UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO

TROY COSTA KOHWALTER

SOFTWARE DEVELOPMENT MANAGER – JOGO DE ENSINO DE ENGENHARIA DE SOFTWARE

NITERÓI
2011

TROY COSTA KOHWALTER

SOFTWARE DEVELOPMENT MANAGER – JOGO DE ENSINO DE ENGENHARIA DE SOFTWARE

Trabalho de Conclusão de Curso Apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. LEONARDO G. PAULINO MURTA
Orientador: Prof. ESTEBAN W. GONZALEZ CLUA

NITERÓI
2011

TROY COSTA KOHWALTER

SOFTWARE DEVELOPMENT MANAGER – JOGO DE ENSINO DE ENGENHARIA DE SOFTWARE

Trabalho de Conclusão de Curso Apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense para obtenção do grau de Bacharel em Ciência da Computação.

Data da Aprovação: _____

BANCA EXAMINADORA

_____

Prof. Dr. LEONARDO G. PAULINO MURTA - Orientador
UFF

_____

Prof. Dr. ESTEBAN W. GONZALEZ CLUA - Orientador
UFF

_____

Prof. Dra. ADRIANA PEREIRA DE MEDEIROS
PURO - UFF

_____

Prof. Dr. LEANDRO A. FRATA FERNANDES
UFF

NITERÓI 2011

Dedico este trabalho aos docentes da Universidade Federal Fluminense, em especial aos meus orientadores Leonardo Murta e Esteban Clua. Que lhes possa ser útil todo o conteúdo oferecido por fruto deste trabalho.

AGRADECIMENTOS

Aos meus pais, Vania e Hans, meu irmão, Bruno e minha sombra, Kira.

Aos meus tios Fernando, Silvana, Sônia, Heitor e meus avós Marly, Aroldo e Evelyn.

Aos professores Leonardo Murta e Esteban Clua, por todo conhecimento fornecido, conselhos e disposição não só durante este trabalho, mas no decorrer da minha vida acadêmica.

Aos professores do Instituto de Computação, Física e Matemática, em especial, Alexandre Plastino, Aline Nascimento, Anna Dolejsi, Anselmo Montenegro, José Raphael, Carlos Martinhon e Valéria Zuma, pelos ensinamentos e convívio.

Aos amigos por todo apoio, conselhos, ajuda e compreensão.

Por fim, gostaria de agradecer a Universidade Federal Fluminense, por me proporcionar tanto aprendizado e experiência.

“Nós não somos, não podemos ser, sobre a concepção de conteúdo. A perspectiva fundamental que eu quero tirar de você é de nós estarmos projetando experiências. Se nada outra coisa, começar a pensar não sobre a criação de conteúdo, mas sobre a criação de ambientes de aprendizagem e de experiências. Tornou-se claro para mim que este é um ponto fundamental. Você tem que começar a pensar em colocar os alunos em um contexto onde eles têm que tomar decisões, entender porque essas decisões são importantes, querer fazer essas decisões, e saber que há consequências dessas decisões." Quinn (2005).

RESUMO

A Engenharia de Software é uma área de conhecimento da computação que se concentra nos aspectos práticos da produção de um software. Cursos de graduação em Ciência da Computação possuem cadeiras de Engenharia de Software, porém o tema é apresentado de forma teórica e poucos trabalhos de implementação utilizando as técnicas e ferramentas aprendidas. A abordagem prática dos conceitos aprendidos durante o ensino de Engenharia de Software auxilia na fixação do conteúdo apresentado, no entendimento e na razão da utilização das técnicas apresentadas. Com base nisso, esta monografia tem como objetivo o desenvolvimento de um jogo educativo que possibilite o usuário fixar os conhecimentos de Engenharia de Software, especificamente de gestão de pessoas, envolvendo a capacitação, carga de trabalho e função desempenhada pelos integrantes da equipe de desenvolvimento. A intenção é familiarizar o jogador com as dificuldades encontradas durante o desenvolvimento do software, levando-o a perceber a motivação do uso de técnicas de Engenharia de Software. Desta forma, esta monografia propõe o *Software Development Manager*, um jogo de simulação, onde o jogador possuirá uma empresa desenvolvedora de software que contará com o auxilio de uma equipe, que é administrada pelo jogador, para desenvolver produtos desejados por clientes. O intuito deste jogo é auxiliar no aprendizado dos conhecimentos de Engenharia de Software de uma forma que se aproveite os benefícios da diversão e entretenimento.

Palavras-chaves: jogos, engenharia de software, gestão de pessoas.

ABSTRACT

Software Engineering is a subject area of computing that focuses on practical aspects of the production of software. The Undergraduate courses of Computer Science have disciplines of Software Engineering, but it is shown usually in theoretically way and with only a few implementation jobs using the techniques and tools learned. The practical approach of the concepts learned during the teaching of Software Engineering aids in determining the content featured in understanding and the reason for using the techniques presented. Based on this, the following work aims to develop an educational game that allows the user to determine the knowledge of Software Engineering, specifically people management, involving the training, workload and role played by members of the development team. The intention is to familiarize the player with the difficulties encountered during software development, leading him to understand the motivation of using the techniques of Software Engineering. Thus, this work proposes the *Software Development Manager*, a simulation game where the player will have a software development company that will count with the help of a team, which is administered by the player, to develop products desired by customers. The purpose of this game is to assist in learning the knowledge of Software Engineering in a way that takes advantage of the benefits of fun and entertainment.

