Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Projeto e Desenvolvimento de um Jogo Digital para Auxílio no Ensino de Gerenciamento de Processos de Software

Sebastião Lúcio Reis de Souza

JUIZ DE FORA
DEZEMBRO, 2018

Projeto e Desenvolvimento de um Jogo Digital para Auxílio no Ensino de Gerenciamento de Processos de Software

Sebastião Lúcio Reis de Souza

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Igor de Oliveira Knop

JUIZ DE FORA

DEZEMBRO, 2018

Projeto e Desenvolvimento de um Jogo Digital para Auxílio no Ensino de Gerenciamento de Processos de Software

Sebastião Lúcio Reis de Souza

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Igor de Oliveira Knop
Prof. Dr.

André Luiz de Oliveira
Prof. Dr.

Marcelo Caniato Renhe
Prof. Dr.

JUIZ DE FORA

4 DE DEZEMBRO, 2018

Resumo

Treinar pessoas no desenvolvimento de projetos de *software* em um ambiente educacional é uma tarefa que demanda custos e tempo de preparação dos alunos. O presente trabalho apresenta a utilização de jogos digitais no ensino de modelos de processo de *software* cujo objetivo geral é desenvolver um jogo digital com a proposta de auxiliar a aprendizagem da metodologia ágil *Scrum* . Trata-se de uma pesquisa exploratória, baseada em seis estudos de caso com apoio de referencial teórico. A fim de garantir o objetivo proposto foram consultadas fontes de pesquisas acerca de modelos de processo de *software* e de desenvolvimento de ferramentas educacionais. Com base no planejamento realizado foi desenvolvido um jogo educacional digital denominado *SEJam* o qual permite o ensino de conceitos da metodologia ágil *Scrum* . Através do uso do jogo, os alunos podem aprender os conceitos da metodologia, simular o acompanhamento do projeto e avaliar os resultados de uma forma rápida e interativa. Como trabalhos futuros estão previstos a atualização do jogo com novos recursos e funcionalidades.

Palavras-chave: *Scrum* , gerência de projetos, jogos digitais educacionais, modelos de processo de *software*, metodologias Ágeis.

Abstract

Training people in the development of *software* projects in an educational environment is a task that demands costs and preparation time for students. The present work presents the use of digital games in the teaching of software process models whose general objective is to develop a digital game with the purpose of aiding the learning of the agile methodology *Scrum* . This is an exploratory research, based on six case studies supported by theoretical reference. In order to guarantee the proposed objective, we consulted research sources about software process models and the development of educational tools. Based on the planning, a digital educational game called *SEJam* was developed, which allows the teaching of agile methodology *Scrum* concepts. Through the use of the game, students can learn the concepts of the methodology, simulate project monitoring and evaluate the results in a fast and interactive way. Updating the game with new features is expected in the future.

Keywords: scrum, projects management, educational digital games, process of *software* development, agile methods.

Agradecimentos

A todos os meus parentes, pelo encorajamento e apoio. Ao professor Igor pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria. Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

Conteúdo

Lista de Figuras

LISTA DE ABREVIAÇÕES

ADDIE	*Analyze, Design, Develop, Implement and Evaluate*
CASE	*Computer Aided* software *Engineering*
DCC	Departamento de Ciência da Computação
ISD	*Instrucional System Development*
NPC	*Non Playable Character*
PMBOK	*Project Management Body of Knowledge*
PMI	*Project Management Institute*
RPG	*Role Playing Game*
RUP	*Rational Unified Process*
UFJF	Universidade Federal de Juiz de Fora
UML	*Unified Modeling Language*

1 Introdução

1.1 Contextualização

Nos últimos anos tem aumentado o número de empresas que utilizam metodologias ágeis em seus projetos (KUMAR; BHATIA, 2012). As metodologias ágeis são modelos de processo de software que tem como característica um ciclo de desenvolvimento compreendendo uma situação atual, a definição de um problema, o desenvolvimento técnico e a integração de uma solução (PRESSMAN, 2011). As metodologias ágeis são utilizadas no apoio ao gerenciamento de projetos e buscam estimular o contato da empresa com o cliente e a entrega incremental de funcionalidades, resultando na maioria das vezes na redução de custos e na entrega bem sucedida do projeto no prazo estabelecido (RUBIN, 2012). Uma metodologia ágil utilizada no gerenciamento de projetos é o *Scrum* . A metodologia ágil *Scrum* foi criada no inicio dos anos 90 para tratar e resolver problemas com características complexas e adaptativas (SCHWABER; SUTHERLAND, 2015).

1.2 Apresentação do Problema e Justificativa

Segundo Group (2013) diversos projetos de *software* apresentam problemas como entrega fora do prazo, falta de orçamento, funcionalidades as quais não atendem corretamente aos usuários e até mesmo cancelamento do projeto. Entre vários fatores, é atribuído como uma das possíveis causas destes problemas a falta de profissionais bem capacitados em desenvolvimento de *softwares*. Ao visualizar a formação destes profissionais no meio acadêmico é possível verificar que o método mais utilizado para o ensino de modelos de processo de *software* consiste na ministração de aulas teóricas expositivas (NAVARRO, 2013).

De fato, é de fundamental importância para a formação do aluno a exposição de conceitos teóricos relacionados a qualquer área utilizando-se meios tradicionais como aulas e livros, porém levando-se em consideração a opinião de educadores, é necessário

mesclar os conteúdos teóricos apresentados em sala de aula com a utilização de atividades lúdicas como jogos educacionais, por exemplo, permitindo que o conhecimento possa ser adquirido e aplicado pelos alunos de forma mais eficiente e intuitiva (FALKEMBACH; GELLER; SILVEIRA, 2006).

Segundo Needleman (2015) o crescimento do mercado de jogos digitais para dispositivos móveis nos Estados Unidos é maior quando comparado ao crescimento de outras mídias digitais. Assim, é possível observar a crescente popularidade dos jogos digitais no cotidiano das pessoas. O emprego dos jogos digitais no contexto de ensino cria um vínculo entre entretenimento e aprendizagem o qual torna o ambiente de ensino mais atrativo e faz com que o processo de aprendizagem seja mais eficiente (TAROUCO; ROLAND, 2004).

Neste contexto, o objetivo deste trabalho é fazer o estudo acerca de jogos digitais educacionais e a criação de um jogo digital para ajudar estudantes e profissionais da área de Tecnologia da Informação no aprendizado da metodologia ágil *Scrum* .

Diante do problema da busca de um método alternativo eficaz para o ensino de modelos de processo de *software* e a crescente popularidade de jogos digitais, justifica-se neste trabalho a realização da pesquisa e o estudo acerca da utilização de jogos digitais no ensino de modelos de processo de desenvolvimento de *software* e o desenvolvimento de um jogo digital educacional para o ensino do modelo *Scrum* . Por fim, destaca-se a viabilidade da criação da ferramenta educacional pretendida, desenvolvida apenas com a utilização de *softwares* gratuitos.

1.3 Objetivos

O objetivo geral a criação de um jogo digital educacional para o complemento ao ensino da metodologia ágil *Scrum* . Esta meta foi alcançada através da realização dos objetivos específicos: revisão bibliográfica acerca de modelos de processo de *software*, com ênfase nas metodologias ágeis *Extreme Programming* (PRESSMAN, 2011) e *Scrum* (SCHWABER; SUTHERLAND, 2015); criação de um protótipo do jogo digital com a função de simular atividades do modelo *Scrum* ; avaliação do jogo digital produzido com o objetivo de extrair informações referentes a problemas de usabilidade do jogo.

1.4 Organização do trabalho

O presente trabalho está dividido em cinco capítulos, incluindo esta Introdução. O Capítulo 2 faz uma revisão bibliográfica acerca de modelos de processos de *software* permitindo a maior entendimento sobre o assunto assim como a revisão sobre a utilização de jogos digitais educacionais como forma de ferramenta pedagógica. O Capítulo 3 apresenta o desenvolvimento do jogo digital *SEJam*, as ferramentas utilizadas na sua implementação e a descrição sobre a utilização e funcionamento da ferramenta. O Capítulo 4 descreve o método de análise da utilização do protótipo e as métricas utilizadas. Por fim, o Capítulo 5 apresenta as considerações finais e propostas de trabalhos futuros.

2 Fundamentação Teórica

Este capítulo tem como objetivo reunir informações relacionadas a modelos de processo de *software* e jogos digitais para o entendimento do desenvolvimento do trabalho nos capítulos seguintes. Serão conceituados os modelos clássicos e as metodologias ágeis *Extreme Programming* (XP) e *Scrum* . Será realizada uma breve introdução ao modelo de *design* instrucional *Analyze, Design, Develop, Implement and Evaluate* (*ADDIE*) e ao final do capítulo é realizada uma abordagem referente a jogos digitais educacionais criados para o ensino de modelos de processo de *software*.

2.1 Processo de *software*

Um processo de *software* é caracterizado pela execução de sucessivos passos seguindo um roteiro predefinido envolvendo uma série de atividades a serem realizadas com o objetivo final de desenvolver-se um *software* (SOMMERVILLE, 2007). Um processo de *software* também é definido pela combinação de uma série de elementos de um conjunto envolvendo atividades de trabalho, ações e tarefas que devem ser executadas quando é solicitado a construção de um artefato de *software*. Sommerville (2007) ainda afirma que um modelo de processo pode variar dependendo da interpretação de cada pessoa, porém na maioria dos casos os modelos possuem algumas atividades em comum classificadas como: especificação de *software*, projeto e implementação de *software*, validação de *software* e evolução de *software*.

A atividade de especificação de *software* envolve especificações acerca de funcionalidades e restrições demandadas pelo *software*. A atividade de projeto e implementação de *software* visa produzir o *software* de maneira a atender todos os requisitos exigidos. A etapa de validação de *software* possibilita a aplicação de testes os quais possibilitam saber, se o *software* desenvolvido atendeu às especificações fornecidas pelo cliente. A última atividade denominada evolução de *software* é responsável pela manutenção do *software*, a qual permite a realização de modificações e atualizações no *software* após o *software* já

ter sido entregue ao cliente. O autor destaca ainda a utilização de ferramentas e *softwares* auxiliares no apoio de determinadas atividades do processo utilizado.

2.2 Modelos de Processo de *Software*

Um modelo de processo de *software* define etapas nas quais atividades e tarefas serão executadas de forma a restringir o processo de desenvolvimento de *software* a determinadas práticas particulares ao modelo adotado (SOMMERVILLE, 2007).

A criação dos primeiros modelos de processo foi realizada com o objetivo de estabelecer uma determinada ordem e estrutura do processo de *software* (PRESSMAN, 2011). Destacam-se como modelos clássicos de processos os modelos Cascata, Desenvolvimento Evolucionário e Modelo de Processo Baseado em Componentes.

2.2.1 Modelo Cascata

O modelo Cascata é um modelo linear no qual as principais atividades envolvidas no processo de desenvolvimento são executadas de maneira sequencial, seguindo etapas determinadas pelo modelo. As etapas do modelo são denominadas: comunicação, planejamento, modelagem, construção e implantação, conforme pode ser observado na Figura 2.1.

No modelo Cascata existe uma relação de precedência entre duas etapas consecutivas indicando que uma determinada etapa somente pode ser iniciada mediante o término da etapa predecessora. O modelo Cascata estabelece o planejamento e o gerenciamento do processo de *software* além de fazer a documentação das atividades em cada etapa. Como contrassenso, observa-se que na prática raramente os projetos seguem a sequência proposta pelo modelo, além disso, o modelo exige a apresentação de todos os requisitos no inicio do desenvolvimento, porém o cliente nem sempre consegue definir de início todas as funcionalidades pretendidas pelo sistema.

Uma desvantagem apresentada pela utilização deste modelo é que uma versão pronta do *software* somente é apresentada na última etapa do processo de desenvolvimento. Outro problema apresentado é a limitação do modelo em não permitir a evolução do *software* com a entrada de novos requisitos durante o processo de desenvolvimento

(PRESSMAN, 2011).

Figura 2.1: Fases do ciclo de vida do modelo Cascata Fonte: baseado em (PRESSMAN, 2011).

2.2.2 Modelo de Desenvolvimento Evolucionário e Incremental

Um modelo evolucionário é baseado no desenvolvimento de uma versão inicial simplificada do *software*, e ao longo do tempo conforme novas funcionalidades ou requisitos sejam estabelecidos, novas iterações são realizadas com o objetivo de incorporar ao *software* funcionalidades especificadas por novos requisitos de *software* (PRESSMAN, 2011). O autor afirma a existência de dois tipos de modelos evolucionários: o Modelo de Prototipação e o Modelo Espiral.

No Modelo de Prototipação o cliente define os requisitos sem a existência de uma especificação detalhada, e um protótipo do *software* e construído para ser submetido à avaliação do cliente e posteriormente utilizado para a elaboração de uma versão mais refinada do *software*. Segundo Pressman (2011), esta abordagem pode ser vantajosa quando existe a incerteza com relação aos requisitos iniciais do sistema.

O segundo modelo, ainda apresentado pelo mesmo autor, denomina-se Modelo Espiral, o qual incorpora o aspecto iterativo do modelo de prototipação aliado à disciplina estabelecida pelo modelo em cascata. O modelo espiral comparado com outros modelos permite o reconhecimento dos riscos envolvidos durante as atividades de desenvolvimento do *software* permitindo aos desenvolvedores fazer a análise desses riscos durante a fase de planejamento (SOMMERVILLE, 2007).

2.2.3 Modelo de Desenvolvimento Baseado em Componentes

Criado no final da década de 90, o modelo de desenvolvimento de *software* baseado em componentes é um modelo que tem como base a utilização e integração de um conjunto variável de componentes reutilizáveis (SOMMERVILLE, 2007). O autor destaca que

as principais diferenças entre o modelo baseado em componentes e outros modelos de processo de *software* encontram-se no início do processo, no qual os requisitos de usuário são refinados a partir da disponibilidade de componentes, além da realização de buscas por componentes adicionais adequados ao projeto. O modelo de desenvolvimento baseado em componentes é composto por quatro fases: análise de componentes, modificação de requisitos, projeto de sistema com reuso e desenvolvimento e integração, definidas a seguir conforme Sommerville (2007):

Na fase de análise de componentes é feita a busca e seleção de componentes que atendam os requisitos de usuário. A segunda fase de modificação de requisitos tem como objetivo adaptar os requisitos de forma a mantê-los compatíveis aos componentes existentes. A fase de projeto visa definir um conjunto de funcionalidades, levando-se em consideração componentes pré-existentes, e na necessidade da criação de novos componentes. A última fase do modelo é a de integração, a qual abrange o desenvolvimento de novos componentes e integração de todas as partes do *software* para a execução da validação do sistema.

2.2.4 Modelo *Rational Unified Process* (RUP)

O modelo *Rational Unified Process* (RUP) é considerado um modelo híbrido que combina conceitos de modelos tradicionais de processo de *software* com os princípios das metodologias ágeis (PRESSMAN, 2011). O *RUP* é um modelo que preza pela comunicação entre os envolvidos no projeto de *software* e utiliza diagramas da linguagem de modelagem *Unified Modeling Language* (UML), como o diagrama de casos de uso com o objetivo de fazer a interlocução das intenções e necessidades dos usuários aos desenvolvedores e projetistas do sistema. O modelo *RUP* é composto por quatro fases: concepção, elaboração, construção e transição, definidas a seguir segundo Sommerville (2007).

Durante a fase de concepção ocorre o levantamento de atores externos e suas respectivas interações com o sistema, e em conjunto é realizado o levantamento de operações e serviços que devem ser oferecidos pelo sistema. Com base no resultado obtido do levantamento, surge a possibilidade de verificar o impacto da imersão do sistema no ambiente de negócio. A fase de elaboração é marcada pela modelagem dos requisitos do sistema

e pela descrição dos serviços a serem oferecidos através da utilização de casos de uso. Nesta fase são definidos também padrões arquiteturais e padrões de projeto de *software*. Na fase de construção são escritos casos de testes de unidade para cada funcionalidade especificada pelos casos de uso. Ao longo da fase de construção o *software* é construído de acordo com as atividades de codificação, refatoração (FOWLER, 2000), integração e testes. Na fase de transição o sistema é transposto do ambiente de testes para o ambiente de negócio, no qual são realizados testes de aceitação pelos usuários finais do sistema. Ao final da fase de transição é pretendido ter como resultado um sistema documentado e pronto para uso (SOMMERVILLE, 2007).

2.2.5 Metodologias Ágeis

De acordo com Sommerville (2007) os métodos ágeis ou metodologias ágeis foram propostos devido à insatisfação de um grupo de desenvolvedores com modelos de processo de desenvolvimento de *software* clássicos, os quais quando utilizados no desenvolvimento de sistemas pequenos demandavam grande quantidade de tempo na elaboração do projeto de *software* superando o tempo de desenvolvimento, agregado ao fato destes modelos clássicos apresentarem limitações com a ocorrência de mudanças nos requisitos do sistema ou de incertezas durante o processo de *software*.

