A Systematic Survey of Games Used for Software Engineering Education

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

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

David 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

S Paul 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

Received: September 7, 2011 Accepted: October 9, 2011 Published: December 1, 2011
doi:10.5539/mas.v5n6p00 URL: http://dx.doi.org/10.5539/mas.v5n6p00

Abstract

Simsoft is a serious game— one that trains or educates— at the centre of a research project designed to see if and how games can contribute to better software engineering management education by helping software engineers and project managers explore some of the dynamic complexities of the field in a safe and inexpensive environment. A necessary precursor for this project was to establish what games already existed in the field and how effective they had been. To this end a systematic review of the literature was conducted using a collection of online science, engineering, education, and business databases looking for games or simulations used for educational or training purposes in software engineering or software project management across any of the SWEBOK knowledge areas. The initial search returned 243 results, which was filtered to 36 papers by applying some simple quality and relevance inclusion/exclusion criteria. These remaining papers were then analysed in more depth to see if and how they promoted education in the field of software engineering management. The results showed that games were mainly used in the SWEBOK knowledge areas of software engineering management and development processes, and most game activity was in Europe and the Americas. The results also showed that most games in the field have learning objectives pitched at the first rung of Bloom's taxonomy (knowledge), most studies followed a non-experimental design, and many had very small sample sizes. This suggests that more rigorous research is needed into the efficacy of games in teaching software engineering management, but enough evidence exists to say that educators could include serious games in their courses as a useful and interesting supplement to other teaching methods.

Keywords: Software engineering, Project management education, Serious games

1. Introduction

1.1 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. To start, 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. 14). Meanwhile, a simulation is a special kind of model that exhibits processes in some way like the system is represents, and that shows how these processes change from state A to state B, between two points in time (J. G. Miller, 1978, p. 83).

Games naturally come in many forms. In a seminal work in the field, *Man, Play and Games*, Caillois (1961) proposed a classification that depends on whether the role of competition (*agôn*), chance (*alea*), simulation (*mimicry*), or vertigo (*ilinx*) is dominant. *Agôn* are 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). Football, billiards, or chess fall into this category. *Alea* are games of chance such as roulette or a lottery; games of mimicry involve the players becoming other characters, such as cowboys and Indians; while *ilinx* are "those which are based in the pursuit of vertigo and which consists of an attempt to momentarily destroy the stability of perception and inflict a kind of voluptuous panic upon an otherwise lucid mind" (Caillois, 1961, p. 23).

The games that this research project deals with are a subset of Caillois's *agôn* classification and they use an adjective— serious— to show they want for more than simple amusement and that they are designed to educate, train, or inform their players (Abt, 1970; Michael & Chen, 2005; Schrage & Peters, 1999).

1.2 The Value of Games

Games have been used to train and educate players for many years in many different fields (see for example, Gee, 2007b; Michael & Chen, 2005; Perla, 1990; Prensky, 2007) and are based on learning and development theories such as problem-based learning (Savin-Baden & Major, 2004), experiential education (Dewey, 1938/1963; Kolb, 1984; Papert, 1980), and decision science (Raser, 1969, pp. 46-55). Yet, to a common extent, games have been found to be more expensive and administratively demanding to develop and use than some other forms of instruction or research (Abt, 1970, pp. 110-111; Babb, Leslie, & Van Syke, 1966, p. 471; Cohen & Rhenman, 1961, p. 151; Petranek, 1994). Still, there are some offsetting advantages.

For example, it has been noted that the human capacity to understand the implications of our mental models and to accurately trace through even a small number of causal relationships is fairly limited (G. A. Miller, 1956; Simon, 1957). Yet, a game is a visible and physical representation of a problem space; a captured mental model. As such, they are places to trial new ideas and to experiment with established theories (Feldman, 1995; McKenney, 1962); to replay these theories as many times as needed; places where time and space can be contracted or expanded (Raser, 1969); places where it is acceptable just to try different things and where more might be learned from failure than success (Booker, 1994).

Even so, there are some dangers to be heeded when using games. Games are just… games, and as such are just one representation of how the world works. Therefore, "it is potentially dangerous to have players leave the gaming environment with the belief that the strategies that were effectively employed in playing the game are directly transferable to the real world" (Watson & Blackstone, 1989, p. 493). Participants should ideally be provided with more information than just the game to help them wisely discriminate between what may or may not work outside the game itself (Andlinger, 1958, pp. 152-158).

It was with these pros and cons aforethought that a game—Simsoft (Caulfield, Veal, & Maj, 2011b)—was developed to see what value games might bring to the education of software engineers and project managers.

1.3 Simsoft

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, 2011a), 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 a small set of fundamental causal relationships of simple software development projects.

In Simsoft game sessions, 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.

A necessary precursor for this project was find out what games already existed in the field of software engineering education, how effective they had been, and how Simsoft might be able to contribute new knowledge. To this end a systematic review of the literature was conducted using a collection of online science, engineering, education, and business databases looking for games or simulations used for educational or training purposes in software engineering or software project management across any of the Software Engineering Body of Knowledge (SWEBOK (Bourque, Dupuis, Abran, Moore, & Tripp, 1999)) knowledge areas.

2. Survey Methods

For this survey, we followed an established procedure for conducting systematic reviews in the field of software engineering (Kitchenham, 2004), which has been used to survey the game field before (Gresse von Wangenheim & Shull, 2009). Given the upward trend in the use of games for software engineering education revealed in that previous survey, it was timely to update and expand the search.

2.1 Data Sources and Search Strategy

To perform this review we used the IEEE Xplore Digital Library, the ACM Digital Library, ScienceDirect, Sage Journals Online, ProQuest, the ISI Web of Knowledge, and the Wiley Online Library. The following pseudo-code search string was adapted for the specific query languages of each library:

where abstract OR title OR keywords contain

((game OR simulation) AND (learning OR teaching OR education OR training))

AND

(software engineering OR software project OR

software process OR software design OR

software testing OR software configuration management OR

software quality OR software management OR

software maintenance OR software construction

OR software requirements OR software engineering tools and methods))

AND

(date >= 1990)

That is, we looked for games or simulations (computer and non-computer based) used for educational or training purposes in software engineering or software project management across any of the SWEBOK knowledge areas. (Despite the distinction made between game and simulation in the introduction, the terms are often used interchangeably in the literature (Maier & Grossler, 2000), therefore *simulation* has been used as one of the search parameters).

2.2 Inclusion and Exclusion Criteria

We limited the results to English-language papers published from 1990 to the present in peer-reviewed journals and conference proceedings. We excluded position papers, papers in which no data was reported (unless they were preliminary papers for completed studies), and those in which the game or simulation was not used to train or educate the players at a tertiary level.

