Scrum do FoG e o Workflow dos Projetos

O Guia Essencial

Versão 1.0 por Leonardo Tórtoro Pereira

1. O que é o Scrum e por que eu deveria me importar?

Honorável membro do FoG que está lendo este documento e está se perguntando: “o que diabos é isso?” A resposta mais curta é: um manual para te ajudar a entender como administrar um projeto dentro do grupo de maneira eficiente e tentando evitar o máximo de riscos possíveis. Além de servir como um guia caso você esteja perdido em alguma reunião ou etapa do projeto, além de uma introdução (bem introdutória) sobre o Scrum, um método lindo que todos deviam aprender.

Atualmente estamos usando uma versão adaptada do Scrum para o nosso grupo, que, devido à nossa grande capacidade criativa, foi chamada de Scrum do FoG. Mas fique à vontade para chamá-la de João, Zé, Chunchunmaru, ou o que achar melhor. E é nessa versão adaptada que vamos focar. Sem mais delongas, vamos ao que interessa. Espero que seja uma leitura proveitosa :)

1.1. O Scrum

O Scrum é um método ágil de gerenciamento de projetos, focado no desenvolvimento de software, mas usado em várias empresas de diferentes setores. Na grande maioria das vezes, seu uso foi extremamente vantajoso, e é por isso que vamos usar no FoG.

O método é focado em times pequenos e baseia-se em cortar tudo que não é útil de um projeto e focar em começar o desenvolvimento o mais rápido possível, e a entregar versões funcionais do projeto o mais rápido possível. Documentação em excesso é algo que só gera dor de cabeça e perda de tempo. Você nunca vai conseguir documentar o projeto perfeitamente antes de começar, ao contrário do que os métodos mais antigos pregavam.

Além disso, ficar desenvolvendo algo por meses sem ver resultado algum para testar, além de ser tedioso e desmotivador, abre espaço para que quando você for mostrar para seu cliente, nada funcione como você e ele queriam. Eu sei que na graduação a gente se acostuma a programar tudo e seguir a regra do “se compilou, então funciona”. Mas para projetos grandes, como jogos, isso pode gerar muitos problemas e, pior ainda, mais trabalho do que devia.

Aliás, outra coisa que o Scrum tenta cortar é reuniões de projetos. Ninguém gosta de reunião. Quase sempre reunião é sinônimo de perda de tempo. A galera vai falar e falar e falar, nada vai ser decidido e no final todo mundo perdeu umas horas que podia estar upando. Por isso o Scrum foca em reuniões extremamente necessárias e objetivas.

Em resumo, no Scrum você tem um Scrum Master, que vai fazer a equipe seguir o Scrum e resolver os problemas que forem encontrando que os impeçam de seguí-lo. É esse o papel dele, sem segredos.

Também tem um Product Owner, um cara que vai agir pensando sempre nos interesses do cliente, e que vai guiar todo o desenvolvimento do projeto para que primeiro sejam desenvolvidas as funcionalidades que dão mais valor pro projeto. Basicamente é o cara chato que vai cortar aquelas funcionalidades que o designer achou super divertido, mas que vai custar muito tempo pra equipe, como adicionar o modo battle royale no seu clone de Super Mario Bros. Na indústria de jogos ele se assemelha bastante ao Produtor. Esse cara é que vai escrever todas as funcionalidades que devem ser desenvolvidas, ordenar a prioridade e falar, para cada entrega (ou Sprint), o que a equipe deve fazer.

Por fim, tem o Time de Desenvolvimento. A galera que coloca a mão na massa e faz o projeto. O papel deles é estimar os custos de cada tarefa, ajudar uns aos outros, desenvolver tudo que o Product Owner estabelecer para aquele Sprint (vamos falar melhor disso depois, mas considere que é o período entre uma versão "*testável*" do produto e outra). O mais importante: eles têm total liberdade de decidir entre si quem fará cada tarefa, como fará, e quando fará, desde que cumpram os prazos do Sprint.

Ok, agora que você tem uma noção superficial do que é o método, desenvolvido principalmente por Ken Schwaber e Jeff Sutherland.

1.2. Os benefícios

Você leu um monte de coisa e deve estar se perguntando: e eu com isso? Como saber se isso realmente é útil? Por que eu deveria parar de fazer as coisas do meu jeito? São todas perguntas muito importantes e todo bom cientista (ou pessoa com bom senso) deveria se fazer antes de acreditar em algo. Pois bem, eu lhe trago números! A grande maioria deles, retirados do maravilhoso livro de Sutherland sobre o Scrum (Sutherland, 2016).

Focando no que importa, o Scrum é capaz de fazer com que os desenvolvedores entreguem muito mais conteúdo com muito menos horas de trabalho. A Figura 1, abaixo, mostra a comparação dele com o Cascata. Um dos principais motivos para esse aumento na produtividade é a filosofia do Scrum de só terminar uma tarefa quando ela estiver pronta.

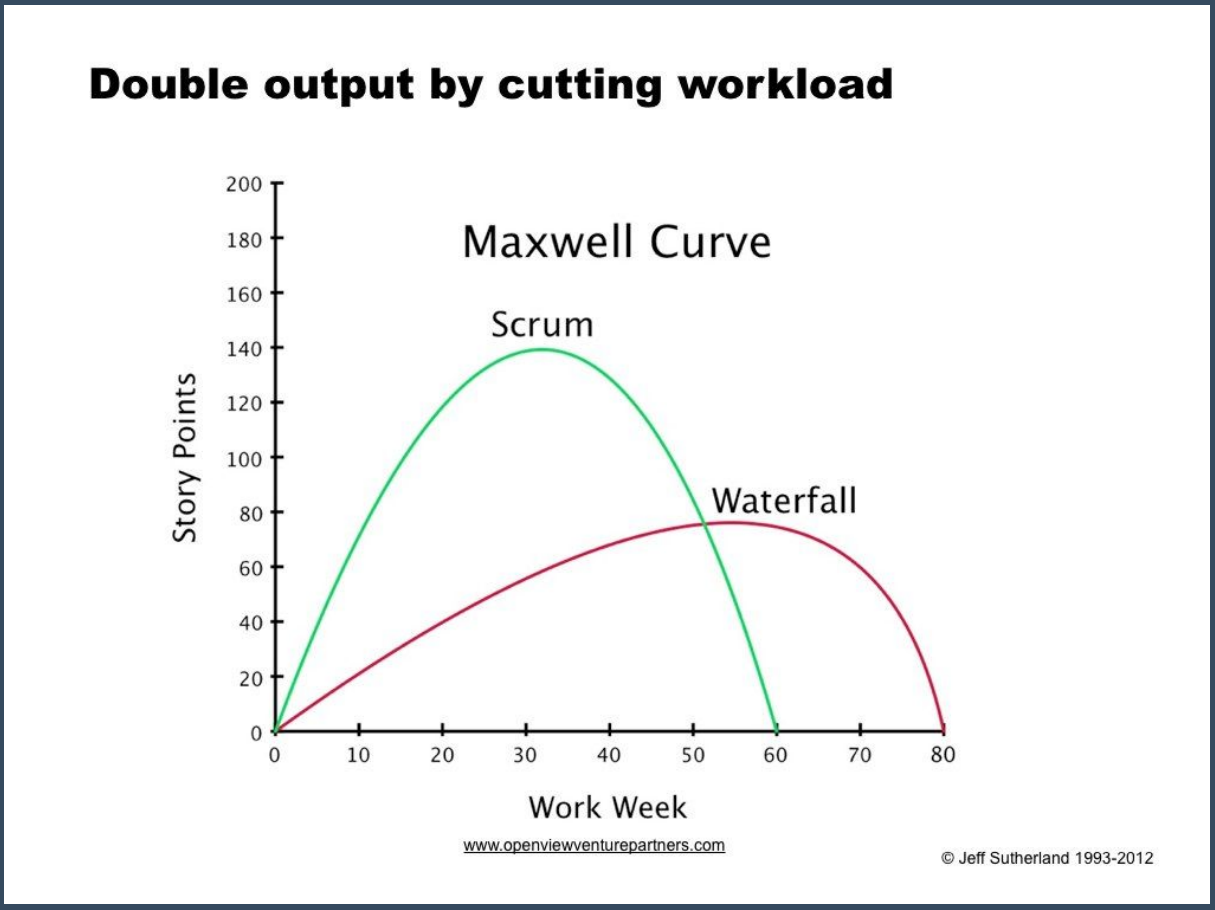


Figura 1: Comparação da produtividade do Scrum com o método Cascata

Fonte: https://34slpa7u66f159hfp1fhl9aur1-wpengine.netdna-ssl.com/wp-content/uploads/2013/02/Maxwell-Curve.jpg