Pressman (2011) define processo ágil como aquele capaz de adaptar-se de forma incremental a mudanças nos requisitos do sistema e imprevistos que surgem durante as fases de desenvolvimento do *software*. O autor afirma ainda que a adaptação incremental do processo somente é possível quando ocorre a contínua comunicação da equipe de desenvolvimento com o cliente durante as etapas de desenvolvimento do *software*.

Os princípios em comum nos métodos ágeis são definidos no "Manifesto para o Desenvolvimento Ágil de *software*". De acordo com Beck et al. (2001) alguns destes princípios são:

- Indivíduos e interações são mais importantes que processos e ferramentas;
- *Software* pronto é mais importante que documentação completa;
- Colaboração com o cliente é mais importante que negociação com contratos;

- Adaptação às mudanças é mais importante que seguir um plano estabelecido.

Foram propostos diversos métodos ágeis sendo os mais comuns os modelos: Desenvolvimento de *Software* Adaptativo, Desenvolvimento Dirigido a Funcionalidades, Desenvolvimento de *Software* Enxuto, Método de Desenvolvimento de Sistemas Dinâmicos, *Extreme Programming (XP)*, Modelagem Ágil, Processo Unificado Ágil e *Scrum* . Nas próximas seções serão abordadas como destaque as metodologias ágeis *Extreme Programming (XP)* e *Scrum* .

2.2.6 *Extreme Programming (XP)*

O *XP* é uma metodologia ágil que utiliza orientação a objetos como paradigma de desenvolvimento e envolve um total de quatro atividades metodológicas: planejamento, projeto, codificação e testes (PRESSMAN, 2011).

Na atividade de planejamento os requisitos de *software* são definidos com a ajuda do cliente compondo as histórias de usuário definidas como um conjunto de tarefas que o *software* deve realizar para atender ao seu propósito (SOMMERVILLE, 2007). Ao longo da atividade de planejamento cada história de usuário recebe um custo definido pela equipe mensurado em tempo de desenvolvimento, e calcula-se a quantidade de histórias que serão realizadas para compor a próxima versão do *software*. As histórias com maior prioridade são escolhidas para serem realizadas primeiro, e em caso de necessidade é permitido a divisão das histórias de usuário em histórias menores.

Na etapa de elaboração do projeto são identificados os componentes estruturais do sistema e suas relações, este conjunto então é modelado como classes relevantes para a versão atual do *software*. Antes da etapa de codificação são definidos testes unitários para cada história de usuário com o propósito de prevenção de erros.

Durante a etapa de codificação é incentivado que os programadores trabalhem em pares, estabelecendo-se que um membro do par seja designado a codificar as histórias de usuário, e o segundo membro designado a analisar o processo de codificação, sugerir melhorias e alterações no código, além de apontar possíveis erros.

O desenvolvimento do código na metodologia XP deve ser acompanhado de integração contínua (FOWLER, 2000) e refatoração (FOWLER, 2000), com o objetivo de

obter organização e simplicidade do código fonte (PRESSMAN, 2011). Além dos testes unitários realizados antes da etapa de codificação, Wells (1999) incentiva a execução diária de testes de integração e de validação do sistema com o objetivo de monitorar o andamento do projeto. Sommerville (2007) aponta a integração do cliente junto à equipe de desenvolvimento, com a responsabilidade de realizar testes de aceitação de funcionalidades do sistema.

As etapas de planejamento, projeto, codificação e teste presentes no ciclo de vida do modelo XP podem ser observadas na Figura 2.2, no qual observa-se o caráter cíclico do processo.

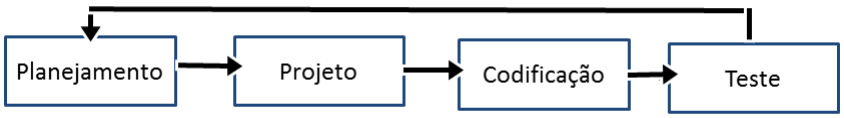

Figura 2.2: Diagrama demonstrando as etapas propostas pela metodologia ágil XP: planejamento, projeto, codificação e teste. Adaptado de (PRESSMAN, 2011).

2.2.7 Metodologia *Scrum*

A metodologia ágil *Scrum* tem como foco o gerenciamento de projetos. É um *framework* o qual permite a resolução de problemas através do emprego de vários processos e técnicas levando-se em conta o conhecimento adquirido a partir de experiências anteriores, com o objetivo de entregar um produto com a maior qualidade possível (SCHWABER; SUTHERLAND, 2015). O autor classifica o processo do *Scrum* como iterativo e incremental, o qual permite o monitoramento e controle de riscos do projeto.

As principais características do *Scrum* são definidas a seguir conforme (SCHWABER; SUTHERLAND, 2015):

1. Times de desenvolvimento pequenos compostos de 5 a 9 pessoas, visando a agilidade e a produtividade da equipe;

2. Divisão do trabalho total em uma lista de tarefas;

3. Entrega de novas funcionalidades ao final de cada iteração do processo as quais serão incrementadas ao produto final;

4. Inspeção e adaptação do sistema ainda em desenvolvimento;

5. Realização de reuniões de planejamento e de acompanhamento da equipe de desenvolvimento.

O autor afirma que o método *Scrum* tem como princípio dividir o processo em iterações de curto prazo denominadas *Sprints*, nas quais cada *Sprint* pode apresentar duração de duas a quatro semanas e este período não pode ser modificado durante sua execução.

Papeis do *Scrum*

Segundo Schwaber e Sutherland (2015) no *Scrum* são definidos três papéis principais definidos como *Scrum Master*, *Product Owner* e *Equipe Scrum*. Mesmo que as pessoas envolvidas atuem diretamente no desenvolvimento do projeto, as responsabilidades pelas tarefas devem ser bem claras e definidas antes do início do trabalho.

O *Scrum Master* atua como um gerente de projetos, tendo como meta o emprego dos valores e práticas do *Scrum*. Sua principal responsabilidade é garantir o andamento do projeto, removendo impedimentos e atuando como um facilitador durante a realização dos eventos do *Scrum* .

O *Product Owner* representa o cliente na equipe e tem como responsabilidade definir, priorizar e gerenciar os requisitos do projeto através do artefato *Product Backlog*. O principal objetivo do *Product Owner* é revisar e aceitar as entregas ao final de cada *Sprint* .

A *Equipe Scrum* é formada por membros da equipe responsáveis pelo desenvolvimento do projeto (programadores, testadores ou engenheiros). Geralmente é composta por 5 a 9 pessoas, uma vez que uma equipe muito pequena pode ter dificuldades para a realização de todas as metas da *Sprint* e uma equipe muito grande dificultaria a coordenação das tarefas pretendidas.

Artefatos do *Scrum*

Ao decorrer das fases e eventos do *Scrum* são gerados artefatos utilizados para aumentar a transparência das informações produzidas (SCHWABER; SUTHERLAND, 2015). O *Product Backlog* é um artefato formado por uma lista de prioridades dos itens necessários para a produção do produto final do projeto. Cada item da lista apresenta atributos como descrição, prioridade e estimativa em dias. O *Product Backlog* é um artefato dinâmico, podendo sofrer alterações ao decorrer do projeto. Os itens do *Product Backlog* podem ser funções, funcionalidades, melhorias e correções.

O *Sprint Backlog* é uma lista contendo um conjunto de tarefas selecionadas a partir do *Product Backlog* para serem realizadas durante uma determinada *Sprint* . O *Sprint Backlog* define o trabalho necessário para entregar a versão incremental do produto.

O *Burndown Chart* é um gráfico que permite visualizar o trabalho total restante para o término de uma *Sprint* . Os dados de entrada do *Burndown Chart* provêm do artefato *Sprint Backlog* e devem ser atualizados a cada dia durante o período da fase de *Sprint* permitindo o *Scrum Master* monitorar o andamento do projeto (COHN, 2010).

No eixo horizontal do gráfico são colocadas marcações correspondentes ao período de desenvolvimento mensurado em iterações ou em dias, e no eixo vertical as marcações representam a quantidade de trabalho restante para o término da *Sprint* ou do projeto. O *Taskboard* é um quadro onde são colocadas informações de acompanhamento das atividades da *Sprint* . O *Taskboard* deve ser atualizado diariamente durante a *Sprint* e deve estar sempre visível para os envolvidos no projeto, a sua função é agregar transparência e visibilidade ao processo de desenvolvimento (SCHWABER; SUTHERLAND, 2015). A Figura 2.3 demonstra como geralmente são distribuídas as atividades no *Taskboard* e a Figura 4.14 apresenta uma representação de *Burndown Chart*.

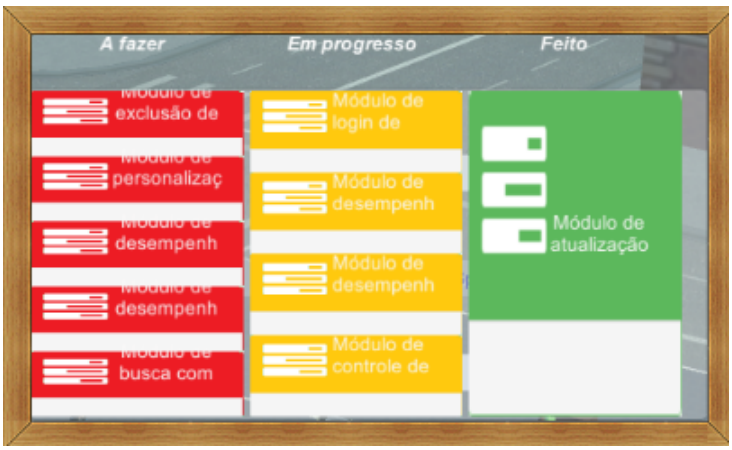

Figura 2.3: Exemplo demonstrativo de um *Taskboard*.

Fases e Ciclo de Vida

O *Scrum* pode ser dividido em três fases: *Pré-Sprint*, *Sprint* e *Pós-Sprint* (HIGHSMITH, 2003). A equipe de trabalho é dividida em times nos quais seus membros têm por definição os papéis de *Product Owner*, *Scrum Master* e *Team Member* (SCHWABER; SUTHERLAND, 2015). A Figura 2.4 descreve o ciclo de vida de uma *Sprint* apresentando os artefatos *Product Backlog* e *Sprint Backlog* produzidos durante a fase de *Pré-Sprint* e as Reuniões de Planejamento e de Retrospectiva da *Sprint* .

Os eventos no *Scrum* são eventos em um período fechado (em inglês, *Time-Boxed*), de forma que todo evento tem uma duração máxima. Este modelo de evento é adotado com o objetivo de criar uma rotina a qual permite a inspeção e adaptação de atividades ou recursos dentro do processo (SCHWABER; SUTHERLAND, 2015).

Os eventos definidos no *Scrum* são a Reunião de Planejamento da *Sprint* , *Sprint* , Reunião diária, Revisão da *Sprint* e Retrospectiva da *Sprint* . A fase de *Pré Sprint* do *Scrum* inicia-se com a primeira parte da Reunião de Planejamento da *Sprint* , a qual é definida a atividade de criação do *Product Backlog* formado por uma lista contendo todos os itens necessários para compor o produto.

O principal responsável pela condução bem sucedida da reunião de planejamento é o *Product Owner*. Após definido o *Product Backlog* é realizada a segunda parte da reunião de planejamento a qual tem como objetivo selecionar itens do *Product Backlog* e estimar o trabalho necessário para a realização e entrega dos itens ao final da *Sprint* .

A Reunião de Planejamento da *Sprint* conta com a participação de todos os

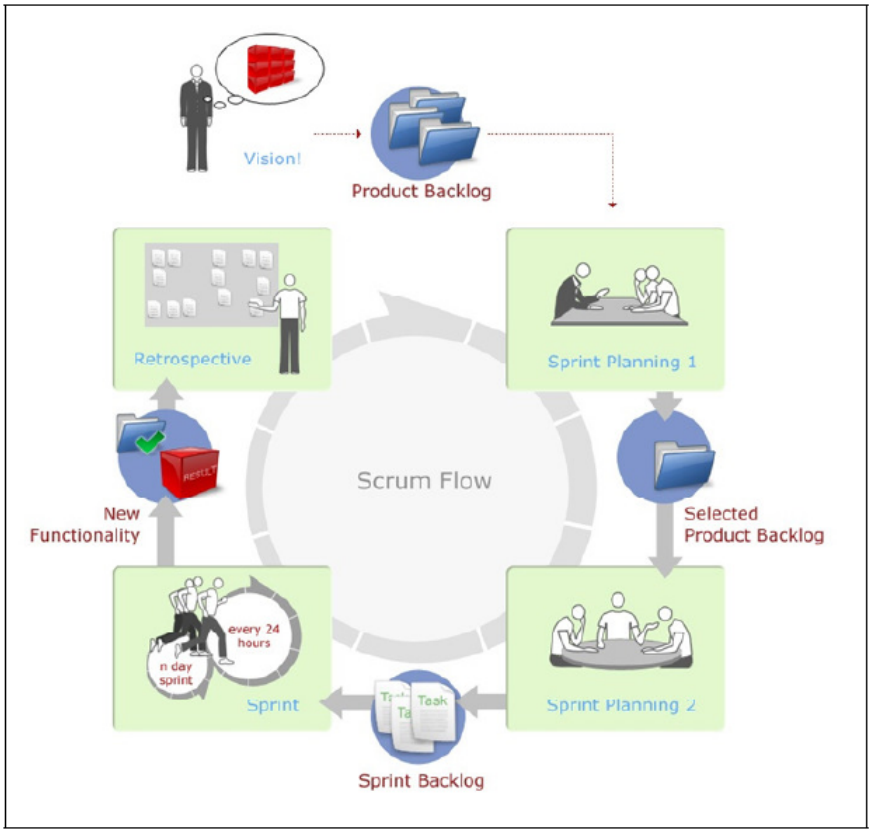

Figura 2.4: Ciclo de vida de uma *Sprint* definida pelo modelo *Scrum* (GLOGER, 2007).

membros da equipe, na qual o resultado obtido ao final da reunião é a composição do artefato *Sprint Backlog* contendo uma lista de funcionalidades retiradas do artefato *Product Backlog* para serem realizadas dentro da *Sprint* .

Durante a fase de *Sprint* ocorre a execução dos itens presentes no *Sprint Backlog*. Um dos eventos presentes dentro da *Sprint* é a Reunião Diária, a qual apresenta duração média de até 15 minutos com o objetivo de permitir ao *Scrum Master* monitorar o progresso da execução das tarefas do *Sprint Backlog*. Durante a Reunião Diária os *Team Members* devem esclarecer as seguintes questões ao *Scrum Master*: "O que eu fiz ontem que ajudou o time de desenvolvimento a atender a meta da *Sprint* ?", "O que eu farei hoje para ajudar o time de desenvolvimento atender a meta da *Sprint* ?" e "Eu vejo algum obstáculo que impeça a mim ou o time de desenvolvimento no atendimento da meta da *Sprint* ?"

As principais vantagens da Reunião Diária são a melhora da comunicação entre membros da equipe, identificação de impedimentos que interferem no desenvolvimento do projeto e a possibilidade de rápida tomada de decisões por parte do *Scrum Master*.

Uma *Sprint* pode ser cancelada pelo *Product Owner* caso as funcionalidades desenvolvidas se tornem obsoletas por influência das mudanças de mercado e de tecnologia.

Na fase de *Pós Sprint* ocorrem os eventos de Revisão da *Sprint* e Retrospectiva da *Sprint* . A Revisão da *Sprint* é um evento cujo objetivo é apresentar os resultados obtidos durante a *Sprint* , apresenta duração média de 4 horas para uma determinada *Sprint* com período de duas semanas, e ao longo da reunião são apresentados os problemas ocorridos durante a *Sprint* em conjunto com as soluções encontradas para a resolução dos problemas. A Retrospectiva da *Sprint* é um evento realizado após a Revisão da *Sprint* com duração média de 3 horas, e visa obter melhorias para a equipe, levando-se em conta a experiência adquirida durante a realização da *Sprint* (SCHWABER; SUTHERLAND, 2015).

2.3 *Design* Instrucional e modelo *ADDIE*

Design Instrucional é um processo sistemático que utiliza técnicas e recursos pedagógicos com o objetivo de melhorar o método de ensino (PAQUETTE, 2002). O *Design* Instrucional define a melhor técnica de ensino a ser utilizada com base em experiências anteriores e pode ser decomposto por um conjunto de processos denominado *Instrucional System Development* (*ISD*) ou Desenvolvimento de Sistemas Instrucionais em uma tradução livre. O *ISD* consiste em uma família de modelos os quais conduzem o desenvolvimento de tecnologias de ensino (MOLENDA, 2003).

