# Comments on UI and Game Play:

--

instructions need sample plays with pictures perhaps a video

--

* People in the story?
* Allow people to take a different role? *A little bit like selecting your role in a RPG. Good ScrumMaster; ScrumMaster trying to harm the team; What’s the fastest you kill your team*
* Game tied into prep work, submit your highest score and what you did to get there. *Game results to be reviewed in a workshop coaching session. Tell players to bring their game log, point out it’s a url*
* Players need a what’s in it for me question
* Facilitiation into the game mechanics
* Ensure all elements of for BluePrint for success show up in the game.

Mike Eng notes on the game:

 ​Mechanics and playability?

* Mechanics: Feels like the game could be extended for a few more rounds, to provide more time to feel the compounding effects of the modifiers.
* Mechanics: How will this be played?  Single/multi player? Local/online?
* Mechanics: There is a focus on capacity/velocity, backlog items done. Should product and sprint goals be a factor (demonstrating that sometimes it's awesome to not complete stories, but still hit your goal).
* Mechanics: Would be neat for the player's team to have certain attributes, which would impact the cost and benefit of improvements.  The player would decide on which improvements to take based on the team's strengths/weaknesses.
* Mechanics: Items seem to be pure positive/negative impact.  Double-edged swords might be interesting
* Mechanics: Idea - Bypass DOD to get more stories "done", but pay for it with increased tech debt drag (or production fire).
* Mechanics: Idea - Skip retro to get a few more points in the sprint, but reduce/disable improvements in the next sprint.
* Mechanics: Nothing to do in sprint 5/6?
* Mechanics: Confucious said: "Best time to plant a build server was last round, the second best time is now".  I can't implement unit testing because I didn't build a build server in round 1... why can't I build one in round 2?

 Is it obvious how the game progresses?

* Visualize product backlog progress
* Show sprint velocity history as a graph
* Visualize customer satisfaction with product (neat to show this diverging from velocity)

 Is it obvious the effect actions are having?

* Some options are obviously negative (e.g., firefighter award), but it is unclear why I would want to pick some beneficial items over the others.
  + One exception that becomes clear later, is the Build Server, as it unlocks unit testing.
* Visually represent active effects
* What's the impact of the PO or stakeholder telling the player to prioritize feature work?
* Maybe stakeholder happiness as well.

 Is the rule text clear enough?

* The significance of the relationship between rounds -> sprints -> weeks is unclear.
* Would it be simpler to just say:
  + Game runs for 6 rounds.  Each round is one sprint

 Anything else around the basics of the playability?

* Assuming the UI is in a very early stage, but more visuals/graphics
* Selecting from tooltips is not great.  Can you show a description on the main page?
* Add more visuals/graphics for the UI - Perhaps bars/graphics to represent the capacity breakdown section.

**Christiaan Verwijs**

I think it's a really good way to teach teams how certain decisions (may) impact their performance over time. There seems to be a lot happening in the background. Here are some thoughts:

* The UX is a bit unclear for me as to how to start playing. I discovered at some point that "play" is a button. A tiny thing, but easy to fix;
* You're doing a good job with the storytelling. I would even do a bit more of this during the steps where you select actions. I frequently felt a bit lost as to what I was trying to resolve again. Maybe merge the screen where you explain the new situation and allow people to select new actions?
* The previous point was most pronounced for me in the first step ("Welcome to the World's Smallest Online Bookstore"), where I wasn't sure what was expected of me
* I wonder; would it be better to hide the working capacity breakdown? Or present it more simply? It confused me a bit, even though I understood it once I took the time to understand it. But its an element that takes a lot of space and takes time to understand, so most people may not take that time and miss the performance metric altogether. Perhaps you could just show the result of the formula and hide the formula itself behind a button?

These are just some thoughts I had while trying it. You're definitely doing something really interesting here! In summary, my feedback would be to simplify the experience for users a bit,

--

**Jeff Kosciejew**

Some suggestions to reward some things on the Rules page:

Background: I'd make it a little more clear what the objective of the game is. And, try to make it a little less obvious how to win. Something like this, perhaps:

Your team is working on the World’s Smallest Online Bookstore, a site that provides the best results (just a few) for every search, not every result on earth, but the best books that match whatever is searched. Your team is a software development team, focused on driving more sales of the best books in the world.

