Proposta de Um Simulador para Auxiliar no Processo de Ensino do Scrum

Pedro Rauiz E. Gestal[1], **Rodolfo M. de Barros**[1]

[1]Universidade Estadual de Londrina (UEL)

Resumo. *Este artigo propõe um modelo de simulação de curta duração (simulações de duas a quatro horas) para o ensino do* Scrum *(Métodologia Ágil de Desenvolvimento) e seus papéis. Este modelo de simulação possui como foco a tomada de decisão rápida e o* feedback*, tanto positivo quanto negativo, a estas decisões. Em conjunto com este modelo, também é proposto um criador de simulações a ser utilizado por professores ou instrutores para criar simulações seguindo este modelo. O artigo também explora brevemente a possibilidade de criar um modelo mais longo de simulação, na qual o projeto simulado também deveria ser implementado.*

Abstract. *This article proposes a short simulation model (two to four hours long simulations) for the teaching of* Scrum *and its roles. This simulation model possesses, as its focus, quick decision making and this decision's both positive and negative feedback. It's proposed, alongside this model, a simulation creator to be used by the teacher to create simulations that follow the proposed model. This article also briefly explores the possibility of a longer model of simulation, where the simulated project would also be implemented.*

1. Introdução

Metodologias ágeis como o *Scrum*, *XP* ou *Feature-Driven Development* vem sido amplamente estudadas. Principalmente devido ao seu uso e sucesso quando aplicadas em um abiente corporativo. O Manifesto Ágil (*The Agile Manifesto*)[Beck et al. 2001] pode ser visto como a linha que a maioria das metodologias ágeis seguem e, este, descreve uma lista de prioridades. A maioria destas prioridades informalizam o processo de desenvolvimento porém mantendo o cliente no centro. Em outras palavras, o cliente está diretamente envolvido no processo de desenvolvimento: ele valida as versões funcionais do sistema e pode requisitar mudanças a cada ciclo.

Parafraseando, as metodologias ágeis demonstram uma maneira de desenvolver *software* onde o programa sendo desenvolvido é visto como um objeto sempre em evolução e mudança. Devido a esta visão, é possível adaptar-se as mudanças no mercado consideravelmente mais fácil. Isto fez com que as metodologias ágeis se tornassem uma metodologia de desenvolvimento ideal no mercado atual; não somente em projetos localizados mas até em projetos distribuídos globalmente [Paasivaara et al. 2008, Paasivaara et al. 2012]. E, portanto, tornando necessário o ensino de pelo menos uma destas metodologias de desenvolvimento para estudantes de Ensino Superior.

Vários artigos podem ser encontrados com relação aos detalhes de tais metodologias. Mas, um debate sobre como ensiná-las também começou. Afinal, a natureza dinâmica de tais metodologias as tornam muito mais difíceis de se ensinar utilizando as

mesmas técnicas aplicadas no ensino de metodologias mais rígidas de desenvolvimento tais como o *RUP* ou alguma outra metodologia em Espiral ou Cascata.

Assim, várias propostas foram feitas para o ensino de tais metodologias [Devedzic and Milenkovic 2011, Suscheck and Ford 2008, Cubric 2008, Lübke and Schneider 2005, El-Khalili 2013]. Abordagens mais teóricas como metáforas [Suscheck and Ford 2008] e criação de *Wikis* por alunos [Cubric 2008] foram exploradas por alguns autores. Também foram testadas ideias mais práticas como visto em Lübke and Schneider [2005], que propos o uso de peças de LEGO para incrementalmente construir um produto em setenta minutos. E, El-Khalili [2013] que discutiu uma aplicação do conceito de *Problem-Based Learning* para o ensino de metodologias ágeis.

Como demonstrado por vários autores, jogos e simulações tem encontrado sucesso no ensino de modo geral com relação a aprendizagem significativa. Portanto este artigo propõe o uso de uma simulação para ajudar no ensino deste conceito altamente dinâmico. Com isto em mente, os autores propõem o uso de uma ferramenta que ajudaria tanto professores quanto alunos a ensinar/aprender o Scrum baseado na exposição a situações e problemas com um contexto real aplicado em cima deles.

2. Revisão Bibliográfica

2.1. Resumindo o Scrum

O Manifesto Ágil [Beck et al. 2001] sugere uma abordagem iterativa ao processo de desenvolvimento de software. O *Scrum*, sendo uma metodologia ágil, segue suas diretrizes e portanto pode ser caracterizado como um framework responsivo para desenvolvimento de software que trabalha com iterações. O *Scrum* possui três integrantes: o *Product Owner*, o *Scrum Master* e a Equipe (*The Team*) e, como visto em [Schwaber 1995], este é dividido em três partes: *Pre-Game*, *Game* e *Post-Game*. Cada um possui seu próprio conjunto de subprocessos.

2.1.1. Pre-Game

No estágio *Pre-Game*, são analisados, os requerimentos iniciais, definidas as *User Stories* e formado um desing de alto-nível inicial. Estas *User Stories* são, então, organizadas em um *Product Backlog*.

User Story é o nome dado a uma estória curta que descreve uma funcionalidade no mais alto nível possível.[Sutherland and Schwaber 2011]

Product Backlog é a lista de todas as User Stories criada a partir da analise inicial. No fim de cada iteração o cliente pode adicionar itens a esta lista.[Sutherland and Schwaber 2011]

2.1.2. Game

Neste estágio são realizadas as iterações ou *Sprints*. Cada iteração tem uma fase de planejamento onde o *Product Backlog* é analisado e, a partir dele, serão escolhidas *User Stories* que se tornarão parte do *Sprint Backlog*. Estas *User Stories* são divididas em tarefas(*Task*) e, estas, após estimadas, são designadas a membros da equipe [Schwaber 1995].