Um modelo popular o qual define uma abordagem para o *ISD* é o modelo *ADDIE*. O termo *ADDIE* é um acrônimo em inglês para as etapas *Analyze*, *Design*, *Develop*, *Implement*, *Evaluate*, respectivamente em português Análise, Desenho, Desenvolvimento, Implementação e Avaliação (MOLENDA, 2003). A Figura 2.5 ilustra as cinco etapas do modelo *ADDIE*, cada fase do modelo *ADDIE* está consolidada e possui um conjunto de passos a serem seguidos de acordo com Strickland (2000).

A fase Análise é destinada ao levantamento de necessidades do público alvo com o objetivo de obter-se maior compreensão do ambiente de estudo, as necessidades dos usuários podem ser identificadas através de entrevistas, questionários ou de formulários. Na fase de desenho as informações levantadas na fase anterior são utilizadas para traçar

estratégias instrucionais e definir os objetivos de aprendizagem, conteúdos abordados e seleção de atividades pedagógicas.

A fase Desenvolvimento é direcionada à produção do objeto de aprendizagem, todo o conteúdo definido para a implementação do planejamento realizado na fase de desenho é criado nessa fase.

O objetivo da fase de Implementação é definir como o objeto de aprendizagem será aplicado e avaliado no ambiente de aplicação.

A fase de Avaliação é realizada de forma iterativa após o término de todas as outras fases do modelo conforme indicado na Figura 2.5. Na fase de Avaliação, procura-se realizar testes de aceitação e de desempenho do objeto de aprendizagem com o auxilio de representantes do público alvo. O objetivo da fase de avaliação é comparar os resultados obtidos com os resultados planejados e identificar os pontos negativos e positivos da ferramenta (STRICKLAND, 2000).

Figura 2.5: Diagrama caracterizando as cinco fases do modelo *ADDIE* Fonte: (OLIVEIRA; CSIK; MARQUES, 2015).

2.4 Trabalhos Relacionados

Nessa seção é apresentada a revisão sistemática da literatura acerca de jogos educacionais digitais que têm como objetivo o ensino de gerência de projetos de *software*, modelos de processo de desenvolvimento de *software* e metodologias ágeis. É apresentada a execução da busca e a extração e análise de resultados, incluindo descrição da ferramenta, objetivos de aprendizagem e estudos correlatos.

A pesquisa foi realizada no período entre agosto 2016 até julho de 2017, com o auxilio da ferramenta de busca Google Scholar[1], definindo como fontes bibliográficas artigos e periódicos escritos na língua inglesa ou portuguesa.

Os critérios utilizados para a análise de um resultado da busca são de que o jogo seja educacional e digital, disponibilidade do jogo para análise ou conter artigo que descreva as características do jogo.

Os resultados que não atenderam aos critérios de inclusão foram excluídos. Na primeira iteração em agosto de 2016, foram utilizadas as palavras "*teaching software engineering education digital tool*", retornando em média 18.000 resultados. Ao analisar os primeiros 50 resultados pela leitura dos resumos, apenas dois resultados atenderam aos critérios de busca: *Scrum Game* e *SimSE*. Paralelamente foi realizada a busca utilizando a tradução "*jogo educacional digital para ensino da engenharia de* software ", retornando 18.000 resultados, ao analisar os 50 primeiros resultados, apenas quatro resultados atenderam os critérios de inclusão: *SE-RPG*, *Scrumed*, *Scrum Scape*, *Scrumming*.

A seguir são apresentados jogos educacionais digitais para o ensino de gerência de projetos de *software* e de modelos de processo de *software* encontrados na realização da busca.

2.4.1 *SimSE*

O *SimSE* é um simulador educacional voltado para estudantes de engenharia de *software* desenvolvido por Emily Navarro (NAVARRO, 2006). O *SimSE* permite a simulação de rotinas e atividades cotidianas a uma empresa de desenvolvimento de *software* na qual o jogador tem como responsabilidade o gerenciamento de uma equipe de funcionários

[1] ⟨http://scholar.google.com.br⟩

atuando como gerente de projetos.

Ao decorrer do jogo, atividades relacionadas ao gerenciamento de projetos de *software* tais como gerenciamento de uma equipe de desenvolvimento, contratação e demissão de funcionários, delegação de tarefas e monitoramento de atividades do projeto são realizadas pelo jogador. O jogo possui interface gráfica 2D como pode ser visto na Figura 2.6. O *SimSE* permite a realização de simulações de modelos de processo de *software* como *XP*, modelo incremental, modelo cascata e modelo *RUP*. O jogo encontra-se apenas no idioma inglês, porém as ações e artefatos criados pelo usuário durante a simulação podem ser escritos em outro idioma.

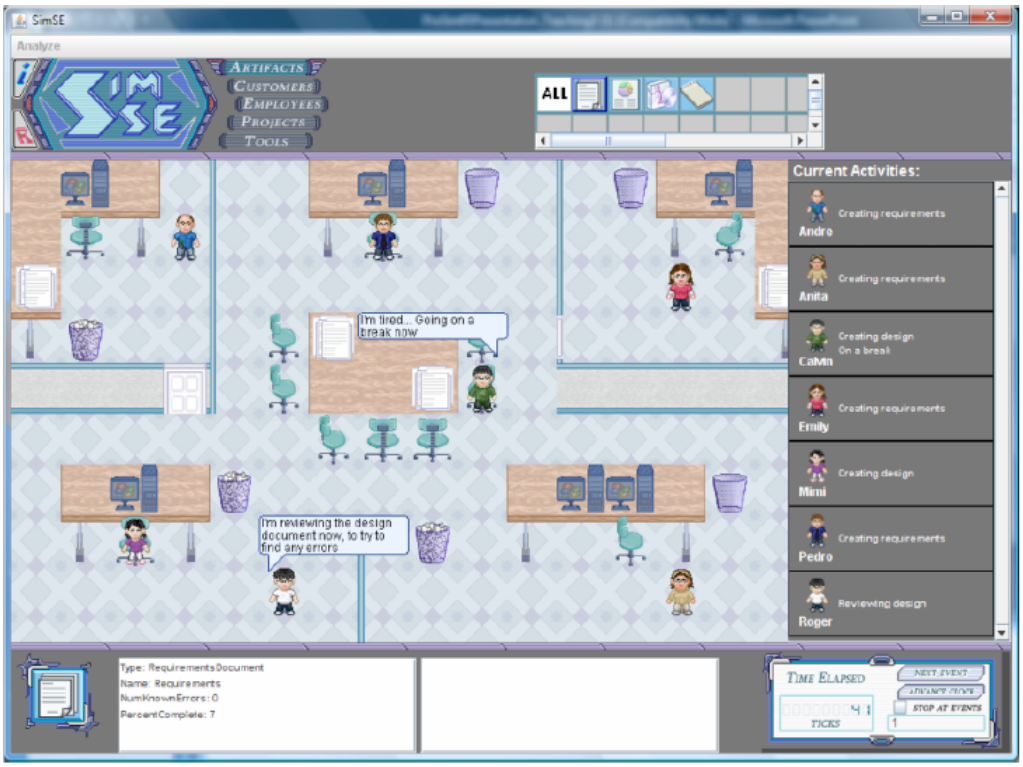

Figura 2.6: Interface do *SimSe* representando o ambiente de trabalho dos personagens.

2.4.2 *Scrum Game*

O *Scrum Game* (GKRITSI, 2011) é um jogo voltado ao ensino da metodologia ágil *Scrum* composto por dois modos: modo de jogo e modo administrativo. O modo de jogo é destinado aos jogadores e o modo administrativo é destinado aos instrutores de *Scrum* . Para iniciar o modo de jogo da ferramenta o jogador deve fazer um registro na ferramenta e em seguida fazer o *login*. No inicio do jogo são disponibilizadas informações de como

jogar o jogo e opções de seleção entre três projetos a serem realizados. Cada um dos projetos apresentados têm diferentes durações mensuradas em quantidade de *sprints*.

O autor destaca que fixou a duração de cada *Sprint* em dois dias. O jogador assume o papel de *Scrum Master* o qual tem a função de gerenciar, coordenar e ajudar o time escolhido a implementar o projeto escolhido. Os times são formados por padrão apenas por três membros, podendo este numero ser alterado no modo administrativo. O objetivo do autor ao reduzir a quantidade de *sprints* e o tamanho do time foi manter a simulação pequena, simples e interessante simplificando várias etapas e artefatos do *Scrum* . Na primeira fase do jogo, o jogador deve especular a duração e o esforço necessário para a realização de cada tarefa por meio das representações do *sprint backlog e product backlog*.

Na fase seguinte é realizada a simulação da *Sprint* e em seguida ocorre a reunião diária no qual o jogador realiza as três perguntas acerca de o que o personagem fez desde a ultima reunião, o que o personagem irá fazer do momento atual até a próxima reunião e quais problemas o membro do time está enfrentando.

Em seguida jogador deve escolher entre três opções diferentes que correspondem ao conselho que poderia resolver o problema enfrentado pelo membro do time. Durante a simulação da *Sprint* o jogador tem a possibilidade de visualizar o gráfico de *burndown* correspondente ao progresso do time. Quando uma *Sprint* é terminada é realizada a simulação da reunião de revisão e o jogador pode avaliar a parte completa do projeto. Caso a qualidade do projeto nesse ponto for inferior a 50% o jogador deve repetir a simulação da *Sprint* até que ele alcance a qualidade desejada, caso contrário ele pode proceder a próxima simulação de *Sprint* , este procedimento é repetido até que todas as *Sprints* sejam concluídas. Quando o projeto é terminado os jogadores têm a opção de visualizar os resultados obtidos.

No modo administrativo é possível visualizar um relatório completo acerca do progresso dos alunos. Em particular, é possível visualizar o projeto e número de jogos que cada jogador completou. O modo administrativo oferece ainda as opções de alterar os projetos existentes com a inserção de tarefas adicionais e alteração do tamanho dos times com a inserção de novos membros. Uma das interfaces do jogo pode ser visualizada na Figura 2.7

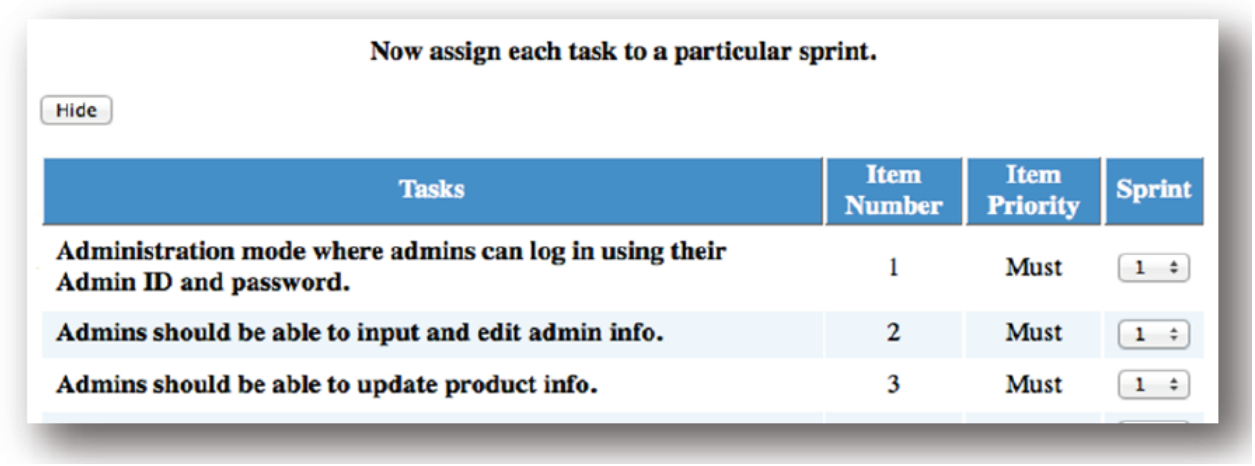

Figura 2.7: Interface do *Scrum Game* representando o artefato *Sprint Backlog*.

2.4.3 *Scrumed*

O *Scrumed* é um jogo de *Role Playing Game (RPG)* criado para ser aplicado em disciplinas de gerência de projetos ou de engenharia de *software* (SCHNEIDER, 2015). O jogo é contextualizado no período medieval o qual possui construções, ambientes e personagens característicos da Europa medieval. O jogador assume o papel de *Scrum Master* (SCHWABER; SUTHERLAND, 2015) e deve ajudar sua equipe formada por personagens não jogáveis ou *Non Playable Characters* (NPC's) a resolverem problemas apresentados pelos clientes presentes no jogo caracterizados como súditos do rei.

Ao decorrer do jogo são apresentados conceitos relacionados à metodologia *Scrum* como *sprints*, reuniões diárias e reuniões de revisão da *Sprint* (SCHWABER; SUTHERLAND, 2015). A Figura 2.8 mostra um dos cenários do jogo. O desenvolvimento do jogo foi realizado utilizando o motor de jogo *Unity 3D* (APS., 2017) e está disponível para *download* gratuito operando nas plataformas *Windows* e *Linux*

2.4.4 *SCRUM Scape*

O *SCRUM-scape* é um jogo de *RPG* no qual o jogador tem como objetivo escapar de uma prisão contextualizada no período medieval. O jogo possui três missões subsequentes as quais abordam respectivamente os papéis, cerimônias e artefatos da metodologia *Scrum* . Em cada missão é solicitado ao jogador responder questões relacionadas a um conceito específico do *Scrum* .

Figura 2.8: Cenário do Scrumed representando o planejamento da *Sprint* .

A cada questão respondida corretamente pelo jogador é concedido o auxílio de um *NPC* para a realização do próximo nível de jogo descrito como uma missão, caso o jogador responda incorretamente à questão apresentada o auxílio do personagem não é concedido. O jogo é finalizado após a resposta de todos os questionamentos ou com a derrota de todos os *NPC's* inimigos (CAMARGO, 2013). A Figura 2.9 mostra um dos níveis do jogo. O desenvolvimento do jogo foi realizado utilizando o motor de jogo *RPG Maker* e está disponível para *download* gratuito operando em computadores *stand alone*.

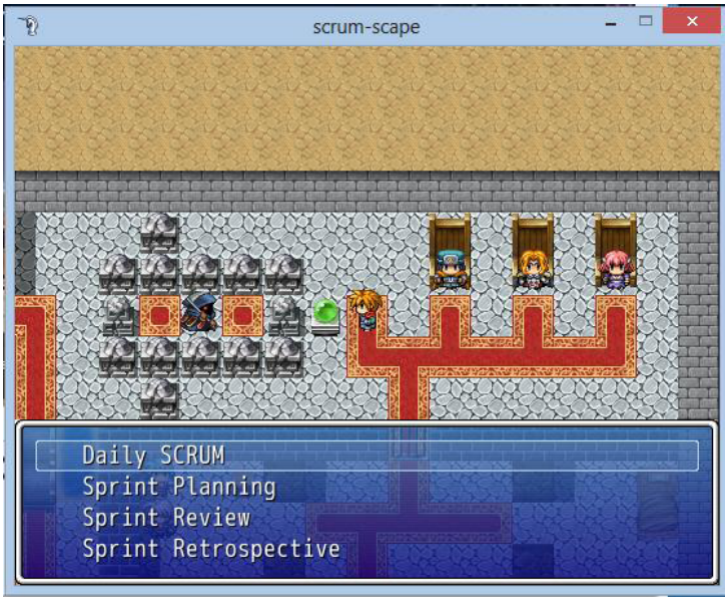

Figura 2.9: Um dos cenários do jogo *SCRUM-scape* ambientando uma missão do jogo.

2.4.5 *Scrumming*

O *Scrumming* é uma ferramenta com o objetivo de permitir ao usuário treinar atividades relacionadas à metodologia Ágil *Scrum*. O principal objetivo da ferramenta é auxiliar no ensino de gerência de projetos com o uso da metodologia *Scrum* . A ferramenta é composta por dois módulos, um administrativo e o outro de simulação. O módulo administrativo da ferramenta permite ao usuário aplicar configurações antes da simulação desejada, ficando a cargo do jogador a inserção de funcionários e atividades da simulação (NETO, 2008).

O módulo de simulação do jogo permite ao usuário assumir o papel de *Scrum Master* (SCHWABER; SUTHERLAND, 2015) tendo como funções definir o numero de ciclos (*sprints*) do processo de desenvolvimento e adicionar ou remover atividades (NOLASCO, 2013). Segundo (NOLASCO, 2013), o *Scrumming* é uma alternativa ao complemento no ensino da metodologia *Scrum* , pois possui interface amigável ao usuário e modulo administrativo de fácil utilização. A Figura 2.10 mostra uma das interfaces do *Scrumming*.