Keywords: games, software engeneering, people management.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ES: Engenharia de Software

UFF: Universidade Federal Fluminense

UFRJ: Universidade Federal do Rio de Janeiro

PUC-Rio: Pontifícia Universidade Católica do Rio de Janeiro

SDM: Software Development Manager

PnP: Problems and Programmers

TIM: The Incredible Manager

JEEES: Jogo de Estratégia para o Ensino de Engenharia de Software

LISTA DE APÊNDICE

SUMÁRIO

1 INTRODUÇÃO

A aprendizagem pode e deve ser um processo divertido. Levando em conta isso, uma opção de tornar a aprendizagem divertida é através da utilização de jogos com intuito de estimular a curiosidade e fornecer motivação no aprendizado. Prensky (2001a) dá doze razões que justificam o fato de jogos de computadores serem potencialmente o passatempo mais atraente na historia da humanidade:

- Jogos são uma forma de diversão. Fornecem entretenimento e prazer;

- Jogos são uma forma de jogar. Fornecem envolvimento intenso e apaixonado;

- Jogos têm regras. Fornecem uma estrutura;

- Jogos têm objetivos. Fornecem motivação;

- Jogos são interativos. Dão o que fazer;

- Jogos são adaptativos. Dão vazão;

- Jogos têm resultados e feedback. Fornecem aprendizagem;

- Jogos têm estados de vitória. Dão gratificação de ego;

- Jogos possuem conflitos, competição e oposição. Dão adrenalina;

- Jogos envolvem resolução de problemas. Estimulam a criatividade;

- Jogos têm interação. Fornecem grupos sociais; e

- Jogos têm representação e histórias. Fornecem emoção.

Existem outros meios que podem prover algumas destas características como, por exemplo, os filmes e livros. No entanto, os jogos são a única maneira de transmitir todos estes aspectos ao mesmo tempo (PRENSKY, 2001a).

Na área de Engenharia de Software (ES), o processo tradicional de ensino é constituído de aulas teóricas e alguns trabalhos práticos para utilizar a teoria aprendida. A finalidade destes trabalhos é de fixar o conteúdo. Porém, estes trabalhos práticos, na maioria das vezes, são trabalhos pequenos e nem sempre estimulam o interesse dos alunos.

A razão destes alunos não mostrarem muita motivação nesses tipos de trabalhos propostos é pelo fato de pertencerem a outro tipo de geração. Esta outra geração, denominada de “Nativos Digitais” por Prensky (2001b), está acostumada a receber informações muito

rápidas. Eles gostam de fazer tarefas em paralelo e preferem gráficos ou interações visuais ao invés das explicações por textos.

Prensky (2002) identificou que um dos problemas no aprendizado é por causa da falta de motivação do aluno. Para aprender qualquer assunto é necessário realizar um esforço, algo que é raramente feito sem ter um motivo. Enquanto o objetivo dos educadores é de transmitir o conteúdo, eles não se preocupam em manter o aluno envolvido. Já em jogos de computadores, o principal objetivo é manter o usuário envolvido e fazer com que ele queira dedicar mais e mais tempo ao jogo.

Levando em consideração estes fatos, a proposta sugerida nesta monografia é a criação de um jogo de ensino voltado para ES. Este jogo, denominado de *Software Development Manager* (KOHWALTER, 2011), tem como principal objetivo auxiliar no aprendizado dos conceitos de gestão de pessoas, envolvendo a capacitação, as funções desempenhadas e a carga de trabalho. Estes conceitos são ensinados nas aulas teóricas de ES, podendo o SDM auxiliar no aprendizado, mantendo o aluno interessado no assunto através da diversão oferecida pelo jogo.

Outro objetivo do SDM é transmitir a importância de cada elemento que aparece no jogo. Estes elementos são de administração da equipe de desenvolvimento e a importância de cada papel no desenvolvimento. Além disso, o jogo permite também transmitir a ideia de que caso um aspecto do produto seja alterado, podendo ser em relação a escopo, tempo, recursos e qualidade, pelo menos outro aspecto vai ser afetado.

Um objetivo secundário é mostrar as necessidades e restrições do cliente. Caso ele peça um produto, a equipe desenvolvedora não deve desenvolver um produto diferente do que foi pedido.

Esta monografia esta dividida em quatro capítulos, além desta introdução.

O Capitulo 2 apresenta outras propostas de jogos de ES que já foram desenvolvidos. Os jogos que são apresentados foram escolhidos devido às semelhanças entre si. Estes jogos são de duas categorias: cartas e jogos de computador. Os jogos de cartas desenvolvidos possuem várias semelhanças entre si, que são apresentados com mais detalhes no capitulo. A escolha dos jogos de computador foi feita para mostrar que vários aspectos de ES já são cobertos por outros jogos.

O Capitulo 3 trata da descrição do SDM, apresentando a abordagem proposta e os diferenciais em relação aos jogos citados no Capitulo 2. Durante a apresentação, são descritos em detalhes todos os elementos e aspectos contidos no SDM.

O Capitulo 4 exibe como o SDM foi implementado, apresentando a ferramenta de desenvolvimento utilizada e inspirações que foram retiradas de outro jogo para a sua interface. Neste capitulo também é mostrado e explicado a interface do SDM e é feita uma discussão de uma avaliação preliminar realizada sobre o jogo.

Por ultimo, o Capitulo 5 finaliza esta monografia apresentando as contribuições que o SDM oferece, as limitações deste jogo e os trabalhos futuros que podem ser feitos em cima do jogo a fim de ir alem a concepção inicial.