2.3 Study Identification and Selection

The initial database searches returned a total of 243 papers. The titles and abstracts were analysed according to the inclusion and exclusion criteria, and any off-topic papers were discarded. This left 36 papers, which were

grouped according to the study they described.

2.3 Data Extraction

Each paper passing the selection process was read in depth and the following data was extracted:

- References to the papers describing the study.
- A brief description of the game and how it was played.
- The experimental design used by the study, which could be either true experimental (random assignment and comparison with a control group), quasi-experimental (comparison with a control group only), or non-experimental.
- The number and type of the players.
- The type of research tool used to collect the data, for example questionnaires, observation, pre- and post-test surveys.
- The primary SWEBOK knowledge area on which the game is focussed. The SWEBOK attempts to characterise and bound the software engineering body of knowledge; the ten knowledge areas are the major topical divisions within the field.
- The expected learning outcomes classified according to Bloom's (1956) cognitive domain taxonomy. The cognitive domain defines six incremental levels of learning objectives that educators may have for their students: *knowledge*: remember previously-learned materials by recalling specific facts, terminology, theories and answers; *comprehension*: demonstrate an understanding of information by being able to compare, contrast, organize, interpret, describe, and extrapolate; *application*: use previously-learned material in new situations; *analysis*: decompose previously-learned material into parts in order find patterns and to make inferences and generalizations; *synthesis*: use existing ideas in different ways to create new ideas or to propose alternative solutions; *evaluation*: judge the validity of ideas or information with a certain context.
- The principal findings of the study.
- The country in which the game sessions were conducted.

Table 1 shows the full data extract of 36 papers describing 26 studies.

3. Survey Results

Figure 1 shows that the preferred medium for games in the field is computer-based (22 out of 26) rather than other types such as board and card games. This way the games are easier to distribute and administer across a large number of players who may be in remote locations. Figure 1 also shows that most of the studies were non-experimental (16 out of 26) meaning they didn't use control groups nor randomly assign participants to different groups.

The survey results show that games have been used in a variety of ways to teach different aspects of software engineering and software project management. Figure 2 shows the distribution of games across the world based on the SWEBOK knowledge area they were designed to address. Most games (21 out of 26) focused broadly on software engineering management or the development process and most activity (21 out of 26) occurred in Europe and the Americas.

Figure 1 shows that overwhelmingly, the learning objectives of the studies pitched at the first rung of Bloom's taxonomy, knowledge. In general, those studies that assessed the degree of learning by the participants found that the participants were sometimes learning new concepts, but they were mainly reinforcing known theories. All the research projects, whether explicitly or implicitly stated, found that games alone were not sufficient pedagogical devices to teach software engineering or project management concepts and would have to be supplemented by other means. Only Navarro (2009) and Hainey et al. (2010) evaluated the effectiveness of games for players of different skills and backgrounds and each found that games were suitable for a wide variety of participants.

It should be noted, however, that apart from Navarro's and Drappa and Ludewig's body of work, many of the research projects in Table 1 had very small sample sizes and few others were developed or repeated beyond that described in the initial papers.

4. Simsoft Compared to Other Games in the Field

Recalling the discussion of model, simulation, and game given at the beginning of this paper: a *model* is a convenient representation (in words, numbers, or other symbols) of some real-world socio-economic or socio-technical system; a *simulation* is dynamic, operational model through which changes over time are revealed; and

a *game* is a simulation that is purposefully run, wholly or partly determined by players' decisions, within some predetermined circumstances. It can be said that software development has been *modelled* (Belady & Lehman, 1976; Boehm, 1981; Boehm et al., 2000; McCabe, 1976; H. Remus & Zilles, 1979) and *simulated* (Abdel-Hamid & Madnick, 1991; Collofello, 2000; Hansen, 1996; Madachy, 2008; Raffo, 1996; Tvedt, 1996; Variale, Rosetta, Steffen, Rubin, & Yourdon, 1994) many times. But, these are not the software engineering perspectives of interest here because:

- They focus primarily on predicting rather than educating. For example, Boehm's COCOMO model (2000) is designed to calculate the cost and effort of a software project based on historical data and what is currently known about the project at hand. COCOMO is used to validate an estimate, not necessarily find out why it is this number.
- They are not interactive or designed for group participation. For example, perhaps the most well-known simulation (Abdel-Hamid & Madnick, 1991) contains over 300 underlying variables, doesn't have a way to interact with the model except through direct manipulation of these variables at a source code level, and still does not describe the development process in detail (Martin, 2002, pp. 32-37).

Given their focus, it is not surprising that these models and simulations fail most, if not all, of Gee's principles of interactive game design (Caulfield, Veal, & Maj, 2011c; Gee, 2007a, 2007b). In contrast, the games described in Table 1 more closely align with Gee's principles. Still, there are differences between these games and Simsoft.

SimSE, the game developed by Navarro (2009) and her colleagues at the University of California, Irvine over a number of years, is perhaps the most advanced game in the field and the only one in Table 1 that has been developed much beyond its initial implementation. SimSE supports a number of different development methodologies (such as rapid prototyping, inspection, and the Rational Unified Process), provides users with a performance report after they complete the game, and has also been tested and verified in a range of controlled classroom settings. Players manage their SimSE project through a rich graphical user interface that shows their team at work along with various management reports and dials. In contrast to Simsoft, SimSE is a single-user game so without players clustering around a single screen, there's little opportunity to discuss and debate project decisions and come to a consensus. SimSE is also heavily focussed on the process of software development– the *how* of software development– whereas Simsoft is also concerned with the *who*.

Like Simsoft, a number of the games in the field have eschewed computers, either completely or partly, in favour of playing cards, boards, and sometimes dice. For example, in Zapata's (2010) game, teams throw a dice, that determines which of a collection of technical questions the team must answer. From here, the team gets a chance to estimate the size of a project component and score points. This slightly convoluted game show format relies more on chance than skill and means that most players are dormant and passive while other teams are having their turn. Chance also plays a role in games like Problems and Programmers (Baker, Oh Navarro, & van der Hoek, 2005)– players draw cards from a shuffled deck– and PlayScrum (Fernandes & Sousa, 2010)– a roll of the dice determines what resources the player can accumulate and what problems may be encountered. Unlike Simsoft, these games are competitive rather than co-operative.

