Universidade de São Paulo
Instituto de Matemática e Estatística

Trabalho de Conclusão de Curso
Monografia Preliminar

Disciplina MAC499
Trabalho de Formatura Supervisionado
Prof. Carlos Eduardo Ferreira

Orientador:
Prof. Flávio Soares Corrêa da Silva

Vinícius Kiwi Daros
Nº USP: 5893980

19 de setembro de 2011

Conteúdo

Parte Objetiva

Introdução

Assim como qualquer atividade realizada em grupo, desenvolver jogos eletrônicos é uma tarefa que pode ser abordada de inúmeras formas. Acompanhando o trabalho do *USPGameDev* durante o ano de 2010, foi possível ver como uma abordagem ingênua sem uma metodologia clara de trabalho pode ser ineficiente, apesar de mesmo assim ter levado a um resultado satisfatório. Mas introduzir uma metodologia de desenvolvimento realmente aumenta a produtividade? Em caso afirmativo, quão expressivos são esses benefícios?

Este estudo visa responder essas questões. Para tanto, será mostrada uma comparação entre duas fases de um mesmo grupo, o *USPGameDev*. Na primeira dessas fases, os desenvolvedores não aplicavam qualquer tipo formal de metodologia, trabalhando de forma relativamente livre. Já na segunda fase, os conceitos de *Scrum* foram introduzidos e o grupo trabalhou seguindo essa metodologia.

Além da comparação do que deu certo e errado entre as duas etapas, também será dada atenção a como o *Scrum* pode ser adaptado ao contexto de desenvolvimento de jogos. A motivação para essa observação reside nas diferenças existentes entre um aplicativo convencional e um jogo. Tais diferenças vão desde os requisitos até as especializações dos envolvidos nos projetos.

O que é Scrum

Scrum é um framework de gerenciamento de projetos usado principalmente no desenvolvimento ágil de software, apesar de poder ser aplicado em projetos de diferentes outras áreas. O processo de trabalho usando Scrum é iterativo e incremental, se baseando em ciclos e objetivando a entrega constante e rápida de versões de software que cada vez atendam melhor às necessidades do cliente.

Nesta seção, será apresentada uma visão geral dos principais conceitos do Scrum, assim como suas práticas e fluxo de trabalho. Apesar desta seção não se restringir ao escopo de jogos eletrônicos, o foco das explicações será projetos de desenvolvimento de software.

História

Em 1986, Hirotaka Takeuchi e Ikujiro Nonaka publicaram um artigo intitulado “The New New Product Development Game”, no qual são descritas as características de gerenciamento do processo de desenvolvimento adotadas por algumas empresas que se destacavam

por lançar produtos inovadores de forma rápida e com grande sucesso. A estratégia dessas empresas se baseava em mover todo o time como uma unidade em direção ao objetivo, chamada então de *abordagem holística ou rugby*.

Em 1990, Peter DeGrace e Leslie Hulet Stahl publicaram o livro "Wicked problems, righteous solutions", onde denominaram essa abordagem por *Scrum* devido à formação do rugby de mesmo nome, representada na figura 1. Essa foi a primeira vez que Scrum foi proposto como modelo de desenvolvimento de software.

Finalmente, em 2002, Ken Schwaber e Mike Beedle escreveram o livro "Agile Sofware Develpment with Scrum" baseando-se nas experiências profissionais que tiveram ao aplicar as propostas de DeGrace e Stahl na prática. Este foi o título que popularizou o Scrum.