2 JOGOS DE ENGENHARIA DE SOFTWARE

A teoria faz parte do aprendizado e é indispensável em todo o processo educacional, mas pode ser mais bem compreendida quando utilizada em conjunto a situações praticas. Uma maneira de aplicar o conhecimento adquirido, nas aulas teóricas, é através da utilização de jogos para o ensino, também conhecidos como jogos sérios. De acordo com Zyda (2005), jogos sérios são definidos como uma competição mental, jogado com um computador de acordo com regras que utiliza o entretenimento para treinamento. Com isso pode-se dizer que jogos sérios são uma categoria de jogos que tem um propósito diferente do entretenimento. Este propósito pode ser, por exemplo, de educação, de exploração cientifica, planejamentos, simulação e visualização.

Este tipo de jogo é focado em resolver problemas, sendo muitas vezes utilizado como uma ferramenta de treinamento. No entanto, nada impede que o jogo sério seja divertido, porem este aspecto pode ser sacrificado para que se consiga transmitir a mensagem desejada. Enquanto jogos de entretenimento são medidos pela sua jogabilidade, os jogos sérios são classificados por categorias que correspondem ao conhecimento que se deseja transmitir. Algumas das categorias que jogos sérios podem ter são de aprendizagem e simulação.

Em ES, os estudantes são expostos a diversos conceitos teóricos e acabam possuindo poucas oportunidades para aplicar todos os conceitos aprendidos na pratica. Devido a este problema, jogos de ensino foram desenvolvidos para abordar os conceitos que são ensinados em ES, bem como auxiliar na fixação do conteúdo. A seguir serão apresentados cinco jogos de ensino de Engenharia de Software: o SimSE (NAVARRO, 2002) que foi desenvolvido na universidade da Califórnia, o Problems and Programmers (BAKER et al., 2003) também desenvolvido na Califórnia, o TIM (DANTAS et al., 2004) desenvolvido na UFRJ, o SimulES (FIGUEIREDO, E. et al., 2007) desenvolvido na PUC-Rio e o JEEES (FIGUEIREDO, K. et al., 2010) desenvolvido na UFF.

2.1 SIMSE

SimSE é um jogo de simulação de Engenharia de Software. Ele foi criado com o propósito de abordar a lacuna existente nas técnicas tradicionais de ensino de ES onde os

estudantes são expostos a diversos conceitos e teorias, mas possuem poucas oportunidades de aplicar estas ideias em prática.

No SimSE, o jogador tem o cargo de gerente de projetos que possui uma equipe de desenvolvedores. Enquanto o jogador gerencia o desenvolvimento do software, ele pode tomar decisões de contratação e demissão, monitorar o processo, atribuir tarefas e comprar ferramentas. A interface gráfica do jogo é exemplificada na Figura 1. Um dos objetivos fundamentais do projeto SimSE é permitir a customização de modelos de processos de softwares que é simulado, podendo assim ser utilizado por professores durante a apresentação de conteúdo sobre ciclos de vida do software.

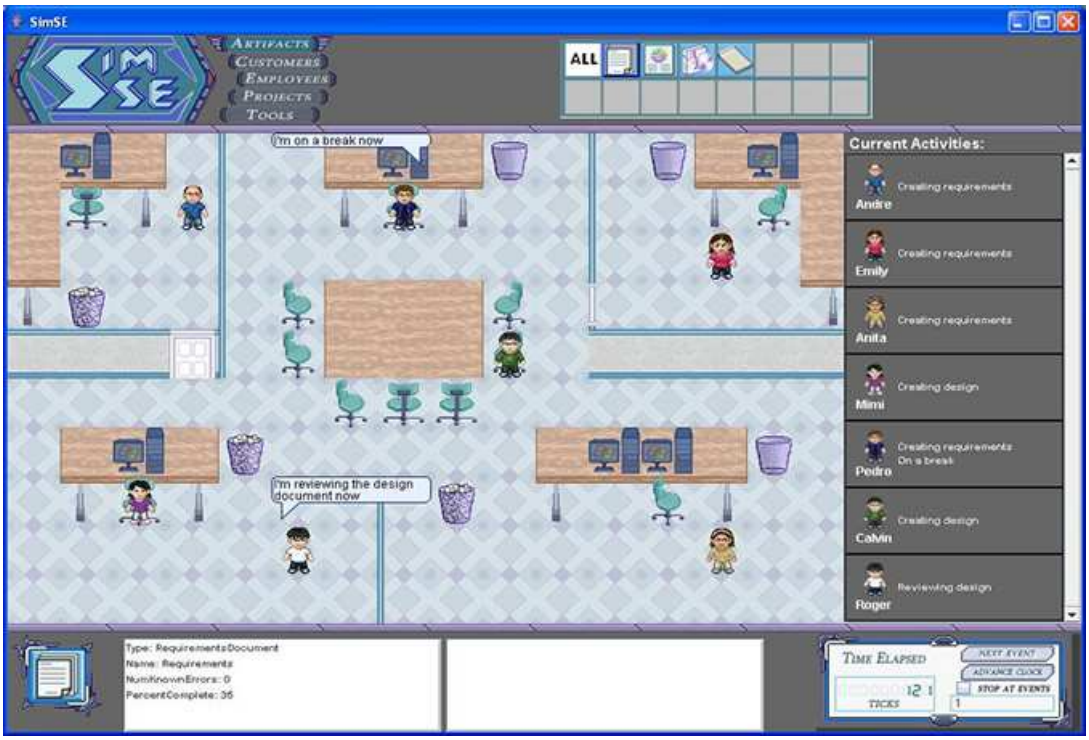

Figura 1- Interface grafica do jogo

SimSE é um jogo altamente interativo, onde tudo que acontece é exibido na interface gráfica em diversas formas, como por exemplo, a comunicação dos funcionários através de caixas de diálogos. Com o foco de fazer o jogo interativo, os autores adotaram um sistema de relógio para controlar a passagem do tempo, baseando-se na mesma ideia de jogos com turnos. Existem duas maneiras de se passar o tempo: uma delas é escolher quantas unidades de tempo deseja-se avançar, e a outra é passar o tempo até que um evento ocorra.