Sprint é uma iteração do desenvolvimento do software. Normalmente gera uma versão possívelmente ”pronta para entrega”do programa [Sutherland and Schwaber 2011].

Task é uma tarefa. Um passo necessário para completar uma User Story [Sutherland and Schwaber 2011].

Sprint Backlog O conjunto de User Stories que foram selecionadas para um Sprint [Sutherland and Schwaber 2011].

2.1.3. Post-Game

O estágio final consiste de uma leva completa de testes, integração, documentação, marketing, dentre outros. Esta etapa, basicamente, consiste em preparar o estágio de lançamento [Schwaber 1995].

2.2. Ensinando o Scrum e outras Metodologias Ágeis

Como descrito anteriormente através do Manifesto Ágil [Beck et al. 2001], estas metodologias são dinâmicas e adaptáveis. Sendo assim, funcionam bem dentro dos confins da administração de empresas, onde respostas a mudanças são mandatórias. O problema que se mantém é: como podemos ensiná-las?

Várias pesquisas sobre maneiras diferentes de se ensinar o Scrum e outras metodologias ágeis e muitas delas encontram uma quantidade moderada ou alta de sucesso. Lubke encontrou bons resultados no ensino de *XP (Extreme Programming)* com sua metáfora de construção incremental utilizando blocos de LEGO [Lübke and Schneider 2005]. Ainda mais recentemente [Krivitsky 2011] desenvolveu uma maneira similar de ensino (também faz uso de blocos de LEGO) específica para o *Scrum* que encontrou resultados bons. Estes artigos discutem uma simulação de uma metodologia ágil através de uma metáfora e encontraram resultados, porém estas técnicas removem o contexto do desenvolvimento de software para que seja possível simplificar os conceitos do *Scrum*, focando, assim, no ensino do processo teórico do *Scrum*.

Contrastando com estas pesquisas, estão outras que são considerávelmente menos experimentais (ativas) e mais teóricas. Suscheck descreve uma metáfora sobre jazz improvisado que mostrou resultados decentes [Suscheck and Ford 2008]. Ainda assim, todas estas pesquisas possuem em comum o fato de extrairem o contexto do desenvolvimento de software para explicar as características do *Scrum*. Em [Devedzic and Milenkovic 2011] é possível ver alguns dos problemas encontrados no ensino de metodologias ágeis.

Nota-se, então, uma falta de um ensino prático contextualizado para ser aplicado em conjunto com estas abordagens que atingiram sucesso no ensino da teoria. Este artigo propõe um modelo para suprir tal necessidade. Também, elabora sobre a possibilidade de um segundo modelo a ser aplicado ao longo de um tempo maior.

2.3. Game-Based Learning - Jogos e Simulações no Ensino

Pivec diz em sua proposta de um modelo de Ensino com Jogos [Pivec et al. 2003] que, fazendo uso de jogos de computador, é possível encorajar alunos a explorarem várias áreas de conhecimento para chegar em uma solução. Também diz que esta abordagem ao

assunto permite ao aluno perceber as consequências de suas decisões e facilita a discussão entre a equipe.

Ainda, Pivec diz que existem alguns ambientes onde o GBL (*Game-Based Learning*) demonstra-se uma valiosa ferramenta de ensino. São características destes ambientes:

- Pensamento Crítico
- Comunicação
- Debate e Tomada de Decisão

Ainda com relação ao GBL, Van Eck [2006] afirma que o motivo da efetividade do GBL é que aquilo que se está aprendendo é, além de relevante, aplicado e praticado no contexto do jogo. Assim, também afirma que aquilo aprendido sobre um contexto relevante e significativo possui efetividade superior com relação às alternativas. Reforçando o dito por Pivec [2003], Gros [2007], menciona que os jogos são mais eficazes no desenvolvimento das habilidades presentes na lista enumerada previamente. Pode-se perceber que existe uma semelhança entre as características mencionadas e a descrição do ambiente proporcionado pelo *Scrum*.

De acordo com Prensky [Prensky 2001] uma das técnicas empregadas no GBL pode ser traduzida diretamente como ”aprendendo com os erros”. Esta considera os erros um ponto no jogo onde o aluno recebe feedback por sua decisão. Pivec diz também que é necessário definir um objetivo claro para os jogos e que os alunos devem ser capazes de avaliar seu processo [Pivec et al. 2003].

Um segundo modelo de GBL foi proposto por Kiili [2005]. Este modelo foca em ajudar o aluno a atingir o que é chamado de estado de “flow”. Neste estado o aluno, se encontraria completamente focado e engajado na atividade que está sendo realizada. Para atingir este estado, Kiili propõe um modelo que consiste de dois *loops* e um banco de desafios. O primeiro *loop*, melhor realizado em grupo, é um *loop* cujo papel é a geração de ideias para solução do proplema apresentado. O segundo é onde suas decisões são testadas e seus resultados observados. Além de afirmar que, para facilitar o alcance do estado de “flow”, um jogo deve dar ao aluno objetivos claros e feedback. Assim como Prensky e Pivec, também comenta sobre a facilidade proporcionada pela capacidade dos alunos de rever suas decisões.