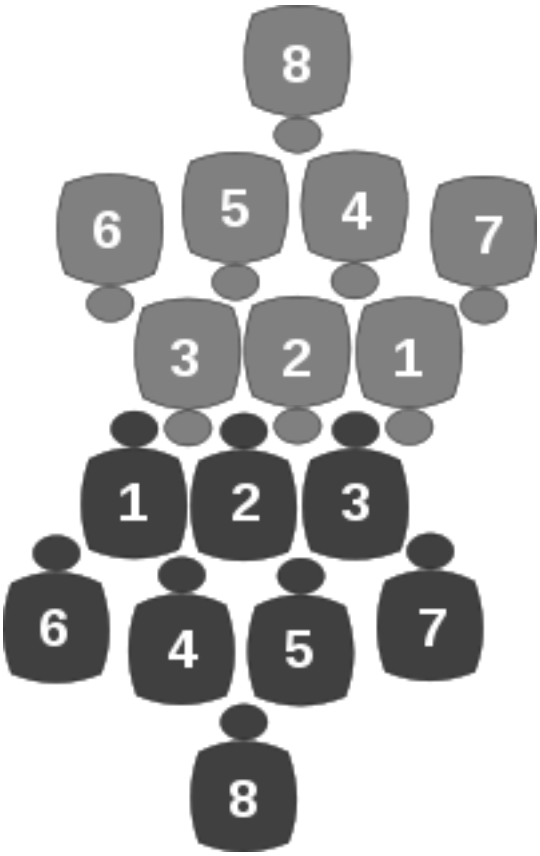

Figura 1: Formação Scrum

Visão geral

Antes de detalhar os elementos do Scrum, será dada uma breve descrição geral do fluxo de atividades envolvidas no processo. Isso se justifica pois ter em mente uma ideia do ciclo de trabalho como um todo ajuda na compreensão da influência de cada prática e como elas interagem entre si. A princípio, poder-se-a ter a impressão de que vários conceitos estarão sendo deixados soltos, sem explicação. Entretanto, todos os conceitos citados receberão atenção individual posteriormente.

Como já citado no início da seção, o Scrum é interativo, o que implica na existência de ciclos de trabalho, denominados *sprints* e que duram de duas a quatro semanas. Em sprint, o *product owner* verifica na lista de funcionalidades a serem implementadas, chamada de *product backlog*, aquelas que agregam maior valor ao produto, isto é, tem maior prioridade. Para cada item selecionado, avalia-se quais tarefas menores precisam ser cumpridas para que a funcionalidade seja implementada. Em cada dia durante o sprint, o time de desenvolvedores se reúne no *encontro diário* (daily meeting) para discutir sobre as tarefas e acompanhar o andamento global do time. Em seguida, trabalha-se objetivando cumprir as tarefas do sprint backlog. Ao fim do sprint, se espera que a nova versão do produto tenha as funcionalidades propostas e que seja utilizável. Nesse momento, o product owner verifica se a nova versão atende ao que foi requisitado, reordena o product backlog e dá início a uma iteração.

Essa sequência de eventos de um sprint é ilustrada na figura 2.

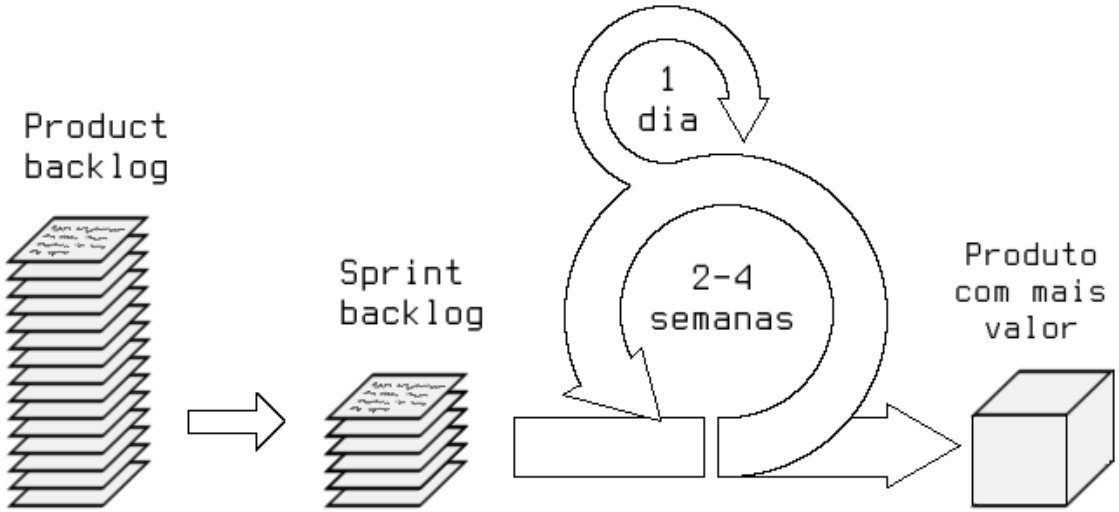

Figura 2: Processo de Scrum

Princípios do Scrum

O Scrum é um framework para gerência de projetos. Assim sendo, é natural presumir que suas práticas possam ser alteradas ou até mesmo não usadas de acordo com as necessidades e experiências de cada time. Entretanto, existem alguns princípios fundamentais que devem ser preservados.

