122nd ASEE Annual Conference & Exposition
June 14 - 17, 2015
Seattle, WA
Making Value for Society

Paper ID #12938

Game Design and Development Capstone Project Assessment Using Scrum

John Glossner, Daniel Webster College

Dr. John Glossner is Associate Professor of Computer Science at Daniel Webster College. He also serves as CEO of Optimum Semiconductor Technologies. Prior to joining OST John co-founded Sandbridge Technologies and served as EVP & CTO. Prior to Sandbridge, John managed both technical and business activities in DSP and Broadband Communications at IBM and Lucent/Starcore. John was also an adjunct professor at Lehigh University. John received a Ph.D. in Electrical and Computer Engineering from TU Delft in the Netherlands, M.S degrees in E.E. and Eng. Mgt from NTU, and a B.S.E.E. degree from Penn State. He has more than 120 publications and 36 issued patents.

Prof. Nicholas Bertozzi, Daniel Webster College

Nick Bertozzi is a Professor of Engineering at Daniel Webster College (DWC) and Dean of the School of Engineering and Computer Science (SECS). His major interest over the past 18 years has been the concurrent engineering design process, an interest that was fanned into flame by attending an NSF faculty development workshop in 1996 led by Ron Barr and Davor Juricic. Nick has a particular interest in helping engineering students develop good communications skills and has made this a SECS priority. Over the past ten years he and other engineering and humanities faculty colleagues have mentored a number of undergraduate student teams who have co-authored and presented papers and posters at Engineering Design Graphics Division (EDGD) and other ASEE, CDIO (www.cdio.org), and American Institute of Aeronautics and Astronautics (AIAA) meetings as well. Nick was delighted to serve as the EDGD program chair for the 2008 ASEE Summer Conference and as program co-chair with Kathy Holliday-Darr for the 68th EDGD Midyear meeting at WPI in October 2013. Nick is currently serving as the Vice Chair of the ASEE EDGD.

Prof. Charles N Stevenson P.E., Daniel Webster College

Game Design and Development Capstone Project Assessment Using Scrum

Abstract

This paper discusses the transition of a senior game design and development course project to a scrum-based methodology from the traditional waterfall process. The paper explains the scrum-based tools used to facilitate the project and observations on the transition. Examples are given of student documentation and the methodology used to generate them. The prototype game will be demonstrated at the presentation. In game design and development, by always producing an iterated playable game, students are able to focus on "finding the fun" in the game. Using the Scrum methodology all course outcomes were achieved.

Introduction

Our Game Design and Development (GDD) program and the complementary Game Programming (GP) program are housed in the School of Engineering and Computer Science (SECS). Game design students focus on art, game design, storyboards, and the development of games. They also take entry-level programming and computer science courses. Game programmers focus on programming, computer science and some game design aspects. The senior year capstone project brings together both GDD and GP students, blending both artists and programmers with the goal of producing a fully working game. Students are grouped into cross-functional teams of nine to ten students and work together for two semesters.