Some of the games in Table 1offer only a very high level of interactivity meaning players can perform just broad project functions and hence only see general project dynamics. In SimVBSE (Jain & Boehm, 2006), SimjavaSP (Shaw & Dermoudy, 2005), MO-SEProcess (Zhu, Wang, & Tan, 2007), Hainey's game (2010), and OSS (Sharp & Hall, 2000) players make their avatar visit certain rooms or characters to ask questions or collect information. In Hainey's game the result of this office tour is a requirements document that is then passed to the project manager avatar for assessment. The tour may have to be repeated if all the requirements haven't been identified. A game interface makes this engaging for a while, but how it relates to real-world software project management is dubious. Providing the same information in a short project description, such as the one that comes with Simsoft, means the player can begin exploring the problem domain sooner. And, with less effort required to create the office environment, more could be devoted to the interesting detail of the project's dynamics.

SESAM (Drappa & Ludewig, 1999; Drappa & Ludewig, 2000) could almost be called a model or simulation rather than a game because a user runs it by typing commands in a complex modelling language and the system responds in kind. In exchange for this complexity, SESAM allows its users to define a wide variety of development methodologies as well as hire and fire staff, assign tasks, and ask developers about their progress. But, without an effective visual interface, playing SESAM is like programming an old VCR: there isn't enough feedback to know what is happening (Norman, 1988, pp. 51-53). It is perhaps not surprising that SESAM has not been developed far beyond that described in the original papers. In contrast, Simsoft's state of play is always visible on the game board.

One feature common to all the projects in Table 1 is the research population they use: the participants are either

undergraduate or post-graduate university, and in one case high school, students. In broader research circles, there is some debate (Camerer & Johnson, 1991; Garb, 1989; Remus, 1986) about whether students make viable candidates for research involving management decisions because they may lack the experience and knowledge to make their responses transferable to the workplace. Simsoft side-steps this still inconclusive debate because its research population is a mixture of university students and project managers and software developers of varying lengths of experience.

In summary, there are four main differences between the approach taken in this research project and others in the area:

- 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 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, such as OSS, mask some fundamental project dynamics.
- The research sample for this project is a mixture of students and experienced professionals rather than wholly students.

5. Conclusions

This systematic survey of games used in software engineering management education has shown that, as a pedagogical device, they are becoming more common, particularly in Europe and the Americas, and students in general enjoyed playing them and felt they got some value from the experience. However, few of the games were developed beyond their initial implementations suggesting their pedagogical value was not demonstrated sufficiently.

From these findings, there are some implications for researchers, educators, and game developers:

- More rigorous research is needed into the efficacy of games in teaching software engineering management. Most of the games in Table 1didn't follow a true experimental design and many had very small sample sizes, meaning the findings should be viewed with some caution.
- Even so, enough evidence exists to suggest that educators should consider using games as part of their courses in software engineering, but as an interesting supplement to other teaching materials and preferably later in the course when the students have had time to gain the knowledge needed to make sense of what the game is trying to teach.
- In many of the games in Table 1, rich graphics and avatars contributed little to meeting the learning objectives of the game and sometimes distracted or frustrated the players. Making the games simpler would shorten the time it takes to create the games and also allow the players to focus more on the content.

These findings have influenced the design and implementation of Simsoft, the serious game behind this research project. For example, Simsoft is a simple, collaborative board game, which has so far been played by combined teams of students and experienced software developers and project managers. Further games sessions are under way to test the efficacy of the current implementation.

References

Abdel-Hamid, T. K., & Madnick, S. E. (1991). *Software Project Dynamics: An Integrated Approach*. Englewood Cliffs: Prentice-Hall.

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

Andlinger, G. R. (1958). Looking Around: What Can Business Games Do?. *Harvard Business Review*. *36*(4), 147–160.

Babb, E. M., Leslie, M. A., & Van Syke, M. D. (1966). The Potential of Business Gaming Methods in Research. *The Journal of Business*. *39*(4), 465–472. http://dx.doi.org/10.1086/294887

Baker, A., Navarro, E. O., & van der, H., Andre. (2003). Problems and Programmers: An Educational Software Engineering Card Game. *Proceedings of The 25th International Conference on Software Engineering (ICSE'03)*. 3-10 May, 2003. Portland: Oregon. http://doi.ieeecomputersociety.org/10.1109/ICSE.2003.1201245

Baker, A., Oh 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

Barros, M. d. O., Dantas, A. R., Veronese, G. O., & Werner, C. M. L. (2006). Model-Driven Game Development: Experience and Model Enhancements in Software Project Management Education. *Software Process: Improvement and Practice*. 11(4), 411–421. http://dx.doi.org/10.1002/spip.279

Belady, L. A., & Lehman, M. M. (1976). A Model of Large Program Development. *IBM Systems Journal*. 15(3), 225–252. http://dx.doi.org/10.1147/sj.153.0225

Bloom, B. S., Masia, B. B., & Krathwohl, D. R. (1956). *Taxonomy of Educational Objectives: The Classification of Educational Goals* (Handbook I: Cognitive Domain ed.). London: Longman.

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

Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., et al. (2000). *Software Cost Estimation with Cocomo II*. Upper Saddle River: Prentice Hall.

Booker, E. (1994, 4 July). Have You Driven a Simulated Ford Lately? *Computerworld*. 28, 76.

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

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

Camerer, C. F., & Johnson, E. J. (1991). The Process–Performance Paradox in Expert Judgment. In K. A. Ericsson & J. Smith (Eds.). *Toward a General Theory of Expertise: Prospects and Limits*. (pp. 195–217). Cambridge: Cambridge University Press.

Caulfield, C W., Veal, D., & Maj, S. P. (2011c). Teaching Software Engineering Project Management–A Novel Approach for Software Engineering Programs. *Modern Applied Science*. 5(5). in press.

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 Management – Issues and Perspectives. *IJCSNS International Journal of Computer Science and Network Security*. 11(7), 50–54.

Cohen, K. J., & Rhenman, E. (1961). The Role of Management Games in Education and Research. *Management Science*.7, 131–166. http://dx.doi.org/10.1287/mnsc.7.2.131

Collofello, J. (2000). University/Industry Collaboration in Developing a Simulation Based Software Project Management Training Course. *Proceedings of Proceedings of the Thirteenth Conference on Software Engineering Education & Training*. Austin: Texas.

Connolly, T. M., Stansfield, M., & Hainey, T. (2007). An Application of Games-Based Learning within Software Engineering. *British Journal of Educational Technology*. *38*(3), 416–428. http://dx.doi.org/10.1111/j.1467-8535.2007.00706.x

Dantas, A. R., Barros, M. d. O., & Werner, C. M. L. (2004). A Simulation-Based Game for Project Management Experiential Learning. *Proceedings of The Sixteenth International Conference on Software Engineering & Knowledge Engineering*. Banff: Alberta, Canada.

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

Dewey, J. (1938/1963). *Experience and Education*. New York: Collier Books.

