CREATING A CUSTOM BOARDGAME USING TABLETOP SIMULATOR TO ORIENT AND TRAIN NEW HIRES

Scott Hollenbeck

A Capstone Project (or Thesis) Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems and Operations Management

University of North Carolina Wilmington

2022

Approved by

Advisory Committee

Judith Gebauer
Committee Member

Ellie Ebrahimi
Committee Member

Jeff Cummings
Chair

Accepted By

_____
Dean, Graduate School

Table of Contents

	Page
Chapter 1: Introduction→	4
Chapter 2: Review of Literature Review and Analysis→	6
Waterfall Methodology	6
Agile Methodology	8
Scrum Framework	10
Jira Board and Ticket Types	12
Chapter 3: Methodology	16
Gamifying Scrum at nCino	16
The Rules - How the game is played	17
Web Based Applications vs Virtual Reality Training	18
The Power of Gamification	19
Game Platform	20
Chapter 4: Outline of Completed Thesis (or Project)	23
Initial Runs of the Game	23
Pilot Test Game with retired business executives	25
Second Test Run with nCino Employees	27
Creating Epics	28
Creation of Event Cards	29
Game Creation	31
Chapter 5: Conclusions and Future Work	36
References	38

Figures

1	Sample Jira Board	11
2	getKanban Gameboard	14
3	Scrum Tale	15
4	InGame Screenshots	17
5	Percentage Chance of Rolling 6 four-sided dice	21
6	Percentage Chance of Rolling 2 four-sided dice	21
7	Initial Game Run	22
8	Final Run of the Game	23
9	As Needed	43
10	As Needed	43

ABSTRACT

CREATING A CUSTOM BOARD GAME USING TABLETOP SIMULATOR TO ORIENT AND TRAIN NEW HIRES. Hollenbeck, Scott, 2022. Capstone Paper, University of North Carolina Wilmington.

Experience with the onboarding process at nCino Inc. resulted in a need for a more efficient and effective way to explain to new hires nCino's project management framework. This paper addresses the inefficiencies of the current process, and presents a more efficient way of accomplishing the task and at the same time, engaging participants to maintain their interest. The research indicates that gamifying lessons is an effective way of doing just that. However, the Tabletop Simulator game engine that was selected proved not to be as effective as hoped given the Scrum framework practices to which nCino wants to expose to its people. On an interim basis, a physical version of the game is currently used at nCino; however the conclusion reached from this project was that a new online game engine would be better suited for nCino's purposes since many new hires will be working virtually and gaming is always of great interest to a young workforce. Journey to Plan-It Scrum will be an effective tool to help nCino accomplish its objectives and the online version will only improve the experience in the future.

CHAPTER 1: INTRODUCTION

In 2012, nCino, a financial technology software company, was founded in Wilmington, NC with a dozen employees to implement cloud software solutions for mid-level banks using Salesforce as a platform. Since then, the company has grown to over 1000 employees and recently experienced the best initial public offering of any tech company in the last twenty years. The company uses Salesforce software as a platform and modifies the software from the sales driven companies for which it was designed to banking software applications.

Because of the rapidly changing nature of the banking industry, nCino had to evaluate their approach to project management in order to adapt rapidly to deliver the right products for their clients. The decision was made to adopt an agile methodology to manage projects using the Scrum framework. This framework is meant to be bent and shaped to suit the needs of the organization. Over the past eight years, nCino has continued to utilize the Scrum framework, adapting it as needed to handle their product development needs. However, the framework has been adapted uniquely to the needs of nCino which requires training approaches for employees to be adapted as well. There is a need to teach new employees efficiently and effectively in the exclusive way that work is done (e.g., nCino does bi-weekly basis) as well as the lessons learned over the last few years. To address this need, I am proposing to implement an online board game to efficiently and effectively increase that understanding of nCino's approach to the Scrum framework that is both fun and engaging. Although the intention is to create a way of on-boarding new employees, I also see this as a way of refreshing knowledge and creating friendly competition amongst current nCino teams.

In the following chapters we will review relevant literature, analyze the different agile frameworks, the methodology implemented, outline the work completed and conclude with future action items resulting from the outcomes of the experience.

CHAPTER 2: REVIEW OF LITERATURE REVIEW AND ANALYSIS

Toward the end of the twenty-first century, there was a growing frustration with the slowness and constraints associated with software development. Most companies were using a framework known as “waterfall”. The waterfall methodology originated in the 1970s and was effective for projects where the requirements could be established upfront with some certainty. Because of that success, this approach was adopted for software development projects. (Hughey, 2009)

Waterfall Methodology

Waterfall, also known as the software development life cycle (SLDC), entails breaking the process of software development into a series of phases, each phase to be completed in linear order, that is, a new phase cannot be started until the previous phase is completed. The phases of the model include (Hughey, 2009):

- **Requirement Gathering and Analysis** − focusing on the key business needs of the organization and the software that needs to be developed to accomplish those needs. The key to success is that intense and thorough planning is required at this phase to capture all project needs.
- **System Design** − determine hardware and system requirements and the overall system architecture.
- **Develop Code and Unit Test** − During this phase the system is broken down into smaller units to allow for more efficient development and testing of software code.

- **Integration and Testing** − Units are integrated and the entire system is tested for any faults and failures.
- **Deployment** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

There are advantages to using the waterfall method. Each phase of the process is specific and done in sequence, which makes it easy to understand as well as manage since the deliverables for each phase is determined in advance. With good documentation and research up front, design errors are caught early. Also, developers can be onboarded early on to work with the documentation. Because waterfall is very structured, it is also easy to determine costs, progress and delivery dates with set milestones. Lastly, verification is a smooth process because the testing scenarios have been defined in advance (Hughey, 2009). Overall, the waterfall methodology is a good option if all of the requirements of the project are understood and there is little chance of scope creep.

However, there are significant drawbacks to using the waterfall method in software development. A critical one occurs during the Requirement Gathering and Analysis phase. In this initial phase, the customer outlines what needs to be accomplished by the development team and based on that information, the requirement specification document is finalized. Because the requirements are established up front and not revisited, there is a strong possibility that the final product may be very different from what the stakeholders had hoped for originally (Balaji and Murugaiyan, 2012). This is only compounded by the fact that the customer doesn't get to experience a working software until late in the process.

Additionally, since the waterfall methodology is linear, it is not suitable in cases where requirements might have a moderate to high risk of changing later in the process.

While the intent was to keep project development under tight control, the structured environment actually can lead to time delays and cost overruns. Developers who tried to change projects in mid-stream had extreme difficulty since the total project was designed upfront with no expectation of changes. In many cases, timelines extended out for many years, or projects were eventually abandoned since it was not possible to adjust the scope of the project. It is primarily for these reasons that the waterfall methodology fell out of favor for software development projects (Balaji and Murugaiyan, 2012).

Agile Methodology

To address the difficulties of the waterfall method, 17 of the greatest minds in the software industry met in 2001 at Snowbird ski resort in the Wasatch mountains of Utah. Over a few days, they emerged with the Agile Manifesto that set out the key values and principles behind what came to be known as the agile methodology (Kent, 2001). The four main values of agile are:

- Individuals and their interactions is favored over processes and tools
- Working software is favored over comprehensive documentation
- Customer collaboration is favored over contract negotiation
- Responding to change is favored over following a plan

This document changed the way the software industry looked at development and from that moment on, agile has become the predominant methodology in software development. It is an iterative approach to software development that encourages continuous collaboration with stakeholders. (Kent, 2001)

Instead of being forced to determine all the requirements up front,, a team would produce software on a continuing basis. By breaking up a project into several development stages and involving constant collaboration with stakeholders, the stakeholders/customer expectations are constantly clarified. This allowed for continuous

improvement and iteration at every stage. It is the breaking down of work into smaller increments that allows for reviewing a project's plans and direction on a continuous basis. This continuous review allows more flexibility ensuring that the software team can adapt to unforeseen problems or scope, change direction to accommodate revised client needs and can be modified to meet the changes of an ever evolving software environment.