Sim, você leu certo. É comum pararmos de desenvolver alguma coisa (vamos falar aqui de um programa, para exemplificar) sem testar ela totalmente, ou até mesmo testando, vendo que existem alguns bugs e falar: "ah... tá funcionando até que bem. Depois eu arrumo, vai ser rápido". O problema é que nosso cérebro não é bom de armazenar algo complexo como um código por tanto tempo. Mesmo que voltemos naquele programa depois de 2 semanas, já vamos ter esquecido o contexto em que aquele bug estava inserido.

Como diria a máxima do Scrum "faça tudo certo de primeira". Alguns dados mostram que é até 20 vezes mais rápido você fazer a mesma tarefa até todos os problemas terem sido arrumados do que deixar para resolver esses problemas depois. Na próxima seção vamos falar melhor disso, mas o Scrum tenta mudar diversos detalhes bem simples mas que melhoram muito a produtividade. E isso quer dizer que você vai ter mais tempo pra ver suas séries e animes no final do dia.

Outro grande benefício do Scrum é reduzir a incerteza do projeto e o gasto de tempo com planejamento inútil. Veja bem, planejamento é importante e muito útil. Mas se exagerarmos na documentação, é inútil. Aí vem outra premissa importante do Scrum "o mapa não é o terreno". Não importa o quão bem você conheça o

projeto, o quão experiente você é (ou acha que é). Você não vai conseguir planejar perfeitamente o projeto antes de começar a executar.

Aliás, mesmo que você, grande deus do desenvolvimento, consiga, sua equipe dificilmente terá a experiência que você tem, para chegar nesse nível, e eles vão cometer erros. Não adianta fazer algo que só funciona para você. O Scrum é todo sobre fazer o que funciona melhor para a equipe, e sempre melhorar como equipe.

Veja bem, qualquer planejamento durante o começo do projeto pode errar a estimativa de tempo tanto para 4x mais do que o tempo real, quanto para 4x menos. E essa margem de erro decresce quanto mais você desenvolve. Abaixo, as Figuras 2 e 3 comparam a evolução da incerteza ao longo do tempo em um projeto que usa o modelo cascata (Figura 2) e o Scrum (Figura 3).

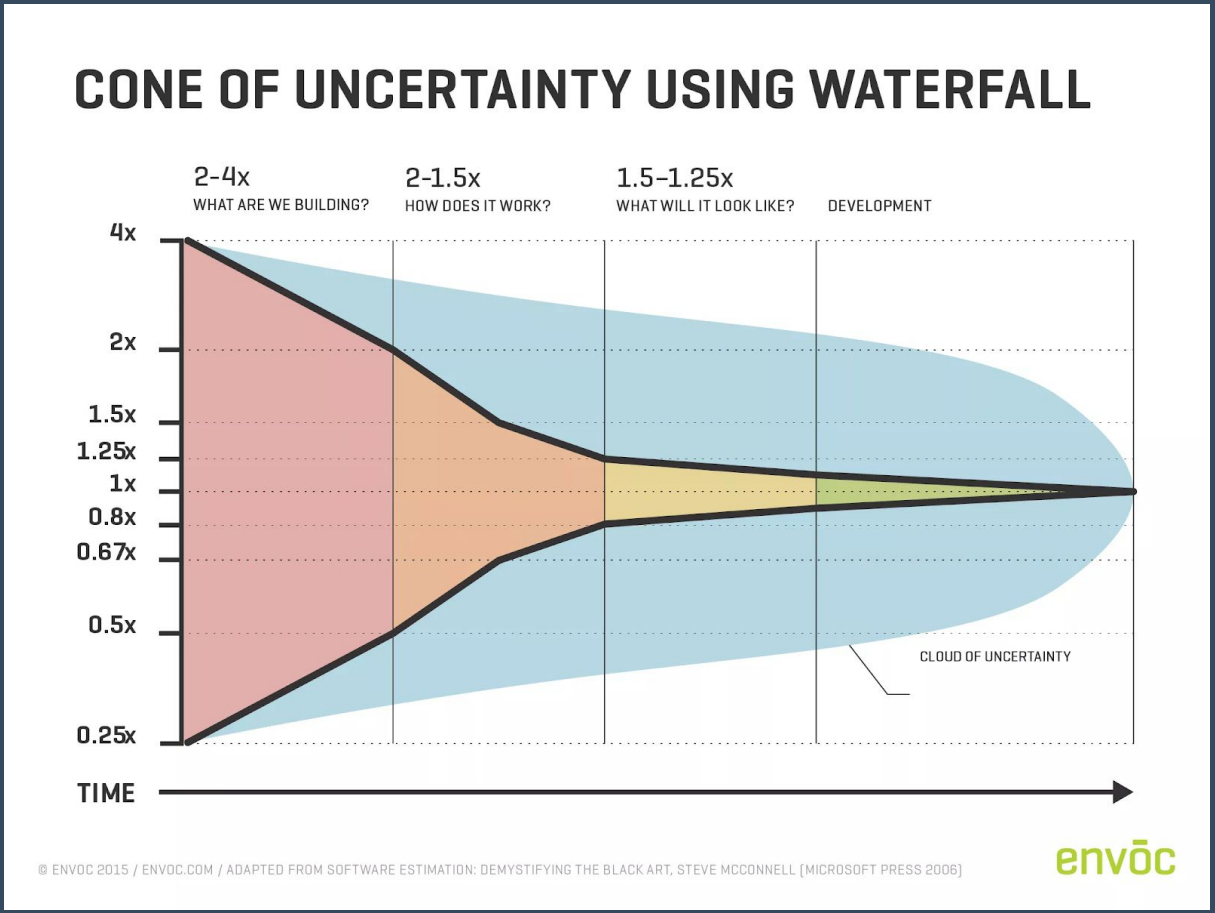


Figura 2: Cone da incerteza usando o modelo Cascata.