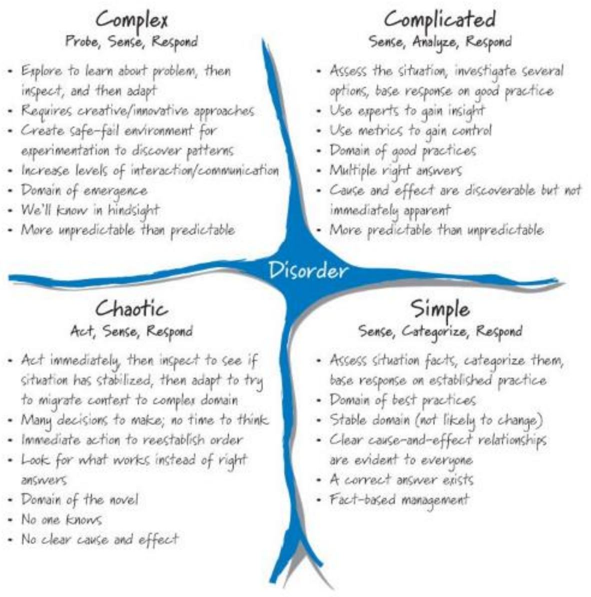

Complex
Probe, Sense, Respond
• Explore to learn about problem, then inspect, and then adapt
• Requires creative/innovative approaches
• Create safe-fail environment for experimentation to discover patterns
• Increase levels of interaction/communication
• Domain of emergence
• We'll know in hindsight
• More unpredictable than predictable
Complicated
Sense, Analyze, Respond
• Assess the situation, investigate several options, base response on good practice
• Use experts to gain insight
• Use metrics to gain control
• Domain of good practices
• Multiple right answers
• Cause and effect are discoverable but not immediately apparent
• More predictable than unpredictable
Disorder
Chaotic
Act, Sense, Respond
• Act immediately, then inspect to see if situation has stabilized, then adapt to try to migrate context to complex domain
• Many decisions to make; no time to think
• Immediate action to reestablish order
• Look for what works instead of right answers
• Domain of the novel
• No one knows
• No clear cause and effect
Simple
Sense, Categorize, Respond
• Assess situation facts, categorize them, base response on established practice
• Domain of best practices
• Stable domain (not likely to change)
• Clear cause-and-effect relationships are evident to everyone
• A correct answer exists
• Fact-based management

Figure 1. Cynefin Complex Domain (used with permission *Cognitive Edge Pte Ltd http://cognitive-edge.com/* as published in *Essential Scrum: A Practical Guide to the Most Popular Agile Process*[1])

Many projects can be partitioned into domains according to their complexity and uniqueness. The Cynefin framework partitions projects into five domains based on their complexity: Simple, Complicated, Complex, Chaotic, and Disorder. In the Simple domain the correct solution is obvious and generally undisputed. Best practices and continuous business process improvement methodologies tend to work well in this domain. In the Complicated domain there may be multiple correct answers that typically require experts in the field to determine the best approach. In the Complex domain a solution can typically only be determined with hindsight. There are often more unknowns than knowns, and finding a solution requires invention. This is a perfect domain in which to use Scrum. In the Chaotic domain crisis intervention is required. This may come about from a program producing errors or an algorithm being incorrect. In the Disorder domain it is impossible to identify any particular domain in which to classify a problem. Any project in this domain has a goal of partitioning the problem such that the subdivided problem will fall into any of the other four domains. In summary, Scrum is well suited to complicated problems with many unknowns. This includes new engineering product development, software development of new products, and particularly large student-led projects.

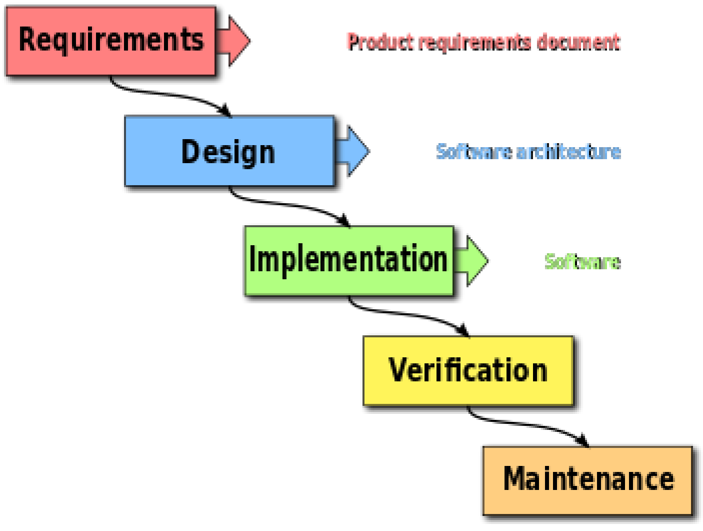


Figure 2. Waterfall Development Process[2]

Waterfall method

Historically our capstone project has been taught using a traditional waterfall software engineering process. As shown in Figure 2, a requirements document is first generated and is the contract between the customer and the design team. Once this document is approved, the Design phase is started. A comprehensive design document is produced that details every item of what will be developed. In the context of the game design project, this includes completion of the Baldwin game design document[3] including:

1) An overview of the game including concept, vision, target audience, look and feel
2) Gameplay and mechanics including game progression, structure, objectives, play flow, physics, movement, objects, actions, combat strategy, options, replays, and game cheats

3) Story, setting, and character including back story, plot, storyboard, world design, character stories, animations, and relationships between characters
4) Levels including for each level objectives, synopsis, introductory materials, map, walkthrough, encounters, critical path, and opening/closing material
5) Interface considerations including menus, rendering system, cameras, lighting, character controllers, sound effects, music, and a help system
6) Artificial intelligence (AI) including opponent AI, enemy AI, non-player character AI, player collision detection and path finding
7) Technical considerations such as platform (PC, mobile), game engine, and scripting languages
8) Game art including characters, style guides, and equipment
9) Miscellaneous including test plans and design software required

The aforementioned items are not exhaustive but meant to provide insight into the large amount of work involved – and this is just the documentation phase. After the requirements document and design document are complete, the implementation phase begins. In the context of game design, this means both the game programming and art assets are implemented. Once all the design is completed, the process moves to the verification phase during which the program and assets are validated to work as specified in the design document. Finally a maintenance phase is entered.

A key point of the waterfall process is that each subsequent stage is only entered upon the completion of the previous stage. Thus any changes made may affect all stages. For example, a project in maintenance phase that requires a change to the requirements should stop all work and regress to the requirements phase.

Scrum methodology

A scrum-based methodology has been shown to be successful in a Cynefin complex environment where there are many undefined and unknown features[1]. Using a traditional methodology may result in changes to specifications late in the process, making the integration of individual parts of the process difficult.

In Scrum, there are three primary roles. The product owner represents the customer and is responsible for prioritizing product features. The development team completes all work in time-boxed units of development called sprints. The scrummaster acts as a coach to both the development team and the product owner. The scrummaster also facilitates removing obstacles impeding the progress of the development team.

A key feature of scrum is a focus on iterative development with the output of each sprint being a potentially shippable product. Game features are described in terms of user stories and estimated using either T-shirt sizes (small, medium, large, extra-large) or story points (1, 2, 3, 5, 8, 13, 21, 50, 100). User story sizing is meant to be relative to other user stories. A numerical value of 2 story points would roughly represent an easy-to-implement user story that is twice as difficult as a numerical value of 1. A key point is that the development team must come to a unanimous vote on the number of story points required for a user story. While there are different estimates the team is encouraged to discuss why their votes are different. Re-votes are taken followed by more discussion. The process repeats until everyone agrees.

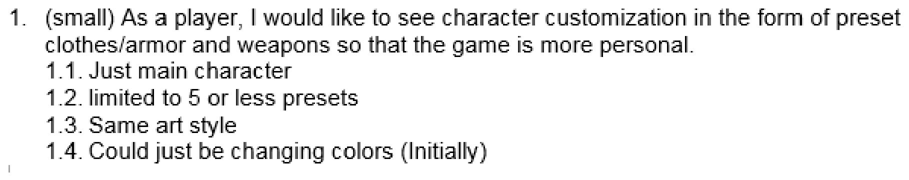
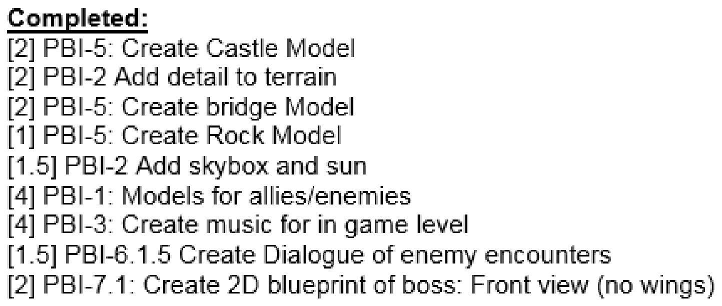

1. (small) As a player, I would like to see character customization in the form of preset clothes/armor and weapons so that the game is more personal.
    1.1. Just main character
    1.2. limited to 5 or less presets
    1.3. Same art style
    1.4. Could just be changing colors (Initially)

Figure 3. Example User Story with Conditions of Satisfiability

Story points are not intended to be directly correlated with time, although some organizations using scrum do estimate user stories with time. The list of all user stories is called the product backlog. Each user story in the product backlog is called a Product Backlog Item (PBI).