While there are a number of benefits to using agile approaches, there are also drawbacks. With the ability to change the development so quickly, the scope of the project could increase ad infinitum as the customer asks for more to be added to the project. Furthermore, this increased scope can make it difficult to predict when a project will be completed. The speed of change in agile makes documentation difficult to maintain since what was relevant today could quickly become obsolete tomorrow. However, even with these drawbacks, the agile methodology provides a superior approach to software development projects compared to the traditional waterfall approach which is why many companies today have chosen this approach.

Scrum Framework

There are many different frameworks intended to guide enterprises in agile practice. Common agile frameworks include Crystal, Extreme Programming and Kanban, each of which serves a specific purpose. However, Scrum is by far the most popular agile framework and the one nCino has adapted to be used in software development projects. Thus, this project's primary focus is the Scrum framework.. The reason for the popularity of Scrum is it is simple to understand, easy to implement and it gets results.

> "Because of its simplicity and high performance. It takes advantage of a need for a sense of achievement, positive feedback, and ownership of work done within a teamwork environment."

Scrum is a project management framework that relies on incremental development completed in repeatable fixed time-box increments. This is usually two to four week periods, called sprints, during which a feature or product is delivered. Each sprint's goal is to build the most important features first and come out with a potentially deliverable product. Within the sprint, the scrum team completes small units of work known as stories.

The backbone of scrum is based on the three pillars of empiricism which refers to working with evidence that is observed, documented data and based on facts rather than thoughts and plans. The three pillars of empiricism are transparency, inspection and adaptation (these are discussed further in the subsequent sections). Scrum implements an empirical process where progress is based on observations of reality. This is not just the responsibility of the scrum master; all team members are expected to examine the facts and think of ways that the team can adapt.

Scrum Ceremonies

The heart of the scrum framework are the ceremonies, which are meetings held at very specific times during a sprint. The scrum ceremonies are:

- Sprint Planning - The event that kicks off the sprint where the team makes a commitment on what work they can accomplish during the sprint.
- Daily Scrum or Stand-Up - A development team daily check-in where the team discusses what they did the day before, what they are going to do today and anything that is impeding their progress.
- Sprint Review - This is held at the end of the sprint where stakeholders evaluate the work that was accomplished and provide feedback.

- Retrospective - An event immediately after the sprint review where the team discusses what went well, what could be improved and highlights action items that the team could use to improve.
- Refinement - During a refinement, technical details are added to tickets, estimates for those tickets are made, andthe stories are put in the product backlog. There is not much guidance from the scrum guide on when or how this ceremony is to be conducted, only that it is up to the scrum team to decide the best way. At nCino, it is common for teams to break their refinements into two meetings, a pre-refinement where delegates are appointed and details are added to stories and a refinement where the entire team participates and estimates are added to tickets and timeboxed to five minutes for each tickets.

In each ceremony, the team is encouraged to inspect and adapt, touching upon the other two major pillars of empiricism.

The Jira Board and ticket types

Transparency means that everyone on the team has a clear understanding of their goals and responsibilities. One of the best tools used to promote transparency at nCino is the Jira scrum board. The board is a way to visualize the tasks that need to be completed for the sprint to be successful and the current status of said tasks. An example of the Jira scrum board can be seen in Figure 1. Here you can see different ticket types populating the board. Clearly stated are the ticket types with the use of the icons (green for stories, red for bugs, grey for technical debt, and orange for PDIs). The estimated size of each ticket is shown. One of our tickets In Progress shows the assignee (yours truly) as an example.

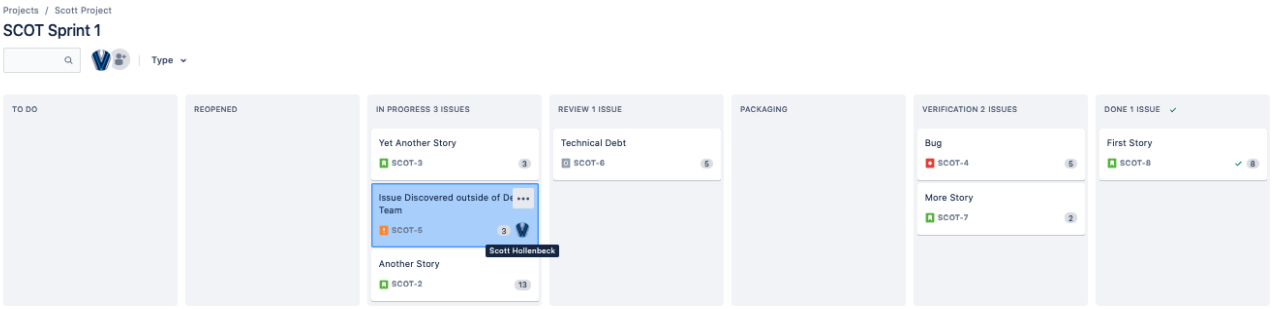


Figure 1. Sample Jira Board

Those tasks in Jira, are represented by tickets that are pulled from the large list of work to be completed known as the backlog and move from left to right, from…

- **Ready** - tickets that are ready to be worked.
- **Development** - tickets that are currently being worked by a developer.
- **Review** - all tickets must be reviewed by at least two developers before being packaged.
- **Packaging** - work that is complete must be run through software that runs tests on the code and also looks for merge conflicts, which occur when two developers are attempting to change the same lines of code.
- **Testing** - where the work is verified by Quality Assurance.
- **Done** - when the work is listed as complete.

There are many types of tickets in Jira. The most common is a story ticket, which is a requirement that needs to be written from the perspective of the end user. Stories are grouped under a larger effort known as an Epic. An Epic can be thought of as the end goal, whether it be a stand-alone deliverable or a self sustaining unit of a larger effort. The stories can be thought of as the work tasks that are pulled together into a sprint from the current backlog that is needed to accomplish that goal.

During the sprint process there are other tickets that may need to be generated to assist in accomplishing tasks. One is a Bug ticket, which is created when an issue is found which prevents the product from functioning properly. A Bug ticket is used to reopen a story to fix a problem that has surfaced. At nCino, we have a special form of bug known as a PDI or Product Development Issue. PDIs are issues that were discovered outside of the development teams; it could be a different department at nCino or something that developed in a customer environment. These 'bug' tickets are then posted on the board as new stories to be addressed.

The last major form of ticket at nCino is Technical Debt or Tech Debt. Technical debt is work that does not provide additional business value but rather cleans up work that may have been done quickly in the past. Because our teams sometimes need to work fast, the best practices are not always followed and corners need to be cut. Like regular debt, if technical debt builds up and best practices are not followed it can make your software difficult to iterate on or weaker overall. At nCino we have an allotment limit or a percent that we dedicate to this quality work because we know the simple truth that if a team accrues too much technical debt eventually the code base can become too cumbersome to work with, which could endanger the entire project. So, while useful, technical debt must be monitored very closely to be sure it does not become so convoluted that it is impossible to work with, and work slows to a trickle or stops entirely.

Each ticket is given an estimate by the team which is a guess usually encapsulated in a number that captures the level of effort and complexity of a ticket. At nCino we use fibonacci sequence to measure our estimation; using fibonacci a story would have a weighted number 1, 2, 3, 5, 8 or 13 etc. attached to it with a story sized at '1' being the

lowest effort and complexity and a ‘13’ being the highest. The general rule of thumb is that you want to chunk the work into the smallest increment possible so that if a team, for example, estimates a ticket at a 13, it should be discussed if the work can be broken down any further.