Outro aspecto mencionado por estes autores [Kiili 2005, Pivec et al. 2003] pode ser visto como a capacidade de um jogo a se adaptar ao nível de habilidade dos jogadores. Por fim, condizente com os modelos discutidos, o modelo proposto de simulação apresenta foco nos aspectos mencionados. Sendo estes:

- Apresentação de Objetivos Claros
- Feedback Imediato
- Capacidade de Reflexão sobre as Decisões Tomadas
- Adaptação à Habilidade dos Jogadores.

3. O Modelo Proposto

O modelo proposto visa dar aos professores uma maneira de explorar, em uma duração de duas a quatro horas, aspectos do *Scrum* fazendo uso das técnicas mencionadas na Seção 2.3. São estes aspectos:

- Estimatitivas de Tempo de User Stories
- Planejamento do Sprint: Formação do Sprint Backlog
- Planejamento do Sprint: Divisão em Tarefas
- Planejamento do Sprint: Estimativa
- Execução do Sprint: Reação a Problemas
- Recuperação a partir de uma Decisão Errada

Este modelo não foca na distribuição das responsabilidades entre os participantes do *Scrum*, mas, sim, nas características mais dinâmicas do *Scrum*: a comunicação entre a equipe e a tomada de decisão em conjunto para cumprir um objetivo imposto por um cliente, representado na simulação preparada por um professor com um Sprite(Imagem 2D, animada). O modelo possibilitará, ao grupo, discutir sobre cada decisão e estimativa tomada, rever decisões tomadas, visualizar reações positivas e negativas do cliente e, por fim, responder a eventos que ocorrem no meio do Sprint.

Propomos, também, que uma ferramenta seja utilizada para criação de simulações seguindo este modelo. Esta ferramenta será discutida mais adiante neste artigo, porém é importante mencionar um pouco sobre ela.

A ferramenta funcionaria como um Motor de Jogos, porém aplicada a estrutura do modelo da simulação. Isto permitiria que professores criem sua propria simulação, alimentando uma estrutura com dados. Além de permitir abstrações gráficas como as seguintes:

- Todos os integrantes da equipe e elementos do Scrum (Sprints, User Stories) poderão ser representados em alguns momentos no jogo como um Sprite referido, a partir de agora neste artigo como Avatar.
- Toda a distribuição de informações da simulação será representada por uma "conversa" entre um cliente e o *Product Owner* e membros da equipe.
- Toda tomada de decisão será representada por uma batalha entre Avatares dos elementos do *Scrum* no qual o problema ocorreu ou do problema em si.

Esta estrutura seria, basicamente, uma abstração do processo do *Scrum* permitindo que o professor defina falas para o cliente, respostas esperadas para algumas perguntas, problemas que ocorrem durante os Sprints, dentre outros.

Tais "ganchos" na estrutura, serão preenchidos pelo professor assim criando uma maneira de se criar simulações com contextos de software diferentes, mas executam sobre a mesma estrutura. Também, estas abstrações — detalhadas mais adiante — visam trazer para a simulação um certo grau de atratividade e dar aos alunos uma sensação de imersão em uma estória.

O modelo pode ser dividido em três partes, assim como o *Scrum*, interligadas como na Figura 1. As duas tarefas realizadas antes do começo dos Sprints são o equivalente ao *Pre-Game* descrito na Seção 2.1.1. Os Sprints seriam a simulação do *Game* apresentado na Seção 2.1.2.

O fim da simulação se daria como uma entrevista de entrega com o cliente, extraindo muitas das complexidades técnicas do *Post-Game* (Seção 2.1.3) que não fazem parte do escopo da simulação. Seria, resumidamente, uma maneira da equipe ver o grau de satisfação do cliente com o resultado, ou seja, um feedback contextualizado da simulação inteira.