Your team is part of a company being funded by a Vulture Capital - that means, if you don’t deliver, our funding will be cut for the entire company. Your goal is to build a high-performing team, while delivering a meaningful ROI to our investors. You'll be faced with some choices in each round, but remember: all the investors care about is the delivery of features. You don't want to let them, your teammates, and the company overall, down.

The goal of this simulation is to help see the effects and impacts of the choices you make. Your team has a limited capacity, and will have to make tradeoffs on a number of aspects, including productivity and team cohesion. While some of the benefits of Agile happen at the individual level, there are many things that affect the relationships between team members and, therefore, the overall cohesion and productivity of the team. The game simulates ten months in the life of a team. Each round represents one month of work.

Rules:

* The game lasts 10 rounds - each round represents a month. *(not sure why you picked 6 weeks - that feels very SAFe-like to me. A month is a more tangible unit of time for most people)*
* Your team starts with a working capacity of 10 points.
  + These points can be spent on delivering Features to your customers or possible Improvements (aka Actions). Each improvement you select costs the team feature development capacity, just like real life. And, also like in real life, these actions may or may not provide a benefit in future rounds. We certainly hope they'll give us a benefit. But not everything works out as planned.
* Improvements fall into three categories:
  + Engineering Practices - designed to improve the quality of your code
  + Communication Practices - designed to improve the team members ability to work together
  + Product Ownership Practices - designed to improve the team's chances of building the right product
* How you balance your investment in these improvements with the delivery of features is entirely up to your team.
* Some improvement practices have prerequisites.
* Speaking of Features, just like real life, it is not certain the features you deliver will be valuable to the customer. At the start of the game, when you have little knowledge of the market and your target customers, there's a strong possibility that what your team commits to and delivers will actually delight the customer.
* The game has gremlins -- that is, challenges that may crop up and impact your team's ability to deliver. You may be able to avoid some of these. But don't forget that your goal is the delivery of valuable software that our customers will love.
* And while your goal is to deliver features the leaders of this company have decided to give you a chance to get the team started on the right foot. They've agreed to allow you to consider some possible Improvements even before you've delivered one Story. But, they do remind you that every improvement that you want to implement needs to be done by the team, so will reduce their capacity to deliver features.

Rounds:

* Each round consists of a series of steps:
* You get a round description. Read the description as a team.
  + Click Start Round
* Select Improvements or Actions.
  + Once selected, the cot of the Actions cost will be deducted from the team's Capacity.
  + You can see your capacity reduce every time you select an Action item.
  + Click Begin Development *(a better term than "Complete Round", IMHO)*
  + And, you might get some feedback from the Senior Leaders of your company, based on your choices.
* The game shows you how much capacity there is remaining to develop your User Stories.
* Click Roll for User Stories
  + The game rolls a die for each User Story to determine how many are actually completed. That's right. Not every Story will necessarily get completed. Just like real life.
  + Have a look at how much work your team completed, and the impact that's having. You may notice some feedback at the bottom, based on your progress. It's worth considering any feedback you receive, since it's coming from the people paying your salary.
* Click Next Round
  + At the end of each round, the game also checks to see if your team was affected by a Gremlin. Example: "Manager yells at a team member"
  + As you progress through the rounds, you'll notice that your Capacity increases or decreases depending on the improvements you have made, or not, and the Gremlins that affect you.
  + After round 1, the game displays the active impacts of Improvements that were previously made.
  + Actions not selected in a round are still available for the team to adopt at a later time. To keep the User Interface tidy, they can be found in the round where they were first available.
* Hint: play the game as often as you want. So if you want to run experiments to see what happens, go for it.