Figure 3 shows an example user story with conditions of satisfiability meant to clarify the scope of the user story. The user stories are prioritized by the product manager in a process called product backlog grooming. Stakeholders and the development team members are encouraged to participate in the prioritization, but the product manager has final authority for the prioritization.


Completed:

[2] PBI-5: Create Castle Model
[2] PBI-2 Add detail to terrain
[2] PBI-5: Create bridge Model
[1] PBI-5: Create Rock Model
[1.5] PBI-2 Add skybox and sun
[4] PBI-1: Models for allies/enemies
[4] PBI-3: Create music for in game level
[1.5] PBI-6.1.5 Create Dialogue of enemy encounters
[2] PBI-7.1: Create 2D blueprint of boss: Front view (no wings)

Figure 4. Sprint Tasks

Before the sprint begins, the development team performs sprint planning. Sprints are a fixed time duration, usually between two and four weeks where all of the work is implemented. The development team decides how many user stories can be completed in a single sprint. Sprint planning transforms the user stories into a set of sprint tasks. Unlike user stories that represent customer features, sprint tasks represent the tasks the development team must perform to implement the user stories. Figure 4 shows an example of sprint tasks at a snapshot of a sprint. The sprint tasks are estimated in ideal hours shown in the brackets. Ideal hours are the uninterrupted amount of time estimated to do the task. Just as with user stories, the development team must come to a unanimous estimate on ideal hours, or they discuss their reasons for differences and vote again. A link to the original user story is denoted by the PBI label. The prioritized list of sprint tasks is called the sprint task backlog. Each sprint task within the sprint task backlog is called a sprint task backlog item.

Once sprint planning is complete, the development work can begin. Sprint tasks are assigned to development team members in user story priority with some allowance for the capabilities of individual participants. On each day from the second day until the end of the sprint, the development team meets for a time-boxed meeting called the daily scrum. To facilitate a short meeting, all participants are required to stand up in a circle.

During the daily scrum each development team member, in no specific order, answers three questions for the team: 1) what did I do since yesterday? 2) What I am going to do today? And 3) is anything impeding my progress? It is important that the team members talk to each other and not to the scrummaster or product owner. The product owner is encouraged to attend but is not required. The daily scrum should last no more than fifteen minutes. Complicated discussions can be side-lined and resumed after the daily scrum by those affected. If any development team member has impediments, those become the responsibility of the scrummaster to resolve.

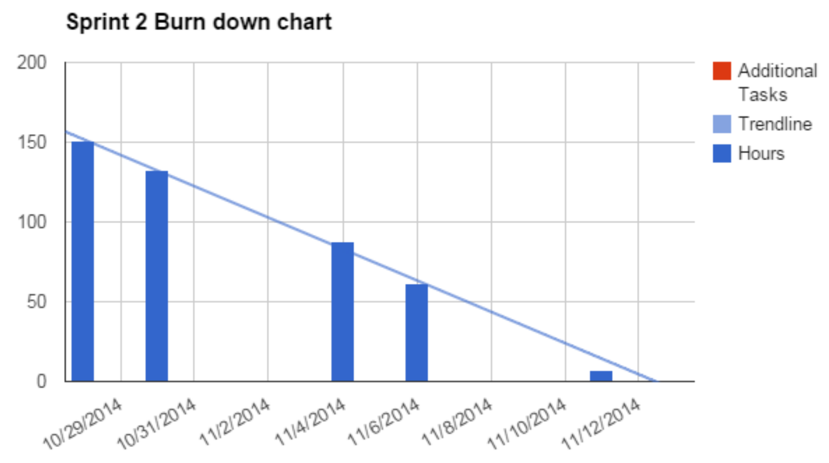


Figure 5. Sprint Burn Down Chart

A key metric discussed at each daily sprint is the team's velocity. During the sprint this is the number of ideal hours completed toward the sprint goal divided by time. This is captured in a burn down chart. Figure 5 shows an example burn down chart from a successful sprint. The vertical axis measures ideal hours for the total sprint. The horizontal axis is the date of the daily scrum meeting where the remaining sprint task hours are measured. Typically the burn down chart measures sprint task hours. It is also possible to measure the "burn down" of user story points. The observation that this is not recorded "daily" is discussed in the section scrum as used in the game design project.