De acordo com Neto (2008) o *Scrumming* tem como publico alvo estudantes e profissionais com nível básico de conhecimento em gerência de projetos e na metodologia *Scrum* . O jogo foi desenvolvido em *Java* e tem como idioma padrão a língua portuguesa (NETO, 2008)

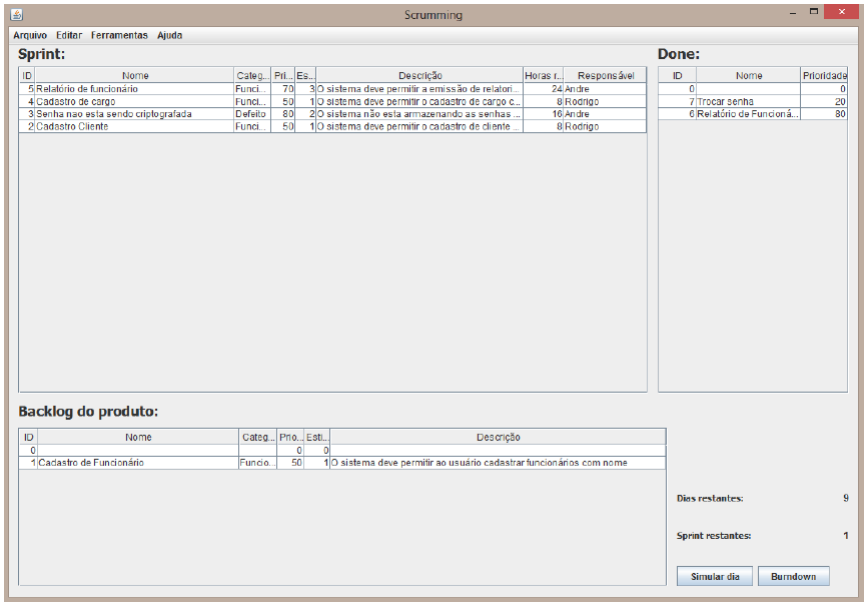

Figura 2.10: Interface do *Scrumming* aprensentando os itens dos artefatos do *Scrum Sprint Backlog* e *Product Backlog*.

2.4.6 *SE RPG 2.0*

O SE-RPG é um jogo do tipo *Role Playing Game* (RPG) no qual é feita a simulação do ambiente de trabalho de uma empresa fictícia de desenvolvimento de *software* , apresentando interfaces gráficas de três ambientes: recepção, sala de reuniões e sala de produção. O jogo visa ensinar conceitos de engenharia de *software* e segue uma sequencia de atividades relacionados com as atividades de uma empresa de *software*, o jogador deve escolher o projeto a ser desenvolvido, um dos modelos de processo de *software* Cascata ou Iterativo, a linguagem de implementação do projeto e a realização de contratação de funcionários. No jogo cada personagem é diferenciado por suas habilidades e o jogador pode escolher os personagens com as características desejadas para o projeto. Durante o jogo são atribuídas tarefas a cada personagem e o progresso de cada atividade pode ser acompanhado pelo jogador, ao final do jogo o jogador conta com o relatório acerca do cumprimento das metas estabelecidas como escopo, prazo e custo, satisfação do cliente e verificação se o modelo de processo foi corretamente escolhido pelo projeto (AMBROSIO, 2008). A Figura 2.11 apresenta o ambiente principal do jogo.

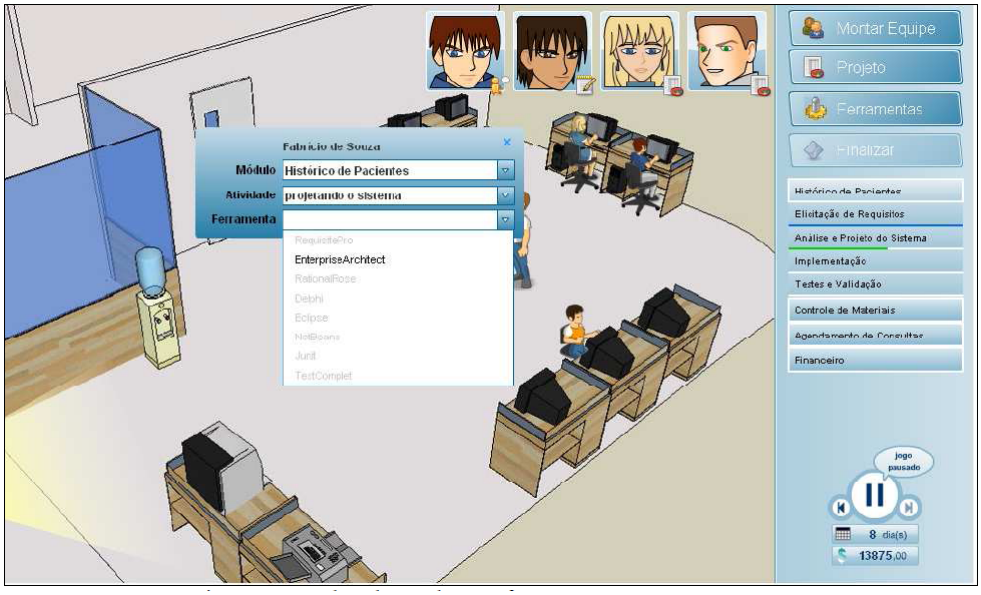

Figura 2.11: Ambiente da sala de desenvolvimento do SERPG2.

2.4.7 Análise dos Jogos Digitais Pesquisados

Nesta seção é realizada a análise acerca dos jogos *SimSE* (NAVARRO, 2006), *Scrum Game* (GKRITSI, 2011), *Scrummed* (SCHNEIDER, 2015), *Scrum Scape* (CAMARGO, 2013), *Scrumming* (NETO, 2008) e SE RPG 2.0 (AMBROSIO, 2008).

O *SimSE* (NAVARRO, 2006) apresenta características atraentes aos jogadores e instrutores. A vantagem desta ferramenta sobre as demais ferramentas analisadas é a possibilidade da modelagem dos processos de *software* pelo instrutor no qual são disponibilizados a confecção de atividades e artefatos possibilitando ao instrutor criar modelos que adequem-se aos seus propósitos. O *SimSE* permite ao jogador interagir com objetos da simulação e apresenta representações gráficas destes objetos e dos personagens de jogo além de um cenário simulando um ambiente de trabalho real. O *feedback* ao jogador ao longo da simulação é dado através de um gráfico que apresenta o progresso das atividades escolhidas pelo jogador. Ao final da simulação é apresentado ao jogador métricas como desempenho durante a simulação e informações adicionais.

Estudo inicial com trinta estudantes do curso de Ciência da Computação e três estudos subsequentes mostraram que o *SimSE* tem potencial para ser uma ferramenta educacional efetiva no ensino acerca de processos de desenvolvimento de *software* (NAVARRO, 2006). A desvantagem desta ferramenta sobre as demais pesquisadas é a necessidade de conhecimentos técnicos por parte do jogador para iniciar a simulação de um modelo.

O *Scrum Game* (GKRITSI, 2011) apresenta um rápido *feedback* ao jogador, pois a avaliação de cada simulação de *Sprint* é feita automaticamente ao seu término. O jogo disponibiliza instruções e explicações acerca da simulação e o usuário pode acompanhar o seu progresso utilizando artefatos presentes no modelo *Scrum* como o gráfico de *burndown*.

O jogo oferece ainda flexibilidade aos instrutores através da possibilidade da edição dos projetos existentes, oferecendo a possibilidade da inserção de tarefas adicionais, ajuste do tamanho dos times, inserção de empregados, edição das habilidades dos membros do time e das perguntas realizadas na simulação da reunião diária. No modo administrativo é possível ainda ajustar o nível de dificuldade do jogo de acordo com o nível de conhecimento dos jogadores.

Segundo Gkritsi (2011) um estudo realizado com 22 participantes, resultou que 86% dos participantes consideraram que seu nível de conhecimento acerca do *Scrum* aumentou após o jogo, 91% consideraram que o jogo poderia ser potencialmente usado como material complementar em aulas, e 76% acharam o jogo agradável de ser jogado. Como desvantagem em relação ao uso da ferramenta, o jogo apresenta elementos de interface de usuário pouco atraentes, pois dispõe na maioria das atividades de elementos textuais.

O *Scrumed* apresenta uma boa narrativa com personagens e modelos em 3D. O destaque da ferramenta é envolver o jogador em um cenário de fantasia onde deve resolver problemas relacionados à história apresentada pelo jogo utilizando a metodologia *Scrum* , com modo de interação individual o jogador tem total liberdade para explorar o cenário e tentar realizar os objetivos propostos pelo jogo. O *feedback* ao jogador é apresentado através dos diálogos dos personagens e a representação de artefatos da metodologia *Scrum* . No final do jogo é mostrado se a execução da *Sprint* (interação) do projeto foi bem sucedida, apresentado neste caso um cenário de sucesso no jogo ou caso contrário apresentando um cenário de fracasso. Uma vantagem do jogo é o baixo nível de dificuldade, permitindo a alunos com nível básico de conhecimento em engenharia de *software* jogar sem problemas.

O jogo encontra-se apenas no idioma português e um estudo realizado com membros do Grupo de Qualidade de *software* do INE/UFSC (GQS) e alunos da disciplina Planejamento e Gestão de Projetos do curso de Bacharelado em Ciências da Computação INE/UFSC mostrou boa eficiência do jogo como ferramenta de apoio a aprendizagem do *Scrum* (SCHNEIDER, 2015). Uma desvantagem do jogo em relação aos outros pesquisados neste trabalho é o *feedback* fornecido ao jogador o qual não apresenta de forma clara se as atividades da metodologia *Scrum* estão sendo executadas de forma correta.

O *SCRUM Scape* apresenta boa narrativa e elementos de ação semelhantes a jogos digitais de *RPG*. Uma vantagem do jogo é a simplicidade da jogabilidade e o desafio proposto ao jogador em evoluir no jogo através das fases que vão sendo liberadas a cada objetivo alcançado. O *feedback* ao jogador é dado à medida em que o jogador responde a questionamentos acerca de conceitos da metodologia *Scrum* , como artefatos e cerimonias com o objetivo da obtenção de vantagens dentro do jogo.

Estudo realizado com dez estudantes da disciplina de Planejamento e Gestão de Projetos do curso de Bacharelado em Ciências da Computação INE/UFSC e sete profissionais da área de Tecnologia da Informação mostrou que para 70% dos participantes o jogo foi eficiente para a aprendizagem do *Scrum* (CAMARGO, 2013). Uma desvantagem do jogo é que dúvidas a respeito da metodologia *Scrum* devem ser resolvidas fora do ambiente de jogo, e o jogo não apresenta nenhuma forma de instrução acerca da metodologia *Scrum* exigindo do jogador algum conhecimento prévio acerca do *Scrum* .

O *Scrumming* com seu módulo administrativo permite ao instrutor configurar a simulação da forma mais adequada para os seus alunos. O modo de simulação do jogo permite ao jogador interagir com os elementos do cenário definidos pelo instrutor. Uma vantagem oferecida pelo *Scruming* é a liberdade dada ao jogador para configurar as atividades e recursos que serão utilizados durante a simulação do *Scrum* .

O *feedback* ao jogador é dado de forma imediata durante a simulação, pois a cada ação do usuário durante a simulação de uma *Sprint* é processado um evento de resposta correspondente à ação realizada pelo jogador através de janelas informativas e ocorre a mudança de estado dos objetos presentes na interface do jogo. Não foi especificado no artigo da ferramenta a realização de avaliação empírica do jogo. Uma desvantagem do jogo é a falta de elementos gráficos que representem de fato os artefatos e eventos do *Scrum* em uma simulação.

O SE RPG 2.0 apresenta personagens, atividades e cenários atraentes, os quais permitem simular um ambiente de trabalho realista. Ao decorrer do jogo, o jogador tem como desafio lidar com os problemas apresentados pelos funcionários, como conflitos com outros funcionários e realocação de funcionários em caso de ferias de algum outro funcionário. O *feedback* ao jogador é dado através de representações gráficas que informam ao jogador o progresso de determinada atividade proposta e ao final do jogo é mostrado se o projeto foi finalizado à tempo. Uma desvantagem no jogo é a falta de mais cenários e a limitação da escolha de apenas dois modelos de processo de *software*, modelos cascata e modelo iterativo.

3 Desenvolvimento do Jogo Digital *SEJam*

Neste capítulo é apresentado o planejamento e desenvolvimento do jogo digital *SEJam* de acordo com as diretrizes propostas pelo modelo *ADDIE*, visto na Seção 2.3 do Capítulo 2.

3.1 Modelo *ADDIE* para o *SEJam*

O planejamento do jogo *SEJam* foi efetuado utilizando como recurso o modelo de desenho instrucional *ADDIE* (MOLENDA, 2003). O modelo *ADDIE* foi escolhido de forma a permitir a definição clara e concisa das etapas do processo de elaboração das atividades providas pelo jogo e as revisões necessárias com o objetivo de garantir a efetividade da aprendizagem proporcionada pelo jogo. A seguir são descritas as fases do modelo *ADDIE* aplicadas ao jogo *SEJam* .

3.2 Análise

Na fase de análise foram definidos os objetivos do jogo estabelecendo-se como meta o levantamento de necessidades de aprendizagem e a definição do público alvo do jogo e os recursos tecnológicos a serem utilizados. As diretrizes para a criação do jogo definem como público alvo alunos e profissionais da área da Ciência da Computação com nível básico de conhecimento em Engenharia de *Software*. O objetivo educacional do jogo será possibilitar o jogador aprender de forma prática e didática as funções dos artefatos, eventos e papéis característicos da metodologia ágil *Scrum* .

É desejável que ao final da simulação proporcionada pelo jogo, a habilidade do aluno saber diferenciar as responsabilidades de cada um dos papéis presentes na metodologia *Scrum* bem como as suas principais funções dentro de um projeto. É um dos objetivos do jogo, o aprendizado sobre os relacionamentos entre os artefatos gerados durante os eventos do *Scrum* . Para o desenvolvimento tecnológico do jogo será definido a

utilização da versão 2017.2.2 do motor gráfico *Unity 3D* (APS., 2017).

3.2.1 Requisitos Funcionais do *SEJam*

Os requisitos funcionais foram definidos a partir da especificação das atividades propostas pelo jogo respeitando-se o aparato tecnológico disponível.

`Apresentação das atividades do ciclo de vida do modelo Scrum` : O modelo *Scrum* dentro do *SEJam* deve apresentar ao jogador as etapas: Reunião de Planejamento da *Sprint* ; Reunião Diária; Revisão da *Sprint* e Retrospectiva da *Sprint* .

`Apresentação dos papéis do Scrum` : O jogo deve introduzir ao jogador funções e responsabilidades dos papéis *Scrum Master*, *Product Owner* e *Team Members* presentes no modelo *Scrum* .

`Apresentação dos Artefatos do Scrum` : O modelo *Scrum* dentro do *SEJam* no jogo deve apresentar ao jogador os artefatos *Product Backlog*, *Sprint Backlog*, *Taskboard* e *Sprint Burndown Chart*.

`Fornecer feedback ao jogador acerca do seu progresso e desempenho na simulação`: O jogo deve permitir ao jogador acompanhar o progresso das tarefas das *Sprints* do projeto através da representação do artefato *Sprint Burndown Chart*. O jogo deve ainda permitir ao jogador acompanhar o progresso das tarefas do projeto através da representação do artefato *TaskBoard* do projeto escolhido.

`Permitir ao jogador visualizar as estatísticas acerca da simulação do projeto`: O jogo deve apresentar o desempenho de cada personagem ao decorrer de cada simulação de dia de desenvolvimento do projeto. O jogo deve apresentar a satisfação do cliente ao decorrer de cada simulação de dia de desenvolvimento do projeto. O jogo deve permitir ao jogador finalizar o projeto apresentando o resultado da atuação do jogador ao final da simulação do desenvolvimento do projeto.

`Permitir priorizar as histórias de usuário que devem ser realizadas no projeto`: O jogo deve permitir ao jogador priorizar quais histórias de usuário devem ser realizadas durante o projeto escolhido através de uma interface representando o artefato *Product Backlog*.

`Permitir atribuir Sprints às histórias de usuário escolhidas para serem`

`implementadas durante o projeto`: O jogo deve permitir ao jogador atribuir uma *Sprint* a cada história de usuário selecionada do *Product Backlog* por meio de uma interface representando o artefato *Sprint Backlog*.

`Permitir a definição de personagens para participar da simulação do modelo Scrum` : Com o objetivo de otimizar o jogo e manter um número mínimo de personagens para a simulação do modelo *Scrum* , o jogador deve estar limitado a escolher exatamente três personagens para integrarem o time *Scrum* , onde em seguida deve-se atribuir os papeis de *Scrum Master*, *Product Owner* e *Time Scrum* aos personagens.

`Disponibilizar projetos fictícios para simulação da aplicação modelo Scrum` : O jogo deve disponibilizar projetos fictícios para serem utilizados na simulação do modelo *Scrum* .

