odyssey with Games. *Simulation & Gaming,* 30(3), 342 – 351. http://dx.doi.org/10.1177/104687819903000310

Michael, D., & Chen, S. (2005). *Serious Games: Games That Educate, Train, and Inform*. Boston: Thomson Course Technology PTR.

Miller, J. G. (1978). *Living Systems*. New York: McGraw-Hill Book Company.

Navarro, E. O., & van der Hoek, A. (2007). Comprehensive Evaluation of an Educational Software Engineering Simulation Environment. *Proceedings of The Twentieth Conference on Software Engineering Education and Training*, July 2007 Dublin, Ireland

Navarro, E. O., & van der Hoek, A. (2009). Multi-Site Evaluation of SimSE. *Proceedings of The 40th ACM Technical Symposium on Computer Science Education* March 3 – 7 Chattanooga, Tennessee.

Naveda, J. F., & Seidman, S. B. (Eds.). (2006). *IEEE Computer Society Real-World Software Engineering Problems: A Self-Study Guide for Today's Software Professional*. Hoboken: John Wiley & Sons.

Osterweil, L. (1987). Software Processes are Software Too. *Proceedings of Proceedings of the 9th International Conference on Software Engineering* Monterey, California.

Patton, M. Q. (2002). *Qualitative Research and Evaluation Methods* (3rd edition ed.). Thousand Oaks: Sage Publications.

Perla, P. P. (1990). *The Art of Wargaming: A Guide for Professionals and Hobbyists*. Annapolis, Maryland: Naval Institute Press.

Polack-Wahl, J. A. (2006). Lessons Learned From Different Types of Projects in Software Engineering. *Proceedings of International Conference on Frontiers in Education: Computer Science & Computer Engineering*, June 26-29, 2006, Las Vegas, Nevada

Prensky, M. (2006). *Don't Bother Me Mom– I'm Learning!*. St. Paul, Minnesota: Paragon House Publishers.

Raia, A. P. (1966). A Study of the Educational Value of Management Games. *The Journal of Business,* 39(3), 339 – 352. http://dx.doi.org/doi:10.1086/294863

Richards, L. (2009). *Handling Qualitative Data* (2nd edition ed.). Thousand Oaks: Sage Publications.

Riddell, R. (1997, April). Doom Goes to War. *Wired,* 5, 113 – 118, 164 – 166.

Robillard, P. N., & Robillard, M. (1998). Improving Academic Software Engineering Projects: A Comparative Study of Academic and Industry Projects. *Annals of Software Engineering,* 6(1), 343 - 363. http://dx.doi.org/10.1023/A:1018925902814

Rossman, G. B., & Rallis, S. F. (1998). *Learning in the Field: An Introduction to Qualitative Research*. Thousand Oaks: Sage Publications.

Salen, K., & Zimmerman, E. (Eds.). (2005). *The Game Design Reader: A Rules of Play Anthology* Cambridge, Massachusetts: The MIT Press.

Savin-Baden, M. (2003). *Facilitating Problem-Based Learning*. Maidenhead: The Society for Research into Higher Learning & Open University Press.

Savin-Baden, M., & Major, C. H. (2004). *Foundations of Problem-Based Learning*. Maidenhead: The Society for Research into Higher Learning & Open University Press.

Schrage, M., & Peters, T. (1999). *Serious Play : How the World's Best Companies Simulate to Innovate*: Harvard Business School Press.

Senge, P. M., Kleiner, A., Roberts, C., Ross, R. B., & Smith, B. J. (1994). *The Fifth Discipline Fieldbook*. London: Nicholas Brealey Publishing.

Sterman, J. D. (1989). Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Environment. *Management Science,* 35(3), 321 – 339. http://dx.doi.org/10.1287/mnsc.35.3.321

Strauss, A. L. (1987). *Qualitative Analysis for Social Scientists*. Cambridge: Cambridge University Press.

Suits, B. (1967). What is a Game?. *Philosophy of Science,* 34(2), 148 – 156. http://www.jstor.org/stable/186102

Tang, C., Lai, P., Tang, W., Davis, H., Frankland, S., Oldfield, K., et al. (1997). Developing a Context-Based PBL Model. In J. Conway, R. Fisher, L. Sheridan-Burns & G. Ryan (Eds.), *Research and Development in*

Problem Based Learning: Integrity, Innovation, Integration (pp.588 – 589). Newcastle: Australian Problem Based Learning Network.