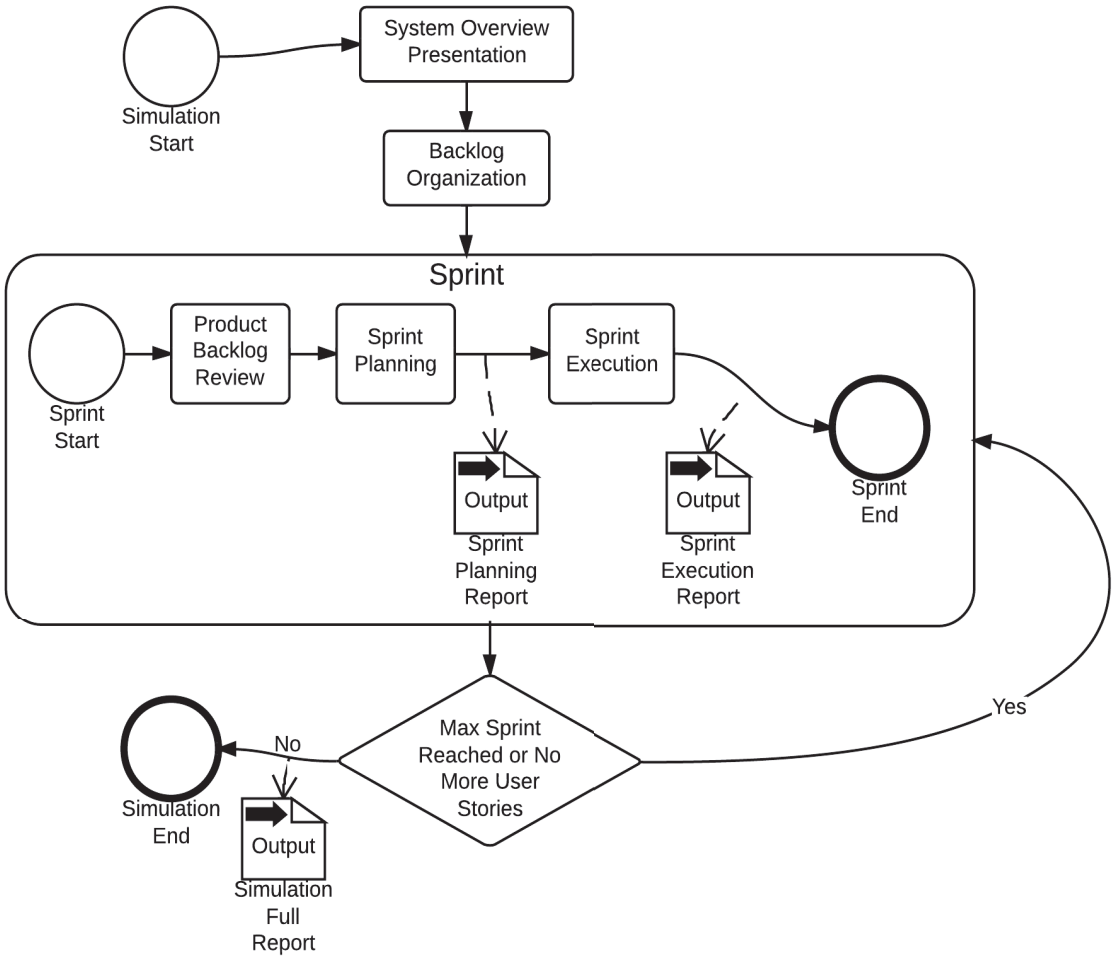

Figura 1. Short Simulation Process Flowchart

Por fim, antes da analise do modelo proposto, é necessário mencionar que o modelo foi imaginado para trabalhar com equipes de cinco a nove alunos, correspondente ao tamanho ideal de uma equipe do *Scrum*. Porém, caso não seja possível formar ao menos três equipes com este número, é recomendado que se diminua o numero de alunos por equipe, até um limite de três.

3.1. O Começo do Projeto - Project Set-Up

Nesta etapa a equipe será introduzida ao projeto cujo processo de desenvolvimento será simulado. Esta introdução será feita pelo Avatar do cliente na simulação. Este personagem irá notificar a equipe de informações pertinentes além de apresentar aos alunos as User Stories. A apresentação das User Stories será feita através de falas naturais, ao invés de uma simples enumeração.

Caberá aos alunos identificarem, através de uma série de perguntas objetivas predefinidas pelo professor, representadas por uma série de batalhas, qual das User Stories apresentadas se encaixa com aquilo que foi dito pelo Avatar do cliente. Cada Avatar na

batalha irá possuir animações de derrota (resposta errada) e vitória (resposta correta); permitindo, assim, dar o feedback imediato necessário com relação à decisão. Porém, as decisões erradas não poderão ser corrigidas imediatamente.

Todas decisões erradas serão armazenadas em uma fila para que a equipe possa refletir em suas decisões erradas e procurar novas soluções. Esta fila será revisada no fim de cada etapa com uma nova leva de perguntas. Isto permite que a equipe divida as responsabilidades de continuar a análise das falas do Avatar do cliente e revisar o motivo do erro cometido. Este modelo de batalha se mantém para todas perguntas existentes na simulação variando, possivelmente, em casos específicos que serão mencionados.

Equipes que não conseguirem identificar as User Stories corretamente não poderão avançar na simulação, pois, obviamente, é necessário entender o que o cliente deseja antes de se começar um projeto. Os integrantes da equipe desfeita são então distribuidos pelas outras equipes na turma, caso existam. Isto da oportunidade aos alunos de continuarem a simulação mesmo falhando esta primeira etapa. Também, simula um abiente real de trabalho onde profissionais podem entrar em equipes já existentes. Por isto, a recomendação feita com relação ao tamanho da equipe.

Entre a tomada das decisões e a revisão das decisões erradas, o Avatar do cliente comentará no desempenho geral da equipe. Finalmente, as prioridades serão adicionadas as User Stories pelo Avatar do cliente, o backlog será automaticamente organizado e os alunos precisarão entrar com uma estimativa. Para simular o efeito contratual do início do desenvolvimento de um projeto a estimativa dada pelos alunos para User Stories serão finais. Esta estimativa deverá estar entre valores predefinidos pelo professor ao montar a simulação. A recomendação, aqui, é de que estes valores sejam distantes um dos outros, pois estimativas variam com a especialidade da equipe em questão. Está pergunta será subjetiva e também será representada por uma batalha. Após dadas as estimativas uma chance de revisar as que não se encontraram entre os valores predefinidos.

Assim, após esta etapa, os alunos iniciarão os sprints. É recomendado que esta etapa leve de dez a quarenta e cinco minutos para se executar. Estes valores, obviamente, dependem do tamanho geral da simulação. A recomendação para professores é de que, quando montarem a simulação, não iniciem o sistema com muitas User Stories e, sim, programem a simulação para adicioná-las entre as Sprints.

3.2. As Sprints

Esta etapa é o foco deste modelo. É iterativa e possui um sistema onde é dividida em três Fases:

- Planejamento de Sprint
- Execução de Sprint
- Entrega de Sprint

Cada uma destas fases possui como objetivo simular uma parte do processo de uma sprint. As fases 1 e 2 são as de maior importância para a simulação, pois necessitarão muito do pensamento crítico dos alunos e do conhecimento, como um grupo, da equipe. A ultima etapa de cada sprint servirá como uma maneira dos alunos receberem uma resposta do Avatar do cliente. Esta resposta seria programada em níveis de compleção das User Stories definidas para a Sprint.