Conclusion for Chapter 2

In this chapter we have compared scrum and Waterfall practices and how they relate to the agile methodology. We then did a bit of a deep dive on how scrum works on a tactical level. In the following chapters we will see how these practices relate to the board game and how the lessons of scrum will be imparted to the players.

CHAPTER 3: METHODOLOGY

Scrum Gamification at nCino

Over the years the Scrum process has evolved at nCino as it learned how to manipulate and bend the scrum framework to improve upon it to meet its specific needs. The true intention of this game is to relay how scrum works and how nCino uses it. Because of this, new hires need to be taught the original concept of the scrum framework, how work flows during sprints, and how nCino has changed the framework to meet its needs.

All of this could be taught by using a very long and drawn out powerpoint presentation. In researching how we might do a better job in new hire training, I believe there are advantages to using a game board format as part of new hire orientation.

There are several board games currently available to teach varying agile frameworks. One is getKanban (as seen in Figure 2). The getKanban Board Game is a physical game designed to teach the concepts and mechanics of Kanban for software development in a class or workshop setting. It includes one board, one set of tickets, dice, marker pens, charts, etc. While getKanban has become a successful board game using an Agile framework, it does not speak to the scrum framework.

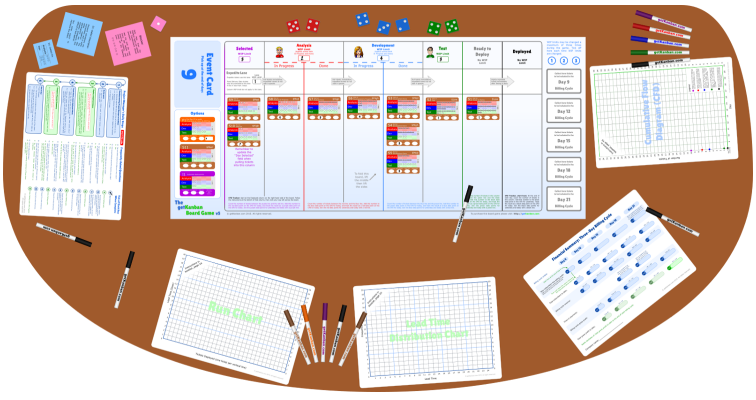


Figure 2 - getKanban Gameboard

A board game designed to teach the scrum framework is Scrum Tale (see Figure 3). This game is designed to provide practice in all the elements of the framework. Authors of the game state "The players simulate Scrum teams writing a crime story simultaneously using Google Docs. Collaborative story writing is very similar to software production: you need to be creative and act as a team to deliver a product. You will also experience challenges with integration, quality and tools that are present during software development without the need of having a technical background. You can observe the mechanics of the Scrum during the stormy process of story writing." While the game is a good reference to understanding scrum terms, it is not effective in experiencing the scrum process. Also, there are lessons learned at nCino that could be applied to customize the game for our employees.

Figure 3 - Scrum Tale

Web Based Applications vs Virtual Reality Training

I conducted a survey with over a dozen employees hired in the last 6 months asking them two simple questions, "What did you wish you had learned before starting

your work? Is there something about the process you wish was taught better?" Out of the developers, testers, UX designers and product managers, no one could give me specific parts of the process that could have been taught better. However, most workers emphasized that experiencing being a part of the actual process was the best way to learn. While not a completely immersive experience, I believe a virtual hands on experience is a vast improvement to the current powerpoint lecture that we currently provide our employees.

The benefits of hands-on learning have been well documented with increased engagement and retention being the most important. A study by Devon Allcoat at the University of Warwick noted "Participants in the traditional and VR conditions had improved overall performance (i.e. learning, including knowledge acquisition and understanding) compared to those in the video condition. Participants in the VR condition also showed better performance for 'remembering' than those in the traditional and the video conditions. Emotion self-ratings before and after the learning phase showed an increase in positive emotions and a decrease in negative emotions for the VR condition. Conversely there was a decrease in positive emotions in both the traditional and video conditions." (Allcoat, D. & von Mühlenen, A, 2018) Similar studies have also seen increased engagement, and especially increased enjoyment. Enjoyment is a very important factor. At a company where having fun is part of our company wide manifesto (nCino, 2022), the increased enjoyment reflects the goals we as a company set out for ourselves.

The Power of Gamification

Not just virtual gaming but research has taught us time and time again that the gamification of learning is very beneficial. Studies have shown that a large percentage of

companies are investing in gamification tools such as virtual reality headsets, across the industry. (Harami, 3055) The major benefits can include engagement and improvement in retention of knowledge so much so that students will often improve their scores on exams (Barata, 10) There is so much literature about the positive benefits of there is actually research that examines the overwhelming findings of the other literature. (Harami, 3055) A virtual gaming environment would improve engagement, remembering lessons learned and increase the fun of what can be a dull onboarding process at a fun experience.

Game Platform

The criteria for the game is online, multi-player and must have some degree of automation. To avoid building from scratch which would take years, Tabletop Simulator was used because it meets all the criteria stated right out of the box. Tabletop Simulator is a video game that allows players to play or create games in a virtual sandbox. The game allows for multiple players to play at once, online and allows for automation. On top of all that, it has built in objects like dice and cards which would save a considerable amount of time (See Figure 4).

Figure 4 - InGame Screenshots

The automation was completed using the language for Tabletop Simulator, Lua. Lua scripts are stored in the json file of the save game as a plain text string. There is no need for an external internet host for Lua scripts, everything is self-contained in the game's save file.

Game Play

The game would be facilitated by a scrum master who would have full knowledge

of the rules of the game but also how they relate to the real life processes implemented at nCino. Similar to a table top role playing game (e.g., Dungeons & Dragons), the facilitator of the game would act as a "game master" who would narrate the details of the story of the game but would also provide more information about aspects of the game if prodded by the players. For example, players could garner more information about the epics that are presented to be completed if they ask the facilitator the right questions which adds another aspect to the game. The facilitator in game would be known as the on board computer.

The lessons are the processes of completing work (based on how nCino handles these processes) which mimics the software development lifecycle as well as the epics and event cards (more details around the epics and events cards are presented in the next chapter). The game is won if the team is able to complete all of the work they set out to accomplish.

It is worth noting that completing the Tech Debt items in the game is the best path to success and the final lesson of the game. Even if a team does not complete the Tech Debt work the game master would reveal to the team how much more they could get done if they had done the tech debt. The lesson is that without addressing tech debt and quality items, the product can become bogged down to the point where the product is too difficult to use or even work on. More details surrounding game play and rules can be found in the appendix.

Conclusion of Chapter 3

Now we have covered the basics of scrum and what lessons of scrum I intend to relay in the boardgame, the fourth chapter will explore what this will actually look like when implemented.

CHAPTER 4: OUTLINE OF COMPLETED PROJECT

Initial Runs of the Game

In simulating the learning experience in the game, one of the most important components is to ensure that work tickets move across the board in an efficient manner. If there are too many tickets in play during the process, the game can become so bogged down that little gets done and the simulation is ineffective. On the other hand, if there are too few tickets, the board will get cleared of work too early and significantly reduce the learning experience and interest of participants. In addition, managing personnel is an important part of the game, and one of the first events to occur in the game is the ability to move development resources to help when the testing component slows down the process. Since dice are being used to simulate the completion of tickets at each stage, it was important to nail down exactly what could be accomplished rolling the dice for effort.

To obtain an initial idea of what should be expected from rolling the die, I ran a few scenarios like the one written about in the Example Round from Appendix B. In the first couple of runs, I experimented with having developers represented as four-sided die and testers as six-sided die. The intent is for the testing column to occasionally become a bottleneck, forcing players to decide on how best to manage resources. As it turned out, the testing team was accomplishing too much using 6 sided-die, thus making simulation unrealistic and not providing the desired challenges. This led to using four-sided die for both developers and testers.