Fonte: https://i1.wp.com/envoc.com/uploads/files/EstimateVariability-01.png

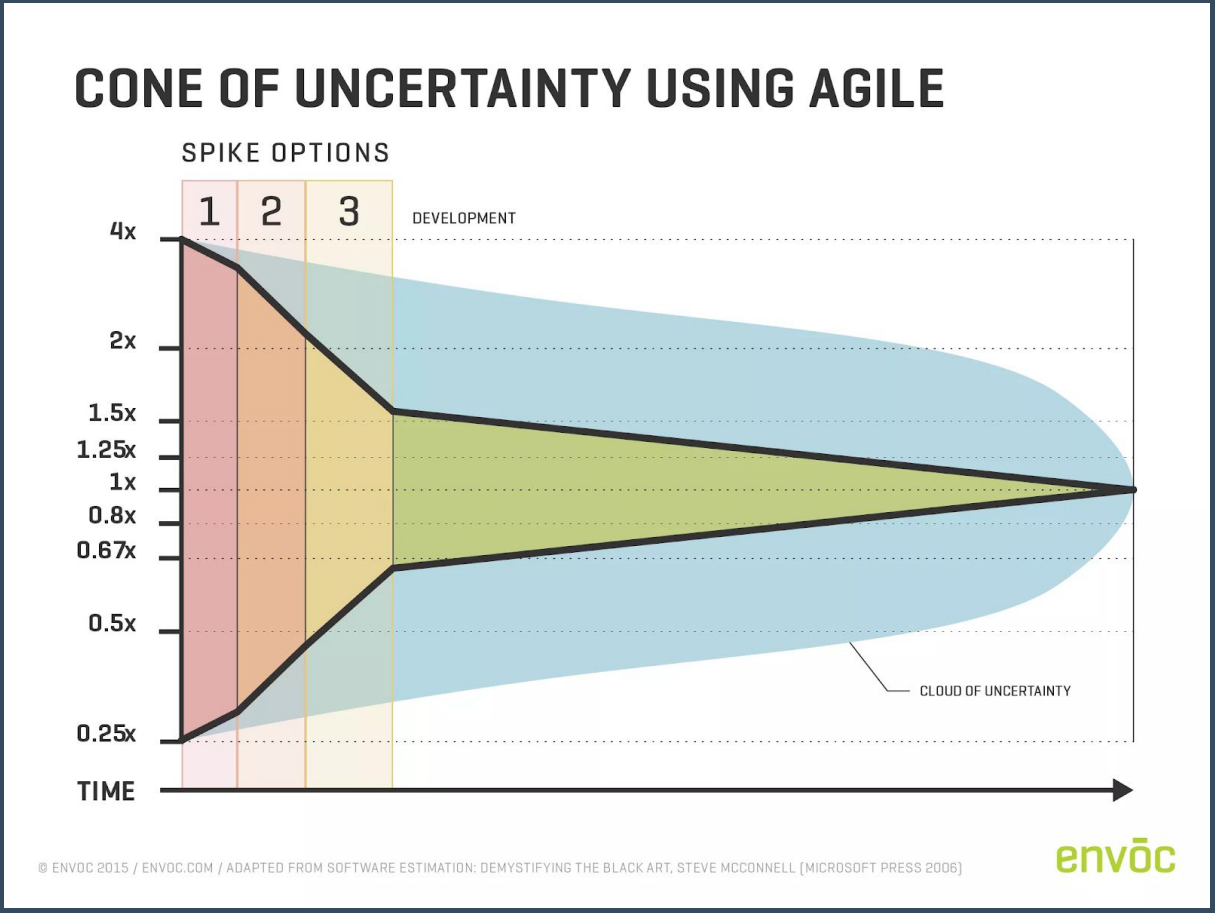


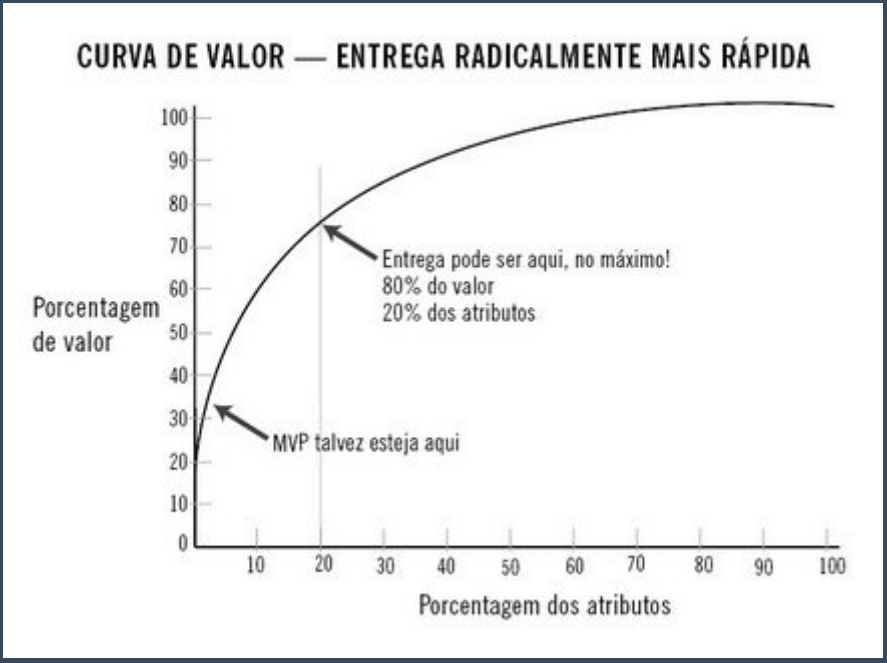
Figura 3: Cone da incerteza usando um modelo Ágil (do qual o Scrum faz parte)
Fonte: https://i1.wp.com/envoc.com/uploads/files/EstimateVariability-03.png

Fica bem claro que o método Ágil (e, por consequência, o Scrum) consegue reduzir muito mais rapidamente a incerteza para um nível aceitável para começar o desenvolvimento, mesmo que ela seja maior do que comparada ao início do desenvolvimento do Cascata. Ele consegue isso pois para ele não importa uma documentação detalhada do projeto. Sua maior documentação é o *backlog* do produto.

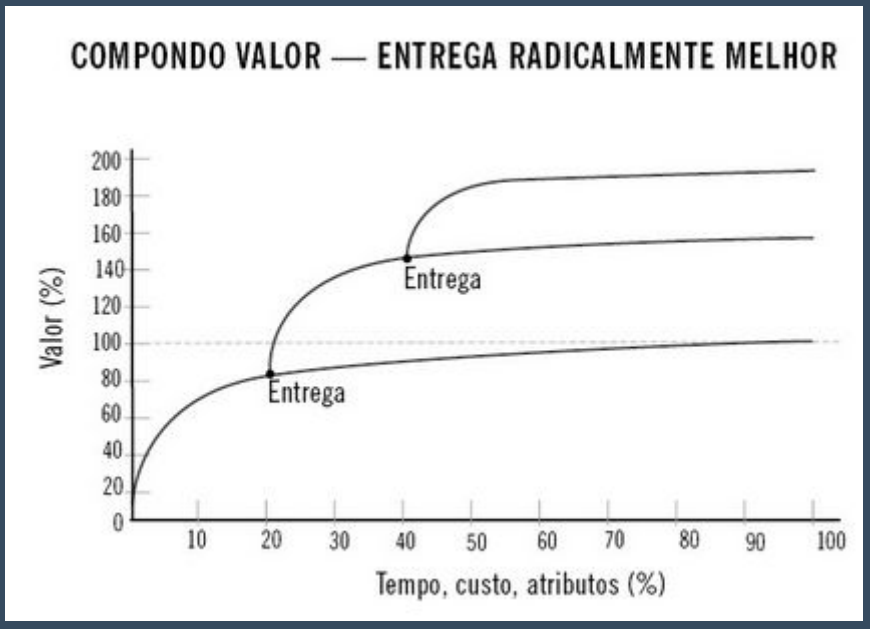
O *backlog* é um conjunto de histórias (vamos falar melhor mais tarde, mas considere como um jeito mais interessante de descrever as funcionalidades do produto) do produto. Elas podem começar abstratas, enquanto a equipe ainda não tem muita certeza de como implementá-las. Mas as mais importantes devem ser detalhadas logo no início do projeto. Uma vez definida a prioridade e detalhadas as histórias importantes, o desenvolvimento começa. É isso. Sem diagramas extensos, planilhas, diagramas de diagramas... Só um backlog definido para o primeiro Sprint.

Ao final do Sprint, um novo conjunto de histórias referente ao próximo deve estar pronto, já incorporando o feedback do que foi feito e dos problemas encontrados pela equipe. Aí essas histórias vão ser usadas pro próximo Sprint, e assim vai. O processo e sua documentação adaptam-se ao longo do projeto, e por isso a incerteza é tão mais fácil de ser reduzida ao começo.