Drappa, A., & Ludewig, J. (1999). Quantitative Modeling for the Interactive Simulation of Software Project. *The Journal of Systems and Software*. 46(15 April), 113–122. http://dx.doi.org/10.1016/S0164-1212(99)00005-9

Drappa, A., & Ludewig, J. (2000). Simulation in Software Engineering Training. *Proceedings of The 22nd International Conference on Software Engineering*. Limerick, Ireland. http://dx.doi.org/10.1145/337180.337203

Feldman, H. D. (1995). Computer-Based Simulation Games: A Viable Educational Technique for Entrepreneurship Classes?. *Simulation & Gaming*. 26(3), 346–360. http://dx.doi.org/10.1177/1046878195263006

Fernandes, J. M., & Sousa, S. M. (2010). PlayScrum - A Card Game to Learn the Scrum Agile Method. *Proceedings of The 2010 Second International Conference on Games and Virtual Worlds for Serious Applications*.

Garb, H. N. (1989). Clinical Judgment, Clinical Training, and Professional Experience. *Psychological Bulletin*. *105*(3), 387–396.

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

Gee, J. P. (2007b). *What Video Games Have to Teach Us About Learning and Literacy*. New York: Palgrave MacMillan.

Gresse von Wangenheim, C., & Shull, F. (2009). To Game or Not to Game?. *IEEE Software*. 26(2), 92 – 94.http://doi.ieeecomputersociety.org/10.1109/MS.2009.54

Gresse von Wangenheim, C., Thiry, M., & Kochanski, D. (2009). Empirical Evaluation of an Educational Game on Software Measurement. *Empirical Software Engineering*. 14(4), 418-452. http://dx.doi.org/10.1007/s10664-008-9092-6

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

Hansen, G. A. (1996). Simulating the Software Development Process. *IEEE Computer*. 29(1), 73–77. http://doi.ieeecomputersociety.org/10.1109/2.481468

Jain, A., & Boehm, B. (2006). SimVBSE: Developing a Game for Value-Based Software Engineering. *Proceedings of Proceedings of the 19th Conference on Software Engineering Education & Training*. http://dx.doi.org/10.1109/cseet.2006.31

Kitchenham, B. A. (2004). *Procedures for Performing Systematic Reviews*. Keele University, Staffordshire.

Knauss, E., Schneider, K., & Stapel, K. (2008). A Game for Taking Requirements Engineering More Seriously. *Proceedings of The Third International Workshop on Multimedia and Enjoyable Requirements Engineering - Beyond Mere Descriptions and with More Fun and Games*. 9 September, 2008. Barcelona, Spain. http://doi.ieeecomputersociety.org/10.1109/MERE.2008.1

Kolb, D. A. (1984). *Experiential Learning: Experience as the Source of Learning and Development*. Englewood. Cliffs: Prentice-Hall.

Madachy, R. J. (2008). *Software Process Dynamics*. Hoboken: John Wiley & Sons.

Maier, F. H., & Grossler, A. (2000). What Are We Talking About? — A Taxonomy of Computer Simulations to Support Learning. *System Dynamics Review*. 16(2), 135–148. http://dx.doi.org/10.1002/1099-1727(200022)16:2<135::AID-SDR193>3.0.CO;2-P

Mandl-Striegnitz, P. (2001). *How to Successfully Use Software Project Simulation for Educating Software Project Managers*. Proceedings of The Frontiers in Education Conference.

Martin, R. C. (2002). *Agile Software Development: Principles, Patterns, and Practices*. Upper Saddle River: Prentice-Hall.

McCabe, T. J. (1976). A Software Complexity Measure. *IEEE Transactions on Software Engineering*. 2(4), 308–320. http://doi.ieeecomputersociety.org/10.1109/TSE.1976.233837

McKenney, J. L. (1962). An Evaluation of a Business Game in an MBA Curriculum. *The Journal of Business*. *35*(3), 278–286.

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

Miller, G. A. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*. *63*(2), 81–97.

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

Navarro, E. O. (2006). *SimSE: A Software Engineering Simulation Environment for Software Process Education.* Unpublished Thesis. University of California, Irvine.

Navarro, E. O., & van der Hoek, A. (2005). Design and Evaluation of an Educational Software Process Simulation Environment and Associated Model. *Proceedings of The Eighteenth Conference on Software Engineering Education and Training*. Ottawa, Canada.

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.

Navarro, E. O., & van der Hoek, A. (2008). On the Role of Learning Theories in Furthering Software Engineering Education. In H. J. C. Ellis, S. A. Demurjian & J. F. Naveda (Eds.). *Software Engineering: Effective Teaching and Learning Approaches and PracticesIGI Global*. IGI Global.

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.

Navarro, E. O., Baker, A., & van der Hoek, A. (2004). Teaching Software Engineering Using Simulation Games. *Proceedings of The 2004 International Conference on Simulation in Education*. January 2003 San Diego, California

Norman, D. A. (1988). *The Psychology of Everyday Things*. New York: Basic Books.

Oh, E., & van der Hoek, A. (2002). Towards game-Based Simulation as a Method of Teaching Software Engineering. *Proceedings of The Frontiers in Education. 2002. FIE 2002*. 32nd Annual, 6-9 Nov. 2002. http://dx.doi.org/10.1109/FIE.2002.1158674

Papert, S. (1980). *Mindstorms*. Brighton. Sussex: The Harvester Press.

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

Petranek, C. F. (1994). A Maturation in Experiential Learning: Principles of Simulation and Gaming. *Simulation & Gaming*. *25*(4), 513–522, http://dx.doi.org/10.1177/1046878194254008.

Pfahl, D., Koval, N., & Ruhe, G. (2001). An Experiment for Evaluating the Effectiveness of Using a System Dynamics Simulation Model in Software Project Management Education. *Proceedings of The Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International*. http://dx.doi.org/10.1109/METRIC.2001.915519

Pfahl, D., Laitenberger, O., Dorsch, J., & Ruhe, G. (2003). An Externally Replicated Experiment for Evaluating the Learning Effectiveness of Using Simulations in Software Project Management Education. *Empirical Software Engineering*. 8(4), 367–395. http://dx.doi.org/10.1023/a:1025320418915

Pfahl, D., Laitenberger, O., Ruhe, G., Dorsch, J., & Krivobokova, T. (2004). Evaluating the Learning Effectiveness of Using Simulations in Software Project Management Education: Results from a Twice Replicated Experiment. *Information and Software Technology*. 46(2), 127-147. http://www.sciencedirect.com/science/article/pii/S0950584903001150

Prensky, M. (2007). *Digital Game-Based Learning*. St. Paul, Minnesota: Paragon House Publishers.