For the game itself:

* I'd be tempted to change the "Team Members on the Same Floor" in light of our current world. Maybe something like "Team Members agree on high-bandwidth. The team agrees to have a communication channel open via Slack throughout the day, and has regular touch-points on Zoom at an established time at the start of the day, and after lunch."
* For "Clarify Product Vision",  
  I'd remove: "Teams that aren’t involved in the creation of their product vision are doomed to build Product that neither Product Owner, Customers, nor Stakeholders want."  
  and replace it with something like: "Teams that aren't involved in the creation of their Product Vision are able to focus on building what they're told to build by the Product Owner"
* Same idea of "Al Work is Done on Main or Trunk". Your descriptions are a little too leading, IMHO, in general. I'd be tempted to write something like "Feature branches allow the developers to optimize for themselves", and leave the "while harming the team" for a debrief or for an impact to the team when that choice is made.
* For things like the "One on One", again, I'd encourage you to consider just describing the activity, and let the impacts speak for themselves when they occur. It's a game mechanic to allow people to make some "wrong" choices, see and feel the impacts of those choices, and then discuss them, debrief them, or play the game again and make different choices.
* In round 5, there's a bug with the "Adopt BDD". When I click on that, it loads a blank page.
* I added the "Problem Solving Award" in Round 8, and when I got to see the impact, It presented it to me as "Problem Solving Award active since 1 rounds" <- should be a singular round (not rounds)

I wouldn't take away the software development aspects. I suspect once the logic is finished, it wouldn't be too difficult to change the labels and descriptions to play the game as an HR, Marketing, or other type of team. That might be fun to have a "What kind of team do you want to be" option at the start - leave all the logic as-is, but change the copy.

The Engineering Practices only make sense if it's an engineering team, and get across a critical point that I think will really benefit teams thinking about.

On the Results page, I'd be tempted to include some debrief questions to spark some thought or conversation. I suspect you've already thought of that, and just haven't got there yet! :)

The last comment, and I suspect this would be more work than it's worth: is there a way to tie an outcome to the output? Specifically, if we deliver 363 User Stories, how many paying customers did that result in? What's the value of the stories delivered? I suspect that's beyond the scope of the game for now, and you've sorta covered that off with not completing every user story in the sprint, kinda.

Hope some of this was helpful. This is great, Mark. Thanks for sharing. I'm happy to play/test this anytime!

# Game Outline

Your team is working on the World’s Smallest Online Bookstore, a site that provides the best results (just a few) for every search, not every result on earth. We’re a vulture capital funded company, so if we don’t deliver, our funding will be cut.

My goal is to help you see the effects of choices/tradeoffs on productivity and team cohesion. While some of the benefits of Agile happen at the individual level, there are many things that affect the relationships between team members, and therefore the overall cohesion and productivity of the team.

Will your team win the award of finishing the most User Stories?

Will it win the award for being most improved (i.e. best final capacity)?

## Ground Rules

* *Rounds represent 6 weeks or three 2 week sprints*
* *Our goal is to play 6 rounds == 36 weeks*
* *Each table will have 5 to 9 team members*
* *In each round you have budget for the amount of work/stuff you can do based on your team’s capacity. Some of that budget must be spent on delivering features, otherwise the business will threaten to let you go. Some of it should be spent on growing the team and their engineering skills, otherwise you don’t get more budget capacity.*
* *There is a risk of failure, and not all teams make it.*
* *There is a sense of risk and randomness. Every round, you roll two dice to see what gremlins will affect you. Human interactions are imperfect and the gremlins will simulate that.*
* *In each subsequent round, all prior actions are still available.*

Drag effects:

* **Moral/Communication Drag** - If nothing has been done to improve the team on the communication level, their capacity will be reduced by 1 Story Points (-1) at the of second round and again in each subsequent round.
* **Engineering/Tech Debt Drag** If nothing has been done to improve the team engineering practices, their capacity will be reduced by 1 Story Points (-1) at the of second round and again in each subsequent round.

Longer term change

* **Product Drag** – if nothing is done to improve Product Ownership i.e. Vision, Product Backlog Refinement etc. then the team’s success chance is reduced.

Much longer term – add Morale, Defects found and Cycle time as measures? See if Google Project Aristotle can help create new actions.