`Permitir ao jogador atribuir papéis do Scrum aos personagens`: O jogo deve permitir ao jogador definir os papéis de *Scrum Master, Product Owner e Time Scrum* respectivamente a cada personagem escolhido para a simulação do modelo para o projeto.

`Permitir simular atividades caraterísticas exercida por cada papel do Scrum` : O jogo deve simular a realização de tarefas específicas a cada papel do modelo *Scrum* presente no jogo utilizando os personagens definidos com os papéis correspondentes.

`Permitir controlar a simulação do dia de desenvolvimento`: O modelo *Scrum* presente no jogo deve permitir ao jogador controlar a simulação dos dias de desenvolvimento do projeto fictício escolhido pelo jogador.

3.2.2 Requisitos Não Funcionais do *SEJam*

A seguir são apresentados os requisitos não funcionais que definem o mapeamento qualitativo do jogo.

`Portabilidade`: O jogo deve possuir versões compatíveis com as plataformas *Android, Windows, Linux e HTML5*.

`Idioma do Jogo`: O jogo deve estar na língua portuguesa, podendo ser traduzido para outras línguas em versões futuras.

`Disponibilidade`: O jogo deve estar disponível de forma gratuita.

`Interfaces Gráficas`: O jogo deve oferecer um ambiente com cenários e personagens tridimensionais. Os modelos tridimensionais presentes no jogo deverão ser modelados com ferramentas de modelagem gratuitas. As animações presentes no jogo deverão ser feitas com ferramentas de animação gratuitas. Como auxílio para a criação, edição e animação de modelos tridimensionais presentes no jogo foram escolhidas as ferramentas *Blender* (FOUNDATION, 2017), *Make Human* (HUMAN, 2017), *Sweet Home 3D* (ETEKS, 2015), *Scketchup Make* (INC., 2017).

`Interfaces de Usuário`: O jogo deve oferecer interfaces de usuário simples e intuitivas.

`Ferramenta de Desenvolvimento e Linguagem de Programação`: O jogo deve ser desenvolvido utilizando o motor de jogo *Unity 3D* na versão 2017.2.2 (APS., 2017) e ter como linguagem de programação C#.

3.3 Desenho

Na fase de desenho foi definida a estrutura de atividades e de conteúdos abordados pelo jogo *SEJam* definindo assim a melhor estratégia de ensino da metodologia *Scrum* ao público alvo do jogo.

A estrutura de conteúdo do *SEJam* foi definida de forma que deve-se expor ao jogador os papéis *Scrum Master*, *Product Owner* e *Time Scrum* previstos na metodologia *Scrum* além de funções e atividades relacionadas a cada um dos papéis. Ao decorrer do jogo devem ser apresentados artefatos específicos do ciclo de vida do *Scrum* e o relacionamento de cada artefato com os papéis definidos pela metodologia conforme Figura 2.4.

Os artefatos abordados pelo jogo são respectivamente *Product Backlog*, *Sprint Backlog*, *Taskboard* e *Sprint Burndow Chart*. As etapas *Reunião de Planejamento da Sprint, Reunião Diária, Revisão da Sprint e Retrospectiva da Sprint* relacionadas ao ciclo de vida do *Scrum* devem ser abordadas com foco em suas principais características.

As atividades do jogo foram divididas em etapas nas quais cada uma faz a simulação de uma fase real no desenvolvimento de um aplicativo, abordando fases como definição de um time de desenvolvimento, escolha de problemas a serem resolvidos e

utilização de um modelo de simulação da metodologia *Scrum* para o gerenciamento do projeto e desenvolvimento do aplicativo.

Na primeira atividade do jogo é proposto ao jogador escolher personagens para participarem da simulação do projeto, e são dadas opções de problemas do mundo real a serem resolvidos pelo projeto. A seguir o jogador deve definir um dos personagens selecionados para atuar como *Scrum Master*, o qual desempenhará funções de gerenciamento como o gerenciamento de funcionários.

O jogador deve escolher um personagem para atuar como *Product Owner*, o qual tem como funções definir quais itens do *Product Backlog* serão realizados, além de definir *sprints* para os itens presentes no *Sprint Backlog* do projeto. As Figuras 3.2 e 3.3 mostram o modelo conceitual acerca de cada papel assumido pelos personagens do jogo. O jogo propõe ao jogador gerenciar as atividades dos personagens definidos como *Scrum Master* e *Product Owner*.

O intuito de permitir ao jogador gerenciar personagens com papéis diferentes tem o propósito de oferecer maior conhecimento acerca das diferenças entre os papéis do *Scrum* . A Figura 3.1 demonstra a possibilidade de atuação de um personagem como *Scrum Master*, *Product Owner* ou como *Team Member*.

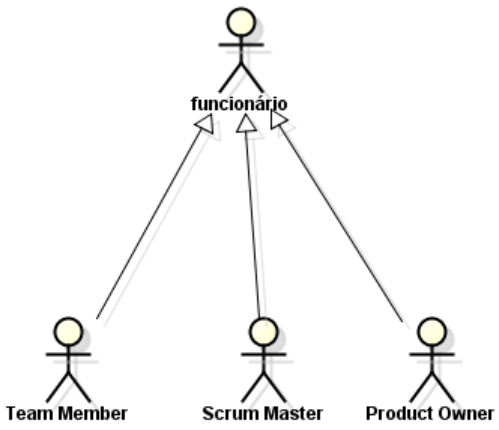

Figura 3.1: Diagrama de Casos de Uso demonstrando a especialização de um personagem como *Scrum Master*, *Product Owner* ou *Team Member*.

Na atividade seguinte é iniciada a simulação do desenvolvimento do projeto com a definição acerca de quais itens do *Product Backlog* serão selecionados para compor o *Sprint Backlog* do projeto. Em seguida o jogador deve atribuir uma *Sprint* para cada item

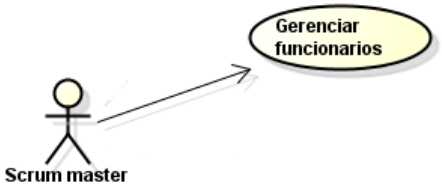

Figura 3.2: Diagrama de Casos de Uso demonstrando as possíveis ações do personagem definido como *Scrum Master*.

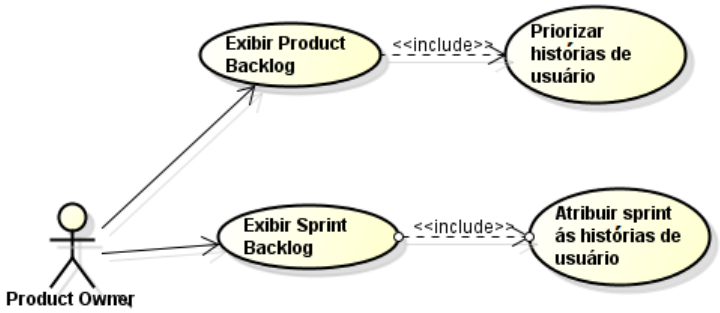

Figura 3.3: Diagrama de Casos de Uso demonstrando as possíveis ações do *Product Owner* no jogo.

do *Sprint Backlog* gerado após a definição do *Product Backlog*. Cada *Sprint* (iteração) do projeto suporta um número máximo de histórias de usuário de acordo com o esforço exigido para realizar cada história. O total de *Sprints* definidas para o projeto é calculada automaticamente pelo jogo de acordo com quantidade de histórias definidas pelo usuário na interface representativa do *Product Backlog* do projeto. Antes de iniciar a simulação de uma *sprint* o jogador deve atribuir pelo menos um personagem para realizar cada uma das etapas de *design*, codificação, integração e testes associadas a cada item do *Sprint Backlog*.

A atividade seguinte oferecida pelo jogo consiste na simulação de uma *Sprint* do projeto definida com a duração de vinte e um dias conforme determinado por Schwaber e Sutherland (2015). Os personagens simulam a realização das etapas de *design*, codificação, integração e testes para cada item do *Sprint Backlog* da *Sprint* corrente. Ao final da simulação de um dia na *Sprint* é realizada a reunião diária onde o jogador tem a opção de visualizar os impedimentos enfrentados pelos personagens durante o andamento da *Sprint*

. Em seguida cabe ao jogador tomar decisões de gerenciamento como, por exemplo, fazer a alocação ou remoção de um personagem responsável por determinada tarefa na *Sprint* corrente. As principais atividades do jogo descritas podem ser observadas na Figura 3.4.

Ao longo da simulação de uma *Sprint* do projeto para cada item do *Sprint Backlog* é respeitada a ordem e execução das etapas de desenvolvimento onde existe uma ordem de execução entre as etapas. É definido pelo jogo um prazo médio de cinco dias para a realização de cada etapa totalizando vinte e um dias dentro da *sprint*.

Para cada personagem participante da simulação é definido um conjunto de quatro perícias *design*, codificação, integração e testes relacionadas às etapas de desenvolvimento dos itens do *Sprint Backlog*. Cada perícia apresenta um valor percentual entre zero e um. Cada item do *Sprint Backlog* possui uma quantidade de pontos associados a cada uma das etapas de desenvolvimento. Após a simulação de um dia de desenvolvimento do projeto é verificada para cada item do *Sprint Backlog* a etapa atual de desenvolvimento do item. Dada a etapa atual de desenvolvimento do item é verificado para cada personagem se este está associado à etapa atual. Caso o personagem esteja associado a etapa atual, o valor da sua perícia associada a etapa é utilizado para incrementar a quantidade de pontos realizados na *sprint*. A quantidade de pontos realizados na *sprint* é incrementada com o resultado do produto dos pontos dos itens do *Sprint Backlog* associados à etapa atual de desenvolvimento pelo valor percentual da perícia do personagem associado a etapa.

Caso um personagem esteja associado a alguma das etapas de desenvolvimento de um item do *Sprint Backlog*, o valor percentual da produtividade do personagem no dia de desenvolvimento do projeto é dado pelo valor da sua perícia correspondente a etapa de desenvolvimento do item do *Sprint Backlog*. Caso a etapa associada ao personagem não esteja em andamento ou o personagem não esteja relacionado a nenhuma etapa, o valor da sua produtividade não é levado em consideração na soma da produtividade total dos personagens.

O índice de satisfação do cliente durante a realização do projeto é definido com base na produtividade dos personagens ao longo da simulação de *Sprint* . O índice de satisfação é incrementado caso a produtividade do personagem participante da simulação corresponda ao valor de 100% no dia de simulação da *Sprint* ou é decrementado caso

contrário.

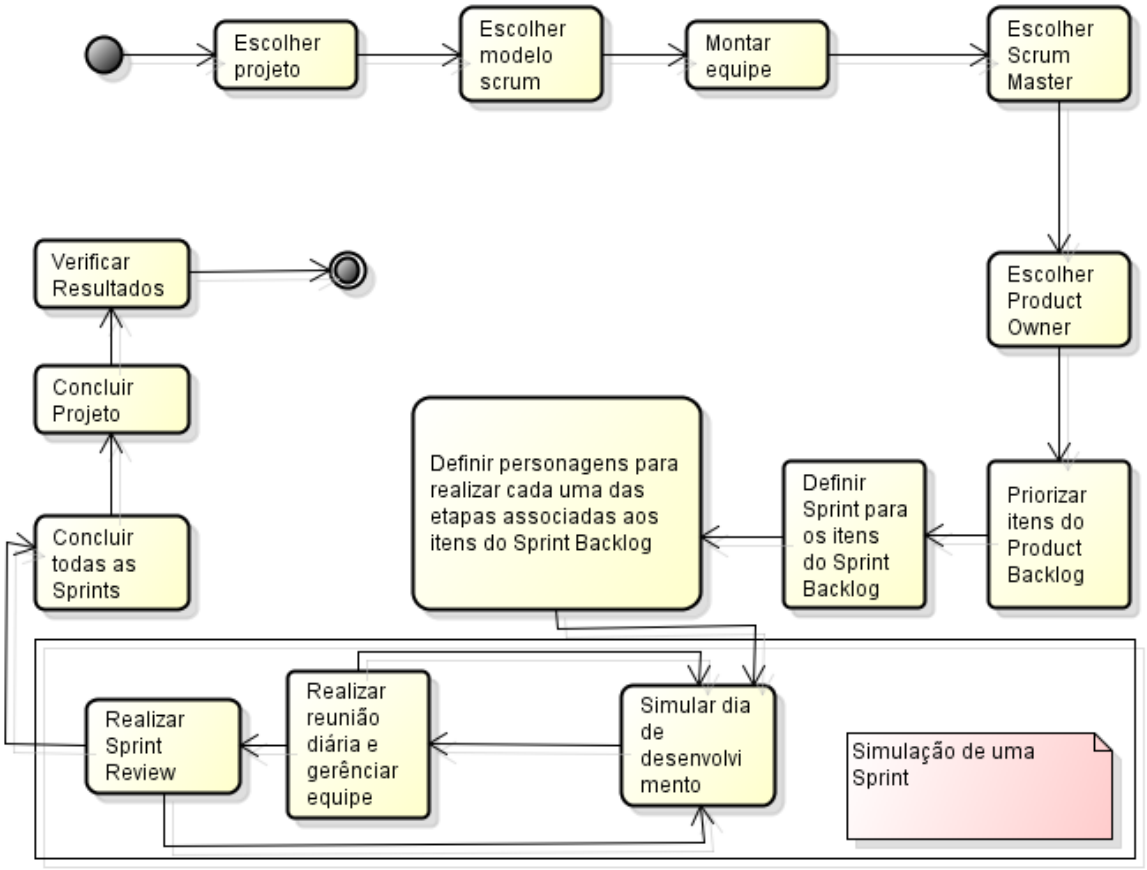

Figura 3.4: Diagrama de estados mostrando a sequencia de atividades propostos pelo jogo *SEJam*

3.4 Desenvolvimento

Nesta seção é apresentada a etapa de desenvolvimento do jogo *SEJam* . Durante a etapa de desenvolvimento foi escolhido o motor gráfico *Unity* (APS., 2017) para construção do jogo atendendo aos requisitos não funcionais definidos na fase de planejamento. São definidas as ferramentas utilizadas para a criação dos modelos tridimensionais presentes no jogo e ao final desta etapa é gerada uma versão incremental do jogo digital *SEJam* .

3.4.1 Ciclo de Desenvolvimento do *SEJam*

O ciclo de desenvolvimento do *SEJam* seguiu as seguintes etapas: análise dos requisitos definidos na fase de análise do modelo *ADDIE* e das atividades propostas definidas na fase

de projeto do modelo *ADDIE*, modelagem dos cenários e de personagens do jogo e codificação. Na fase de análise do modelo *ADDIE*, responsável por determinar características do jogo como definição de conteúdo e funcionalidades do jogo foi definido que o jogo deve disponibilizar três personagens para a simulação do modelo *Scrum* os quais devem assumir papeis de *Scrum Master*, *Product Owner* e *Team Member* respectivamente.

O modo de interação do jogador com os personagens será através de seleção do personagem desejado sendo em seguida exibido uma interface correspondente a função empregada pelo papel do personagem. As animações correspondentes a cada personagem serão coordenadas por meio de eventos disparados por ações do jogador. Os personagens poderão executar três tipos de animações dependendo do contexto apresentado pelo jogo incluindo as animações de andar, digitar e esperar.

Para a implementação do jogo foi pesquisado o ambiente de modelagem e motor de jogo *Blender* (FOUNDATION, 2017), *software* gratuito e de código aberto o qual disponibiliza opções para o desenvolvimento de animações, modelagem tridimensional e possui funcionalidade direcionada para criação de jogos 3D. Outro motor gráfico estudado foi o *Unreal Engine* (TECHNOLOGY, 2017), direcionado para jogos de ação de ambientação 3D.

Por fim, para a criação do jogo *SEJam* foi pesquisado e escolhida a versão gratuita do motor de jogo *Unity* (APS., 2017) . A escolha da ferramenta justifica-se pela vasta coleção de funcionalidades e recursos, como a possibilidade de criação e utilização de *scripts* codificados nas linguagens de programação *C#* , sistema completo para a criação de interfaces de usuário e a possibilidade de gerenciar as animações de modelos tridimensionais (HOCKING et al., 2015). Outros fatores os quais justificam a escolha da ferramenta são a facilidade de aprendizagem, grande disponibilidade de tutoriais e de vídeo aulas na *web* e possibilidade de criação de jogos para diversas plataformas.

3.4.2 Modelagem dos Personagens e Cenários do Jogo

Para a criação dos personagens do jogo foi utilizada a ferramenta *open source Make Human* (HUMAN, 2017). O *MakeHuman* permite a geração de modelos humanoides em 3D. A ferramenta permite que os modelos criados sejam exportados nos formatos obj

e dae, facilitando a utilização e edição dos modelos em outros *softwares* como *Blender* e *Unity*. A Figura 3.5 apresenta um dos personagens do jogo modelado na ferramenta *MakeHuman*.