As each sprint task is completed, the development team decides how to prioritize and assign sprint tasks with the understanding that higher priority user stories are worked on first. Should the development team run out of sprint tasks before the end of the sprint, then additional user stories may be added. Should the development team have incomplete or partially completed user stories, the entire user story is returned to the product backlog list.

At the end of the sprint, two meetings are held. The first is the sprint review. At the sprint review the entire team attends, including the development team, scrummaster, product owner, stakeholders, and customers. A development team member reviews what user stories went into the sprint and what user stories were completed. A demonstration of the potentially shippable product is given. Customer feedback is then received for consideration in future sprints.

After the sprint review meeting, the entire scrum team holds a sprint retrospective meeting. The scrummaster, development team, and product owner all attend. The purpose is to determine what went well in the sprint and what needs improvement from a development process perspective. Generally the team will select one or two observations to improve in the next sprint.

After the sprint retrospective the process begins again with product backlog grooming.

Scrum as used in the game design project

The two-semester game design and development class meets twice a week for 110 minutes each class. Typically a single team of 9 to 10 students is involved. The objective of the course is to develop a portfolio that includes game concepts and art as well as to expand experience in teamwork, game engine skills, and software engineering skills.

At our school we desire to give students ownership of their projects, including making decisions about the type of projects they work on. In the game design and development project, this means they can choose the style of game (RPG, FPS, etc.), target platforms (XBOX, PC, etc.) as well as character stories and plots. Since this is all unknown at the start of the project, this corresponds to the Complex domain in the Cynefin classification and thus is a perfect candidate for scrum. Since scrum is being used in a university setting, some modifications have been implemented. The process is described in this section.

When first transitioning to scrum, a few classes are used to teach the scrum methodology while the students begin to brainstorm a vision for their game. In both Computer Science (CS) projects and GDD projects, multiple competing visions often emerge. In this case the students are asked to vote. If no clear choice emerges, discussion is invited before a re-vote. Gradually the options are eliminated until one project is chosen. It has been observed that the vision statements may be combined or characteristics of one game vision adopted into a revised vision statement prior to re-voting.

The students function primarily as the development team in the scrum methodology. The instructor facilitates the roles of product owner and scrummaster. Students are involved in some of the product owner roles such as writing user stories, product backlog grooming, etc. While ultimately the instructor owns the product backlog, input from the students should be encouraged.

An initial exploratory sprint lasting about a week is run to familiarize students with the process. Prior to beginning the exploratory sprint, the students spend one or two class periods creating user stories. They are encouraged to put themselves in the perspective of the game player. When estimating user stories in a classroom environment, we prefer t-shirt sizes so that there is no confusion that the estimates may correlate to time.

The exploratory sprint “dry run” is not graded and serves a dual purpose of exposure to the scrum process and a dynamic exploration of the game design proposal. In one particular case, students were allowed to "fail fast" whereby during the exploratory sprint they realized their initial game concept could not be implemented within the two-semester sequence. The entire team agreed to select the second place vision statement as their game project. Using traditional software engineering methodologies may not have allowed the students sufficient time change the game proposal. We discuss one such case in the results section. Using Scrum and the exploratory Sprint, they were able to choose a new game concept without academic penalty. It also helps students learn that their initial design idea might not always be the best choice.