Another issue that had to be dealt with was the number of ticket estimates to include in the game. In my initial runs of the game, I found that having varying levels of

estimates made it very difficult to predict how tickets would flow. In reality, ticket estimates vary wildly from project to project and I tried mixing the epics with stories that spanned from 1 to 13, and also allocating the points between development and testing effort. This proved problematic in trying to predict bottlenecks and, in addition, revealed that an excessive amount of work would be required in programming the game. To alleviate this, I changed all tickets to be 5 points for each story, with 3 set aside for development and 2 for testing, making it much easier to anticipate the flow of work.

In deciding how many story points to use as the basis of the game, different combinations of numbers of dice to use for development effort and testing effort were examined. Figure 5 shows that the average for development maximizes at 15 tasks, and for testing maximizes at 5 tasks (Figure 6) for a total of 20 per round (a day's effort). The plan is to have a game cover 10 rounds, or 10 days, which equals the number of days in a normal sprint round at nCino. In a game, the average total number of rolls or tasks for a team would be about 200 points (20 points per round x 10 rounds). The biggest take away from all of this is that die with more sides means a greater range in what your players will roll while a die with fewer sides makes your outcomes smaller and thus more predictable. To be predictable, I would need to go with the smallest denomination of dice: the four sided die.

Roll: 6d4
Statistics
Min: 6
Max: 24
Avg: 15.00
Std Dev: 2.739

Roll	Freq	Prob	Bar
6	1	0.0%	
7	6	0.1%	
8	21	0.5%	
9	56	1.4%	
10	120	2.9%	
11	216	5.3%	
12	336	8.2%	
13	456	11.1%	
14	546	13.3%	
15	580	14.2%	
16	546	13.3%	
17	456	11.1%	
18	336	8.2%	
19	216	5.3%	
20	120	2.9%	
21	56	1.4%	
22	21	0.5%	
23	6	0.1%	
24	1	0.0%	

Besides the links above, I have also seen these ideas:

Figure 5 - Percentage Chance of Rolling 6 four-sided Dice

Roll: 2d4
Statistics
Min: 2
Max: 8
Avg: 5.00
Std Dev: 1.581

Roll	Freq	Prob	Bar
2	1	6.3%	
3	2	12.5%	
4	3	18.8%	
5	4	25.0%	
6	3	18.8%	
7	2	12.5%	
8	1	6.3%	

Figure 6 - Percentage Chance of Rolling 2 four-sided dice

Pilot Test Game with retired business executives

The first official run of the game was with retired business executives in various fields with hands-on experiences of running companies.All participants are considered to be subject matter experts in their respective fields with numerous years of experience running projects.. Half of the group were employed by the military, one of them was on a

board of directors for a major company, a nuclear inspector for NATO, and an entrepreneur that ran his own company.

Initial impressions were that it was a good exercise. I started by letting the group know that I would time box the event to be under an hour but we ran for ninety minutes and no one complained. They were engaged and everyone participated in their own way. Figure 7 shows the initial game run.

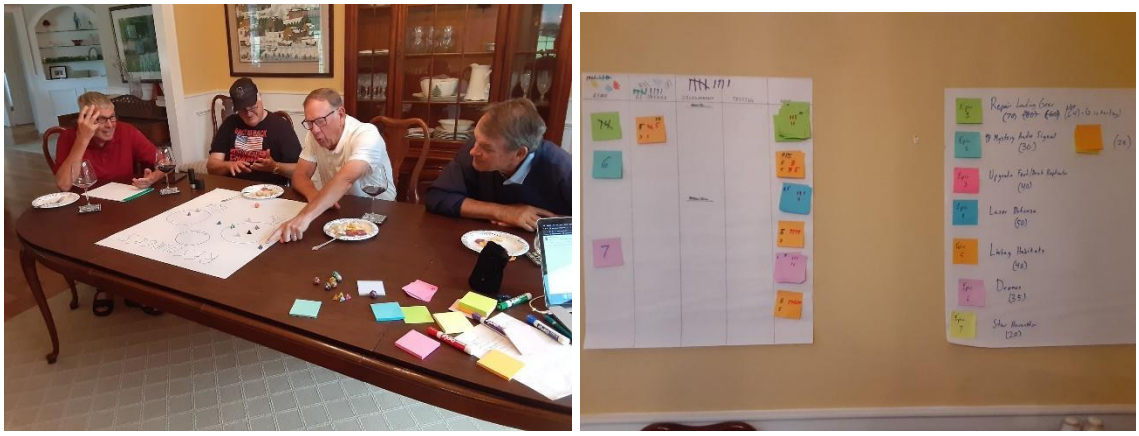


Figure 7 Initial Game Run

There were many action items and improvements suggested. The descriptions for the epics as listed above were confusing and needed a lot more clarification but this presented an interesting twist on the game where being curious and inquiring more, presented the team with more information. For example, Epic 7 the star harvester, provides unlimited fuel for the ship but when the team asked questions and discovered that the ship would not be passing by a star and that the ship had enough fuel for the trip, it quickly became apparent that the work for that epic would be useless in game. Because of this, I changed the script for epics to include notes to the game

One of the participants pointed out that word spreads fast in the office and eventually teams would become wise on which epics were the most important and which

to avoid. His suggestion was to create a longer list of epics to mix and match them going forward. Also, there are two consequential epics that if completed before the eight-day meteor event would result in the team avoiding extra work. The participants liked this, but were hoping for more consequences for deciding on certain epics.

The 20-sided testing die at the end of each round was a huge hit and created a good amount of tension when the team was trying to get work done. The 20-sided die was used to confirm if work that was moving from verification to done had any bugs or defects. If a team rolled a 1-3 on the 20-sided testing die, that ticket would be reopened because a "bug would be found" and the work would have to restart from development. To spice things up, mid-game it was decided that if the team rolled a 20, all testing work would move to the done column. This added a new level of excitement to the game and another reason for the team to celebrate. For those reasons, it will be added and used going forward.

If the team was able to complete all the work before the end of the game, the question was posed as to whether they would be allowed to pull in another epic or would the game just end.

Second Test Run with nCino Employees

The second run was done with nCino employees over zoom. This team included a scrum master, tech writer, tester, developer and recruiter. The team really enjoyed themselves and found the game useful. However, since they had working knowledge of scrum, there was less explanation upfront as it was intuitive to the team how to play and were able to complete the game in fifty minutes. This may not be the case for a team playing this game as part of orientation into the company.

Personally, what made the game a lot more fun to run this time around was throwing out the script and improving a little on the spot. To do this effectively, the game would need to be administered by someone who had a good understanding of scrum – namely a scrum master. They would learn the rules but also the lessons that are trying to be imparted but there would be room for them to improv similar to a role playing game.

The feedback was that it was very informative, particularly for the recruiter who had to join to play. Other team members commented that it was a fun and engaging learning experience. There was unanimous consensus that this would be an improvement over the traditional powerpoint presentation.

Creating Epics

Originally, it was decided that teams entering the simulation would be assigned what epics they were to work on. In the initial runs, it was noted that this method did not provide much of a challenge and team members began losing interest, defeating the purpose of the learning experience. By allowing teams to decide the epics they felt were most important to work on, the players enjoyed it more, because they took ownership of the project, and had more fun instead of just rolling dice to move work across the board. As noted above during the pilot test game with business executives and the second test run, a big suggestion was to increase the amount of choices in the epics. For the purposes of teaching newly hired employees, there would need to be a set run of epics that the teams would be able to choose from. However, if found to be popular, a greater list of epics would be generated to change the experience slightly from team to team. This would keep the game from becoming stale and would not give new employees an advantage if they are told ahead of time by current employees what they should expect.