Apesar de o SimSE permitir a customização dos modelos de processos, eles devem ser escolhidos previamente para que se possa gerar uma simulação do modelo escolhido, não podendo ser alterados ao decorrer do jogo. No entanto, devido a esta customização, diversos modelos diferentes podem ser criados para simular diferentes métodos de desenvolvimento. Com esta funcionalidade de customização, o jogo permite que se adapte a novos modelos de desenvolvimento.

2.2 PROBLEMS AND PROGRAMMERS

Problems and Programmers (*PnP*) é um jogo de cartas onde o foco principal é o ensino de Engenharia de Software. Através de uma simulação de um processo de desenvolvimento de software, desde a concepção até a conclusão, os jogadores aprendem táticas para evitar problemas de ES durante o desenvolvimento do produto.

Neste jogo, os jogadores são gerentes de projetos que trabalham numa mesma empresa, cujo objetivo é competir entre si para completar seus respectivos projetos em menor tempo. No entanto, jogadores que adotam estratégias para minimizar o tempo ou façam decisões de riscos serão prejudicados, enquanto jogadores que optam por seguir os conceitos de ES serão recompensados. Para isto, o jogo depende de diferentes cartas para cada tipo de situação. Estas cartas são classificadas em:

- Cartas de Projeto: representam o projeto em que os jogadores competirem para desenvolver;
- Carta de Programadores: representam a mão de obra para o desenvolvimento do projeto;
- Cartas de Problemas: podem ser de diferentes tipos, sendo estes problemas de requisitos, de personalidade, de design e problemas aleatórios;
- Cartas de Conceitos: representam decisões opcionais que podem ser feitas durante o ciclo de vida do software.

Exemplos destas cartas estão representados na Figura 2.

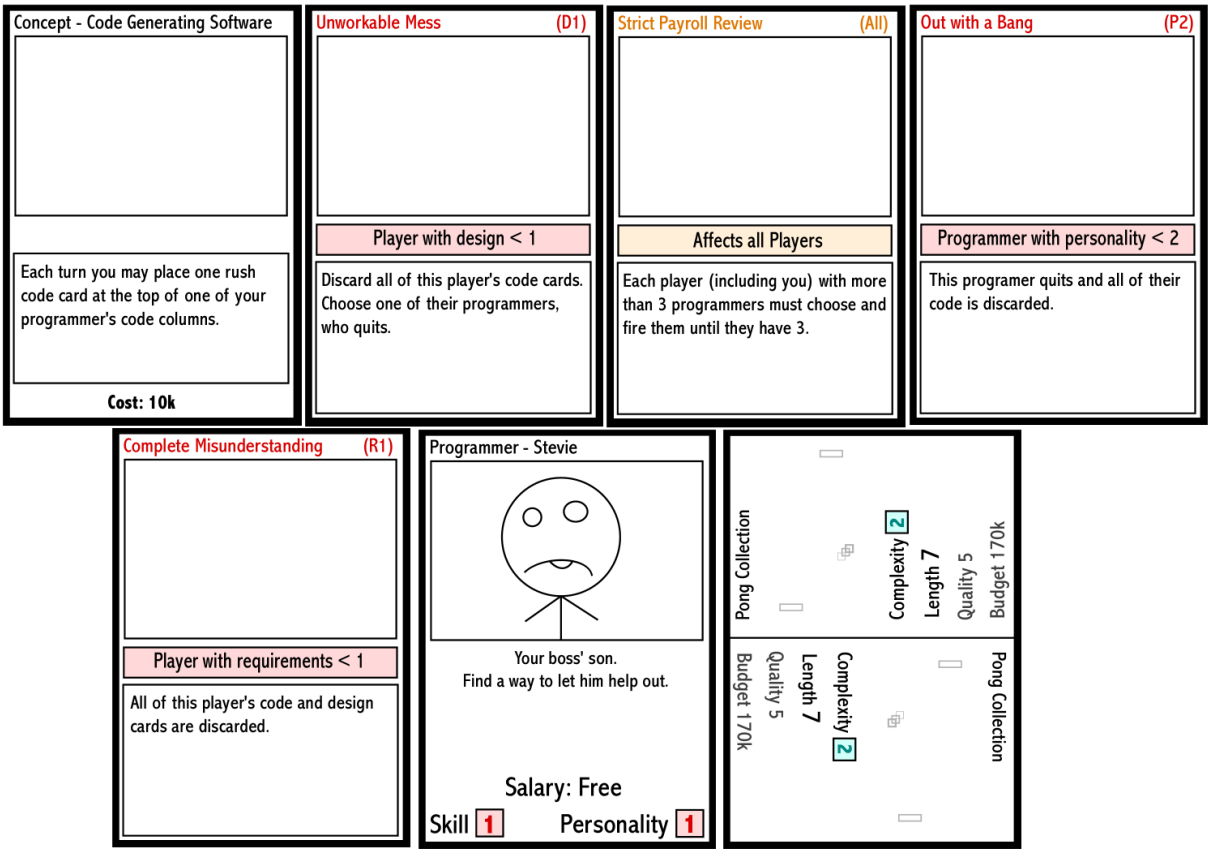

Figura 2- Exemplos de Cartas do jogo

Apesar de apresentar importantes fases do desenvolvimento do projeto de software, como por exemplo, documentação, implementação, inspeção e testes, o PnP deixa a desejar em alguns aspectos, tais como a estruturação das cartas e as regras, que tornam o jogo difícil de jogar com diversas pessoas e impossibilita de ser jogado sozinho. Com estas deficiências a dinâmica do jogo acaba ficando comprometida.