Por fim, outro grande benefício do Scrum é ter algo de alto valor (no nosso caso, um jogo jogável e legal) em pouco tempo. Ao ordenarmos as histórias por maior valor e menor risco no *backlog* do Sprint, e sempre obedecendo às máximas "faça tudo certo de primeira" e "Fazer pela metade é igual não fazer nada", ao fim de cada Sprint o que é entregue é funcional (jogável) e com valor. Estima-se que com o Scrum é possível entregar um produto com 80% do valor em 20% do tempo (ou, com 20% das tarefas planejadas).



Fonte: https://cdn-images-1.medium.com/max/800/1*b7E8lzEYhet5-e1uO03pZg.png



Fonte: https://cdn-images-1.medium.com/max/800/1*aMup0G8w-WanhMZampaoBA.png

2. Qual o ciclo de vida de um projeto do FoG?

Antes de falarmos sobre como o Scrum funciona para ajudar o projeto a ser desenvolvido no prazo e com o menor corte de *features* possível, é importante dar uma visão geral do ciclo de vida dos projetos do FoG e algumas orientações básicas para que todas as equipes sejam bem sucedidas em cada etapa. Um bom planejamento leva a um projeto de tamanho ideal e sem muita confusão de tarefas, facilitando o desenvolvimento e a vida de todos, especialmente a do Product Owner (PO).

Desde o segundo semestre de 2018, tentamos estabelecer um padrão para a vida de todos os projetos do FoG. Algumas alterações podem surgir de acordo com as especificidades do projeto e das coordenações futuras, porém, a grande maioria dos projetos deve seguir esse ciclo de vida.

Todo projeto, tanto no FoG como na indústria, segue 4 fases de desenvolvimento diferentes e que precisam ser seguidas para evitar transtornos ao longo de sua produção. Elas são: pré-produção, produção, testes e pós-produção. No FoG, a pré-produção é separada no momento de *brainstorm* e criação da ideia do projeto (Seção 2.1), na sua devida documentação inicial (Seção 2.2) e no *pitch*, apresentado aos coordenadores e POs interessados (Seção 2.3).

Ao aprovar o projeto, a fase de desenvolvimento é iniciada e, em seu cronograma, existem os testes e a publicação. Portanto, ambas as etapas de produção e testes encontram-se na Seção 2.4, e o final da etapa de testes culmina na publicação do jogo (Seção 2.5).

Após ser publicado, o projeto encontra-se na etapa de pós-produção, que envolve a documentação e organização de códigos e assets para possível uso futuro, e na criação do documento de post-mortem (Seção 2.6). Enfim, encerra-se a vida do projeto ao término desta etapa.

2.1. A gênese

Os projetos do FoG, que antigamente tinham um fluxo completamente livre para início e fim, começaram a seguir um cronograma um pouco mais estrito desde 2018. Isso para evitar 2 problemas: o primeiro é que, sem uma data clara de término, muitos projetos sofriam de *feature creep* e extensões de tempo para terminar "detalhes" (que muitas vezes eram features inteiras novas) que, muitas vezes, nunca eram terminados de verdade. Com isso, muitos projetos que estavam com seu *core* desenvolvido em questão de meses, alongavam-se por anos no grupo, desmotivando os membros e levando a lançamento de jogos com muito pouco polimento e fator de entretenimento para os jogadores.

O outro problema é que, como os outros membros ainda estavam em outros projetos quando um deles terminava, muitas vezes os membros do projeto acabado

eram redistribuídos em grupos existentes (o que causava *overhead* para explicar todo o projeto ao membro novo e redistribuir tarefas) ou os mesmos membros faziam um novo projeto (e algumas vezes até atraíam membros de projetos em desenvolvimento, causando um desfalque naquela outra equipe).

Sendo assim, foi definido e acordado pelo grupo que os projetos do FoG devem seguir, como regra geral, um cronograma de duração semestral: durante as primeiras semanas de aula de cada semestre são realizadas sessões de *brainstorm* para definir novas ideias de projetos. Os membros dividem-se entre esses projetos e/ou candidatam-se para ser POs de projetos. O projeto faz sua documentação, apresenta ao coordenador de dev, seu suplente e os candidatos a PO interessados e, se aprovado, será desenvolvido ao longo do semestre. Todos os projetos (salvo exceções definidas pela coordenação) tem como *deadline* o fim do semestre em que foram criados (com o período de férias incluso, porém, não recomendado devido à dificuldade de comunicação e à necessidade de descanso dos membros - saúde mental é importante e deve ser levada a sério!).

Portanto, a gênese de todos os projetos atuais dá-se durante uma (ou mais) sessão(ões) de *brainstorm* ao início do semestre. É obrigação de todos os membros da área de desenvolvimento do FoG participar e, normalmente, é feito durante um final de semana. O criador da ideia deverá convencer outros membros a entrarem em seu projeto, usando suas habilidades de persuasão e comunicação.

Assim que o projeto consegue um número considerável de membros (pode variar, mas, por exemplo, podemos estar falando de 3 a 4 pessoas), eles podem começar a focar em criar uma documentação inicial adequada para seu projeto.

2.2. A documentação

Existem muitas possibilidades e variações de documentação para projetos de desenvolvimento de jogos. Para um projeto do FoG começar com boa organização e altas chances de sucesso, são pedidos dois documentos: *one-sheet* e *ten pager*.

Você pode conferir exemplos e mais detalhes sobre eles nessa apresentação ou no grande livro de Scott Rogers sobre game design (Rogers, 2012). Abaixo segue uma versão resumida de ambos os documentos.

2.2.1. O *one-sheet*

Esse documento, como seu nome do inglês diz, costuma ocupar não mais que uma folha. É um documento simples, breve e normalmente focado em visuais bonitos. Seu objetivo é atrair a atenção de possíveis interessados no seu jogo (publicadoras, jogadores, jornalistas, etc.) com o mínimo de informação textual possível.

O documento pode ocupar somente a frente ou frente-e-verso da folha, e contém algumas informações essenciais e atrativas como:

- Nome do jogo
- Para quais sistemas será lançado
- Idade dos jogadores alvo
- Classificação etária pretendida
- Resumo da história do jogo, focada em *gameplay*
- Os diferentes modos de jogabilidade
- Argumentos-chave para a venda (quais seus diferenciais)
- Produtos Competitivos (opcional e não tão usual)

  Também é aconselhável colocar logo, *concepts*, *screenshots* e/ou artes do jogo, para deixá-lo bem atrativo.

2.2.2. O *ten pager*

Apesar do nome sugerir que o documento tenha dez páginas, essa é só uma estimativa. Ele precisa ter o conteúdo necessário para dar um entendimento geral e inicial sobre o jogo para seus leitores, mas sem entrar em detalhes. Para escrevê-lo, pense que você poderia mostrá-lo a quem irá financiar seu jogo ou um desenvolvedor interessado em entrar para sua equipe: ele deve conter todas as informações relevantes para entender como será o desenvolvimento e o produto final, mas ser agradável e prático de ler.

Um exemplo de distribuição de conteúdo por página (a ordem pode ser alterada para a qual você acreditar ser melhor para descrever os conteúdos):

1. Título e logo do jogo
2. Resumo do jogo

   Visão geral contendo tema, contexto, gênero, mecânicas centrais, resumo da história, plataformas pretendidas, público alvo, monetização, escopo, influências, *elevator pitch*, objetivos do projeto (para a equipe, para o FoG, ou do próprio desenvolvimento) e uma descrição resumida do projeto.
3. Personagens

   Descrever o(s) personagem(ns) principal(is) de maneira sucinta e focando em como suas personalidades e passado afetam a jogabilidade e a história do jogo. Aqui também é um bom momento para colocar os controles, caso o jogador controle diretamente um avatar/personagem :). Coloquem imagens sempre que possível!
4. Jogabilidade

   Resumo da jogabilidade, detalhando como é a sequência de interações do jogador com o mundo do jogo: vão existir capítulos? Ou é uma sequência de níveis/*rounds*/*waves*? Algum minigame ou jogabilidade diferente para uma parte específica? Algum uso ou necessidade

específica da plataforma em que o jogo vai lançar? Não esqueçam de usar diagramas, mockups e imagens para ajudar no entendimento!