3.2.1. Fase 1: Planejamento de Sprint

A primeira fase da Sprint inclui as seguintes etapas: *Product Backlog Review* e o *Sprint Planning*. A primeira etapa é onde será realizada uma revisão do product backlog. Representada por uma conversa similar a primeira etapa do processo. Nela os alunos precisarão identificar User Stories a partir do que será dito pelo Avatar do cliente. Assim como no *Project Set-Up*, os alunos terão que estimar as novas User Stories.

Na segunda etapa, representada pela Figura 2, a equipe deverá realizar as seguintes tarefas:

- Criar o Backlog da Sprint
- Definir Tarefas
- Estimar Tarefas

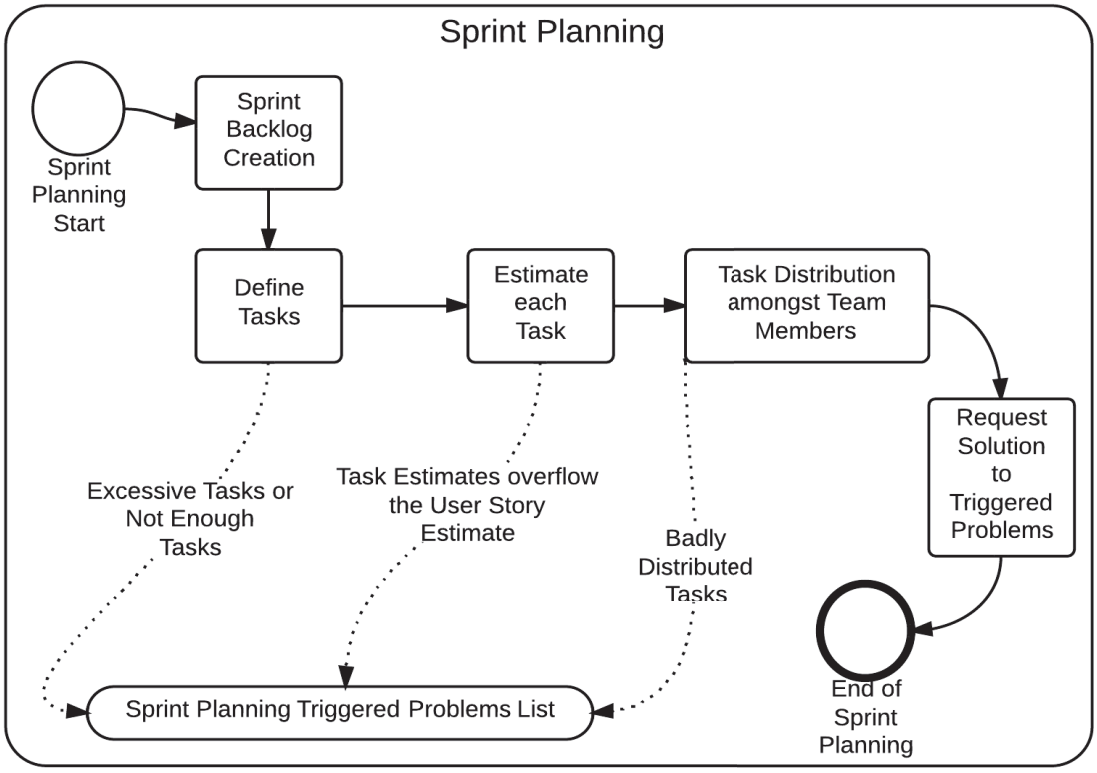

Figura 2. Sprint Planning Simulation Flowchart

A criação do backlog do sprint será caracterizada como uma conversa entre os vários Avatares dos Jogadores. Onde as User Stories são percorridas e cada jogador poderá votar em uma User Story para ser adicionada ao Backlog da Sprint. Dito isto, é necessário esclarescer que informações sobre as User Stories estarão disponíveis para os alunos nesta etapa da simulação. Inicialmente, todas as User Stories do Backlog do Produto estarão disponíveis para serem adicionadas ao Backlog da Sprint. Porém, caso a equipe selecione uma User Story que depende de uma outra ainda não realizada, o Avatar do Cliente aparecerá e informará a equipe que tal User Story depende de outra.

Com as User Stories definidas, os alunos avançaram para uma série de perguntas abertas onde eles dividirão cada User Story em tarefas. Os professores poderão definir o número de tarefas mínimas para cada User Story. Esta definição será representada por uma discução entre a equipe, nela os alunos entrarão com, para cada tarefa, um nome descritivo. Todavia, antes de tal discussão, a equipe enfrentará uma batalha para definir o número de tarefas em que deseja dividir esta User Story. Caso os alunos falhem em entrar com um valor acima do predefinido, deverão enfrentar a batalha novamente.

Finalmente, os alunos enfrentaram uma série de batalhas para estimar as tarefas de cada User Story. Aqui a batalha contra cada Avatar de User Story será definida da seguinte maneira:

1. Os alunos começam com vida igual à estimativa para a User Story
2. Equipe entra com uma estimativa de tempo para a primeira tarefa.
3. Perdem vida igual ao valor entrado.
4. Repetem os itens 1 e 2 até não haverem mais tarefas ou suas vidas acabarem.