*This isn’t an action in of itself – rather the default that will happen for any uncommitted capacity.*

Commit to User Stories- **Cost**: 1 per Story Effect: Roll one die per User Story committed. 3 or higher = commitment met, User Story is completed

In an ideal world I would balance the game choices so that each round has one improvement from each of: SM, PO, Eng, Collaboration and one item for remote teams

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Title** | **Action Description** | **Cost** | **Explanation** | **Effect Next and Future Rounds** |
| **Round 1** | | | | | |
| Team Room | Set up a team room with walls | 5 | Teams form faster in a Team room because we can use eye contact to ask for permission to chat and most conversations happen in the real world | +2 Capacity per round, for 5 rounds |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Title** | **Action** | **Cost** | **Explanation** | **Next & Future** |
| **Round 2** | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Title** | **Action** | **Cost** | **Explanation** | **Next & Future** |
| Move Water Cooler | Move the water cooler/coffee near the team space. | 1 | Social conversation happens around the water cooler/coffee and tea. If team members have to leave the team space then the social benefit is diffused | +1 Capacity  *If you already have a team room this is not applicable.* |
| **Round 3** | | | | |
| **Round 4** | | | | |

Round 1:

Protected from Outside Distraction

Clarify Product Vision

Working Agreements

Build Server

Team Members On Same Floor

Round 2 – Failed to Deliver

Remote Team Avatars

All Work is done on Main or Trunk

Unit Testing

Social Time

Problem Solving Award *Needs a rename*

Backlog Refinement

Round 3: Work Harder

Observe People + Relationships

One on One

Pair Programming

Story Mapping

Refactoring

Improve Retrospectives – Change Retro agenda

Round 4 –TeamBottlenecked

Informal Cross Skilling

Send Team Members on a testing course *Rename for clarity*

Personal Productivity Bonus

New Person on the Team

TDD

Limit WIP slows team down for now and speeds them up later

How to work Psychological Safety into the game

Round 5

Work with PO to ensure PB is prioritized, up to date and remains a manageable size

Include stakeholders in updating/creating product vision and strategy

Establish Sprint Goal

Create a public impediments or slowdown list

BDD – Consider moving to round 6 – should require refactoring and be more expensive

Round 6 Go Live Soon – costs for round 6 are unimaginative

Overtime – built into the round

Improve Forecasting – move to an earlier round since its impact is mainly preventative *Reduces stakeholder interruption gremlin*

Sprint Backlog or Kanban board

Daily Scrum more effective

Learning Time/Coding Dojo

Retrospective Action Items made Concrete.

Round 7 – Go live

*Pre-Allocate Capacity for fires*

*Sacrifice One Team Member*

Work to eliminate long ignored technical problems – requires public impediment list – May also help with 25% of Sprint in Meetings

Improve DOD with Continuous Delivery

Ensemble or Mob Programming – will reduce the effect of GREMLIN\_POOR\_QUALITY

Improve Measurement system – consider measuring Morale; …

Round N introduce handling of Dependancies

DevOps – popular but not enough meaning

Remote team watercoolers – see TheBrain for more ideas

https://agilepainrelief.com/glossary/security-in-scrum

Gremlins as they currently exist:

'GREMLIN\_MANAGEMENT\_YELLS'

  | 'GREMLIN\_EMERGENCY\_ON\_OTHER\_TEAM'

  | 'GREMLIN\_NOT\_PULLING\_THEIR\_WEIGHT'

  | 'GREMLIN\_NOT\_AT\_DAILY\_SCRUM';

Interruptions From stakeholders – mitigate with including them in strategic work, also forecasting

New Story from PO MidSprint – Backlog Refinement or Story Mapping reduce. Combined avoid

Unreadable Code TDD or Pair Programming reduce – to eliminate combine

Product Backlog a Mess – WORK\_WITH\_PO\_LIMIT\_PB\_SIZE, Story Mapping reduces, Backlog Refinement reduces

Skip a Retro – mitigation – IMPROVE\_RETROSPECTIVES\_CHANGE\_AGENDA

And IMPROVE\_RETROSPECTIVES\_CONCRETE\_ACTIONS

Work not from the Sprint Backlog –Shadow Work

*Some Gremlins are made worse by PROBLEM SOLVING or BYPASS DONE*

Dissatisfied Users

PBI Not clear – *Backlog Refinement avoids – Close enough to others that we might skip*

*– Sprint Goal mitigates to a point*

Integration Problems with another team

Computer SDD Failure or Other technical problem

*Waiting for another team*

Dependency – another team injected work in your product backlog that harmed you

Poor Communication

Overtime

Unnecessary Meetings occupying up 25% of the Sprint

\*\*Check GREMLIN percentages add up to 100\*\*

Recheck the interruptions gremlin – does it have desired negative effect

Check all action effects to see if any others had the === 1 bug where 0 fell through

Test if actions protective effect fixes defects after the fact. – Not desirable

Action Ideas:

* All action code should be checked to see what types of action they affect

Put Agile Engineering practices picture on the site. Link to from game. Give its own link URL. Promise a clickable one will come later

Celebrate successes within the team

Collective Code Ownership

10 minute build and test

Firefighter award? Under a name that intrigues and catches attention?