5. Mundo do Jogo

   Onde é situada a história do jogo? Quais ambientes serão apresentados? O que o jogador fará em cada lugar? Qual a relevância de cada lugar para a história? Qual o sentimento que você deseja invocar em cada lugar? Música a ser usada? Como tudo isso será conectado dentro do mundo do jogo e como será transmitida essa conexão ao jogador? Coloque mapas e/ou diagramas de fluxo para ilustrar a navegação do jogador por esse mundo

6. Experiência do jogo

   A ideia é descrever a *gestalt* do jogo, ou seja, o sentimento de "todo". Qual o sentimento que o jogo evocará do jogador ao juntar toda a experiência de arte, som, narrativa, efeitos sonoros, cinemática, mecânicas, câmera, etc.? A *gestalt* do seu jogo será de humor? Horror? Desafio? Como cada elemento irá ajudar a transmitir esse sentimento?

   Aqui também é um bom lugar para colocar um diagrama de fluxo para navegação dos menus do jogo.

7. Mecânicas

   Descreva as principais mecânicas do jogo, os principais *hazards*, *power-ups* e colecionáveis. Quais as variações? O que eles fazem? Quais as vantagens ou desvantagens que trazem?

   Se seu jogo possuir um sistema de economia, também cite aqui. Como o jogador irá coletar dinheiro e comprar as coisas? Qual a interface de compras?

8. Inimigos

   Quais inimigos são encontrados no jogo? O que torna cada um único? Como é possível derrotá-los? Foco nos chefes. Quem são? Onde aparecem? Como derrotá-los? O que é ganho ao derrotá-los? Eles têm alguma relevância para a narrativa?

9. Cinemáticas (*cutscenes*)

   Vão existir? Como serão apresentadas? Como serão criadas? Quando serão vistas pelo jogador?

10. Materiais Bônus

    Vão existir materiais bônus ou desbloqueáveis? Quais incentivos eles darão para que o jogador jogue de novo o jogo? DLC? Episódios?

Essas são as bases para o *ten pager*. O documento deve ser sucinto e de fácil leitura, não se esqueça! Se quiser alguns exemplos, veja a mesma apresentação que recomendei anteriormente. Como esse é um documento um pouco mais longo, ele não costuma ser exigido pela coordenação logo antes da defesa do projeto no *pitch*.

Porém, é bom fazê-lo o quanto antes. Assim, todos os membros do projeto tem uma noção clara do escopo, telas, mecânicas, artes, músicas e jogabilidade que deverão ser desenvolvidas.

2.3. O pitch

O projeto foi definido. O grupo foi formado. O one-sheet foi criado. Agora, é a hora de defender o seu projeto frente à coordenação de desenvolvimento e POs interessados. Para isso, você vai fazer uma apresentação curta, objetiva e muito clara apresentando seu projeto para os interessados: o *pitch*. A duração do *pitch* é muito variada e depende do contexto. Aqui no FoG nós costumamos deixar um *pitch* mais longo, de 20 minutos, para que o grupo consiga apresentar muito claramente o escopo do projeto e seu cronograma, para que a coordenação consiga observar e corrigir quaisquer problemas com *feature creep* desde o início.

O que focar no *pitch* vai de acordo com a equipe e o tipo de jogo, mas algumas coisas são essenciais. Primeiro de tudo, é preciso demonstrar confiança, conhecimento do projeto, e empolgação com a ideia. Lembre-se: você precisa convencer os outros a aceitarem seu jogo. Além disso, é sempre bom apresentar as informações relevantes do projeto (usualmente, as mesmas do one sheet e da parte do resumo do jogo - página 2 - do ten pager).

Ou seja, apresentem o nome do projeto, plataformas, público alvo, sistema de monetização, resumo da história (focada em jogabilidade), personagens, mundo, inimigos, jogabilidade, mecânicas, *gestalt*, colecionáveis, e o que mais vocês acharem essencial, relevante e diferencial do jogo.

A apresentação da equipe e das capacidades e papéis dos membros é muito importante para mostrar que vocês têm as habilidades necessárias para terminar o projeto a tempo. Afinal, cada equipe é diferente e tem seu ritmo próprio. Um bom líder precisa aceitar esse fato e adaptar o projeto e o cronograma à equipe para ter bons resultados. Falando em cronograma, também é requisitada a apresentação do cronograma do desenvolvimento do jogo, tanto numa visão geral em tabela como uma descrição das atividades (em um nível mais alto) mês a mês ou quinzena a quinzena.

Mais detalhes sobre *pitch* podem ser encontrados nesta apresentação, e esta outra aqui contém um template para um *pitch* de projeto do FoG.

Se o seu projeto foi aprovado, parabéns! Agora é iniciar o desenvolvimento com força total!

2.4. O desenvolvimento

O desenvolvimento do projeto é a fase em que todos os seus *assets* e códigos devem ser gerados e testados. Como veremos na Seção 3 com mais

detalhes, o grupo deve reunir-se semanalmente para discutir as atividades realizadas, definir quais as próximas tarefas de cada membro para a semana seguinte e discutir quaisquer problemas e impedimentos para o bom progresso das atividades.

Os projetos têm em torno de 4 meses para serem desenvolvidos (a duração do semestre letivo), com um ou dois meses extras de férias que o grupo pode tentar usar para terminar pontos do jogo que não conseguiram acabar a tempo. Porém, o trabalho nas férias é instável, já que a comunicação à distância dificulta algumas atividades e, especialmente, não é muito recomendado pois todos precisam descansar para começar bem o próximo semestre. Tentem usar esse período de férias apenas para emergências e a parte de pós-produção, que é bem mais curta.

Durante o desenvolvimento cada membro trabalhará com sua parte do projeto: arte, som, programação, design, enredo, dublagem... Mas é importante sempre manter a comunicação ativa e constante com o grupo e reportar o mais cedo possível quaisquer dúvidas ou complicações encontradas.

Lembre-se: seu grupo (e o FoG como todo) está ali para ajudá-lo a terminar seu projeto. Não hesite em perguntar e se comunicar com todos. Mesmo se ninguém do seu projeto souber como resolver seu problema, alguém do FoG provavelmente terá uma resposta ou, ao menos, um bom material para indicar.

A etapa de desenvolvimento varia muito de projeto a projeto, mas é sempre bom tentar seguir o cronograma o máximo possível e tentar identificar o quanto antes indícios de que o escopo não conseguirá ser completado, para que o PO possa ajudar a reformular o projeto. Não é sempre que isso acontece (mas pode acontecer), porém é importante corrigir o problema assim que ele surgir.

Dito isso, não se desespere ao encontrar um problema. O *Scrum* prega que para cada vez que um problema é encontrado deve-se buscar soluções, e não culpados. Se existe o hábito de jogar a culpa em outra pessoa na sua equipe (ou, inclusive, se você tende a fazer isso), tente ajudá-los a livrarem-se desse hábito o mais cedo possível. Ele é muito prejudicial à produtividade.

Aliás, não ser legal com os outros é um grande problema para a produtividade. Se alguém não está sendo educado com os outros ou está causando intrigas no grupo, peçam ajuda ao RH ou à outros membros da coordenação :). Voltando à solução de problemas de projeto, existem três perguntas que podem ajudar a resolvê-los:

- Por que isso aconteceu dessa maneira?
- Por que não percebemos aquilo?
- Como podemos trabalhar melhor/mais rápido?