O objetivo desta batalha é sobreviver com menos de 15% ou 20% de vida. Isto se deve ao fato da vida do Avatar da Equipe ser a estimativa que eles definiram para a User Story. O motivo que deverão restar 15% a 20% da vida da equipe por batalha é para representar a boa prática de se estimar com uma margem de segurança. Apesar de esta etapa ser a maior de todas as etapas da simulação, na realidade, o ideal é que o planejamento de uma Sprint não seja longo. Portanto, cada equipe terá três vidas para esta etapa, ou seja, poderão tentar três vezes caso falhem durante o processo. Demonstrando o quanto é custoso o replanejamento de uma Sprint, User Story ou Tarefa.

3.2.2. Fase 2: Execução de Sprint

Esta fase tem como foco a solução rápida de problemas e será representada por uma única grande batalha entre o Avatar da equipe e o Avatar de um Sprint. Nesta batalha, serão apresentados, problemas predefinidos (se existentes) para cada User Story. Cada erro a solucionar um problema ocorrido ao longo de uma User Story causa dano a equipe. Cada acerto causará dano ao Sprint. A User Story que for de maior prioridade, caso possua algum problema predefinido, será a que mais causará dano ao sprint se resolvido erroneamente. A Figura 3 exemplifica este processo.

O objetivo desta etapa é causar o máximo de dano à Sprint. Os alunos serão vitoriosos caso, ao final da Sprint, possuam mais vida que o Avatar da Sprint. Após a luta existirá uma recapitulação das respostas dadas e a oportunide de revanche será dada à equipe.

As batalhas desta revanche são contra User Stories individuais e proporcionam uma maneira dos alunos reavaliarem suas decisões. Assim como as outras batalhas propostas, o feedback, positivo ou negativo, é dado imediatamente ao aluno. É necessário esclarescer que User Stories não derrotadas na revanche permanecerão não completadas e trarão uma resposta negativa do cliente.

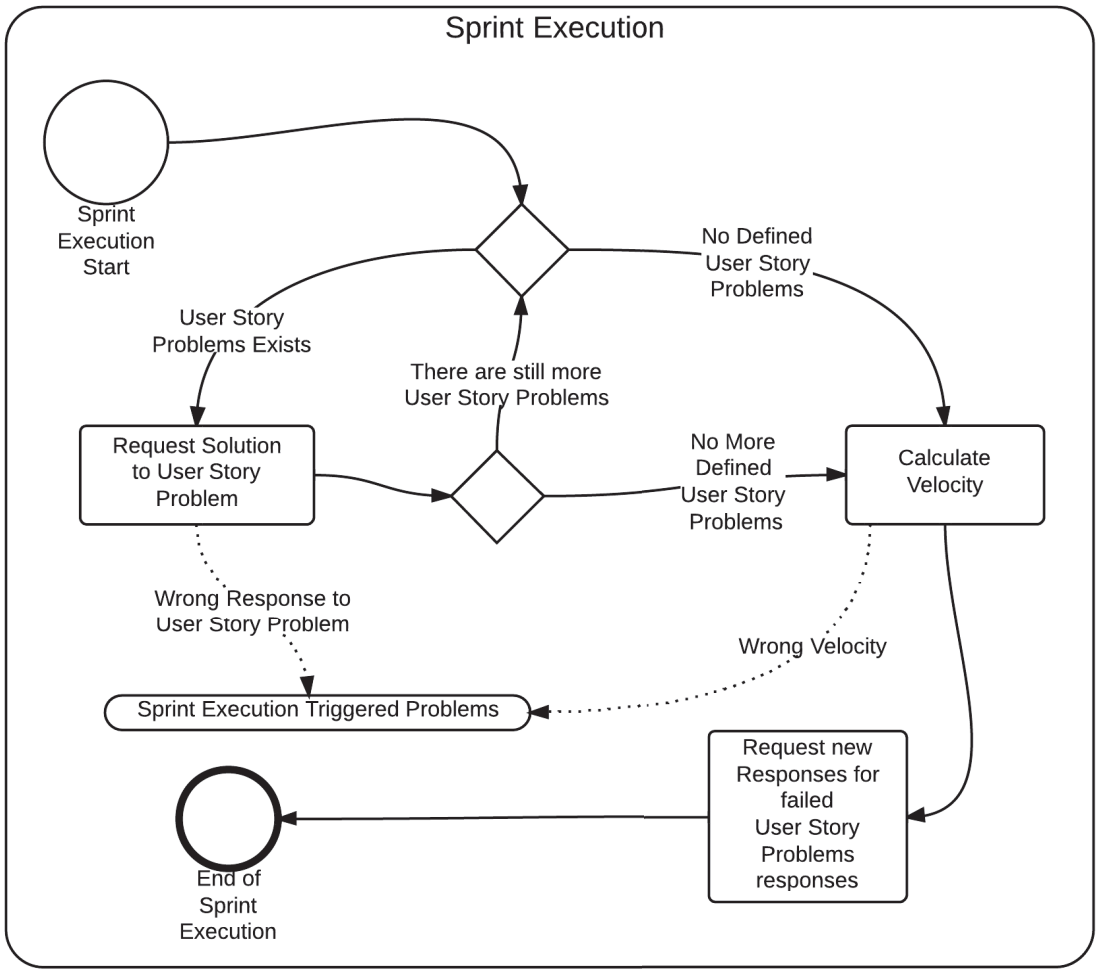

Figura 3. Sprint Execution Simulation Flowchart

Finalmente, será realizado o calculo de *velocity*. Neste calculo a equipe deverá levar em consideração somente as User Stories completas. O simulador, então, calcula a velocidade esperada (melhor *velocity* possível) e a compara com a real. A partir disto, o Avatar do *Scrum Master* dará à equipe 4 alternativas sendo somente uma correta. A equipe deverá escolher a real *velocity*. Para isto, terá quantas tentativas forem necessárias.