2.3 TIM

The Incredible Manager (*TIM*) é um jogo de simulação de ES com o foco de gerenciamento de projeto. O jogador tem a função de gerente em uma empresa onde deverá planejar e controlar projetos de desenvolvimento de softwares.

Como gerente do projeto, o jogador tem as opções de formar uma equipe de desenvolvimento, estimar duração de cada tarefa, atribuir tarefas para os desenvolvedores, negociar planos de projetos com os stakeholders, controlar quantas horas a equipe trabalhará por dia, determinar o esforço que será gasto para a garantia de qualidade do software e

contratar ou demitir desenvolvedores, onde todas estas opções estão disponíveis através da interface gráfica, como apresentado na Figura 3.

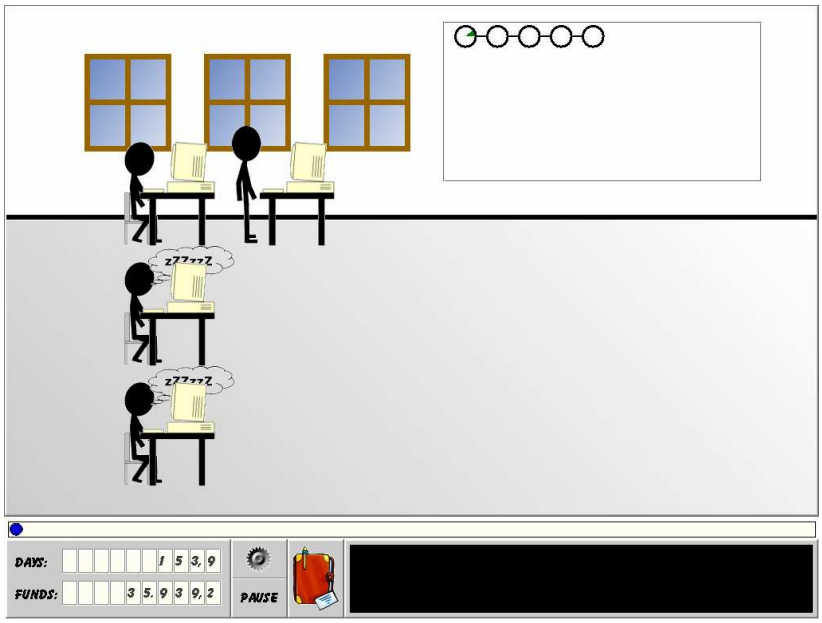

Figura 3- Interface gráfica

Os desenvolvedores podem ser afetados por status que variam de acordo com diversos fatores. O status pode ser de: pânico, que ocorre quando a tarefa esta atrasada, projeto esta atrasado, esta sem verbas ou sem tempo; cansado, quando o desenvolvedor esta fazendo muitas horas extras; e dormindo, que ocorre quando o desenvolvedor não possui nenhuma tarefa para ser realizada. A Figura 4 ilustra estes diferentes estados.

Figura 4- Estados do desenvolvedor

Em TIM, o jogador compõe um plano de desenvolvimento para o projeto. Este plano é utilizado para o desenvolvimento do projeto, que é realizado por etapas denominadas de tarefas. Durante a formulação do plano, é escolhido o desenvolvedor que será responsável por cada tarefa e realiza-se a estimativa da quantidade de horas que serão gastas para a das

tarefas. Com isto, pode-se perceber que o jogo se baseia fortemente no planejamento e na execução das tarefas que serão realizadas para desenvolver o projeto.

No entanto, assim como no jogo SimSE, são apresentados apenas dois tipos de papeis que podem ser desempenhados durante o desenvolvimento do projeto: o gerente, que é o próprio jogador, e de desenvolvedores, que são os funcionários da equipe. Porém, estes não são os únicos papeis que são desempenhados no desenvolvimento de um software. Existem outros que não foram mencionados, mas igualmente importantes, como por exemplo, analistas de sistemas e arquitetos de softwares.

2.4 SIMULES

SimulES é um jogo de cartas sobre o ensino de ES baseado no jogo PnP com o foco em evolução, sendo portanto um jogo similar ao PnP. Os participantes, de quatro a oito jogadores, devem disputar para terminarem um projeto de software, sendo o vencedor aquele que concluir o projeto primeiro.

Os jogadores, no decorrer do jogo, irão aprender conceitos importantes de ES, como Evolução de Software e técnicas contemporâneas de ES. Os recursos utilizados são: um tabuleiro onde cada jogador colocará seus engenheiros de software nas colunas e nas linhas os artefatos, como mostrado na Figura 5; cartas que podem ser de Projetos, de Problemas, de Conceitos, de Engenheiros de Software e de Artefatos, como são mostrados na Figura 6, e um dado de seis lados.