Além das mudanças de cronograma, podem (e provavelmente vão) surgir algumas mudanças no design do projeto: ideias melhores para a história, mudanças no design ou personalidade de personagens, mecânicas que não ficaram tão divertidas quanto planejado, itens que acabaram tornando-se desnecessários...

Tudo isso é normal. É uma atividade importante refinar o plano ao longo do projeto. Inclusive, é uma prática internalizada pela metodologia do *Scrum*.

A documentação nunca será perfeita e sempre haverá o que mudar nela ao longo do projeto. Como diria Sutherland: *"O mapa não é o terreno"*. Por isso, a pré-produção normalmente nunca acaba, de fato. O documento de design deve ser atualizado a cada mudança de projeto e estar sempre refletindo fielmente o jogo produzido.

Outra consideração importante nessa etapa é sobre como cada integrante deve realizar cada uma de suas tarefas no projeto. Usando, mais uma vez, o *Scrum* como base, ficam algumas regras de ouro para tudo na sua vida:

- Fazer pela metade é igual não fazer nada.
  Não importa o quão cansativa parece uma tarefa, ou o quão empolgado você está para fazer a próxima parte do projeto. Se você não terminar aquela funcionalidade/*asset*, mesmo que ele esteja "90% pronto", ele não está terminado para ser colocado no jogo e, portanto, para o projeto, você ainda não terminou a tarefa. Isso pode gerar atrasos no cronograma dos outros e inclusive no seu. Afinal, você poderia ter pego uma tarefa menor para realizar antes dessa, caso seu horário estivesse reduzido naquela semana, e, ao terminá-la, gerar algo novo para o jogo.
- Faça direito de primeira
  Encontrar erros no código, arrumar *sprites* que não ficaram do jeito que você queria, ou aquela parte da música que não ficou tão boa costuma ser um processo chato. Porém, não deixe para fazer isso depois. Quando você voltar, depois de uma(s) semana(s) , seu cérebro provavelmente vai ter esquecido de boa parte do que você estava fazendo e você irá demorar muito mais para terminar aquilo do que se tivesse suportado um pouco mais a "tortura". Aliás, estatísticas sugerem que pode demorar até 20 vezes mais. Você realmente vai querer ficar 20 vezes mais tempo achando um *bug*? Por isso, sempre que terminar uma *feature* ou *asset*, faça questão de testar e verificar com a equipe se está tudo ok antes de seguir adiante.
- Realizar muita coisa ao mesmo tempo emburrece
  Faça uma tarefa por vez, até acabar. Dedicar-se a mais de uma atividade simultaneamente torna-o mais lento e piora seu desempenho. Não, ninguém é *multitask* de verdade.
- Trabalhar demais dá mais trabalho
  Trabalhar em excesso leva à fadiga e resulta em erros, o que faz com que você tenha que consertar aquilo que acabou de terminar. Ao invés de trabalhar até tarde todo dia e/ou nos fins de semana, trabalhe em um ritmo sustentável durante a semana. Mesmo que você esteja muito empolgado com o projeto e gostando muito do que está

fazendo, tire um tempo, todo dia, para descansar. Esse tempo pode variar de pessoa a pessoa, mas seja sincero consigo mesmo e respeite seu tempo, sua mente e seu corpo. Também não esqueça de tirar férias ao fim do projeto para voltar com as energias renovadas para o próximo :)

- Torne o trabalho visível
  O uso correto de softwares de criação de listas e gerenciamento de projetos como Trello ou HacknPlan ajudam a deixar o processo de desenvolvimento claro e transparente: todos sabem o que todos estão fazendo. Isso ajuda a equipe a identificar quem está com dificuldades em cumprir uma tarefa (e, então, ajudar essa pessoa) e a saber se alguma tarefa está atrasada. Além disso, é interessante enviar mensagens no canal do projeto sobre progressos importantes nas tarefas para incentivar a equipe ao ver o jogo tomando forma. Nada melhor para motivar a equipe do que ver que os outros membros estão igualmente empolgados e empenhados para criar o jogo.

Desenvolver um jogo não é uma tarefa fácil. Mas essas dicas podem ajudar, e muito, a tornar essa tarefa mais produtiva e divertida. E falando em tornar o trabalho visível, é ideal, ao fim de cada *sprint*, ter uma versão funcional do produto. Para nós, isso quer dizer uma nova *build* jogável. Ao fim do primeiro *sprint*, idealmente, a equipe terá o *core* do jogo feito e pronto para ser testado. A partir daí, podemos começar a trabalhar na publicação.

2.5. A publicação

Publicar projetos no FoG é bem simples! Nós usamos (na data que esse documento foi escrito) o itch.io, uma plataforma gratuita para distribuição digital de jogos. Para publicar um jogo lá, basta criar uma página, enfeitá-la com artes bonitas, criar um texto de divulgação para um possível jogador ler e entender melhor sobre o jogo, e fazer *upload* de uma *build*. Antes de publicar a página, converse com a frente de publicação para que eles adicionem o jogo na lista correta de jogos do FoG, revisem o texto e, assim, maximizem o alcance do seu jogo tão precioso.

Idealmente, a cada *sprint* a equipe deve subir uma *build* nova, com novos elementos de jogabilidade inseridos. Façam o possível para que ela esteja o mais livre de *bugs* possível e possa ser jogada! Além disso, para cada *build*, é aconselhável escrever um *devlog*: um pequeno texto contando sobre a experiência da equipe em desenvolver aquela nova *build*, o que foi adicionado, curiosidades e quaisquer outras coisas que possam interessar a seus fãs :) Ao fim do projeto, a *build* de *release* deve ser lançada junto com seu *devlog* final.

Para jogos *mobile*, nós temos também uma conta na Google Play Store. Para publicar seu jogo aqui, fale diretamente com a frente de publicação para eles te ajudarem.

Publicada a versão final, está na hora de dizer adeus ao projeto :,)

2.6. A morte e o além-vida

Ao fim do semestre, publicado o projeto, é hora de comemorar o sucesso! Porém, não podemos esquecer de dar a ele uma despedida digna e honrada. Para isso, temos a etapa de pós-produção. Essa é a época de organizar o projeto, arrumar os *assets* nas pastas corretas, documentar trechos de código que ainda não o foram e criar o documento de *post mortem*.

Este documento é uma análise técnica do projeto terminado. Não existe um formato específico e nem uma lista de conteúdos exatos a serem abordados. Porém, neste documento a equipe deve relatar os principais pontos do projeto que acreditam terem sido bem-sucedidos e queiram deixar como sugestão para outros seguirem, e, então, pontos que deram errado ou poderiam ter sido melhores, e o que a equipe sugere para evitar passar pelos mesmos problemas.

O *post mortem* é um documento muitas vezes deixado de lado, porém extremamente relevante! É muitas vezes na escrita deste documento que a equipe reflete sobre o projeto e adquire um aprendizado a ser levado pelo resto de sua vida sobre boas e más práticas. Aliás, um ótimo meio de aprender sobre o desenvolvimento de jogos, conhecer melhor a indústria e aprender mais sobre como escrever este documento, é ler os *post mortems* de outros jogos. O Gamasutra possui vários!

Ok, depois de escrito o documento e tudo ter sido organizado, o projeto pode descansar em paz (*RIP*). Os membros devem aproveitar para comemorar, descansar, e chamar todos os seus amigos para jogar. Divulgação é sempre algo bom :)

Agora que você conhece o ciclo de vida dos projetos do FoG, é hora de voltarmos a falar diretamente do *Scrum*. Na próxima seção vamos falar sobre todas as reuniões que devem ocorrer durante a produção do jogo e os papéis de cada membro.

3. As reuniões e os papéis de cada membro

Nesta seção vamos descrever o que deve ser feito por cada membro em cada reunião durante o desenvolvimento do projeto. Sempre seguindo os conceitos do *Scrum* e adaptando-os à nossa realidade de grupo de extensão.