Raffo, D. M. (1996). *Modeling Software Processes Quantitatively and Assessing the Impact of Potential Process Changes on Process Performance (TQM).* Unpublished Thesis, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Raser, J. R. (1969). *Simulation and Society: An Exploration of Scientific Gaming*. Boston: Allyn and Bacon Inc.

Remus, H., & Zilles, S. (1979). Prediction and Management of Program Quality. *Proceedings of The 4th International Conference on Software Engineering*. Munich, Germany.

Remus, W. (1986). Graduate Students as Surrogates for Managers in Experiments on Business Decision Making. *Journal of Business Research*. *14*(1), 19–25.

Rodriguez, D., Sicilia, M. A., Cuadrado-Gallego, J. J., & Pfahl, D. (2006). e-Learning in Project Management Using Simulation Models: A Case Study Based on the Replication of an Experiment. *Education, IEEE Transactions on*. 49(4), 451-463. http://dx.doi.org/10.1109/TE.2006.882367

Rusu, A., Russell, R., Robinson, J., & Rusu, A. (2010). Learning Software Engineering Basic Concepts Using a Five-Phase Game. *Proceedings of The 40th ASEE/IEEE Frontiers in Education Conference*. October 27-30, 2010 Washington, DC. http://dx.doi.org/10.1109/FIE.2010.5673327

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 Worlds Best Companies Simulate to Innovate*. Harvard Business School Press.

Sharp, H., & Hall, P. (2000). An Interactive Multimedia Software House Simulation for Postgraduate Software Engineers. *Proceedings of The 22nd International conference on Software engineering* Limerick. Ireland. http://dx.doi.org/10.1145/337180.337528

Shaw, K., & Dermoudy, J. (2005). Engendering an Empathy for Software Engineering. *Proceedings of The 7th Australasian Conference on Computing Education*. Newcastle, New South Wales, Australia.

Simon, H. A. (1957). *Models of Man Social and Rational: Mathematical Essays on Rational Human Behavior in a Social Setting*. New York: John Wiley & Sons.

Suits, B. (1967). What is a Game?. *Philosophy of Science*. *34*(2), 148–156.

Taran, G. (2007). Using Games in Software Engineering Education to Teach Risk Management. *Proceedings of The Software Engineering Education & Training*. 3-5 July 2007. http://dx.doi.org/10.1109/CSEET.2007.54.

Tvedt, J. D. (1996). *An Extensible Model for Evaluating the Impact of Process Improvements on Software Development Cycle Time.* Unpublished Unpublished Ph.D. dissertation, Arizona State University, Phoenix, Arizona.

Variale, T., Rosetta, B., Steffen, M., Rubin, H., & Yourdon, E. (1994). Modeling the Maintenance Process. *American Programmer*. 7(3), 29–37

Wang, A. I., Fsdahl, T., & Morch-Storstein, O. K. (2008). An Evaluation of a Mobile Game Concept for Lectures. *Proceedings of The 21st Conference on Software Engineering Education and Training*. 14-17 April 2008. http://dx.doi.org/10.1109/CSEET.2008.15

Wang, T., & Zhu, Q. (2009). A Software Engineering Education Game in a 3-D Online Virtual Environment. *Proceedings of The First International Workshop on Education Technology and Computer Science*. 7-8 March, 2009 Wuhan, Hubei, China. http://doi.ieeecomputersociety.org/10.1109/ETCS.2009.418

Watson, H. J., & Blackstone, J. H. (1989). *Computer Simulation* (2nd edition ed.). New York: John Wiley & Sons.

Ye, E., Chang, L., & Polack-Wahl, J. A. (2007). Enhancing Software Engineering Education Using Teaching Aids in 3-D Online Virtual Worlds. *Proceedings of The Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports*. 10-13 Oct. 2007. http://dx.doi.org/10.1109/FIE.2007.4417884

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

Zapata, C. M., & Awad-Aubad, G. (2007). Requirements Game: Teaching Software Project Management. *CLEI Electronic Journal*. 10(1).

Zhu, Q., Wang, T., & Tan, S. (2007). Adapting Game Technology to Support Software Engineering Process Teaching: From SimSE to MO-SEProcess. *Proceedings of The Third International Conference on Natural Computation*. http://dx.doi.org/10.1109/ICNC.2007.159

Table 1. Full data extract of games used in software engineering education

ID	Study	Description	Experimental Design	Sample Size (if known) and Population	Data Collection Tool	SWEBOK Knowledge Area	Bloom Learning Outcome	Observed Learning Outcomes
GS-01	University/Industry Collaboration in Developing a Simulation Based Software Project Management Training Course. (Collofello, 2000).	A single-player game, based on a system dynamics model with an iThink user interface that models a software project. Players attempt different management exercises (risk management, life cycle model comparison, critical path scheduling, etc.) that follow the lecture material.	Non-experimental	16 students	Questionnaire	Software engineering management Software engineering process	Knowledge	Learning was not assessed.
GS-02	[1] Quantitative Modeling for the Interactive Simulation of Software Project (A. Drappa & Ludewig, 1999) [2] Simulation in Software Engineering (Anke Drappa & Ludewig, 2000)	SESAM (Software Engineering Simulation by Animated Models) is a model of a software project. Users run the model loaded with its initial project state and then tweak it to simulate different scenarios before running it again. Players take the role of a project manager and must plan and control a simulated project. Rather than a graphical user interface, players control the game by typing commands in a modelling language. Players analyse their performance through an after-game analysis tool.	[1] Non-experimental [2] True Experimental	[1] 10 undergraduate project management students [2] 19 second-year computer science students	[1] n/a [2] Pre- and post-game tests Project plan	Software engineering management	Knowledge	[1] A qualitative assessment that the players experienced something similar to a real project, including panic when the deadlines were approaching. [2] Students in the experimental and control groups improved their performance in successive game sessions.
GS-03	An Interactive Multimedia Software House Simulation for Postgraduate Software Engineers (Sharp	Case studies are presented through a simulated office environment and then completed outside of the game environment.	Non-experimental	Post-graduate distance education software engineering students.	Questionnaire	Software requirements Software design Software construction Software	Knowledge	Learning was not assessed.