3.2.3. Entrega de Sprints

Após as batalhas do Sprint, os resultados serão processados e uma resposta será dada através do Avatar do cliente. Esta resposta pode ser positiva, negativa ou neutra em função dos resultados das batalhas enfrentadas.

Com isto, um Sprint será finalizado. Este processo se repetirá até o momento em que não houverem mais User Stories ou um limite predefinido de sprints seja atingido. Ao atingir um dos casos acima, os alunos serão enviados à próxima etapa da simulação.

3.3. Finalização do Projeto - Project Wrap-Up

Inicialmente, nesta última e final etapa da simulação, o Avatar do cliente fará comentários finais com relação ao desempenho da equipe. Também haverá a possibilidade de rever todas decisões tomadas durante a simulação.

Esta etapa é, basicamente, uma recapitulação. O simulador será capaz de gerar um documento com todas as decisões tomadas. E, a recomendação é que professores peçam às equipes justificativas para cada decisão tomada; sendo esta decisão errada ou não.

Isto finaliza este modelo curto da simulação do *Scrum*.

4. O Criador de Simulações Proposto

Nesta seção será discutido a ferramenta proposta para criação de simulações. Os objetivos desta ferramenta são:

- Permitir fácil criação/alteração de simulações.
- Organizar as várias User Stories e Sprints existentes em uma simulação.
- Criar simulações diferentes que sigam o mesmo modelo.

Esta proposta tem como principal justificativa facilitar o balanceamento da simulação para turmas de níveis diferentes. A personalização de simulações que sigam o modelo permitirá aos professores manter a dificuldade da simulação balanceada.

A criação de User Stories, Sprints e Problemas serão independentes. A ideia é dar aos professores uma maneira simples de organizar os elementos de suas simlulações.

Primeiramente, será analisado o processo de criação de Problemas. Estes problemas serão enfrentados pelos alunos durante a execução da Sprint descrita na Seção 3.2.2.

1. Definir uma descrição para o problema. Esta descrição deverá ambientalizar os alunos.
2. Definir um grau de dificuldade (puramente organizacional).

Além de definir problemas os professores poderão definir User Stories. O processo de criação de User Stories se daria da seguinte maneira:

1. Definir nome e prioridade.
2. Definir falas do Avatar do Cliente para sua introdução. Deverá ser claro suficiente para descrever a User Story e o mais breve possível.
3. Definir falas do Avatar do Cliente para sua finalização. (Boa, Média e Ruim). Estas, serão ditas ao alunos com base em seu desempenho (porcentagem de acerto) nos problemas existentes para estas User Stories.
4. Definir estimativas de tempo mínima e máxima.
5. Definir número de tarefas mínimas esperadas.
6. Ligar qualquer problema criado que desejar a esta User Story.

Por fim, os professores poderão ligar User Stories a Sprints. Para exemplificar, um professor pode ligar a User Story 10 ao Sprint 3. Durante a execução da simulação e quando a equipe chegar no Sprint 3, esta User Story será introduzida pelo Avatar do Cliente. Qualquer User Story adicionada em uma Sprint estará no Backlog até o fim da simulação ou ser completada. Vale mencionar que o número de Sprints programados pelo professor não será utilizado como número de Sprints totais da simulação. Os professores poderão definir um número máximo de Sprints extras para se completar a simulação.

5. Considerações Finais e Conclusão

Em uma analise inicial, apresentou-se uma ideia inicial do modelo para um grupo de professores juntamente com um protótipo. Foram questionados sobre os objetivos do jogo, a jogabilidade e as tecnicalidades do *Scrum*. Após a apresentação e coleta de dados realizada na forma de questionários que obtiveram resultados variando de: não concordo, concordo parcialmente e concordo, percebeu-se que o jogo proposto agregaria valor ao processo de ensino do *Scrum*.