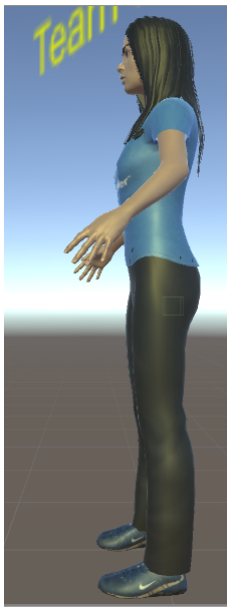

Figura 3.5: Personagem tridimensional modelado na ferramenta *MakeHuman* em vista lateral.

Para a criação das animações dos personagens do jogo foi utilizada a ferramenta de código aberto *Blender* (FOUNDATION, 2017). Conforme definido anteriormente o *Blender* é *software* gratuito o qual disponibiliza diversas funcionalidades e ferramentas para o desenvolvimento de animações em modelos tridimensionais.

Para a criação do cenário principal do jogo que consiste em um escritório foi utilizada a ferramenta *open source Sweet Home 3D* (ETEKS, 2015). O *Sweet Home 3D* é uma ferramenta de design interior que permite criar o interior de ambientes com o auxilio de visualização 2D e 3D e disponibiliza modelos de mobílias, janelas, escadas, portas e paredes. A Figura 3.6 apresenta o cenário principal do jogo visualizado na ferramenta *Sweet Home 3D*.

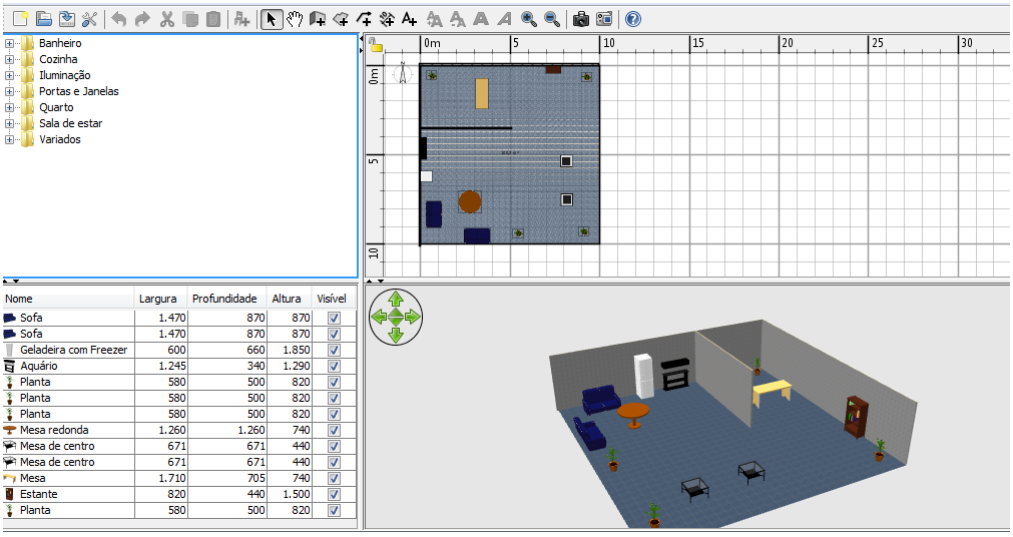

Figura 3.6: Modelo do cenário principal do jogo visualizado na ferramenta *Sweet Home 3D*.

Para a criação do cenário secundário do jogo composto por construções e ruas foi utilizado o recurso gratuito *Modular City Kit* (ARNDT, 2013). Na Figura 3.7 é possível visualizar os modelos utilizados para compor o cenário secundário do jogo.

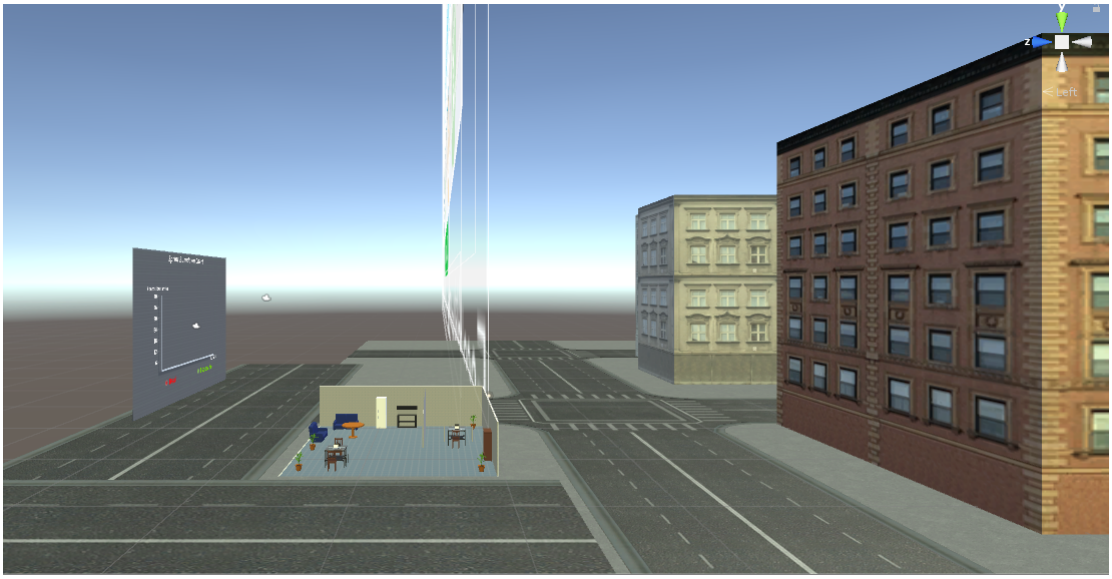

Figura 3.7: Modelos de prédios e ruas criados para compor o cenário secundário do jogo *SEJam* .

3.4.3 Interfaces de Usuário do *SEJam*

O projeto de interface de usuário do *SEJam* foi desenvolvido utilizando a linguagem de programação *C#* em conjunto ao ambiente gráfico de desenvolvimento do *Unity*.

Para exibir informações ao jogador e permitir a captura de informações do mesmo foram criadas interfaces de usuário. Diferente das interfaces de usuário apresentadas em *softwares* comuns ou *web sites* onde as interfaces compreendem todo o ambiente, as interfaces de usuário do jogo também conhecidas como *heads-up display* (HUD), apenas revestem a cena tridimensional do jogo, ou seja, está em uma camada sobreposta à camada responsável por exibir os elementos tridimensionais da cena do jogo. Para a criação das interfaces de usuário do jogo foi utilizado o objeto *Canvas* fornecido pelo próprio *Unity*.

O *Canvas* permite a organização de vários elementos como botões, caixas de texto e rótulos de forma orgânica e responsiva. O *Unity* também permite configurar elementos como textos, botões, imagens, caixas de texto entre outras opções disponíveis. Na Figura 3.8 é possível observar a estrutura de uma das interfaces criadas utilizando os componentes *HUD* do *Unity*.

Figura 3.8: Itens de interface de usuário agregados ao componente *Canvas* visualizado no editor do *Unity*.

3.4.4 Classes de Controle e de Gerenciamento do *SEJam*

Para o desenvolvimento do jogo foram criadas cinco classes responsáveis por persistir os dados e o estado de determinados objetos durante a execução do jogo, são elas *ScrumProject, UserStory, Employee, Question e Score* e sete classes de gerenciamento e de controle, responsáveis por conduzir o ciclo de operações do jogo de acordo com o modelo *Scrum* , além de controlar a exibição e atualização de interfaces de usuário. As classes definidas

são: ScrumManager, CanvasManager, Static,LineGraphManager, GameOverController, EmployeeAI e Touch Character. A Figura 3.9 apresenta a representação visual das classes criadas para o jogo *SEJam* .

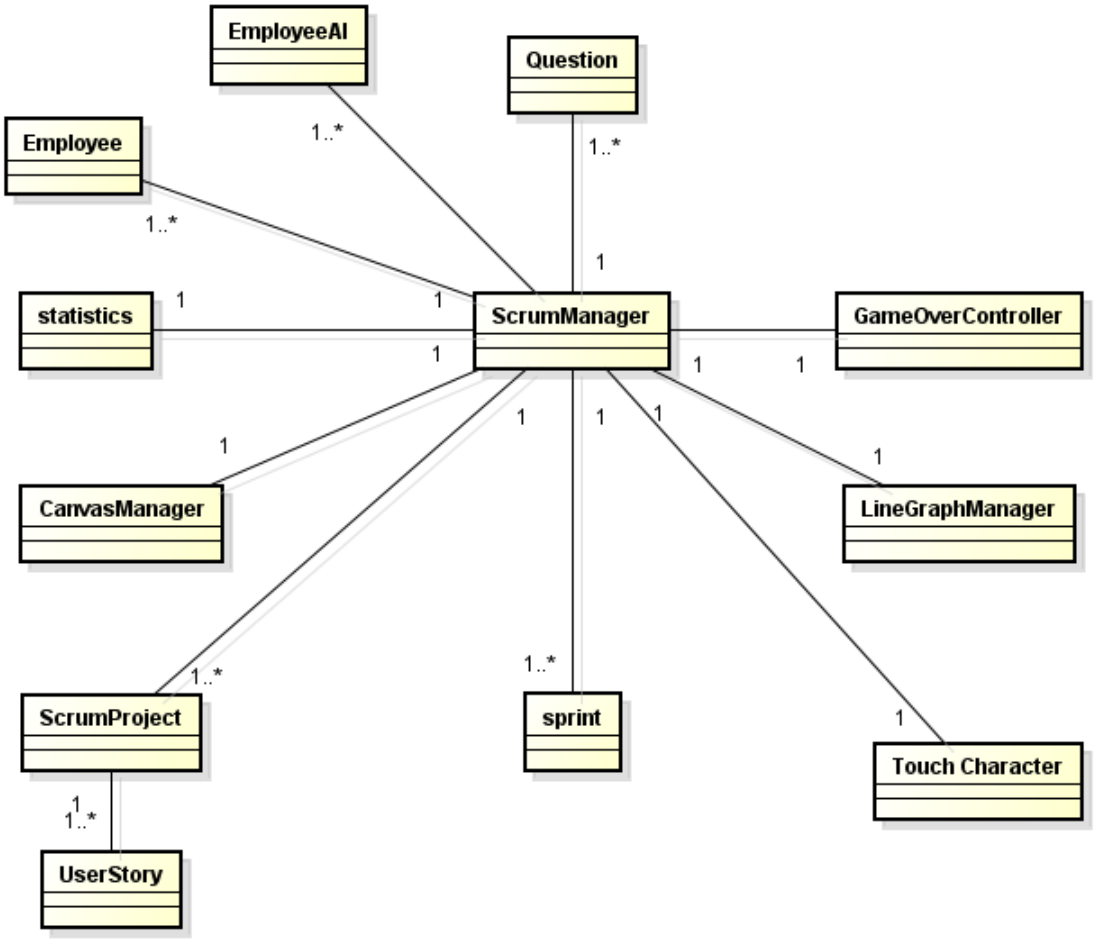

Figura 3.9: Diagrama de Classes do jogo *SEJam* .

A classe `ScrumProject` é responsável por armazenar os atributos acerca dos projetos disponíveis no jogo. Os principais atributos armazenados por esta classe são o identificador do projeto, a descrição e as histórias de usuário (`UserStory`).

A classe `UserStory` é uma classe responsável por armazenar os atributos acerca das histórias de usuário de um determinado projeto do jogo. Armazena informações como descrição, pontos (esforço) e a *Sprint* associada.

A classe `Employee` armazena as características e atributos de um determinado personagem do jogo. Entre os atributos definidos para o personagens destacam-se o nome, perícias (*design*, codificação, integração e testes), papel do *Scrum* (*Product Owner*,

Scrum Master ou *Team Member*) e produtividade.

A classe `Question` é responsável por armazenar os atributos das questões apresentadas ao jogador durante a simulação do evento de reunião diária do modelo *Scrum* do jogo.

A classe `Score` é responsável por calcular pontuação do jogador ao final do jogo.

A classe de gerenciamento do jogo `ScrumManager`, controla os principais eventos ao decorrer da simulação do modelo *Scrum* . Esta classe é responsável por controlar e notificar as classes responsáveis pelas ações do usuário e controle de personagens. A principal responsabilidade desta classe é conduzir a progressão do jogo ao decorrer das fases apresentadas.

A classe `CanvasManager` é responsável por controlar a exibição das interfaces de usuário do jogo.

A classe `Statistics` controla as estatísticas do jogo como produtividade dos personagens, pontos realizados no projeto e velocidade dos *Team Members* com os dados fornecidos pela classe *ScrumManager*.

A classe `LineGraphManager` permite desenhar linhas através de recursos gráficos do *Unity* de acordo com os parâmetros fornecidos pelo programador (CHAKRABORTY, 2015). A função desta classe no jogo é atualizar as informações relacionadas ao progresso dos *Team Members* no gráfico de *Burndown* presente em uma das interfaces do jogo.

A classe `GameOverController` é responsável por persistir os dados relacionados ao desempenho do jogador ao final de uma simulação da execução um projeto.

A classe `EmployeeAI` é responsável por controlar o estado do personagem durante a ocorrência de eventos disparados por alguma ação do usuário. Esta classe possui métodos responsáveis por controlarem as animações do personagem como andar, digitação e parado.

A classe `Touch Character` é responsável por controlar as entradas de usuário via *mouse* ou *touch screen* para o caso de dispositivos móveis.

3.5 Implementação

Na fase de implementação do modelo *ADDIE* para o *SEJam* foi definido como método de avaliação da ferramenta o método de inspeção de usabilidade Percurso Cognitivo. O percurso cognitivo foi escolhido de forma a avaliar a usabilidade da ferramenta. O Percurso Cognitivo é um método de avaliação de Interação Humano-Computador (IHC) baseado na engenharia cognitiva o qual através da exploração, avalia a facilidade de aprendizado de um sistema interativo (ROCHA, 2003). Nesta etapa será avaliada a facilidade de aprendizado do jogo, interatividade e a eficácia da ferramenta em cumprir os objetivos educacionais estabelecidos na fase de análise do modelo *ADDIE* para o *SEJam* .

O Percurso Cognitivo propõe a realização das atividades de preparação, coleta de dados e interpretação e consolidação dos resultados. Na fase de preparação são coletadas características sobre os potenciais usuários do sistema e a definição das tarefas que farão parte da avaliação. Na fase de coleta de dados, as interfaces selecionadas do sistema são percorridas de acordo com as ações necessárias para executar a tarefa proposta. Na fase de interpretação e consolidação dos resultados são apresentados os problemas encontrados na interface e a recomendação de correção dos problemas.

3.5.1 Preparação

A execução da avaliação do *SEJam* utilizando o percurso cognitivo foi realizada pelo próprio desenvolvedor do jogo percorrendo as interfaces da ferramenta com o objetivo de verificar a facilidade de aprendizado oferecida pelas interfaces do jogo e o rastreio de problemas presentes nas interfaces apresentadas. Conforme definido na fase de análise o público alvo do jogo é formado por alunos e profissionais da área de ciência da computação nível básico de conhecimento em engenharia de *software*.

Para a realização da avaliação foi selecionado o conjunto das principais interfaces de usuário e funcionalidades associadas às atividades propostas pelo jogo. Para cada tarefa proposta foi realizada a coleta de dados, reunindo informações das possíveis ações do usuário e as dificuldades encontradas. Por fim foi realizada a interpretação e consolidação dos resultados obtidos após a aplicação do método.

3.5.2 Coleta de Dados e Interpretação

Para a coleta de dados foram percorridos os principais cenários de interação do jogo *SEJam* , o percurso foi realizado de acordo com a sequência de ações necessárias para a realização de cada tarefa. Para cada interface analisada foi definida a descrição da tarefa associada, uma lista de ações necessárias para a execução correta da tarefa e a descrição de como o usuário faria a execução da tarefa e os possíveis problemas que seriam encontrados. O processo de coleta de dados do Percurso Cognitivo em detalhes pode ser visto no Apêndice I para o *SEJam* .

A realização do Percurso Cognitivo para a ferramenta *SEJam* permitiu verificar de forma analítica a facilidade de aprendizado oferecida pelo jogo e encontrar problemas de usabilidade nas interfaces presentes no jogo. O principal objetivo ao aplicar o método foi a detecção de problemas de usabilidade das interfaces. Durante a aplicação do método foi realizada em conjunto a verificação cumprimento dos objetivos educacionais acerca de conceitos presentes na metodologia *Scrum* .

Com o *feedback* obtido após a aplicação do método Percurso cognitivo é possível fazer correções e ajustes na ferramenta. Para cada problema de usabilidade apresentado nas interfaces do jogo foram catalogadas sugestões para a correção dos problemas. A seguir é apresentado o relato dos resultados consolidando os problemas encontrados nas interfaces do *SEJam* e recomendações de correção.