Os *sprints*, período entre uma entrega funcional do projeto e outra, costumam ser quinzenais a mensais no *Scrum* original. Essa é uma característica mantida no FoG. Normalmente é preferível um *sprint* mensal, no FoG, pois não temos tanto tempo semanal para desenvolver o projeto a ponto de que em duas

semanas possa ter sido, de fato, realizado algo digno de ser chamado de uma nova versão do projeto. Porém, recomenda-se que o PO tente participar quinzenalmente de reuniões com o grupo. Mesmo que seja o Standup Diário quando não for a quinzena em que acontecerá a Retrospectiva de Sprint.

Porém, os *Standups* Diários são impraticáveis no FoG, uma vez que os membros não têm condições de encontrarem-se todo dia para discutir o projeto. Mesmo de tivessem, pouca coisa (ou nada) teria sido feita entre um dia e outro para ser discutida, já que todos possuem graduação, estágio, IC, pós-graduação, etc., além do FoG. Por isso, nossos *Standups* são semanais. Também conhecido por nós como reunião semanal.

E agora vamos aos detalhes de cada reunião. É importante lembrar que a função do Scrum Master é garantir que todas as boas práticas citadas na seção de desenvolvimento e todas as reuniões descritas nesta seção sejam cumpridas por todos os membros, da maneira correta e mais eficientemente possível.

3.1. Planejamento de Sprint

Essa é a primeira reunião de planejamento do projeto, e deve ser repetida ao fim de cada *sprint*. Nela, o PO, junto com o Time de Desenvolvimento deve definir os objetivos do próximo *sprint*: quais funcionalidades deverão estar presentes na próxima *build* e, portanto, quais histórias do *backlog* serão transformadas em tarefas na lista de *to do*. Essa decisão deve ser feita sempre priorizando as funcionalidades que agregam mais valor ao jogo (ou seja, elementos mais importantes para o *core* do jogo) e menor risco (ou seja, se algo der errado ao adicionar aquela funcionalidade ao jogo, não haverá grandes prejuízos). É importante ordenar as histórias do *backlog* pensando nessa prioridade, tanto no Trello como no HacknPlan.

É provável que muita coisa seja descartada do *backlog* por motivos de atraso no projeto ou, simplesmente, porque aquela funcionalidade não parece mais tão atrativa assim. Não se preocupe caso isso aconteça e sempre esteja preparado para modificar algumas coisas do projeto se isso ocorrer.

Logo na primeira reunião de planejamento o PO deve montar o *backlog* referente a, pelo menos, o 1º *sprint* do projeto. Então, ele terá até o próximo *sprint* para montar o *backlog* do restante do projeto. Assim o Time de Desenvolvimento poderá começar a colocar a mão na massa enquanto ele se aprofunda na criação de histórias e planejamento de todos os itens necessários para cada funcionalidade.

Durante o planejamento, é importante que o PO mantenha os “pés no chão” e seja racional. Metas impossíveis de realizar-se são deprimentes para a equipe e, além do estresse, podem causar desânimo e diminuir drasticamente a produtividade. Para ajudar a fazer boas estimativas existem diversas técnicas, uma das mais famosas é o *planning poker*. Ela consiste em, resumidamente, pedir a opinião da estimação do tempo para desenvolvimento da tarefa para todos os

membros da equipe ao mesmo tempo, ao virar cartas voltadas para baixo (para evitar a influência da opinião do outro), e usando métricas que facilitam diferenciar o tempo estimado da tarefa.

Muitas vezes utilizam-se cartas com os números na sequência de Fibonacci, em que o número escolhido indica o número de horas para realizar a tarefa. Caso exista discrepância de mais de 2 números entre as cartas mostradas (por exemplo, 2 e 8), aqueles que colocaram esses valores explicam seus motivos e o time todo joga de novo as cartas (sem mostrar os valores até ser pedido que todos revelem, simultaneamente). Isso é repetido até a diferença ser de apenas 1 número. Então, a média dos valores apresentados é definida como a estimativa de horas para a tarefa. Existem vários *apps* que podem ser usados ao invés de baralhos físicos, como esse aqui.

Ok... Mas o que é isso de histórias e tarefas que tanto foi falado e que se deve estimar? Vamos falar delas nas próximas subseções.

3.1.1. História de usuário

Uma história (de usuário) é um item do *backlog* do produto e representa uma parte do produto a ser implementada. Elas devem conter uma descrição detalhada do que deve ser implementado. Elas podem descrever qualquer funcionalidade, de complexa (como um sistema de habilidades) a simples (como a UI do menu principal), mas precisam conter tudo que pretende-se desenvolver. É comum escrevê-las do ponto de vista de um usuário do produto (no nosso caso, um jogador), como: “Como um jogador, eu quero ser capaz de matar inimigos ao pular em cima deles”, ou “Como jogador, quero ser capaz de acessar a tela de seleção de níveis a qualquer momento de uma fase ao acessar o menu de pausa”.

É importante, para a clareza, simplicidade e eficiência do *backlog*, que as histórias sigam o padrão INVEST: Independente, negociável, valiosa, estimável, sucinta e testável. Ou seja, uma história não deve depender da outra. Ela pode ter parte de sua implementação negociável, deve agregar valor ao produto, deve poder ter seu tempo para desenvolvimento estimado pela equipe (ou seja, um objetivo bem tangível e definido), deve ser descrita de maneira sucinta (ninguém quer perder 5 minutos lendo uma das 200 histórias do projeto) e precisa ter uma definição de pronta, para que o Time de Desenvolvimento possa testar se ela está feita e o PO verificar se ela pode ser realmente considerada como terminada.

3.1.2. Tarefa

As tarefas são subdivisões a nível técnico das histórias. Para cada história, durante o planejamento de *sprint*, o PO, junto com o Time de Desenvolvimento, definem quais tarefas são necessárias para realizar todas as histórias daquele *sprint*.

Por exemplo, para a história "*Como um jogador, eu quero ser capaz de matar inimigos ao pular em cima deles*", as possíveis tarefas seriam:

- adicionar colisão entre os pés do jogador e a cabeça do inimigo
- fazer inimigo receber dano na colisão (estamos considerando que o sistema de dano já foi implementado em outra história e, portanto, não precisa ser re-implementado aqui, apenas chamado)
- reproduzir animação de dano por pulo
- criar a animação de dano por pulo
- criar efeito de som para dano por pulo (Estamos considerando que todas as mecânicas e *assets* relacionadas à morte de um inimigo já foram implementadas anteriormente e somente serão chamadas, não implementadas).

Ao definir uma tarefa, é importante refletir sobre 4 pontos:

- A tarefa é pequena o bastante ou pode ser quebrada em tarefas menores?
- Ela possui uma definição clara de quando está pronta? (Quando pode ser movida para *done*)
- Ela cria um valor visível para o jogo?
- Sua descrição tem informações suficientes para um desenvolvedor ler e fazê-la?

3.2. *Standup* "Diário"

Ao começar sua primeira *sprint*, os membros do Time de Desenvolvimento devem se reunir semanalmente em nossa adaptação do *Standup* Diário. Essa é uma reunião que deve ser rápida, cerca de 15 minutos. Nela, deve-se pedir para que cada membro diga apenas essas três coisas:

- O que vocês fez semana passada para ajudar a equipe a concluir o *sprint*?
- O que você vai fazer essa semana para ajudar a equipe a concluir o *sprint*?
- Há algum obstáculo que esteja impedindo você ou a equipe de alcançar a meta do *sprint*?

Não se deve criar nenhuma tarefa nova, nem decidir alterações no *sprint*, nem criar histórias de usuário... NADA! Apenas discutir quais tarefas foram feitas, quais cada membro irá escolher para fazer para a semana atual, e reportar quaisquer dificuldades e tentar achar uma solução (lembre-se, nada de encontrar culpados e criar intrigas, apenas encontrar obstáculos e livrar-se deles. O Time de Desenvolvimento está nisso junto e, se o projeto falhar, todos serão prejudicados igualmente).