Tabuleiro do Jogador	**Engenheiros de Software**			
	Engenheiro 1	Engenheiro 2	Engenheiro 3	Engenheiro 4
Requisitos				
Desenhos				
Códigos				
Rastros				
Ajudas				

Figura 5- Tabuleiro do SimulES

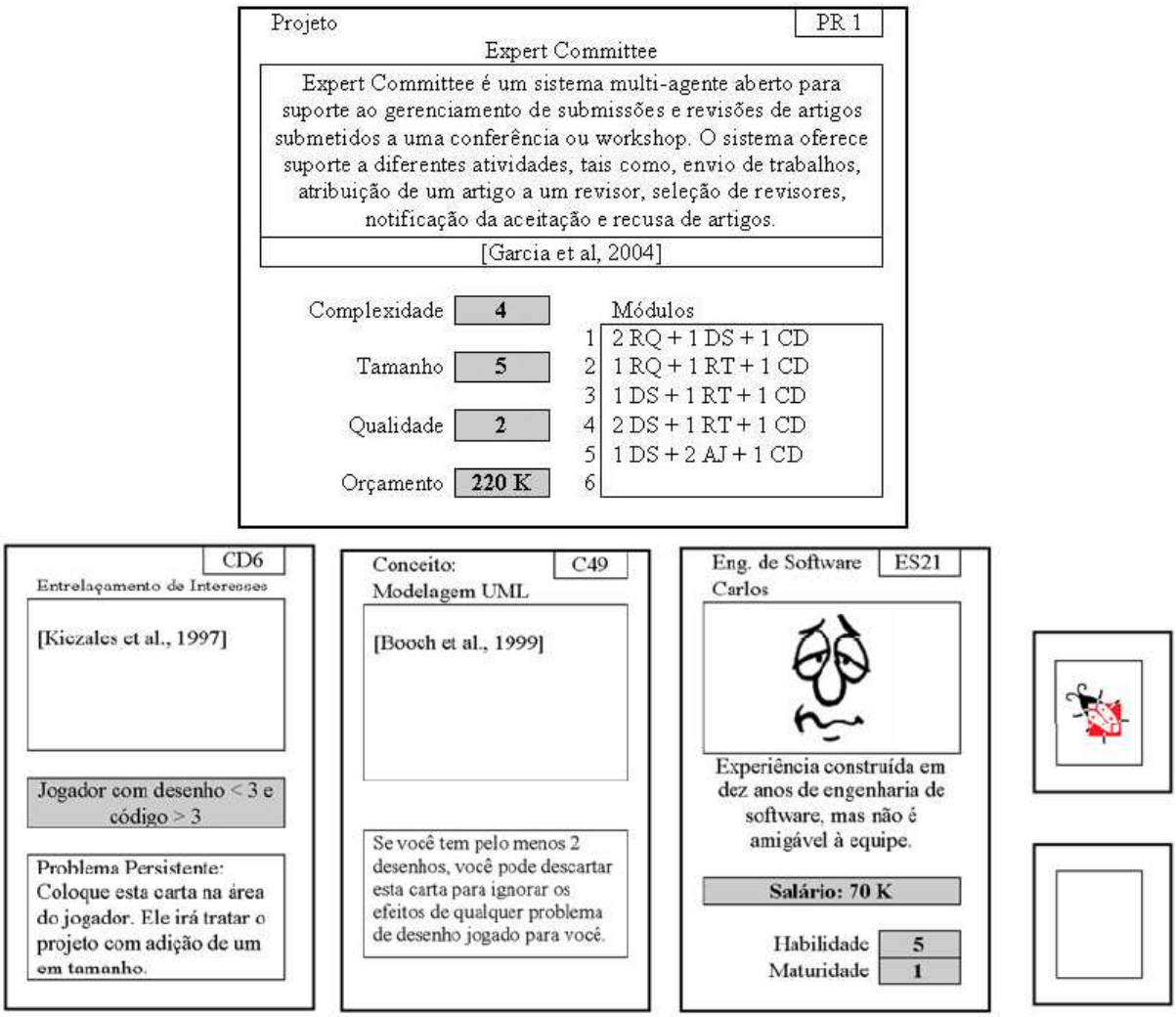

Figura 6- Exemplo de Cartas do SimulES

A proposta do SimulES é aprimorar a dinâmica do jogo PnP e identificar suas fraquezas para resolvê-las, tornando assim um jogo mais agradável de se jogar. Algumas soluções foram feitas através de alterações na regra do jogo, na remodelagem das cartas, tornando-as menos abstratas, e na implementação de um tabuleiro. Um problema presente no SimulES, assim como seu antecessor, é que não é possível que este jogo seja jogado por menos de quatro pessoas, tornando-o assim limitador ao usuário que deseja aprender conceitos de ES através do jogo porem não possui nenhum companheiro naquele momento.

2.5 JEEES

O Jogo de Estratégia para o Ensino de ES (JEEES) é um jogo de cartas educacional onde os autores tiveram inspiração por elementos do jogo SimulES. O objetivo

deste jogo é ensinar conceitos e práticas de Gerência de Configuração(GC) (ESTUBLIER, 2000) através da simulação de um ambiente de desenvolvimento de um projeto. Ele é composto por um tabuleiro, conforme exibido na Figura 7, cartas de diversos tipo, e de dados de seis lados.

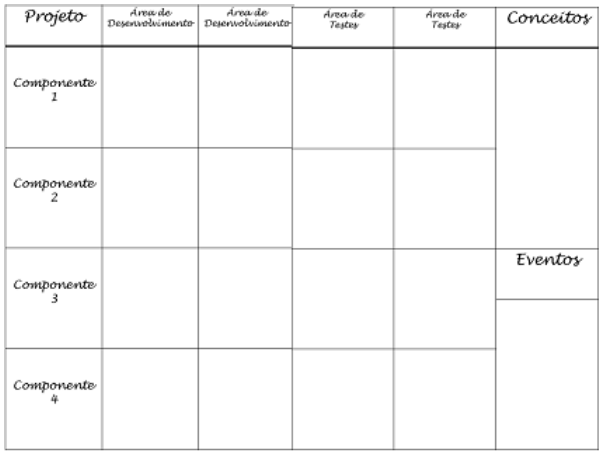

Figura 7- Tabuleiro do JEEES

As cartas são classificadas em cartas de Projetos, que possuem o objetivo que o jogador deve alcançar; de Equipe, onde representa a mão de obra contratada para realizar o projeto; de Eventos, que descrevem uma provável situação que pode ocorrer durante o desenvolvimento; e de Conceitos que descrevem elementos que podem ser adicionados ao projeto em desenvolvimento, como exibido na Figura 8. É através destas cartas que os conteúdos são transmitidos para os jogadores.