Elemento da Interface ou Atividade Proposta: Definição do *Product Backlog*, Definição do *Sprint Backlog*.

Problema Encontrado: Não está indicado de forma clara a seleção do personagem definido com o papel de *Product Owner* para acesso às opções *Product Backlog*, *Sprint Backlog*. Após da definição do *Product Backlog* e do *Sprint Backlog* não é exibida nenhuma interface de aviso que indica se a tarefa foi bem sucedida ou não.

Recomendação de Correção: Fornecer uma instrução ou indicação de que o usuário deva selecionar o personagem definido como *Product Owner* para acesso às opções *Product Backlog*, *Sprint Backlog*. Após a definição do *Product Backlog* e do *Sprint Backlog*, fornecer uma indicação de que a tarefa foi bem sucedida ou não.

4 Funcionamento do Jogo digital *SEJam*

Este capítulo aborda o funcionamento e o desenvolvimento de enredo do jogo *SEJam* . Serão exploradas as fases e as rotinas presentes no jogo assim como os resultados obtidos pelo jogador ao final do jogo. O *SEJam* fornece um modelo para a simulação do ciclo de desenvolvimento de um projeto de acordo com a metodologia *Scrum* . A Figura 4.1 mostra a tela inicial do jogo, onde o jogador deve selecionar o botão *novo jogo* para iniciar uma nova simulação do modelo *Scrum* .

Figura 4.1: Tela inicial do jogo para início da simulação do modelo *Scrum* .

A Figura 4.2 apresenta a tela de seleção de um projeto proposto para a simulação. O projeto descreve a resolução de um problema por meio de um dispositivo computacional. A simulação do desenvolvimento do projeto será efetuada pelo jogador com o auxílio do modelo *Scrum* presente no jogo. Após a escolha do projeto é apresentada ao jogador a interface de escolha de personagens conforme mostrado na Figura 4.3. Na interface de escolha de personagens é solicitado ao jogador escolher três personagens os quais são representados graficamente com nome e perícias.

Figura 4.2: Tela de seleção de projeto para simulação do desenvolvimento de acordo com o modelo *Scrum* .

Figura 4.3: Tela para seleção de três personagens para a simulação do modelo *Scrum* .

4.1 Fluxo de Operações do Jogo

As atividades presentes no jogo foram definidas de acordo com Salen e Zimmerman (2004), sendo classificados em atividades operacionais, construtivas e implícitas.

As atividades operacionais definem as operações centrais permitidas ao jogador realizar com os elementos do jogo, como por exemplo, a exibição de um menu ao selecionar um dos personagens definido como *Product Owner*.

As atividades construtivas definem como é realizado as operações aritméticas efetuadas pelo jogo, como por exemplo, o cálculo da produtividade de cada personagem durante a simulação de dia de trabalho.

As atividades implícitas contidas no jogo ocorrem durante a simulação de um dia de desenvolvimento do projeto, no caso de o jogador escolher corretamente a opção de alocar um personagem para uma atividade correspondente à sua perícia. Ao escolher a opção correta o jogo incrementa a satisfação do cliente e mantém constante o percentual da produtividade do personagem.

4.2 Definição de Papéis do *Scrum* e Cenário do Jogo

Após o jogador realizar a escolha de três personagens para a simulação do modelo *Scrum* é solicitada a definição do papel de *Product Owner* para um dos personagens escolhidos conforme pode ser visto na Figura 4.4. Em seguida é solicitado ao jogador definir um personagem para atuar como *Scrum Master* do projeto durante a simulação do modelo *Scrum* conforme pode ser observado na Figura 4.5. O terceiro personagem restante é automaticamente definido pelo jogo com *Team Member*.

Figura 4.4: Interface para escolha de personagem para atuar no papel de *Product Owner*.

Figura 4.5: Interface para definição de um personagem para atuar no papel de *Scrum Master*.

O principal cenário do jogo é a representação tridimensional de um escritório localizado no centro comercial de uma cidade fictícia, representando o ambiente de uma empresa de desenvolvimento de *softwares*.

O escritório é composto por três ambientes: sala de reuniões, local onde é realizada a simulação do evento *Reunião Diária*, sala do gerente de projetos, ambiente que o personagem definido como *Scrum Master* realiza atividades diárias e a sala de expediente onde os personagens definidos como *Product Owner* e *Team Member* são alocados durante a simulação de dia de desenvolvimento na *Sprint* , conforme pode ser visto na Figura 4.6. O cenário do jogo apresenta ainda elementos tridimensionais estáticos como construções.

Figura 4.6: Cenário principal do jogo contendo a representação de um escritório e de uma cidade fictícia.

4.2.1 Simulação do Planejamento da Sprint e Criação do *Product Backlog* e *Sprint Backlog*

Para seguir o fluxo natural de execução de operações do jogo, o jogador deve selecionar o personagem definido como *Product Owner*. Ao selecionar o personagem será exibida uma interface contendo as opções *Product Backlog* e *Sprint Backlog* conforme pode ser visto na Figura 4.7. A opção *Product Backlog* (Figura 4.8), quando selecionada exibe ao jogador uma interface contendo itens representando histórias de usuário as quais representam todos os requisitos do projeto escolhido para a simulação do modelo *Scrum* .

Ao selecionar um item na interface é possível visualizar a descrição da história de usuário e os pontos pertencentes à história de usuário selecionada. A opção *Sprint Backlog* (Figura 4.9) quando selecionada exibe ao jogador uma interface com as histórias de usuário selecionadas na interface de *Product Backlog*. A opção *Sprint Backlog* apenas é ativada após o jogador ter selecionado histórias de usuário na interface *Product Backlog*.

Figura 4.7: Interface responsável por mostrar o menu do *Product Owner*.

Figura 4.8: Interface responsável por mostrar o *Product Backlog*.

Figura 4.9: Interface responsável por mostrar o *Sprint Backlog*.

4.3 Simulação da *Sprint* e do Evento *Reunião Diária*

A simulação de uma *Sprint* do projeto somente é permitida caso o jogador tenha terminado de atribuir *sprints* para as histórias de usuário na interface *Sprint Backlog* (Figura 4.9) e tenha alocado personagens para realizar as atividades presentes em cada história de usuário da *Sprint* corrente, por meio da interface *Taskboard*.

Caso o botão `Simular Dia` localizado na parte inferior central da tela de jogo esteja ativado, quando selecionado, é exibida a animação correspondente à simulação de um dia de trabalho dos personagens da Figura 4.10. Ao final da animação é exibido ao jogador a interface responsável por representar o evento *Reunião diária* como na Figura 4.11. Nesta interface é possível gerenciar os personagens.

Figura 4.10: Animação dos personagens do jogo durante simulação de trabalho.

Na interface apresentada na Figura 4.11 pode ser observado um botão denominado `Gerenciar` associado a cada personagem. Ao selecionar este botão o jogador pode verificar o impedimento que está sendo enfrentado pelo personagem a ser gerenciado, conforme pode ser observado na Figura 4.12. Na parte superior esquerda do cenário do jogo é possível visualizar o nível atual de satisfação do cliente conforme pode ser observado na Figura 4.11. O nível de satisfação do cliente é definido de acordo com o somatório do desempenho de cada personagem na simulação de dia da *Sprint* , variando de 0% a 100%.

Figura 4.11: Interface representando o evento *Reunião Diária.*

Figura 4.12: Interface contendo um impedimento do personagem.

4.4 Estatísticas do Projeto e *Burndown Chart*

O projeto proposto para simulação de desenvolvimento possui uma quantidade determinada de pontos atribuídos como custo das histórias de usuário. O objetivo do jogador durante a simulação do modelo *Scrum* é concluir o projeto com pelo menos 50% destes pontos realizados.

A produtividade de cada personagem ao final da simulação de dia de desenvolvimento é calculada através da capacidade do personagem em executar as tarefas presentes nas histórias de usuário da *Sprint* .

Ao selecionar o botão estatísticas localizado na parte inferior da tela, o jogador tem a opção de visualizar as estatísticas do projeto através de uma interface mostrada na Figura 4.13. Outra opção oferecida ao jogador para verificar o progresso da *Sprint* é através de um gráfico representando o *Sprint Burndown Chart* do projeto (Figura 4.14). A linha verde presente no gráfico representa a meta ideal dos pontos a serem executados pela quantidade de dias da *Sprint* corrente do projeto. A linha vermelha no gráfico indica quantos pontos realmente foram executados pelos *Team Members* durante os dias da *Sprint* corrente. A opção para visualizar o gráfico pode ser obtida selecionando o botão *Burndown Chart* localizado na parte inferior da tela.

Figura 4.13: Interface responsável por mostrar as estatísticas do projeto.

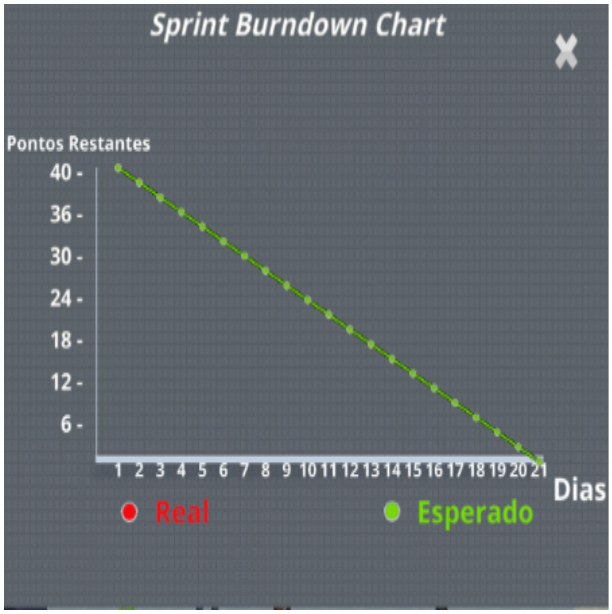

Figura 4.14: Gráfico representando o *Sprint Burndown Chart*.

4.5 *Taskboard*

É oferecido ao jogador a opção de visualizar a representação do artefato *Taskboard* (Figura 4.15 e Figura 4.16) por meio do botão *Taskboard* localizado na parte inferior da tela. Ao selecionar o *Taskboard* o jogador tem a opção de alocar personagens para cada tarefa contida nas histórias de usuário apresentadas. O jogador pode atribuir os personagens a

cada uma das tarefas de: *design*, codificação, integração e testes. Ao decorrer da simulação da *Sprint* estas tarefas serão executadas sequencialmente para cada história de usuário. É permitido alocar mais de um personagem a uma mesma tarefa e alocar um mesmo personagem para mais de uma tarefa. No *taskboard* é possível verificar em qual estado de desenvolvimento encontra-se cada história de usuário do projeto. Os possíveis estados para as tarefas presentes nas histórias de usuário são os estados: a fazer, em andamento e pronto.

Figura 4.15: Interface representando o *Taskboard* do projeto.

Figura 4.16: Interface representando o *Taskboard* do projeto.

4.6 *Simulação dos eventos Revisão e Retrospectiva da Sprint*

Ao final da simulação de uma *Sprint* do projeto é apresentada a interface responsável por representar os eventos *Revisão da Sprint* e *Retrospectiva da Sprint*. Nessa interface o jogador tem a possibilidade de avaliar os resultados obtidos ao final da *Sprint* atual. A interface *Revisão da Sprint* pode ser observada na Figura 4.17.

Figura 4.17: Interface representando o evento *Retrospectiva da Sprint*.

4.7 Término da Simulação e Avaliação de Resultados

Após a realização de todas as *sprints* do projeto é exibido ao jogador os resultados obtidos durante a simulação. Na interface apresentada (Figura 4.18), é possível visualizar a percentagem concluída do projeto e o nível de satisfação do cliente ao final do projeto. O jogador obtém uma simulação bem sucedida, ou seja, ganha o jogo caso conclua ao menos 50% dos pontos de história de usuário do projeto, caso contrário é considerado perdedor. Ao final da simulação do desenvolvimento do projeto o desempenho do jogador é salvo para avaliações futuras.

Figura 4.18: Interface apresentando os resultados obtidos ao final de uma simulação do desenvolvimento de um projeto.

5 Conclusão

5.1 Considerações Finais

Este trabalho apresentou o desenvolvimento do protótipo de um jogo digital com o propósito de auxiliar alunos de Ciência da Computação e áreas correlatas, no aprendizado de práticas da metodologia ágil *Scrum* .

Para atender o objetivo geral proposto neste trabalho foi realizado o estudo de modelos clássicos e ágeis acerca de processo de *software* e o estudo acerca do modelo de *Design* Instrucional *ADDIE*.

Foi realizada uma pesquisa na literatura para obter o estado da arte sobre jogos digitais educacionais voltados ao ensino de modelos de processo de *software* constatando-se uma carência de jogos abordando esta temática. Após a revisão teórica foi iniciado o planejamento do jogo educacional com a aplicação de diretrizes propostas pelo modelo *ADDIE* descritas na seção 2.3 e o desenvolvimento do jogo descrito na seção 3.4.

O desenvolvimento do jogo *SEJam* foi realizado utilizando o *Unity 3D* e o protótipo foi capaz de simular os papéis, artefatos e ações da metodologia. Um estudo de interface com usuário foi conduzido para resolver problemas de uso. O mesmo encontra-se no Apêndice I.

Apesar da avaliação limitada a poucos usuários, o protótipo apresentou um resultado satisfatório como uma ferramenta inicial para o complemento do ensino da metodologia ágil *Scrum* quando comparada com as outras ferramentas pesquisadas. O jogo pode ser utilizado como objeto de aprendizagem por alunos de disciplinas de gerência de projetos ou de engenharia de *software*. A principal contribuição do jogo para o aprendizado do usuário é oferecer um simulador para reforçar os termos da metodologia *Scrum* . A implementação é acessível nas plataformas *Android*, *Windows* e *Html5* permitindo aprender a metodologia de uma forma interativa e lúdica.

5.2 Limitações e Trabalhos Futuros

Durante o desenvolvimento do trabalho foi possível atingir parte dos objetivos parciais mas o objetivo geral ainda carece de trabalhos subsequentes para ser conquistado completamente.

A avaliação realizada, bem como observação do uso foi feita com um número pequeno de pessoas. Acredita-se que seu uso por alunos de disciplinas relacionadas à engenharia de *software* e gerência de projetos irá gerar dados mais consistentes das reais limitações de interface bem como quais são de difícil entendimento.

O modelo de processo utilizado está simplificado, utilizando equações algébricas com valores os quais são alterados durante as iterações das *Sprints* . Modelos de processos com dinâmicas mais complexas podem ser adaptados para capturar melhor efeitos observados em projetos.

Adicionalmente, o modelo não captura o uso de recursos financeiros, a variável de interesse inicial é apenas no tempo gasto pela equipe. Acrescentar restrições de custo recursos como energia elétrica, maquinário, treinamentos, custos trabalhistas e outros podem acrescentar uma camada importante para gestão de riscos.

A interface gráfica e interação com o usuário exige um conhecimento prévio do funcionamento do sistema e ainda está pouco intuitiva. Um sistema de tutorial pode ser implementado para acompanhar um novo usuário nas primeiras partidas.

Como o protótipo pode ser implantado em diversos ambientes diferentes, a coleta de dados remotos de uso poderia ajudar a realizar uma telemetria do progresso dos alunos para também atuar como ferramenta de avaliação dos usuários dentro de uma disciplina ou implementar desafios para que os alunos compitam entre si para otimizar uma pontuação final.

Bibliografia

AMBROSIO, F. K. Se• rpg 2.0: Uma nova versão do software engineering-roleplaying game. *Universidade Regional de Blumenau. Blumenau*, 2008.

APS., U. T. *Unity 3D*. 2017. Disponível em: ¡⟨http://www.unity3d.com⟩¿. Acesso em: 10 ago. de 2016.

ARNDT, J. *Modular City Kit Free*. 2013. Disponível em: ¡⟨https://www.youtube.com/watch?v=IiXrnBp17K0⟩¿. Acesso em: 20 ago. de 2018.

BECK, K. et al. Manifesto for agile software development. 2001.

CAMARGO, A. S. Jogo de rpg para ensinar scrum. *Universidade Federal de Santa Catarina. Florianópolis*, 2013.

CHAKRABORTY, S. *How to create line graph in unity*. 2015. Disponível em: ¡⟨bit.ly/1VKypjl]⟩¿. Acesso em: 10 ago. de 2016.

COHN, M. *Succeeding with agile: software development using Scrum*. [S.l.]: Pearson Education, 2010.

ETEKS. *Sweet Home 3D*. 2015. Disponível em: ¡⟨http://www.sweethome3d.com⟩¿. Acesso em: 10 ago. de 2016.

FALKEMBACH, G. A. M.; GELLER, M.; SILVEIRA, S. R. Desenvolvimento de jogos educativos digitais utilizando a ferramenta de autoria multimídia: um estudo de caso com o toolbook instructor. *RENOTE*, v. 4, n. 1, 2006.