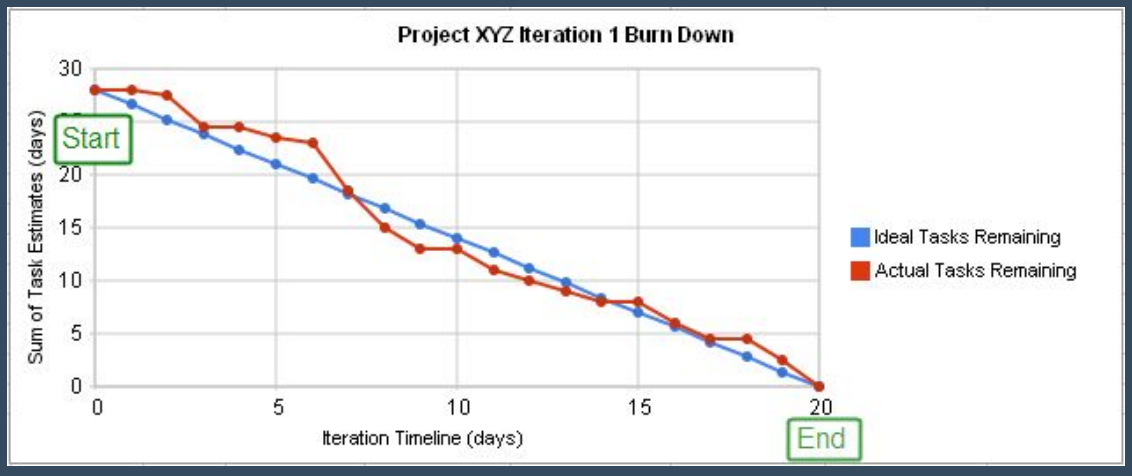
3.3. Revisão de Sprint

Ao fim de cada *sprint* é hora da reunião de revisão e, em seguida, de retrospectiva. Ambas devem ser feitas entre o Time de Desenvolvimento e o PO. Inclusive, elas podem ser feitas no mesmo dia, seguidas pela próxima reunião de planejamento de *sprint*, para economizar tempo. Além disso, o ideal é que a cada *sprint* do projeto seja colocada uma *build* nova do projeto no itch.io/Play Store. É importante lembrar a equipe de fazer isso durante a revisão, caso ainda não tenha sido feito.

A primeira coisa a se fazer na revisão é verificar se a equipe conseguiu cumprir todas as tarefas do *sprint* em questão. Se não conseguiu, não se desespere! Leva tempo até a equipe acostumar-se a trabalhar bem em conjunto e, também, cada equipe tem uma velocidade. É importante ajustar os *sprints* à velocidade das equipes, mas sempre tentando melhorar e otimizar o trabalho de todos. O Scrum Master é a pessoa indicada para ajudar a resolver quaisquer impedimentos da equipe em relação a seguir o Scrum. Se você notar algum problema nesse sentido, avise à coordenação de desenvolvimento!

Para medir a velocidade da equipe durante o *sprint*, é importante somar os pontos de todas as histórias concluídas. Assim, é possível usar a métrica para estimar quando o projeto acabará. Mas lembre-se que a velocidade costuma aumentar ao longo do projeto pelos motivos citados acima.

Além disso, é interessante fazer um *burndown chart*. É um gráfico simples que mede quantas tarefas são feitas em cada período de tempo no projeto e quantas ainda restam. No dia 0 inicia-se com todas as tarefas e, a cada dia/semana/mês, mede-se quantas restam. A Figura abaixo mostra o exemplo do gráfico para um projeto hipotético com 28 tarefas e 20 dias para ser desenvolvido.



Fonte:
https://en.wikipedia.org/wiki/Burn_down_chart

Caso a medida de velocidade mostre que o projeto está grande demais para a equipe lidar, é importante que três perguntas sejam feitas:

- Existe algo que possamos fazer de maneira diferente para acelerar?
- Podemos nos livrar de alguns itens do *backlog*? Há algo que possamos delegar a outros?
- É possível não fazer algumas coisas? Podemos reduzir o escopo do projeto, mesmo que minimamente?

Ainda é importante verificar se todos trabalharam de maneira relativamente similar ou algum membro da equipe está trabalhando muito acima da média de modo a corrigir erros e problemas de outros. Esse tipo de "heroísmo" não é bom para o projeto. Isso indica que as tarefas podem estar sendo grandes demais para a equipe, ou que alguns membros não estão sentindo-se felizes com o projeto e deixando de trabalhar ou que esse membro "herói" não sabe repartir tarefas.

Por fim, é válido lembrar que problemas técnicos do projeto não são encargos do PO para resolver. Ele tem como responsabilidade criar histórias, dividi-las em tarefas adequadas, definir as *sprints* e re-estruturar o escopo do projeto quando necessário. Seu objetivo final é garantir que seja entregue o melhor produto possível para o cliente (no nosso caso, o jogador), e que a equipe trabalhe em seu máximo de produtividade. Muitas vezes o PO não tem conhecimento técnico para ajudá-los.

O Time de Desenvolvimento pode procurá-lo para pedir ajuda com dúvidas técnicas, caso não saiba a quem recorrer. Mas o PO deve encaminhá-los a alguém dentro do FoG competente para responder a dúvida. Ele não deve assumir que é responsabilidade dele sanar toda e qualquer dúvida técnica que surja. Cuidado com a sobrecarga e acúmulo de tarefas.

3.4. Retrospectiva de Sprint

Essa reunião também ocorre ao final de cada *sprint* e tem um viés mais humano, em contrapartida da revisão. Ela é uma reunião para a equipe fazer uma auto-avaliação ao invés de avaliar o desenvolvimento do projeto. É interessante fazer 4 perguntas a cada membro do Time de Desenvolvimento:

- 1 a 5: como se sente em relação ao seu papel no projeto?
- 1 a 5: como se sente em relação ao projeto/fog como um todo?
- Por que se sente assim?
- O que te deixaria mais feliz no próximo sprint?

Lembrem-se, ao responder essas perguntas, mais uma vez que não deve-se arrumar culpados ou criar intrigas. O objetivo dessa reunião é encontrar e livrar-se de obstáculos. Apenas isso.

Terminada a retrospectiva, é hora de realizar o próximo Planejamento de Sprint e repetir o ciclo, até o projeto acabar. Na próxima seção vamos colocar um

exemplode calendário semanal de um projeto, para exemplificar as reuniões que discutimos.

3.5. O calendário

Vamos exemplificar o cronograma de reuniões de um projeto do FoG. Supondo que o semestre tenha 4 meses de aula + 1 de férias, e que durante as férias o grupo foque apenas na pós produção, nosso calendário de reuniões deverá abordar apenas os 4 meses de desenvolvimento. Portanto, vamos considerar meses de exatamente 4 semanas, totalizando 16 semanas de desenvolvimento. O calendário contém as reuniões que deverão ser realizadas a cada semana, sendo *P* o Planejamento de *Sprint*, *D* o *Standup* Diário, *Rv* a Revisão de *Sprint* e *Rt* a Retrospectiva de *Sprint*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P																
D																
Rv																
Rt																

E é isso. Isso é tudo que você precisa saber para fazer projetos bem sucedidos ao saber em detalhes como funciona o *workflow* dos projetos do FoG e como nosso Scrum é estruturado. Para se aprofundar mais você sempre pode ler as fontes da referência e que foram indicados ao longo do texto.

Boa sorte e vida longa ao FoG!

Referências

[1] Sutherland, Jeff. Scrum: a arte de fazer o dobro do trabalho na metade do tempo. Leya, 2016.
[2] ROGERS, Scott. "Level UP: um guia para o design de grandes jogos." São Paulo: Blucher (2012).