Skip Retrospective get 2 additional story rolls?

We could code actions so that their effect is negative in the round after and then slowly increasing positive. *Example Eliminate Feature Branches capacity effect “-1” one round in the future, +1 after two rounds.*

Missing Gremlins that affect the product itself – market changes

Indecisive PO

Gremlins that could be avoided by Vision/Strategy and PB Refinement – one example unhappy customers or stakeholders.

Gremlin for Team Member who wants to be hero or work alone.

Gremlin for Team Member as a Bottleneck.

Skim SBE stories for more Gremlin ideas

Pair Programing would help both Emergency on other team and Team member leaves org.

Notes for game rules:

* More emphasis on resilience

**Distributed Teams**

## Round V – Sprint 13

“We’re live, congratulations. In addition, if we’re unlucky some teams will have to deal with a fire.”

*Facilitator’s notes*

* *Teams that do nothing to protect themselves from interruption due to fires automatically lose half their capacity this round.*
  + *Teams that pre-allocated time to deal with fires roll a die: 1-3 no additional losses; 4-5 lose one additional story; 6 lose two additional stories*
  + *Teams that dedicated one sacrificial team member roll a die: 1-3 no additional stories lost – same outcomes pre-allocating time to deal with the fires*
  + *Teams that pre-allocated capacity and sacrificed a team member, congratulations, you took care of all of the fires that came up.*
* *Teams that invested in Unit Testing two rounds ago gain 1 capacity.*
* *Check to see if anyone gains benefits from Team Room; Social Time; Moving Desks closer together; Working Agreements;*

## Round VII – Sprint 13

“Wahooo, we’re making money. We’re starting to get great customer feedback with all sorts of new ideas. Even though it’s mid-sprint and you’ve already committed, the company needs some changes.”

## Gremlins or Chance Events

These can happen in any round. At the end of every round, roll 2 dice.

|  |  |  |  |
| --- | --- | --- | --- |
| **Event** | **Effect** | **Die Roll** | **Mitigating Factors** *known only to the facilitator* |
| Team Member/Management Relationship | The team perceives their peer is being treated differently than they are and feel this is unfair.  Capacity -2 until something changes. If nothing is done in the next round, reduce capacity by 1 more point. | 2 | Teams that have established social interaction reduce this effect to -1 Capacity.  Establishing one-on-ones reduces the effect by 1 point. |
| Management yells at a team member in public for not pulling their weight | The team member who was yelled at feels their personal status was reduced. The whole team fears they will be next.  Capacity -2 | 3 | Teams whose ScrumMaster has been protecting them from distractions may ignore this. |
| We’ve had an emergency on another team, we need your best tester for a while. | Capacity - 3 for three rounds | 4 | Teams whose ScrumMaster has been protecting them from distractions reduce the number of rounds by 1.  Informal or Formal cross-training also reduces the impact by 1 Capacity Story Point.  These are cumulative effects. |
| One of the people on your team isn’t pulling their weight - not even close. However, nothing seems to get done about it. | The rest of the team is pulling together and taking this person’s work on, but it’s harming morale and productivity. Capacity - 2 | 5 | ScrumMaster who conducts one-on-one’s spots this issue early and reduces the impact to 1.  Informal or Formal cross-training also reduces the impact to 1. (Note: this is not cumulative). |
| One team member is consistently late or misses Daily Scrum. In addition, they do most of their work on their own. | Other team members are annoyed that this person feels that they’re more important than the team. Capacity -1 until something changes. | 8 | ScrumMaster who is already conducting one-on-one’s action avoids this problem  Same with, Working agreements also work because they empower team members to raise the issue. |

[reative Commons License](http://creativecommons.org/licenses/by/4.0/)  
Team Games by Mark Levison - [Agile Pain Relief Consulting](http://agilepainrelief.com)is licensed under a [Creative Commons Attribution 4.0 International License](http://creativecommons.org/licenses/by/4.0/).