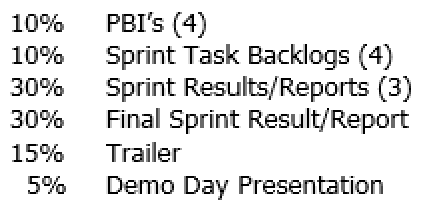
With a vision statement unanimously agreed upon, the classroom structure becomes: 1) write, prioritize, and estimate user stories, 2) convert user stories to prioritized, estimated sprint tasks, 3) sprint to complete development work, 4) hold a sprint review meeting, and 5) hold a sprint retrospective meeting.

Writing user stories and sprint planning each take one to two hours of class time, primarily in the beginning of the project and less after students are familiar. Google documents have been found to be indispensable for this process, as it allows concurrent editing of the document. Since there is no actual customer for the project, we hold the sprint review meeting and sprint retrospective meeting in a single 110-minute class period, but the instructor specifically announces the end of the review and transition to the retrospective.

At the beginning of the first semester, we use about 30 minutes of class time during every class to go over the scrum methodology. We suspend these lectures during sprints and pick them up in between each sprint. By the time students complete the exploratory sprint, they have enough knowledge to enter their first graded sprint.

Because the class meets twice a week, the "daily" scrum is held only on class days. Students are given a few minutes at the start of class to update the burn down chart, and each student answers the three questions: 1) what have I done since the last scrum meeting? 2) What am I going to do today? And 3) is anything impeding me. It is often uncomfortable for students to stand in a circle, but they will get used to it. There is also a tendency for the students to "report" to the instructor. This should be highly discouraged, and before the start of each scrum meeting, the student should be reminded that he or she is talking to the development team and not the instructor. It is sometimes helpful for the instructor to not make eye contact with the student speaking.

Occasionally an impediment will arise during a scrum meeting. If it is valid, the instructor should take an action item to resolve it. One case involved software limitations. The instructor requested that the school install software on lab machines. This was done quickly and removed the student's impediment. Often false impediments such as "not enough time" or "other class demands" may arise. The instructor (as scrummaster) should gently remind the students that these are not valid impediments.

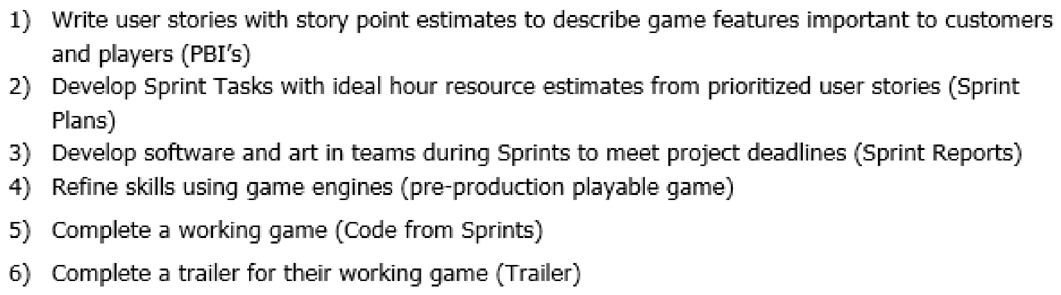


10%	PBI's (4)
10%	Sprint Task Backlogs (4)
30%	Sprint Results/Reports (3)
30%	Final Sprint Result/Report
15%	Trailer
5%	Demo Day Presentation

Figure 6. Game Design Project Course Evaluation

Course evaluation is primarily based on running three or four sprints (depending upon the semester) and the outcome of the sprints. Figure 6 shows the second semester course evaluation metrics. Four product backlog item lists containing the list of prioritized user stories are turned in, each worth 2.5% of the grade per team. Similarly, four sprint task backlog lists (one per team) are graded for a small percent of the grade. A single group grade is also given for the Trailer and demo day presentation. The Trailer is a short video advertisement for the game the students are developing. It makes up 15% of the overall grade.