	& Hall, 2000)					testing		
GS-04	How to Successfully Use Software Project Simulation for Educating Software Project Managers (Mandl-Striegnitz, 2001)	Participants play two sessions of SESAM (GS-02) and their tutor analyzed their performance and provided feedback in between.	Non-experimental	40 undergraduate software engineering students	Questionnaire	Software engineering management	Knowledge	Players improved their performance in the second session but still had problems monitoring their project and tracking progress.
GS-05	An Experiment for Evaluating the Effectiveness of Using a System Dynamics Simulation Model in Software Project Management Education (D. Pfahl, Koval, & Ruhe, 2001)	A three-phase (design, implementation, test) waterfall project modeled using System Dynamics. Key project variables were project duration, effort consumption, product size, and quality after testing. Participants were separated in two groups: one group managed a simulated software project with the aid of a System Dynamics model (Abdel-Hamid, 1989); the other group used COCOMO (Boehm, Abts, Brown, Chulani, Clark, Horowitz, Madachy, Reifer & Steece, 2000).	True Experimental	12 post-graduate software engineering students	Pre- and post-test questionnaires	Software engineering management	Knowledge	Pre- and post-session surveys indicated that participants were improving their knowledge of project management patterns and behaviors. Those using the simulation models performed better that those using COCOMO.
GS-06	An Externally Replicated Experiment for Evaluating the Learning Effectiveness of Using Simulations in Software Project Management Education (Dietmar Pfahl, Laitenberger, Dorsch, & Ruhe,	Same as for GS-05.	True Experimental	[1] 12 graduate and post-graduate students majoring in computer science. [2] 13 senior under-graduate students majoring in computer	Pre- and post-test questionnaires	Software engineering management	Knowledge	The results confirmed the initial findings in which students using the System Dynamics model generally performed better in the pre- and post-test questionnaires than those using COCOMO.

	2003) Evaluating the Learning Effectiveness of Using Simulations in Software Project Management Education: Results From a Twice Replicated Experiment (Dietmar Pfahl, Laitenberger, Ruhe, Dorsch, & Krivobokova, 2004)			science, electrical engineering, and computer engineering.				
GS-07	Problems and Programmers: An Educational Software Engineering Card Game (Baker, Navarro, & van der, 2003) An Experimental Card Game for Teaching Software Engineering Processes (Baker, Oh Navarro, & van der Hoek, 2005) Teaching Software	A competitive card game called Problems and Programmers in which students play the role of project manager in a waterfall project. All players lead the same project. Players must balance several competing concerns including budget and the client's demands regarding the reliability of the final software. Who finishes first, wins.	Non-experimental	28 undergraduate students who had completed an introductory software engineering unit	Questionnaire	Software engineering managementSoftware engineering process	Knowledge	Players self-assessed their level of learning in a post-game survey. Most said the game was not good at teaching new knowledge or reinforcing existing knowledge.

	Engineering Using Simulation Games (Navarro, Baker, & van der Hoek, 2004)							
GS-08	Engendering an Empathy for Software Engineering (Shaw & Dermoudy, 2005)	Players act as a project manager to deliver a product within time and budget constraints. SimjavaSP uses discrete-event simulation as the game engine.	Non-experimental	Undergraduate software engineering students	Post-test questionnaire	Software engineering management	Knowledge	The degree of learning was self-assessed by the participants and was found to be positive.
GS-09	Model-Driven Game Development: Experience ad Model Enhancements in Software Project Management Education (Barros, Dantas, Veronese, & Werner, 2006) A Simulation-Based Game for Project Management Experiential Learning (Dantas, Barros, & Werner, 2004)	Uses simulation to support decision-making on software project management. In the game, The Incredible Manager, the player sets project parameters such as staffing and work hours and executes the project for a period of time. The simulation can be stopped so the parameters can be tweaked.	Non-experimental	7 post-graduate students in a software project management course, 8 undergraduate and post-graduate students from a software development laboratory, 9 other undergraduates.	Questionnaire	Software engineering management	Knowledge	Players self-assessed their level of learning in a post-game survey. Most said they had learned something new but only one person completed their project successfully.
GS-10	SimVBSE: Developing a Game for Value-Based Software Engineering (Jain & Boehm, 2006)	Focused on value-based software project management: every requirement, use case, object, test case and defect is treated as equally important; earned value is used to track project cost and schedule; a separation of	n/a	n/a	n/a	Software engineering management	Knowledge	n/a

		concerns is practiced, in which the responsibility of software engineers is confined to turning software requirements into verified code. The player's avatar visits different game rooms and collects information from stakeholders about the current project and how to proceed.						
GS-11	SimSE: A Software Engineering Simulation Environment for Software Process Education (Navarro, 2006).	Same as for GS-14.	True Experimental	19 under-graduate software engineering students	Pre- and post-test questionnaires	Software engineering management Software engineering process	Knowledge	All groups improved their knowledge, but those in the control groups outperformed those who had played SimSE in the post-test. When players play SimSE for longer periods, their scores improved. But, many dropped out due to boredom or frustration before this point.
GS-12	e-Learning in Project Management Using Simulation Models: A Case Study Based on the Replication of an Experiment (Rodriguez, Sicilia, Cuadrado-Gallego, & Pfahl, 2006)	A replication of the GS-05	True Experimental	11 second-year undergraduate students taking a software engineering module	Pre- and post-test questionnaires	Software engineering management	Knowledge	According to the post-test and qualitative results, students using the simulation appear to have understood the software engineering principles it was trying to teach better than those in the control group
GS-13	Using Games in Software Engineering Education to Teach Risk	A competitive board/card game that focuses on risk management. Players take the role of project manager and have to develop a product and sell it in the market.	Non-experimental	150 on-campus and distance students.	5-question questionnaire	Software engineering management	Knowledge	Players said they understood the learning objectives of the game. The degree of learning was not assessed.

	Management (Taran, 2007)	The player with most money at the end wins. A dice is used to simulate eventuated risk events.						
GS-14	Towards Game-Based Simulation as a Method of Teaching Software Engineering (Oh & van der Hoek, 2002) Design and Evaluation of an Educational Software Process Simulation Environment and Associated Model (Navarro & van der Hoek, 2005) SimSE: A Software Engineering Simulation Environment for Software Process Education (Navarro, 2006) Comprehensive Evaluation of an Educational Software Engineering Simulation Environment (Navarro & van	A single-player game for multiple development methodologies (waterfall, RUP, rapid prototyping) in which the player takes the role of a project manager leading a team of developers. The team must complete a virtual software project by hiring staff, assigning tasks, monitoring progress, purchasing resources. At the end of the game the player receives a score and can analyse their results with an explanatory tool.	Non-experimental	29 under-graduate software engineering students	Post-test questionnaires	Software engineering management Software engineering process	Knowledge	Players felt the game reinforced what they already knew but provided little new knowledge. Players are demonstrating aspects of learning theories such as learning by doing, situated learning, discovery learning, learning through failure, and Keller's ARCS. SimSE is most effective when used with other teaching methods.