Creation of Event Cards

The real lessons learned in the Planet Scrum virtual board game come from the event cards. a major change to the gameplay that happens at the end of each round and are based on real life events that affected nCino during the days of its' infancy. To fully understand how informative they can be, I am highlighting a couple of their event cards and what information players can glean from them.

Anyone Can Test (as seen in Appendix D, event card 1)

` In the early days of the company we struggled with a quality issue. We used a literal army of manual testers but when we would release our product, our customers would discover outstanding bugs that oftentimes significantly hindered their work. It got so bad that we began to garner a reputation of delivering buggy software and, worryingly, our customers began delaying their acceptance of our new software until they were confident the inevitable bugs were dealt with. An automated testing framework would certainly have assisted in this effort, however, the time it takes to develop and mature such a program can take months if not years. We needed to change the way we tested.

The solution for our problem required a drastic shift to how we had been operating. Up until this point, it was policy that, because of their time value, our developers do not and would not test. However, we changed this policy once we realized that developers have insights into testing that a manual tester would not normally consider. Currently, we regress our product with all roles participating.

The results were dramatic. After implementing a testing policy that included developers in the process, the bugs found in delivered products became almost nonexistent. We went from releasing a toxic package that not many customers would

want to adopt immediately to a pristine product with very little necessary feedback from our customers.

Over the Shoulders (as seen in Appendix C, event card 7)

One of the tenets of an agile methodology is "Responding to Change" over "Following a Plan" – it is more important to react and modify your work based on new information than to blindly follow what was originally prescribed. This might seem like an obvious conclusion but keep in mind the waterfall methodology, which was the default way to create software for the majority of the 20th century, would stress the opposite conclusion.

Feedback loops – processes to validate the work that has so far been completed – are a tenant of scrum. To find the areas in our work that need changing, to get feedback on our plan Scrum has a built-in feedback loop, scrum has a prescribed ceremony of the sprint review session where stakeholders review the work that has been created and give actionable feedback to the team.

At nCino, we developed an even quicker feedback loop which we refer to as an "Over the Shoulder". Before a developer finalizes their work and hands it over to a tester for verification, she/he will pull in the tester and product manager and demo the feature.

This has significantly enhanced the productivity of our development department. This quicker feedback loop has resulted in major bugs being caught early, and the developer can act on those bugs while that work is still in development without having to cycle back to "old work" from his current effort and not wait for it to recycle back from testing.

Another advantage of *Over the Shoulders* is the product manager, through the interim product demos, can evaluate whether the agreed to end-product is still appropriate

or whether changes need to be made to the original specs. Our product team can see what is being delivered and can see immediately if the work is not exactly what was previously envisioned or find criteria that may be missing and draw up plans for additional work. Over the Shoulder fine-tunes the vision of our product, reduces the time spent on development and increases the overall quality of the final product.

GAME CREATION

Lua Scripting

Tabletop simulator is an excellent environment with a lot of physics baked into the software i.e., rolling dice, shuffling cards. To extend the functionality, tabletop simulator allows for changes to be made through its scripting language lua.

There are two types of scripts that can be altered in the game: an overall global script and individual object scripts. The global script is in charge of tasks that occur in the background and general game management. Object scripts affect any physical object within the game and there can be as many object scripts as there are objects within the game.

Additionally, there are built in objects to the game such as dice or cards that can be added to any game as well as custom objects that users can create. A custom object example for Journey to Plan-It scrum would be the board and cards that have artwork created specifically for this game. All objects have global unique identifiers which makes it possible to call those objects when coding.

Lua Example

An example of scripting can be seen in Appendix E. Figure 9 from the appendix is an example of the UI scripting. In the game I have created a checker that acts as a button and given a list of attributes including the size of the button, the font size on the button,

the position relative to the checker and lastly a function that the button calls – dealDeck1.

Figure 10 from the appendix shows the coding behind the dealDeck1 function. It starts by calling the global unique identifier which is called from the global settings of the script editor. It then pulls a card from the deck of cards using the built in API function getObjectFromGUID. It then runs a for loop that draws 5 cards from the deck and moves them to a position on the board.

Enhancement Intentions

From a development standpoint I had a number of tasks that I wanted to automate. These were the tasks that I set out for myself to enhance the game:

1. I set out to create a script that would return the dice to the starting position so that each round they would be easy to access since they would be at the same spot each time.
2. Move and flip cards on click
3. After the dice had been rolled, the cards would move automatically based on how much work had been completed.
4. Each card would have a level of effort and that level would be checked off based on what the dice role would be
5. Math to be collected would include how many points were completed in total, how much work was added, how much was removed and a leaderboard with a running high score of past games

As I dug into the scripting language and familiarized myself with the UI I found it was impossible to change the states of cards which means that I would not be able to check off work that had been done on the cards. Worse, it was not possible to pull or read

data from the cards so collecting data on how much work had been completed and tallying scores would have to be done by hand.

Final Runs Through of the Game Online

There were two sessions of the game run using Tabletop Simulator with one scrum team made up of one product manager, three developers and one tester and one team of random new hires with a variety of positions. Both were conducted over zoom.

The immediate issue that arose is that recently our IT department had restricted access and employees could no longer download the software necessary to play the game on their computers. However, in the first run, one of the team members had Tabletop Simulator on their personal computer and had someone who could run the game while I was free to just concentrate on the narrative and script. Despite the extra hand, the game ran over the allotted hour that was planned for a total of 90 minutes. The feedback from the first scrum team was that while they enjoyed the game, it did not seem as

engaging as expected since they were not able to interact with the interface.

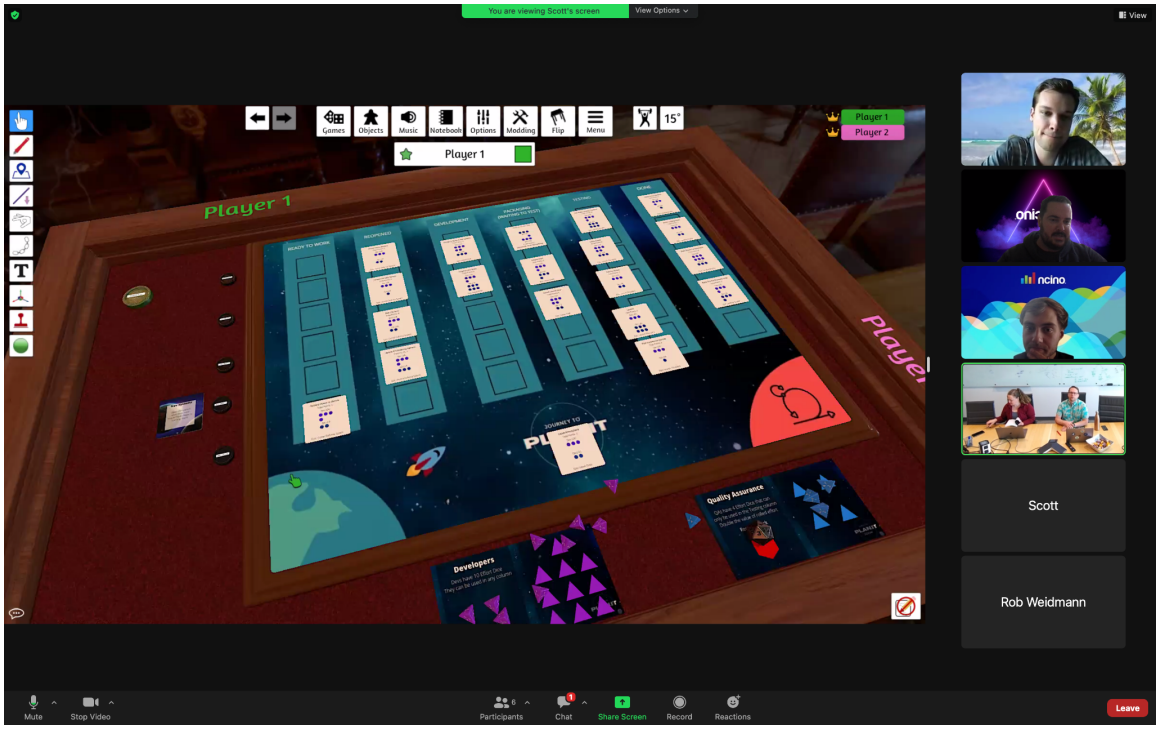


Figure 8

The second session, made up of two folks from sales, one from customer support and one from quality assurance, was much more of a slog. Because the new hirers needed a longer explanation of the rules and ran for nearly a couple hours.