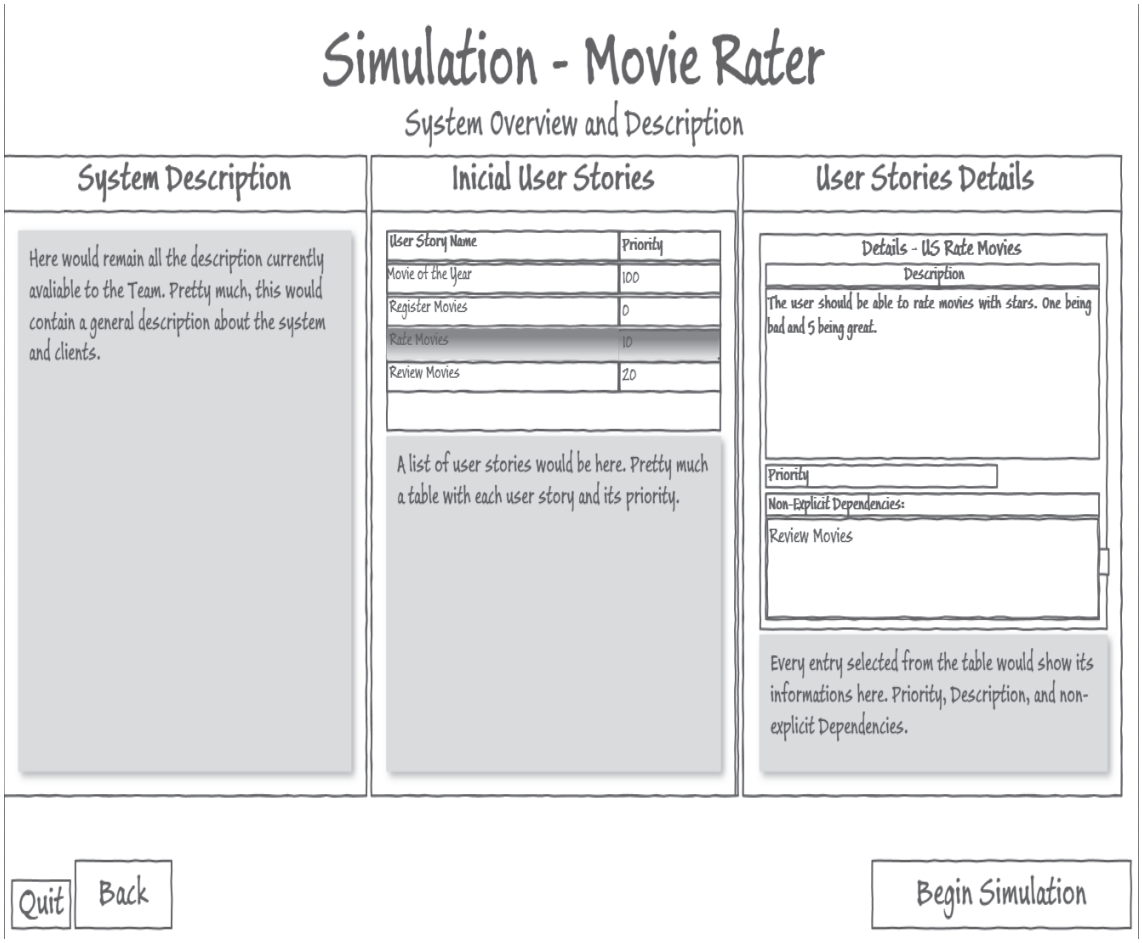

Figura 4. Protótipo Gerado antes da Análise Inicial

A Figura 4 mostra a tela da apresentação do sistema do protótipo inicialmente gerado. Vale mencionar que este protótipo foi gerado antes da análise inicial e focava em uma simulação puramente textual. Após a analise o modelo foi reavaliado e, com isso, foi gerado o modelo apresentado na Seção 3; este, visando uma apresentação que estimularia os alunos. Para este propósito, foi decido utilizar representações de batalhas e revanches com Avatares caracterizados de cada elemento do *Scrum*.

Tem-se que, como fundamentado na Seção 2.2, metáforas são uma maneira muito utilizada para se ensinar os conceitos do *Scrum*. Dito isto, a prática desta metodologia

é mais complexa e dinâmica que sua teoria e, portanto, seria interessante analisar a possibilidade de tal modelo. A intenção deste modelo é permitir que os professores criem simulações diferentes para estimular os alunos a praticar os conceitos do *Scrum* dentro do contexto de desenvolvimento de software.

Apesar deste modelo cobrir muitas partes do *Scrum*, futuramente, estaremos analisando a possibilidade de um modelo de simulação a longo prazo. Este contendo o máximo possível de aspectos do *Scrum*. Porém, antes de analisar este segundo modelo, estaremos verificando a viabilidade do modelo proposto neste artigo.

Referências

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). Manifesto for agile software development.

Cubric, M. (2008). Agile learning & teaching with wikis: building a pattern. In *Proceedings of the 4th International Symposium on Wikis*, WikiSym '08, pages 28:1–28:2, New York, NY, USA. ACM.

Devedzic, V. and Milenkovic, S. R. (2011). Teaching agile software development: A case study. *IEEE Trans. on Educ.*, 54(2):273–278.

El-Khalili, N. H. (2013). Teaching agile software engineering using problem-based learning. *Int. J. Inf. Commun. Technol. Educ.*, 9(3):1–12.

Kiili, K. (2005). Digital game-based learning: Towards an experiential gaming model. *The Internet and Higher Education*, 8(1):13–24.

Krivitsky, A. (2011). A multi-team, full-cycle, product-oriented scrum simulation with lego bricks.

Lübke, D. and Schneider, K. (2005). Agile hour: teaching xp skills to students and it professionals. In *Proceedings of the 6th international conference on Product Focused Software Process Improvement*, PROFES'05, pages 517–529, Berlin, Heidelberg. Springer-Verlag.

Paasivaara, M., Durasiewicz, S., and Lassenius, C. (2008). Using scrum in a globally distributed project: a case study. *Softw. Process*, 13(6):527–544.

Paasivaara, M., Lassenius, C., and Heikkilä, V. T. (2012). Inter-team coordination in large-scale globally distributed scrum: do scrum-of-scrums really work? In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '12, pages 235–238, New York, NY, USA. ACM.

Pivec, M., Dziabenko, O., and Schinnerl, I. (2003). Aspects of game- based learning. In *I-Know*.

Prensky, M. (2001). Digital natives, digital immigrants. *On the horizon*, 9(5).

Schwaber, K. (1995). Scrum development process. In *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA*, pages 117–134.

Suscheck, C. A. and Ford, R. (2008). Jazz improvisation as a learning metaphor for the scrum software development methodology. *Softw. Process*, 13(5):439–450.

Sutherland, J. and Schwaber, K. (2011). The scrum papers: Nuts, bolts, and origins of an agile process.