FOUNDATION, B. *Blender*. 2017. Disponível em: ¡⟨https://www.blender.org/download/⟩¿. Acesso em: 10 ago. de 2016.

FOWLER, M. Refactoring: Improving the design of existing code. *Pearson Addison-Wesley*, 2000.

GKRITSI, A. Scrum game: Ann agile software management game. *Master Thesis in Software Engineering, University of Southampton, UK*, 2011.

GLOGER, B. *The Zen of Scrum*. 2007. Disponível em: ¡⟨http://www.glogerconsulting.de/⟩¿. Acesso em: 20 jul. de 2016.

GROUP, S. Chaos manifesto 2013: Think big, act small. *The Standish Group International Inc*, v. 176, 2013.

HIGHSMITH, J. Agile project management: Principles and tools. *Cutter consortium*, v. 4, p. 1–37, 2003.

HOCKING, J. et al. *Unity in action*. [S.l.]: Manning Publications, 2015.

HUMAN, M. *Make Human*. 2017. Disponível em: ¡⟨http://www.makehuman.org⟩¿. Acesso em: 10 ago. de 2016.

INC., T. *Sketchup Make*. 2017. Disponível em: ¡⟨http://www.sketchup.com⟩¿. Acesso em: 10 ago. de 2016.

KUMAR, G.; BHATIA, P. K. Impact of agile methodology on software development process. 2012.

MOLENDA, M. In search of the elusive addie model. *Performance improvement*, v. 42, n. 5, p. 34–37, 2003.

NAVARRO, E. *SimSE: A Software Engineering Simulation Environment for Software Process Education DISSERTATION*. Tese (Doutorado) — University of California, Irvine, 2006.

NAVARRO, G. Gamificação: a transformação do conceito do termo jogo no contexto da pós modernidade. *Biblioteca Latino-Americana de Cultura e Comunicação*, v. 1, n. 1, p. 1–26, 2013.

NEEDLEMAN, S. E. *Mobile-Games Revenue Growth Is Outpacing Other Content, for Now*. 2015. Disponível em: ¡⟨http://blogs.wsj.com/digits/2015/02/18/mobile-games-revenue-growth-is-outpacing-other-content-for-now⟩¿. Acesso em: 10 jul. de 2016.

NETO, E. I. Ferramenta educacional para ensino de práticas do scrum. *Porto Alegre*, v. 25, 2008.

NOLASCO, A. Z. Análise e comparativo de ferramentas para simulação de projeto de software no treinamento de gerentes. *Centro de Ensino Superior de Juiz de Fora. Juiz de Fora*, 2013.

OLIVEIRA, J. M. d.; CSIK, M.; MARQUES, P. Módulo 1-o modelo dsi. Escola Nacional de Administração Pública (Enap), 2015.

PAQUETTE, G. Modelling and delivering distributed learning environments. In: *TelE-Learning*. [S.l.]: Springer, 2002. p. 251–258.

PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7ª edição. *Ed: McGraw Hill*, 2011.

ROCHA, H. V. Design e avaliação de interfaces humano-computador. *Unicamp, Campinas*, 2003.

RUBIN, K. S. Essential scrum: a practical guide to the most popular agile process. *Ann Arbor: Pearson Education*, 2012.

SALEN, K.; ZIMMERMAN, E. *Rules of play: Game design fundamentals*. [S.l.]: MIT press, 2004.

SCHNEIDER, M. F. Scrum'ed: um jogo de rpg para ensinar scrum. *Universidade Federal de Santa Catarina. Florianópolis*, 2015.

SCHWABER, K.; SUTHERLAND, J. Guia do scrum. *Línea. Consultado el*, v. 12, 2015.

SOMMERVILLE, I. Engenharia de software, 8ª edição, tradução: Selma shin shimizu mel-nikoff, reginaldo arakaki, edilson de andrade barbosa. *São Paulo: Pearson Addison-Wesley*, v. 22, p. 103, 2007.

STRICKLAND, A. Addie model. *Retrieved February*, v. 5, p. 2005, 2000.

TAROUCO, L. M. R.; ROLAND, L. C. *Jogos educacionais*. 2004.

TECHNOLOGY, U. *Unreal Engine*. 2017. Disponível em: ¡⟨http://www.unrealtechnology.com/html/technology/ue2.shtml⟩¿. Acesso em: 10 ago. de 2016.

WELLS, D. *XP - Unit tests*. 1999. Disponível em: ¡⟨http://www.extremeprogramming.org/rules/unittests.html⟩¿. Acesso em: 20 jul. de 2016.

Apêndice I: Percurso Cognitivo para o *SEJam*

Tarefa: Selecionar a opção para iniciar o jogo.

Protótipo de Interface: Figura 4.1

Sequência correta de ações: Selecionar o botão novo jogo presente na parte central da interface.

O usuário tentará atingir a meta correta?

Resposta: Sim, é apresentado de forma clara um botão na interface o qual indica o inicio do jogo.

Uma vez encontrado o elemento de interface, usuários o reconhecerão que ele produzirá o efeito desejado?

Resposta: Sim, existe um botão identificado com o *label* novo jogo o qual permite o usuário avançar para a próxima etapa do jogo.

Após a ação correta ser executada, o usuário perceberá que progrediu em direção à solução da tarefa?

Resposta: Sim, a mudança de interface apresentada após a seleção do botão, indica ideia de continuidade na operação realizada pelo usuário.

Tarefa: Escolha de um projeto para Simulação do modelo *Scrum* .

Protótipo de Interface: Figura 4.2

Sequência correta de ações: Visualizar as informações disponíveis nos elementos de interface representando os projetos, selecionar o elemento de interface representativo do projeto desejado para a Simulação.

O usuário tentará atingir a meta correta?

Resposta: Sim, são apresentados elementos de interface o qual cada elemento representa um projeto disponível no jogo. Cada elemento possui informações relacionadas ao projeto associado.

O usuário irá perceber que a ação correta está disponível na interface?

Resposta: Sim, após selecionar um elemento representando um projeto o usuário é conduzido para a próxima interface.

Uma vez encontrado o elemento de interface, usuários o reconhecerão que ele produzirá o efeito desejado?

Resposta: Sim, cada elemento possui uma identificação indicando o projeto associado e depois de selecionado o usuário é conduzido a uma nova interface.

Após a ação correta ser executada, o usuário perceberá que progrediu em direção à solução da tarefa?

Resposta: Sim, o usuário é conduzido a uma nova interface após a seleção do elemento de interface representando um projeto.

Tarefa: Escolha de Personagens.

Protótipo de Interface: Figura 4.3

Sequência correta de ações: Escolher um personagem selecionando o elemento de interface correspondente ao personagem desejado, repetir o passo anterior até completar o total de três personagens, em seguida o jogador é redirecionado para a próxima interface.

O usuário tentará atingir a meta correta?

Resposta: Sim, são apresentados elementos de interface representando os personagens disponíveis.

O usuário irá perceber que a ação correta está disponível na interface?

Resposta: Sim, quando o jogador seleciona um personagem um ícone informativo associado ao personagem escolhido é exibido.

Uma vez encontrado o elemento de interface, usuários o reconhecerão que ele produzirá o efeito desejado?

Resposta: Sim, cada elemento representando um personagem possui uma foto indicando o personagem associado.

Após a ação correta ser executada, o usuário perceberá que progrediu em direção à solução da tarefa?

Resposta: Sim, após selecionar três personagens o usuário é conduzido a uma nova interface.

Tarefa: Escolha do *Scrum Master* e do *Product Owner*.

Protótipo de Interface: Figura 4.4 e Figura 4.5

Sequência correta de ações: selecionar na interface apresentada um dos três personagens escolhidos na etapa anterior para ocupar o papel de *Product Owner*, após a escolha do *Product Owner* na nova interface apresentada escolher um dos personagens apresentados para ocupar o papel de *Scrum Master*.

O usuário tentará atingir a meta correta?

Resposta: Sim, são apresentados elementos representando os personagens disponíveis para o papel de *Product Owner* e *Scrum Master*.

O usuário irá perceber que a ação correta está disponível na interface?

Resposta: Sim, após a escolha dos personagens o jogador é conduzido ao próximo passo do processo.

Uma vez encontrado o elemento de interface, usuários o reconhecerão que ele produzirá o efeito desejado?

Resposta: Sim, cada elemento representando um personagem possui uma foto indicando o personagem associado.

Após a ação correta ser executada, o usuário perceberá que progrediu em direção à solução da tarefa?

Resposta: Sim, após a escolha do personagem existe uma mudança de estado nas interfaces apresentadas indicando que a ação foi bem sucedida.

Tarefa: Definição do *Product Backlog*.

Protótipo de Interface: Figuras 4.6, 4.7, 4.8

Sequência correta de ações: Selecionar o personagem definido com a função de *Product Owner*, na interface apresentada a seguir, selecionar a opção identificada como *Product Backlog*, na próxima interface apresentada, selecionar o campo *checkbox* dos elementos representando as histórias de usuário desejadas. Ao finalizar o processo, selecionar o botão identificado com o símbolo de fechar para sair da interface.

O usuário tentará atingir a meta correta?

Resposta: Sim, após visualizar a identificação em forma de texto sobre o personagem indicando a função como *Product Owner*, o usuário irá selecionar o personagem, conduzindo a exibição de uma nova interface contendo opções relacionadas ao papel *Product Owner*, incluindo a opção definida como *Product Backlog*. Após a seleção da opção *Product Backlog* no menu do personagem é exibido a interface representativa do *Product Backlog* contendo histórias de usuário as quais apresentam um campo *chekbox* no qual o usuário deve modifica-lo para selecionar o item. Não, o usuário não consegue deduzir que deva selecionar o personagem definido como *Product Owner* para encontrar a opção de definição do *Product Backlog*.

O usuário irá perceber que a ação correta está disponível na interface?

Resposta: Não, a partir da seleção do personagem e da seleção da opção *Product Backlog* presente no menu de opções do personagem, não existe na interface representativa

do *Product Backlog* um botão identificado a conclusão do processo de definição do *Product Backlog*.

Uma vez encontrado o elemento de interface, usuários o reconhecerão que ele produzirá o efeito desejado?

Resposta: Sim, a partir da seleção do personagem definido como *Product Owner* os elementos estão identificados com *labels* indicando a suas funções. Não, o usuário visualiza o personagem definido com *Product Owner*, mas não é capaz de deduzir que deva selecioná-lo para a definição do *Product Backlog*.

Após a ação correta ser executada, o usuário perceberá que progrediu em direção à solução da tarefa?

Resposta: Não, não foi verificada uma interface de aviso que a ação foi bem sucedida.

Tarefa: Seleção do personagem definido como *Product Owner* para definição do *Sprint Backlog*.

Sequência correta de ações: Selecionar o personagem definido com a função de *Product Owner*, na interface seguinte apresentada selecionar a opção identificada como *Sprint Backlog*, na interface seguinte apresentada para cada elemento representando uma história de usuário, selecionar o elemento *dropdown* correspondente ao item e atribuir uma *Sprint* disponível para a história de usuário. Ao finalizar o processo, selecionar o botão identificado com o símbolo de fechar para sair da interface.

Protótipo de Interface: Figuras 4.6, 4.7, 4.9

O usuário tentará atingir a meta correta?

Resposta: Sim, após visualizar a identificação em forma de texto sobre o personagem indicando a função como *Product Owner*, o usuário irá selecionar o personagem , conduzindo a exibição de uma nova interface contendo opções relacionadas ao papel *Product Owner* incluindo a opção definida como *Sprint Backlog*. Após a seleção da opção *Sprint Backlog* no menu do personagem é exibido a interface representativa do *Sprint Backlog* contendo histórias de usuário com o campo *Sprint* vazio o qual o usuário deve modifica-lo atribuindo uma *Sprint* disponível. Não, o usuário não consegue deduzir que deva selecionar o personagem definido como *Product Owner* para encontrar a opção de definição do *Sprint Backlog*.

O usuário irá perceber que a ação correta está disponível na interface?

Resposta: Sim, a partir da seleção do personagem definido como *Product Owner* os elementos estão identificados com *labels* indicando a suas funções. Não, o usuário visualiza o personagem definido com *Product Owner*, mas não é capaz de deduzir que deva selecioná-lo para a definição do *Product Backlog*.

Uma vez encontrado o elemento de interface, usuários o reconhecerão que ele produzirá o efeito desejado?

Resposta: Sim, os elementos estão dispostos na interface de forma plausível.

Após a ação correta ser executada, o usuário perceberá que progrediu em direção à solução da tarefa?

Resposta: Não, não foi verificada uma interface de aviso que a ação foi bem sucedida. Após executar suas ações o usuário apenas é conduzido a uma nova interface ou o jogo retorna para um estado anterior.

Tarefa: Iniciar Simulação de um dia de desenvolvimento na *Sprint* .

Sequência correta de ações: Selecionar o botão Simular dia localizado na parte central do cenário principal do jogo, visualizar o progresso da Simulação do dia por meio da barra de progresso apresentada na parte superior do cenário, na interface seguinte representando o evento Reunião Diária selecionar o botão `Gerenciar` definido a cada personagem participante da Simulação, na interface a seguir selecionar uma das opções

disponíveis para resolução do impedimento do personagem na Simulação, após escolha da opção de resolução do impedimento do personagem selecionar o botão com o símbolo de *play* para concluir a tarefa.

Protótipo de Interface: Figuras 4.6, 4.10, 4.11 4.12 O usuário tentará atingir a meta correta?

Resposta: Sim, após seleção de o botão Simular dia, é apresentado ao usuário uma sequencia de interfaces as quais conduzem as tarefas contidas na atividade de Simulação de dia de desenvolvimento da *Sprint* .

O usuário irá perceber que a ação correta está disponível na interface?

Resposta: Sim, na parte central inferior da tela existe um botão correspondente para inicio da atividade de Simulação de dia de desenvolvimento. A seguir o usuário é conduzido a realizar as demais atividades da tarefa através de interfaces e caixas de diálogo.

Uma vez encontrado o elemento de interface, usuários o reconhecerão que ele produzirá o efeito desejado?

Resposta: Sim, o botão correspondente a Simulação de dia de desenvolvimento está bem identificado na interface e a interface e os elementos representando o evento Reunião Diária estão dispostos de forma organizada.

Após a ação correta ser executada, o usuário perceberá que progrediu em direção à solução da tarefa?

Resposta: Sim a cada ação do usuário ou evento do jogo durante a tarefa, o usuário é notificado caso tenha progredido de forma correta ou não.

Tarefa: Visualização do *Taskboard*.

Sequência correta de ações: Selecionar o botão *Taskboard* localizado na parte inferior da tela, na interface seguinte visualizar o *status* das atividades presentes em cada historia de usuário definida para a simulação da *Sprint* corrente.

Protótipo de Interface: Figuras 4.6 e 4.16

O usuário tentará atingir a meta correta?

Resposta: Sim, existe um elemento *tooltip* identificando a função do botão *Taskboard*. Após a seleção do botão *Taskboard* é exibido a representação do *Taskboard*.

O usuário irá perceber que a ação correta está disponível na interface?

Resposta: Sim, a partir da seleção do botão *Taskboard*, a sequencia de passos mostra-se intuitiva com as opções disponíveis na tela.

Uma vez encontrado o elemento de interface, usuários o reconhecerão que ele produzirá o efeito desejado?

Resposta: Sim, após a seleção do botão *Taskboard* é exibido um elemento *tooltip* informando a função do botão selecionado.

Após a ação correta ser executada, o usuário perceberá que progrediu em direção à solução da tarefa?

Resposta: Sim, após a seleção de o botão *Taskboard* a interface correspondente ao *Taskboard* da *Sprint* é apresentado.

Tarefa: Visualizar *Burndown chart*.

Sequência correta de ações: Selecionar o botão *Burndown Chart* localizado na parte inferior da tela, visualizar o *Burndown Chart* em uma nova interface.

Protótipo de Interface: Figura 4.14

O usuário tentará atingir a meta correta?

Resposta: Sim, existe um elemento *tooltip* identificando a função do botão *Burndown Chart*. Após a seleção do botão *Burndown Chart* é exibido a representação do *Burndown Chart*.

O usuário irá perceber que a ação correta está disponível na interface?

Resposta: Sim, a partir da seleção do botão *Burndown Chart*, a sequencia de passos mostra-se intuitiva com as opções disponíveis na tela.

Uma vez encontrado o elemento de interface, usuários o reconhecerão que ele produzirá o efeito desejado?

Resposta:Sim, após a seleção do botão *Burndown Chart* é exibido um elemento *tooltip* informando a função do botão selecionado.

Após a ação correta ser executada, o usuário perceberá que progrediu em direção à solução da tarefa?

Resposta: Sim, após a selecionar o botão *Burndown chart*, uma nova interface representando o *Burndown Chart* é apresentada.