The majority (60%) of the student's grade is based on the outcome of the sprint. To assess individual contributions to a sprint, we use CATME developed by Purdue University[4]. In this methodology students evaluate both themselves and their teammates. CATME is able to identify conflicts within groups as well as dysfunctional teams. Students who wish to dispute the peer rating must keep a journal of activities. Students receive a team grade that is individually adjusted according to their CATME assessed contribution. The team grade is assessed 80% by sprint results and 20% by the sprint report write-up. Even a sprint that completes all user stories may not receive full credit if the burn down chart is not linear. Thus, students who wait until the project deadline and complete all tasks with one heroic effort jeopardize the overall sprint grade.



1) Write user stories with story point estimates to describe game features important to customers and players (PBI's)
2) Develop Sprint Tasks with ideal hour resource estimates from prioritized user stories (Sprint Plans)
3) Develop software and art in teams during Sprints to meet project deadlines (Sprint Reports)
4) Refine skills using game engines (pre-production playable game)
5) Complete a working game (Code from Sprints)
6) Complete a trailer for their working game (Trailer)

Figure 7. Game Design and Development Project Course Outcomes

Observations and results

The Game Design and Development program is relatively new at our school. In the incubation period students were primarily from the GDD program, and course outcomes #4 and #5 shown in Figure 7 were not fully achieved. With the inclusion of GP students in the capstone project and the transition to a scrum-based project methodology, we have achieved all program outcomes prior to the end of the second semester. In particular, students using the scrum-based methodology have had a working, potentially shippable game since the end of their second sprint. This is a significant improvement over previous years.

Having a working prototype at an early stage also allowed students to more easily envision and write user stories. Iterating simultaneously on the game's vision statement and user stories allowed the game to evolve. At the beginning of the project, it was difficult for students to see everything they wanted to develop within the game. Allowing inspect and adapt iterations at the end of every sprint allowed them to refine their game.

As the semester progressed, having fixed time periods of work helped the students to set goals and prioritize development. While not all Sprints were as successful as that shown in Figure 5, the general capabilities of the group were better estimated for each sprint, and the user stories selected for each sprint were generally completed on time.

Related work

Scrum has been used extensively in industry – including the gaming industry[11]. Scrum has also been used with positive outcomes in academia[5,6]. Scrum was first described by Takeuchi and Nonaka for product development in automotive and printer industries[7]. It was quickly adopted for software development and shown to be effective[8]. It has been shown that 70% of projects using the waterfall process fail to meet their objectives[9]. It is also noted when the two processes are combined, the failure rate of "scrummer-fall" or "water-scrum" may also be 70%[10].

Conclusions

In the game design and development course we followed a pure scrum methodology. Using this methodology students were successful in iterating a working game that was always in a potentially shippable state. Students were able to change their game objectives and refine features over the course of the project, resulting in all expected outcomes being achieved.

Bibliography

1. Rubin, Kenneth S., Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley Professional; 1st edition (August 5, 2012), ISBN-10: 0137043295.
2. Waterfall method. http://en.wikipedia.org/wiki/Waterfall_model
3. Baldwin game design document available from http://baldwindesignconsulting.org
4. http://cateme.org
5. Grimheden, M.E., Mutual learning experiences – Mechatronics capstone course projects based on Scrum, American Society for Engineering Education, 2012.
6. Reichlmayr, T., Working towards the student scrum – Developing agile android applications, American Society for Engineering Education, 2011.
7. Takeuchi, Hirataka and Nonaka, Ikujiro, "The New New Product Development Game", Harvard Business Review, January, 1986.

8. Sutherland, Jeff and Schwaber, Ken, The SCRUM Development Process," at Object-Oriented Programming, Systems, Languages & Applications (OOPSLA) Conference '95 in Austin, Texas.
9. Serena Corporation, An Introduction to Agile Software Development, June 2007. Available at http://www.serena.com/docs/repository/solutions/intro-to-agile-devel.pdf
10. Denning, Steve, "Scrum Is A Major Management Discovery", Forbes magazine, April, 2011. Available at: http://www.forbes.com/sites/stevedenning/2011/04/29/scrum-is-a-major-management-discovery/
11. Agile Game Development with Scrum by Clinton Keith Addison-Wesley Professional; 1st edition (June 2, 2010), ISBN-10: 0321618521, ISBN-13: 978-0321618528