	der Hoek, 2007)							
GS-15	Enhancing Software Engineering Education Using Teaching Aids in 3-D Online Virtual Worlds (Ye, Chang, & Polack-Wahl, 2007)	Two exercises were performed in Second Life, an online virtual environment.[1] Groupthink exercise: groups of students are given a software specification and must reach a design consensus. Afterwards, individuals are asked questions about the specification and points are awarded for correct answers.[2] SimSE exercise: the game from GS-14 was modified to run in Second Life.	Non-experimental	[1] 29 undergraduate students [2] 26 undergraduate students	Questionnaire	Software engineering processSoftware requirements Software engineering management	Comprehension	Most students said the exercises helped them understand the fundamentals of software specification activities and the principles of software development processes.
GS-16	Requirements Game: Teaching Software Project Management (Zapata & Awad-Aubad, 2007)	Teams of 4 or 5 players take on roles such as project manager, developers, designers, or analysts. For a given case-study, the players must produce documentation such as an ER diagram, sketches of at least 3 GUIs, and an estimation of the effort required, and then build the application in, say, Microsoft Access. A facilitator plays the role of a client giving more instructions or clarifications. Fines may be imposed for time or budget over-runs.	Non-experimental	47 systems engineering undergraduate students. 8 systems engineering Masters students. 30 systems, industrial, and administrative engineering undergraduate students.	Performance in the game alone	Software requirements	Knowledge	Not assessed
GS-17	A Game for Taking Requirements Engineering More Seriously (Knauss, Schneider, & Stapel, 2008)	A web-based game that can be completed in about 10 minutes. Software requirements are visualized as a bag of balls that flow from the customer to an analyst, a designer, and a developer depending on the development process chosen. Alternate flows may be taken (such as the client speaking	n/a	n/a	n/a	Software requirements	Knowledge	Not assessed

		directly to the developers to clear up misunderstandings), which can change the rate of flow.						
GS-18	On the Role of Learning Theories in Furthering Software Engineering Education (Navarro & van der Hoek, 2008)	Same as for GS-14.	Quasi-experimental	11 under-graduate students who had passed an introductory software engineering course.	Observation and post-test interview	Software engineering management Software engineering process	Knowledge	Players demonstrated aspects of learning theories such as learning by doing, situated learning, elaboration, discovery learning, learning through failure, Keller's ARCS, and learning by reflection.
GS-19	An Evaluation of a Mobile Game Concept for Lectures (A. I. Wang, Fsdahl, & Morch-Storstein, 2008)	The lecturer acts as a game show host and students answer multiple choice questions about a particular software design issue through their laptop or mobile phone. Players have to answer correctly to get to the next round. The winner is the last person standing.	Non-experimental	20 software engineering Masters students.	Questionnaire Performance in the game	Software design	Knowledge	Players felt the system made them pay closer attention during the lecture and that they learned more than through a traditional lecture.
GS-20	Multi-Site Evaluation of SimSE (Navarro & van der Hoek, 2009)	Same as for GS-14. SimSE was run in game sessions in which the original game designers were not directly involved.	True Experimental	Site 1: 14 students in a senior research seminar course, most of whom had passed a software engineering course. Site 2: 19 under-graduate software engineering students. Site 3: 48 under-	Post-test questionnaire s, performance in SimSE, and final course grades.	Software engineering managementS oftware engineering process	Knowledge	Students seemed to learned the concepts the game is designed to teach.The game was suitable for students of varying abilities and backgrounds.SimSE is most effective when used with other teaching methods.

				graduate software engineering students.				
GS-21	Empirical Evaluation of an Educational Game on Software Measurement (Gresse von Wangenheim, Thiry, & Kochanski, 2009)	In X-MED, the player takes the role of a measurement analyst and defines and executes a measurement exercise based on a given development scenario. A score is calculated based on the number of correct decisions made, and the player is presented with an analysis of their performance.	True Experimental	15 computer science post-graduate students	Pre- and posttest questionnaires	Software engineering management Software engineering process	Knowledge	The results don't conclusively point to a positive learning effect, although most players' subjective evaluation was that the game helped them understand the topic.
GS-22	Adapting Game Technology to Support Software Engineering Process Teaching: From SimSE to MO-SEProcess (Zhu, Wang, & Tan, 2007) A Software Engineering Education Game in a 3-D Online Virtual Environment (T. Wang & Zhu, 2009)	A game based on SimSE (GS-14) using the rapid prototyping profile and deployed to Second Life.	Non-experimental	52 software engineering students	A six-question post-test questionnaire.	Software engineering process	Knowledge	Players self-assessed their level of learning in a post-game survey. Most said the game had helped them understand the software development process better.
GS-23	PlayScrum- A Card Game to Learn the Scrum Agile Method (Fernandes & Sousa, 2010)	Focused on the Scrum (Schwaber, 2004) agile development process. Further development of Problems and Programmers (Baker et al., 2005). Played by 2 to 5 people. Cards are used to represent tasks,	Non-experimental	13 post-graduate students.	Questionnaire	Software engineering management Software engineering process	Knowledge	Students improved their performance in successive game sessions. Players analyze their performance through an after-game analysis tool

		problems, developers, and artifacts. The winner is the person who performs all tasks with the least number of errors. A roll of a dice determines the flow of the game.						
GS-24	Evaluation of a Game to Teach Requirements Collection and Analysis in Software Engineering at Tertiary Level (Hainey, Connelly, Stansfield, & Boyle, 2010) An Application of Games-Based Learning Within Software Engineering (Connolly, Stansfield, & Hainey, 2007)	Players take on specific roles (project manager, systems analyst, systems designer, team leader). The systems analyst moves their avatar through the game world to collect requirements by asking questions of game characters. When the analyst thinks they have all requirements, they prepare a requirements document and send it to the project manager, who must decide whether to proceed with the project.	True Experimental	55 university students and 37 higher-education students (92 in total). The majority had little or no instruction in requirements collection or analysis.	Pre- and post-test questionnaires	Software requirements	Knowledge	Comparison of pre- and post-game test scores showed an increase in knowledge. Control groups who did not play the game also showed in increased in knowledge. The game was found to be a good supplement to existing courses. Higher education students gained more from the game (better post-game scores) and were more accepting of the teaching technique than further education students.
GS-25	Learning Software Engineering Basic Concepts Using a Five-Phase Game (Rusu, Russell, Robinson, & Rusu, 2010)	Players take the role of a requirements engineer in a waterfall development (requirements, design, implementation, testing, maintenance phases) software project. The player's avatar must ask questions of on-screen characters to determine the right requirements. Subsequent phases use arcade-style graphics to kill 'computer bugs' or to 'shoot'	Non-experimental	Developed by teams of under-graduate software engineering students and used by a class of nine middle and high school students with	Pre- and post-test questionnaires	Software engineering management	Knowledge	Comparing pre- and post-game surveys most participants said they gained a better understanding of software development.

		answers in a multiple choice quiz.		limited or no computer science background.				
GS-26	A Classroom Game for Teaching Management of Software Companies (Zapata, 2010)	Players take turns in rolling a dice and answering a technical question about software development. If the answer is right, the player's team has the chance to solve a project estimation problem. The team with the most correct responses to the questions and estimation problems wins.	Non-experimental	40 systems engineering students	Post-test questionnaire	Software requirements	Knowledge	Players self-assessed their level of learning in a post-game survey. Most said they had learned something new.