Watt, K. E. F. (1977). Why Won't Anyone Believe Us?. *Simulation,* 28(1), 1 – 3. http://dx.doi.org/10.1177/003754977702800102

Wolfe, J. (1978). The Effects of Game Complexity on the Acquisition of Business Policy Knowledge. *Decision Sciences,* 9(1), 143 – 155. http://dx.doi.org/10.1111/j.1540-5915.1978.tb01373.x

Zagal, J. P., Rick, J., & Hsi, I. (2006). Collaborative Games: Lessons Learned from Board Games. *Simulation & Gaming,* 37(1), 24 — 40. http://dx.doi.org/10.1177/1046878105282279

Zapata, C. M. (2010). A Classroom Game for Teaching Management of Software Companies. *Dyna,* 77(163), 290 – 299.

Zyda, M. (2007). Creating a Science of Games. *Communications of the ACM,* 50(7), 26 – 29. http://dx.doi.org/10.1145/1272516.1272535

Table 1. Comparison of players pre- and post-game test scores

Role and Experience (in years)	n	Average pre-test score (out of 8)	Average pre-test score (out of 8)	Difference Between Pre- and Post-Game Scores
Students	**17**	**4.64 (SD = 0.861)**	**5.41 (SD = 1.460)**	**+0.77**
0 to 1 years	17	4.64 (SD = 0.861)	5.41 (SD = 1.460)	+0.77
Software Developers	**30**	**5.53 (SD = 0.995)**	**6.33 (SD = 1.107)**	**+0.80**
0 to 1	0			
2 to 5 years	14	5.57 (SD = 1.089)	6.07 (SD = 1.268)	+0.50
5 to 10 years	11	5.72 (SD = 1.009)	6.818 (SD = .0750)	+1.098
10 to 15 years	5	5.00 (SD = 0.707)	6.00 (SD = 1.224)	+1.00
15+ years	0			
Project Managers	**12**	**4.66 (SD = 1.497)**	**5.42 (SD = 2.020)**	**+0.76**
0 to 1	0			
2 to 5 years	6	4.5 (SD = 2.073)	5.00 (SD = 2.529)	+0.50
5 to 10 years	1	5.00 (SD = NA)	6.00 (SD = NA)	+1.00
10 to 15 years	4	4.75 (SD = 0.957)	5.75 (SD = 1.892)	+1.00
15+ years	1	5.00 (SD = NA)	6.00(SD = NA)	+1.00
	59	5.10 (SD = 1.155)	5.88 (SD = 1.486)	+0.78

Table 2. Participants' responses when asked whether they thought Simsoft was easy or difficult to play

Role and Experience (in years)	n	Average Response
Students	17	3.17 (SD = 0.528)
0 to 1 years	17	3.17 (SD = 0.528)
Software Developers	30	2.93 (SD = 0.253)
0 to 1	0	
2 to 5 years	14	2.92 (SD = 0.267)
5 to 10 years	11	3.00 (SD = 0.000)
10 to 15 years	5	2.80 (SD = 0.447)
More than 15 years	0	
Project Managers	12	2.58 (SD = 0.514)
0 to 1	0	
2 to 5 years	6	2.83 (SD = 0.408)
5 to 10 years	1	3.00 (SD = NA)
10 to 15 years	4	2.25 (SD = 0.500)
More than 15 years	1	2.00 (SD = NA)
	59	2.93 (SD = 0.449)

Table 3. Players' evaluation of game features

Feature	Average (1 = very bad, 5 = very good; or 1 = strongly disagree, 5 = strongly agree)
Written instructions	Average = 4.44, SD = 0.771
The game was interesting	Average = 4.37, SD = 0.963
Realistic scenario	Average = 4.37, SD = 0.692
Navigation around the game	Average = 4.22, SD = 0.744
Game logic was apparent	Average = 4.18, SD = 0.730
Useful to work in teams	Average = 4.15, SD = 0.714
Prefer game-board version	Average = 3.98, SD = 0.754

ISSN 1913-1844 E-ISSN 1913-1852