Inspeção e adaptação: Deve-se constantemente inspecionar os artefatos e processos em busca de variáveis desfavoráveis, assim como mudanças no contexto do desenvolvimento, conhecimento e necessidades do cliente. Uma vez encontrada alguma dessas situações, ajustes devem ser feitos o quanto antes, de modo a minimizar desvios de rota e garantir que o projeto sempre esteja progredindo em direção a um produto de valor crescente para o usuário;

Timeboxing: As atividades realizadas usando-se Scrum sempre possuem prazos, seja em semanas, dias ou minutos. Esses prazos são essenciais para que tanto desenvolvedores quanto os demais envolvidos no projeto possam se sincronizar;

Priorização: Nem todas as funcionalidades a serem implementadas tem o mesmo nível de importância e algumas delas nem mesmo geram valor real ao usuário do produto. Por essa razão, as tarefas a serem cumpridas devem ser classificadas por níveis de prioridade;

Auto-organização: Os times de desenvolvimento devem ter o poder e a responsabilidade de se auto-organizarem para que consigam cumprir dentro do prazo as tarefas com as quais se, comprometeram. Para tanto, eles tem autonomia para modificar, adotar ou abolir quaisquer atividades e processos.

Product Backlog

O backlog do produto consiste em uma fila de todas as funcionalidades, requisitos e quaisquer outras atividades que representam um trabalho a ser feito ou a ser implementado dentro do sistema. Essa fila é ordenada por prioridade, sendo que os itens mais prioritários ficam no início. Quanto mais próximo ao início está um item, maior sua urgência. Mas também, mais discussão houve a seu respeito e maior é seu nível de detalhamento.

Os itens do backlog, também chamados de *histórias*, devem ser priorizados seguindo os critérios abaixo:

1. **Valor:**
   O principal critério para escolher qual história tem maior prioridade é o valor que ela traz para o usuário final. Além disso, algumas histórias não trazem valor em si, mas aumentam a produtividade do time. Isso também é considerado uma forma de valor;

2. **Custo:**
   O custo é medido basicamente pelo esforço em se desenvolver uma funcionalidade. Assim, pode haver situações nas quais alguma história pode gerar grande valor, porém seu custo a inviabiliza. Em geral, quando duas funcionalidades tem o mesmo valor, dá-se maior prioridade àquela cujo custo é menor;

3. **Risco:**
   Risco implica em incerteza com relação ao valor e ao custo. Por esse motivo, tarefas mais arriscadas ganham maior prioridade, de modo que as incertezas sejam sanadas;

4. **Conhecimento:**
   Em alguns momentos, não se tem informação suficiente para estimar uma história. Nesses casos, pode-se fazer um *Spike* - tarefa de curto prazo - para obter mais conhecimento sobre os custos e riscos. Por exemplo, fazer um protótipo para verificar a dificuldade em implementar troca de pacotes por rede é um spike que pode ajudar muito a estimar o esforço necessário para desenvolver um jogo multiplayer.

As histórias no backlog do produto são de altíssimo nível, não carregam muitos detalhes e apenas descrevem o resultado desejado após sua conclusão. Por conta dessas características, essas histórias são denominadas por *histórias épicas*. A título de ilustração, segue abaixo um exemplo de história que poderia estar no backlog do produto:

> *Como jogador, quero que o cenário da cidade tenha várias pessoas andando de um lado para o outro, pois com isso passa-se melhor a sensação de se estar em uma metrópole.*

Sprint Backlog

Assim como o product backlog, o sprint backlog é uma fila de tarefas a serem cumpridas. Entretanto, existem algumas particularidades.

Primeiramente, as tarefas no sprint backlog são menores e mais simples, pois são partes de uma história épica que foi quebrada. Além disso, essas tarefas possuem uma estimativa de quanto tempo será necessário que cada uma delas seja cumprida, sendo que soma dessas estimativas não deve superar a capacidade de trabalho do time dentro de um sprint.

Outra característica particular do sprint backlog reside no fato de que nenhuma tarefa pode ser adicionada à fila durante o andamento do sprint. Desse modo, uma vez que o time tenha se comprometido a entregar certas funcionalidades em determinado prazo, os desenvolvedores ficam protegidos de receber exigências não planejadas e que atrapalhem o andamento do projeto. Com isso, tem-se a certeza de que o único momento no qual novas tarefas são adicionadas ao sprint backlog é durante a reunião de planejamento do sprint.

Cartões de Tarefas

Sprint

Priorização

Planejamento

Acompanhamento

Review

Retrospectiva

Release

Papéis

O processo de desenvolvimento envolve pessoas com diferentes habilidades e que desempenham diferentes funções. A fim de maximizar a produtividade da equipe, o Scrum promove uma distinção bem definida das responsabilidades que cada pessoa deve ter. Essa distinção é feita através de papéis a serem desempenhados. Os três papéis adotados no Scrum são product owner, scrum master, time de desenvolvimento. Abaixo, cada um deles é descrito com mais detalhes.