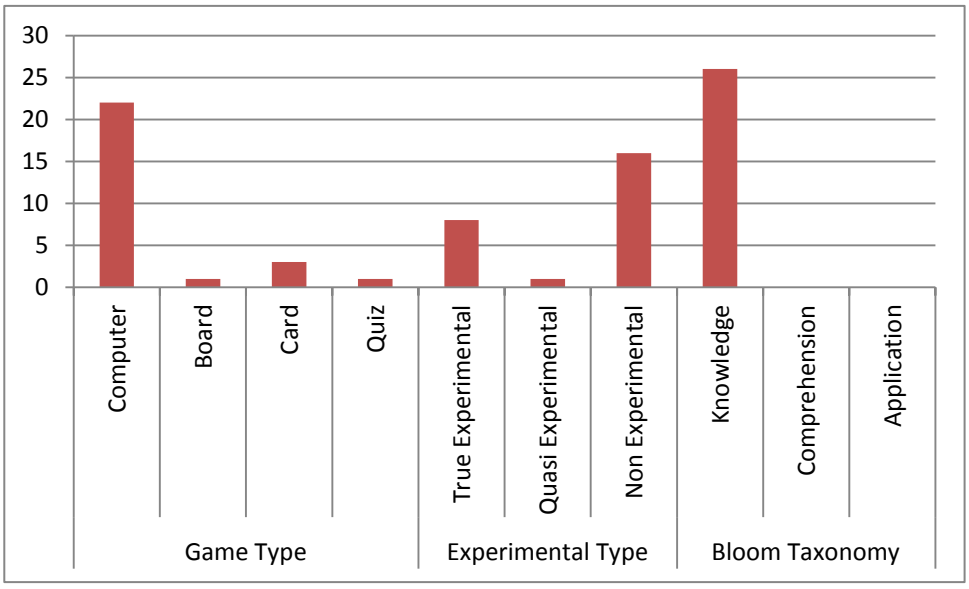

Figure 1. Game surveys classified by game type, experimental type, and Bloom taxonomy.

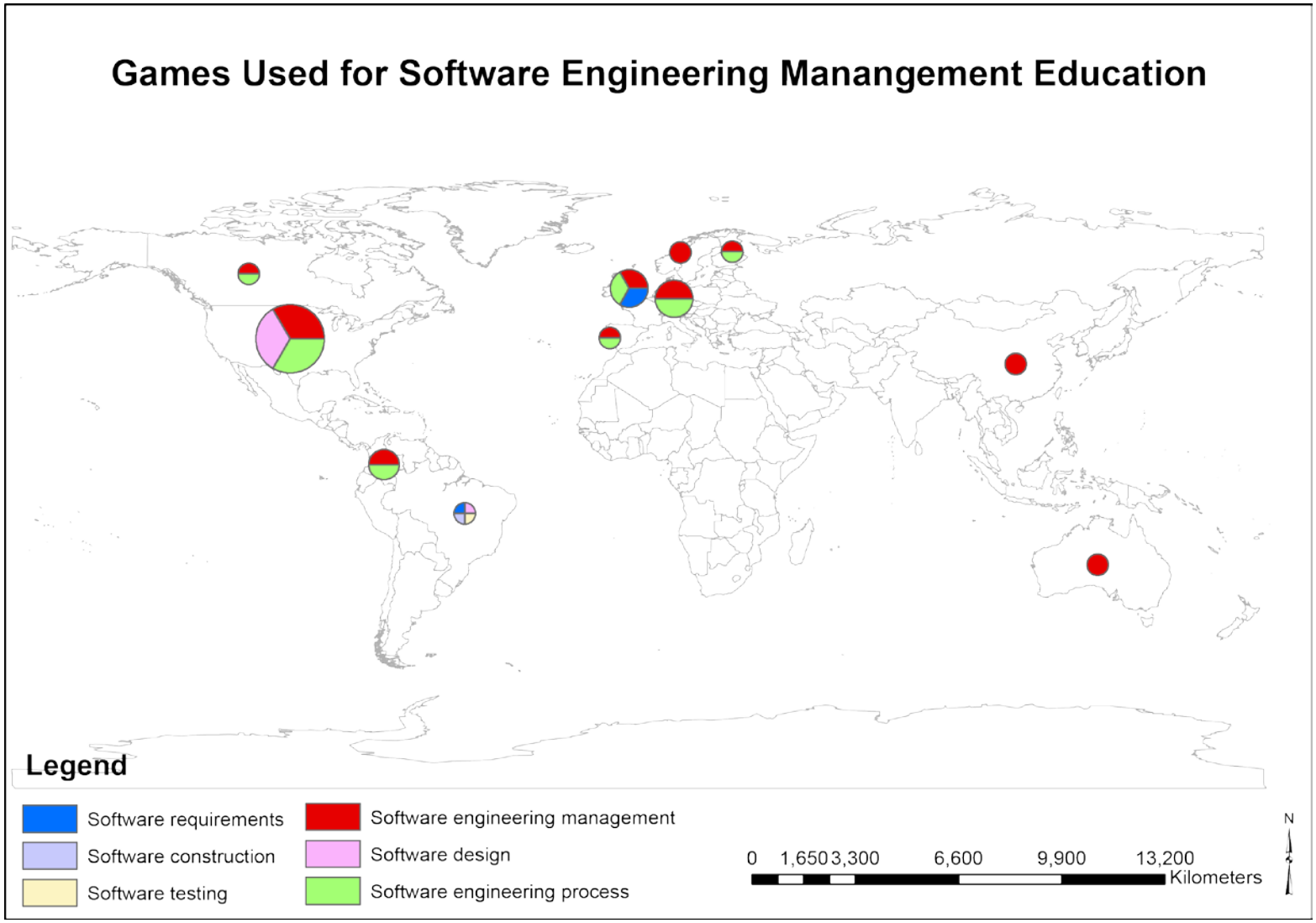

Figure 2. Games used for software engineering education by location and SWEBOK knowledge area