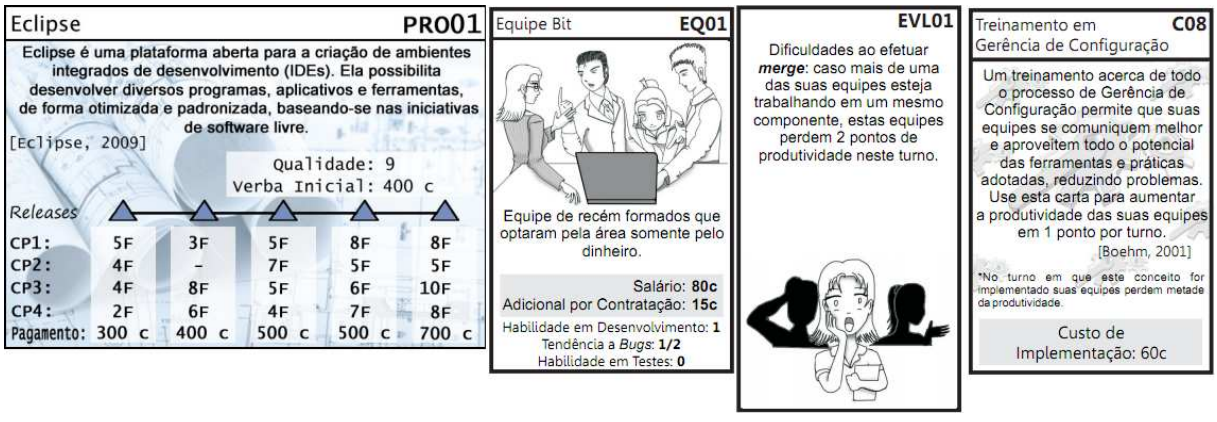

Figura 8- Exemplo de Cartas do JEEES

Os jogadores têm como meta concluir um projeto de software através da conclusão e entrega de todos os *releases*, caso estejam dentro do limite aceitável, que foram descritas na carta de projeto. Quem terminar o projeto primeiro ganha o jogo.

Diferente dos demais jogos de cartas descritos neste capitulo, o JEEES é altamente adaptável. Caso seja desejado, pode-se alterar o foco de ensino do jogo, que na sua versão inicial é para o ensino de Gerencia de Configuração, através da inclusão ou exclusão de novas cartas sem ter que fazer alterações nas regras do jogo. Isto permite criar decks de cartas específicos para cada assunto de ES que se deseja ensinar. Outra grande diferença é que o JEEES possibilita ser jogado individualmente ou em grupos, que pode ter seu tamanho variado dependendo do numero de cartas. Porem, uma das desvantagens dos jogos de cartas, incluído os anteriores citados, é o grande tempo de duração do jogo devido ao monitoramento manual de verificar se esta sendo jogado de acordo com as regras. Por exemplo, o tempo médio de duração do JEEES com um grupo de três jogadores é de duas horas.

2.6 CONSIDERAÇÕES FINAIS

Neste capitulo foram expostas as justificativas do porque jogos sérios são importantes e suas devidas vantagens ao serem utilizados para o ensino. Também foram apresentados cinco jogos de Engenharia de Software, onde cada um é focado em um aspecto de Engenharia de Software, como meios alternativos para auxiliar no ensino. No entanto, apesar de suas qualidades em suas respectivas áreas de interesse, estes jogos possuem algumas deficiências, nas quais já foram mencionadas em suas respectivas seções.

Pode-se perceber que, todos os jogos que foram citados possuem um aspecto em comum: focam em uma área de conhecimento de ES. Esse aspecto é positivo, quando se tem o propósito de auxiliar o aprendizado deste assunto específico. Porém nenhum jogo mostra uma visão abrangente dos conceitos de ES ou outros elementos essenciais para o desenvolvimento de software. O que é apresentado em cada abordagem é apenas o conceito que deseja-se abordar e os requisitos necessários para poder apresentá-lo. Com base nisso, pode-se dar um passo para trás, em relação à profundidade do conteúdo, e criar uma nova abordagem para o ensino de ES onde tenha o objetivo de abranger primeiramente uma área maior dos conhecimentos de conceitos de ES. Afinal, para poder se especializar em algum assunto, deve-se antes ter uma visão do todo.

3 O *SOFTWARE DEVELOPMENT MANAGER*

No jogo *Software Development Manager* (SDM), o jogador possui uma equipe de funcionários que são utilizados para desenvolver softwares solicitados por clientes. Cabe ao jogador decidir estratégias de desenvolvimento e definir os papéis de cada funcionário da equipe para o desenvolvimento do software. Os softwares requisitados pelos clientes possuem requisitos que devem ser respeitados durante o desenvolvimento. Quando um software é concluído e entregue ao cliente, é realizada uma avaliação de qualidade do software e o pagamento de conclusão do produto, ajustado de acordo com o resultado da avaliação.