Product owner

O product owner é, dentro do time, o representante das necessidades dos clientes. Ele serve de interface de comunicação, passando de forma clara para os desenvolvedores o que precisa ser feito e apresentando para os clientes as questões técnicas envolvidas no processo de produção.

Por esse motivo, o product owner é o responsável por lidar com o product backlog. É ele quem define quais tarefas devem entrar na fila, assim como avaliar a prioridade de cada uma delas antes de cada sprint. Ninguém mais deve realizar essas atividades. Com isso, tem-se uma situação na qual ninguém faz requisições de funcionalidades diretamente aos desenvolvedores e nem o time gasta esforços implementando itens que não foram previamente discutidos e cuja necessidade não tenha sido verificada. Ou seja, para que um item seja incluído no product backlog, é preciso convencer o product owner de que será agregado valor ao produto.

Como responsável pelo product backlog, o product owner deve garantir que todas as histórias sejam claras e que o time entenda exatamente o que se espera de cada uma delas. Além disso, outra obrigação do product owner é participar do planejamento dos sprints e releases, sempre direcionando o andamento do projeto rumo a versões que atendam as necessidades mais críticas dos clientes.

Outra obrigação do product owner é garantir que o time como um todo compartilhe a mesma visão que se tem sobre o projeto. Isto é, tanto programadores quanto designers e artistas precisam conseguir visualizar o mesmo resultado esperado, ou de forma mais próxima possível. Assim, cabe ao product owner certificar-se que as histórias representam a mesma coisa para todos os integrantes da equipe.

Dadas essas responsabilidades, a pessoa que for desempenhar esse papel precisa ter uma visão ampla das necessidades do projeto, além de ser capaz de traduzir as necessidades dos clientes em tarefas e conseguir tirar as dúvidas dos desenvolvedores em relação a questões de negócio.

Scrum master

O scrum master é a pessoa responsável por introduzir e colocar em prática o Scrum. Primeiramente, ele deve mostrar aos envolvidos no projeto como o Scrum funciona, quais os principais conceitos, quais práticas e atividades precisam ser realizadas e a motivação para a adoção desse método de trabalho. Ao longo do desenvolvimento, o scrum master será a referência para as outras pessoas e deve tirar todas as dúvidas de modo a compartilhar o máximo possível seu conhecimento.

Atuando como um verdadeiro instrutor, o scrum master ajuda cada um a entender o papel que tem no desenvolvimento do produto e de quais formas podem contribuir melhor, ou até mesmo como não atrapalhar os demais. À medida que essa mentalidade está distribuí e as pessoas passam a ser mais conscientes das maneiras como podem atuar melhor, maximiza-se o valor que o time consegue agregar ao produto a cada sprint e a velocidade do desenvolvimento aumenta, assim como a qualidade dos resultados parciais.

Outra incumbência de grande importância, senão a mais importante, atribuída ao scrum master é garantir que o time esteja sempre no ambiente mais favorável a realização de suas atividades e ao cumprimento de suas tarefas. Esse é um trabalho que envolve desde

minimizar interferências externas, tais como gerentes exigindo que funcionalidades não planejadas para o sprint sejam implementadas para o dia seguinte, tanto quanto garantir que o time tenha material de escritório como cartões para escrever as histórias, lousas onde colar os cartões, softwares instalados nas máquinas de trabalho, dentre muitos outros.

Além disso, o scrum master deve se esforçar para rapidamente remover quaisquer impedimentos que eventualmente surjam. Entende-se por impedimento qualquer dificuldade que bloqueie o progresso do time no desenvolvimento do projeto ou que não permita a realização de alguma prática do Scrum. Por exemplo, não ter acesso à internet no local de trabalho é um grande impedimento.

O Scrum master também deve monitorar o progresso do grupo e, com base nesse acompanhamento, sugerir a implantação, modificação ou até mesmo abolição de algumas práticas. Com esse embasamento, é possível de questionar de forma consciente algumas regras sugeridas pelo Scrum e fazer as adaptações e flexibilizações das práticas de modo a elas se moldarem melhor à equipe e cumprirem seu objetivo primário, aumentar a produtividade e não atrapalhar.

Em suma, a função do scrum master é guiar o time em direção ao aprendizado e mecanismos do Scrum. Com isso, a necessidade de uma pessoa desempenhando esse papel tende a diminuir.