The feedback for the second session was understandably mixed. Similar to the first session, the virtual interface was not as positive considering no one else but myself could move objects in the virtual environment which made a game that should be fun and interactive much more boring.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

Lessons learned while developing the “Journey to Plan-it Scrum” game have been invaluable in developing nCino’s scrum onboarding program.. In the physical version of the game new hires, working in groups of four, have quickly grasped the nCino Scrum concepts in software development. New hires have indicated the process has been both enjoyable and constructive, as well as, allowing for interaction among people who may have met for the first time.

Of the overall project, I created a game, artwork, and basic scripting from scratch. I was assisted in online tutorials found on YouTube to complete the scripting portion. But at the same time, issues surfaced in developing the online version of the game. These included:

- The game is much too slow, taking 90 to 120 minutes to complete versus an ideal maximum of 60 minutes
- Playing cards could not be modified when there they need to be given changing circumstances
- Outcomes could not be digitally recorded which prevents game stats and final scores could not be recorded in game
- Obstacles to implementing a virtual reality system proved not feasible
- More epics would need to be created for the game to stay fresh for players

For these reasons, the online experience needs a complete revamping to be successful. I am using alternative software and scripting languages, for example, Unity, which is a robust engine that can address a lot of the issues outlined above.

The major lesson learned from this experience was the mistake of choosing interface over functionality. Tabletop simulator is a sleek and fun experience particularly with multiplayer games. However, the major items I wanted to accomplish were not possible because the software treated playing cards as static objects that cannot be modified.

Also, my original intention of making this a virtual reality experience proved not possible by the fact that VR technology is not ubiquitous enough to make it possible for random employees to participate in that way.

Even with those needed improvements, I am happy to report that the game has been adopted by our onboarding team for new hires. For the time being, games are run using a physical board until an online experience can be completed.

REFERENCES

Balaji, S., and Dr. M. S. Murugaiyan. "WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC." *International Journal of Information Technology and Business Management*, vol. 2, no. 1, 2012, p. 5. *https://mediaweb.saintleo.edu/*, https://mediaweb.saintleo.edu/Courses/COM430/M2Readings/WATEERFALLVs%20V-MODEL%20Vs%20AGILE%20A%20COMPARATIVE%20STUDY%20ON%20SDLC.pdf.

Hughey, Douglas A. "Comparing Traditional Systems Analysis and Design with Agile Methodologies." *Comparing Traditional Systems Analysis and Design with Agile Methodologies*, 2009, http://www.umsl.edu/~hugheyd/is6840/waterfall.html.

Kent, Beck. "Manifesto for agile software development." *Manifesto for agile software development*, 2001, www.agilemanifesto.org. Accessed 15 1 2021.

ERIC Clearinghouse for Science, Mathematics and Environmental Education, 1929 Kenny Road, Columbus, Ohio

AllcoatD., & von MühlenenA. (2018). Learning in virtual reality: Effects on performance, emotion and engagement. Research in Learning Technology, 26. https://doi.org/10.25304/rlt.v26.2140

nCino (2022). Culture and Careers, nCino. https://www.ncino.com/culture-careers/

J. Hamari, J. Koivisto and H. Sarsa, "Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification," 2014 47th Hawaii International Conference on System Sciences, 2014, pp. 3025-3034, doi: 10.1109/HICSS.2014.377.

Barata, Gabriel. Gamification '13: Proceedings of the First International Conference on Gameful Design, Research, and Applications October 2013 Pages 10–17 https://doi.org/10.1145/2583008.2583010

APPENDIX A

The Rules - How the game is played

For policy reasons it is not possible to use real examples of nCino work to be the focus of a game that people outside of the company would be playing. For that reason, the backstory of the game has been made more imaginative to capture the attention of the player:

The year is 2050, and nCino has built a billion dollar rocket that it has launched to the mysterious Planet Scrum. Not much is known about this planet other than there is a strange audio signal emanating from it. As project managers aboard the nCino rocket, it is your job to delegate tasks by choosing what projects the rocket crew should work on and determining the priority of the work.

The game board is similar to the jira board that is used at our company (see Figure 1). The goal is to move the tasks through development and testing lanes to the done or completed column. Each task has a development and testing level of effort score that is accomplished by rolling dice. Players are given one four-sided dice for each resource they are rolling for. The project managers have two scrum teams under their direction, which is how it normally works at nCino with pairs of teams working on a particular feature. Three four-sided dice represents three developers per scrum team and two four-sided dice represents two QA teams.

The goal is to get as much accomplished in ten rounds with each round signifying one day in scrum which would normally be ten working days. Each round the team would lay out the work that needs to get done and then roll testing and development dice to see what is accomplished. The rolling of the dice outcome represents the variability of work that is accomplished each day.

Between rounds event cards would be pulled and this is where the players will experience the lessons the nCino company has learned in its eight years of existence. The topics for the event cards come from interviews I conducted. The interviewees included a senior project manager, senior developer, manager, tech writer, quality assurance engineer and a scrum master. Here are some of the lessons learned that will become event cards:

- In the first few years of nCino we had quality issues and it hurt our reputation. Customers would wait before taking new updates because they would be riddled with bugs. Anyone can test but we learned after a few iterations that everyone *should* test. Our regression periods, the end of the release testing period where all completed features are tested at once, that were usually just QA engineers testing, became an all hands on deck situation where developers and product managers joined in the testing effort. After that, product quality skyrocketed with very few bugs found after release. In the board game, we would give teams the option to use one of the development dice as testing dice for a round; that is, you can move one of your developers into testing to enhance workflow.
- PDIs. Product development issues are bugs that are found by customers and reported to us. In the game we would introduce one off tickets that would need to be completed in a certain amount of time, 2-3 rounds for example; the short time frame is given to simulate real world experience.
- Brooke's Law - as defined by the internet: Brooks's law is an observation about software project management according to which "adding manpower to a late software project makes it later". It was coined by Fred Brooks in his 1975 book The Mythical Man-Month. According to Brooks,

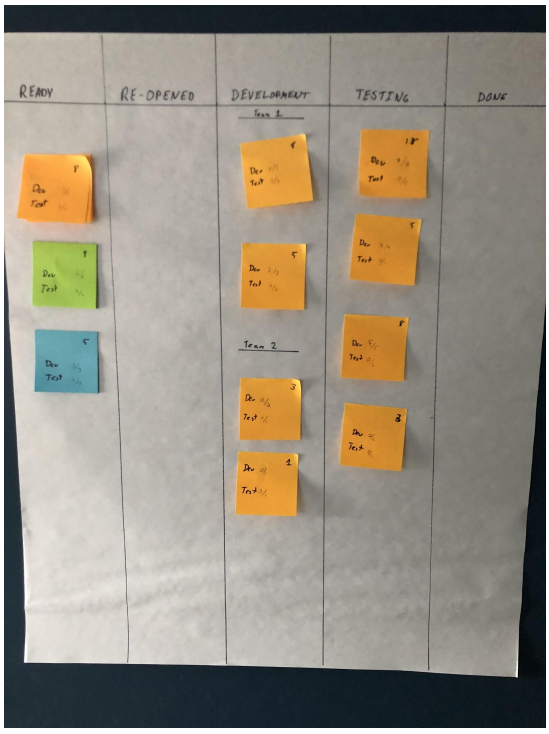
an incremental person, when added to a project, makes it take more, not less time. I have personally seen this at nCino. The added person has a lot of questions to bring them up to speed on the project, thus slowing down the developers. That new person may also make rookie mistakes that will need to be fixed by the more experienced team members. To account for this phenomenon in game there would be an event card to announce that a developer is being added to the team, which should be met with excitement, until the players find out for the first round they will lose a development dice for one round while that developer is being on boarded.