Time

Adaptações para o contexto de games

USPGameDev

Antes de prosseguir para as seções relacionadas a análise do uso e não uso de Scrum, é importante conhecer um pouco sobre o contexto onde essas observações foram feitas. Por esse motivo, os próximos parágrafos darão uma breve descrição do que é o USPGameDev.

História

Em 2009, o orientador pedagógico da Escola Politécnica da USP (Poli), Giuliano Salcas Olguin, iniciou um trabalho de acompanhamento e avaliação do curso de Bacharelado em Ciência da Computação (BCC) do Instituto de Matemática e Estatística da USP (IME). Um dos objetivos desse trabalho era fomentar a iniciativa dos alunos para trabalhos de extensão. Nesse cenário, em novembro daquele ano, surgiu a proposta de reunir alunos do BCC e da Engenharia de Computação da Poli interessados em estudar e desenvolver jogos eletrônicos de modo a constituir um grupo de estudos dessa área. Assim, foi criado o USPGameDev, cuja composição inicial era formada por aproximadamente vinte pessoas, sendo apenas um aluno de engenharia e os demais alunos do BCC.

Durante o ano de 2010, o grupo se dedicou exclusivamente a duas atividades: Construir um framework para o desenvolvimento de jogos em duas dimensões para computador, posteriormente batizado de *UGDK*, e implementar um de jogo, intitulado *Horus Eye*, utilizando-se esse framework. O UGDK levou por volta de sete meses para ficar pronto

para o uso e o Horus Eye teve sua primeira release após aproximadamente três meses seguintes.

Entre dezembro de 2010 e o início de abril de 2011, o grupo passou por um período de seleção de novos membros e uma fase de baixa atividade. Apesar desse intervalo de pouca produção, a chegada de novos integrantes trouxe ao USPGameDev uma maior diversificação de talentos. Com isso, o time passou a cobrir competências outrora deficitárias, como roteiro e artes gráficas.

Local de trabalho

No início, o grupo não tinha um local fixo para se encontrar e fazer as reuniões de trabalho. Assim, sendo a maioria dos integrantes alunos do BCC, usava-se de improviso salas de aula do IME que estivessem desocupadas. Apesar do espaço físico satisfatório, as salas de aula só eram lugares apropriados para reuniões voltadas para definição de propostas, debates de ideias e planejamento. Para trabalhar no desenvolvimento de fato, as principais desvantagens das salas eram: falta de computadores e tomadas suficientes para ligar os notebooks pessoais dos participantes; cadeiras universitárias não acomodavam adequadamente os notebooks; falta de acesso à internet.

Depois de cerca de um mês, foi autorizado o uso da sala de reuniões da Orientação Pedagógica (OP) da Poli. Nessa sala, as necessidades da equipe eram muito melhor atendidas. Havia mesas grandes o suficiente, tomadas e era possível acessar a internet através de uma rede sem fio. Entretanto, eventualmente a sala não estava disponível, pois sua finalidade não era atender ao grupo, mas sim à OP.

No final de outubro de 2010, depois de grande empenho do Giuliano em resolver essa questão, o USPGameDev recebeu uma sala fixa e exclusiva no prédio do Biênio da Poli. Essa nova sala acomodava muito bem o grupo, porém o sinal da rede sem fio era intermitente e por várias vezes completamente nulo. No início de 2011, com ajuda do Prof. Carlos Eduardo Ferreira do IME, novos equipamentos de rede foram comprados e puderam ser instalados na sala de modo a resolver esse problema.

Não se enfrentou mais questões referentes ao local de trabalho até junho de 2011, quando começou uma reforma na sala a fim de transforma-la em um laboratório. Até o fim das obras, que ocorreu no começo de agosto, as reuniões voltaram à situação nômade da época de criação do grupo. Com tudo, ao fim da reforma, a sala do USPGameDev passou a conter uma infraestrutura que atendia plenamente suas necessidades: acesso à internet, tomadas, mesas adequadas, computadores e quadros brancos nas paredes.

Pessoal

Horus Eye

Sem Scrum

Quais eram as espectativas

Como foi o trabalho

O que deu certo

Problemas enfrentados

Os problemas mais imediatos que surgiram ao tentar reunir um grupo de aproximadamente vinte alunos em reuniões periódicas dentro do espaço da universidade foram relacionados a infraestrutura.

Resultado obtido

Com Scrum

Quais eram as espectativas

Como foi o trabalho

Resultado obtido

O que deu certo

O que deu errado

Comparações

Conclusão

Bibliografia

Parte Subjetiva

Desafios e frustrações

Relação entre disciplinas e o TCC

Próximos passos