- It's not uncommon to have an epic's scope increase because of a requirement that was unforeseen and or added after the development had started. In the game tickets might be added to certain epics.
- On the other end of the spectrum, it's also been known to happen that an epic is not needed at all and thus work was wasted. At least one of the epics in the game will be revealed to be unnecessary around the midpoint of the game.
- The Power of OTS. At nCino the term is Over the Shoulder or an OTS. It is when the product team checks the work as it goes from testing to done but is better if done when going from in development to packaging. It has proven to be very useful at nCino to catch known issues early or even things that the Product Manager forgot to include in the initial acceptance criteria. In game, it would allow a ticket to pass from testing to done without a bug check.

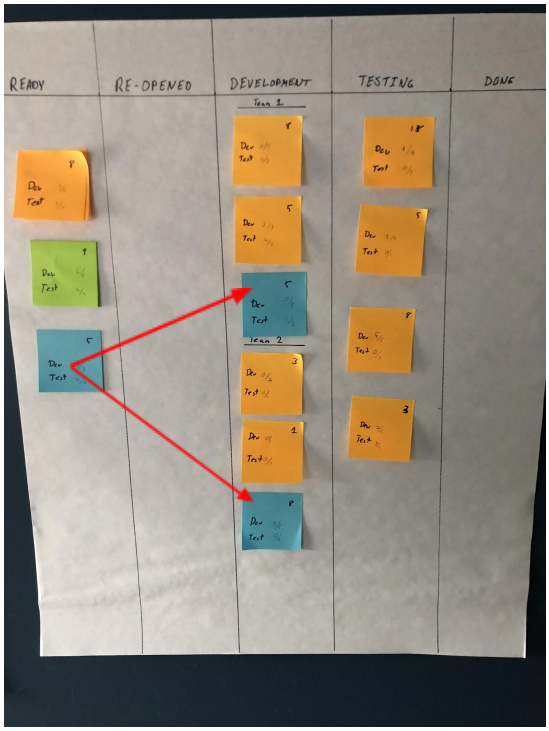
- QA automation is still not a fully realized part of the nCino product but we know that we will need it to test our product quicker and get new features and updates

APPENDIX B

Example Round



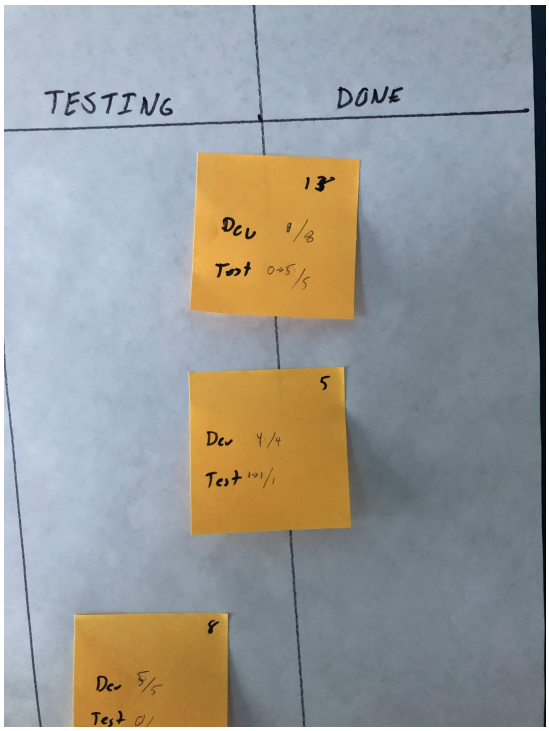
The game starts at the beginning of a new sprint. The players are taking over as product managers of a team at the beginning of a new sprint. The last product managers did not do a very good job as there is work still be completed on the board (indicated by the orange post in development and testing) and it is the goal of every team to move all work to done by the end of a sprint.



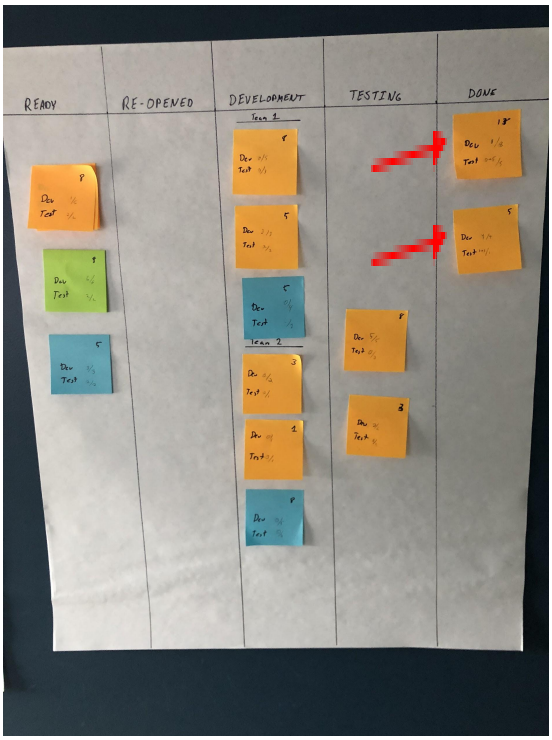
In the Ready column you will find three stacks of tickets each representing an epic. The team in this scenario has determined that the third epic in blue is the most important so it has prioritized that work. There are two scrum teams of 3 developers each (6 developers total). You can only bring in a ticket for each developer working tickets for six total.

With the work selected, 4 sided-dice are rolled to represent testing and development effort. With start with testing and move left across the board.

There are 2 testers so 2 four sided dice are rolled. The output is 3 & 3 for 6 total.



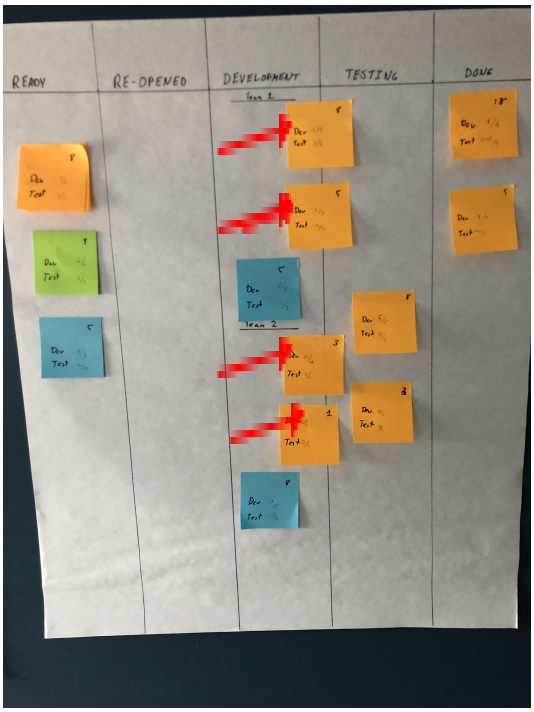
Each card has a level of effort score for development and testing that need to be completed. The testing effort for these were 6 total so the top two cards are tested, but, not so fast, we now need to determine if any bugs were found in that testing effort. A bug check is done by rolling one 20 sided die and if a 4 or higher is rolled the ticket passes. However, if a 1-3 is rolled, a bug is found and the ticket is sent back to development and the level of effort score for development and testing are reset.



A 4 and a 12 are rolled, both tickets pass their bug checks and are moved to the done column.

Next up, the developer dice is rolled to determine their effort. Two teams of three dice each.

The first team's result: 4, 1, 2 = 7 total
The second team's result 1, 3, 3 = 7 total.



The game board splits the developers' efforts amongst two sections while testing is combined.

Seven development moves four tickets total towards testing.

This is the end of the round or as it would be put in the game, this is the end of the day's work. The game lasts 10 rounds or days which is the actual amount of working days in a two week sprint.

APPENDIX C

Epic 1: Repair Landing Gear (70 points level of effort)

Shortly after takeoff, the landing gear was damaged. Belly landings carry the severe risk that the ship may flip over, disintegrate, or catch fire. Plus, they're slightly terrifying. Without the landing gear, we will not be able to land safely on Planet X, so it must be repaired.

Epic 2: Decode a Mystery Audio Signal (30 points level of effort)

The communications officer has detected a mysterious audio signal, possibly a distress signal or warning message coming from Planet X. Right now, it sounds like a bunch of bleeps and bloops. That can't be right! Technicians must tune the audio receiver before we can discern the meaning of the transmission.

Epic 3: Rebuild the Food and Drink Replicator (40 points level of effort)

Currently our food replicators can only produce radishes. Which is great, if you love radishes. With some modifications, we might be able to widen our menu to include our favorite comfort foods from home. Mom's spaghetti and meatballs, anyone? Let's make those upgrades before Sunday dinnertime.

Epic 4: Install a Laser Defense System (50 points level of effort)

It's space! There could be dangers in every corner. Does space even have corners?! If we had lasers, we could shoot the corners. Maybe we should plan for shields, too, but first, lasers!

Epic 5: Create Living Habitats (40 points level of effort)

Sure, other species probably view us as fragile meat sacks, but we're meat sacks with ingenuity and gumption! We know we need to make a sustainable, livable habitat, first in space, and then on Planet X. Let's make sure we have a renewable source for air and water.

Epic 6: Exploratory Drones (35 points level of effort)

Not everyone wants to wear a red shirt. That's why we create drones to scout unknown and potentially unfriendly territory for us. Time to create a few rovers to do some recon.

Epic 7: Invent a Star Harvester (20 points level of effort)

Unlimited cosmic power can be ours if only we can Figure out how to harvest it.

Epic 1 (repairing the landing gear) would be the only required epic with some tickets already in progress when the game begins. Epic 2 (mystery audio signal) and Epic 7 (invent a star harvester) are useless. Epic 3 (rebuilding the food replicator) becomes useless as the game progresses. And the remaining epics hold some value in the outcome of the story.

Order of Game Events

Originally, epics would be decided for the teams ahead of time with a number of tickets already in progress when the game began. This proved to not be very challenging in the initial runs of the game.

1. *"Sprint Planning"*
    a. *Introduce epics - The GM walks through each of the epics with the group.*
    b. *Determine Velocity - The GM states that the predicted velocity for the team is 180-220 points*
    c. *Choose Epics - The team works to Figure out which epics they will choose*
2. *Gameplay*
    a. *Choose Priority of Work - Each day starts with the team deciding the top priority*
    b. *Roll for Development*
    c. *Roll for Testing*
    d. *Bug Check*
    e. *Pull an Event Card*

APPENDIX D

Event Cards

Event cards are pulled at the end of each day (round).

Event 1: Anyone Can Test

At nCino we learned that not anyone can test but we learned after a few iterations that everyone should test. Going forward developer resource dice can be moved at the beginning of each day to help with

Event 2: PDI, Breached Hull

A meteorite has breached the hull of the ship. A 13 point ticket is added to the board and must be completed in the next couple days or crew members will begin passing out due to lack of oxygen. Lose one development resource dice for every day past the two day deadline.

Event 3: Uneventful Day

Nothing of note happens on this day.

Event 4 : Wasted Effort

One epic has been determined to be unnecessary. Remove the smallest epic from the game.

Event 5:

Scope Increase - It's not uncommon to have an epic's scope increase because of a requirement that was unforeseen and or added after the development had started. Add two tickets to the now smallest epic.

Event 6: Hooray! More Resources… oh wait

Two developers are added to the team. Now you are going to be introduced to Brooke's Law which states that adding manpower to a software development slows them down. Loose one development resource (that will teach your new developers). The following round add two development resources

Event 7: The Power of OTS

OTS or "Over the Shoulder".

Event 8: PDI, Meteor

A meteor is headed straight for the ship. If you have completed the

APPENDIX E

Figure 9

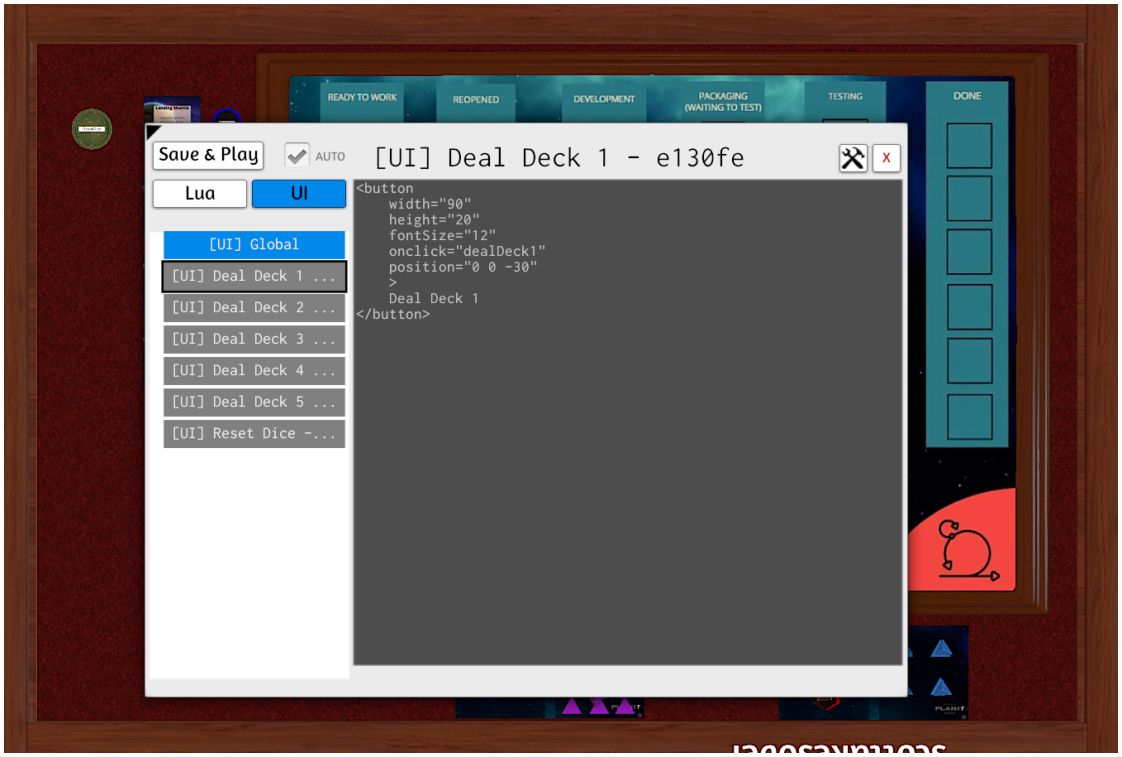


Figure 10