Neste capitulo é apresentado à abordagem utilizada para o desenvolvimento do jogo SDM. Depois de apresentar os aspectos contidos na abordagem proposta para o desenvolvimento deste novo jogo, serão apresentados os pontos de diferencial das propostas dos demais jogos de ES apresentados no capítulo 2 e da abordagem proposta para o SDM.

Inicialmente é apresentado o funcionário no SDM que representa a mão de obra do jogador. Nesta seção que descreve os funcionários, são detalhados diversos aspectos. Estes aspectos são de papéis que podem ser desempenhados para o desenvolvimento do produto, de cargos que eles ocupam, de atributos que influenciam no desempenho das tarefas e das especialidades que podem ter. Além disso, é explicado como o processo de treinamento para o funcionário funciona e uma explicação dos elementos utilizados para controlar o rendimento do funcionário. Estes elementos são compostos de horas de trabalho, estamina e moral.

Em seguida é feita uma descrição da equipe do jogador entrando em detalhes sobre as restrições impostas pelo jogador e alterações dos integrantes da equipe. Depois da equipe, é descrito como é o processo de desenvolvimento de um produto no SDM. Nesta seção também se descreve o que é o produto, as suas características e restrições impostas pelo cliente. Antes de finalizar a descrição do desenvolvimento, é explicado como são realizadas as negociações e desenvolvimento de protótipos no jogo. Após estas descrições, é definido o papel do jogador. A Figura 9 apresenta o diagrama de classes do SDM.

Finalizando o capitulo, é realizado uma analise do diferencial do SDM em relação aos jogos de ES citados no Capitulo 2. Esta analise informa as diferenças nas abordagens adotadas. Por ultimo, se encontra as considerações finais deste capitulo.

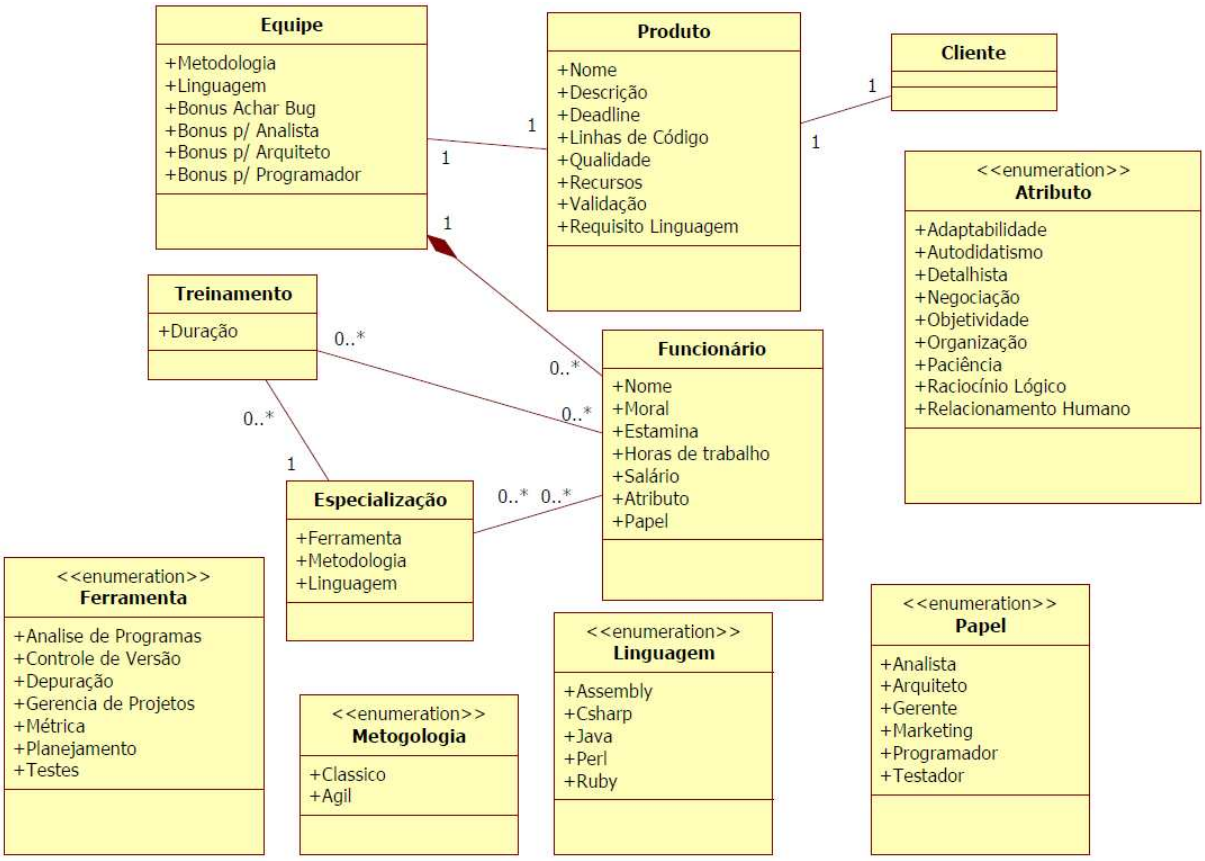

Figura 9- Diagrama de classes do SDM

3.1 FUNCIONÁRIOS

Os funcionários em SDM representam a mão de obra trabalhadora do jogador. São eles que desempenham as tarefas de planejamento, desenvolvimento e controle de qualidade do software através de tarefas que são atribuídas pelo jogador. Além das tarefas que podem ser atribuídas, os funcionários possuem algumas características que são utilizadas para determinar a afinidade do funcionário a exercer cada tarefa. Os demais aspectos relacionados ao funcionário, tais como papéis, cargo, atributos, especialidades, treinamentos, horas de trabalho, estamina e moral são detalhados nas próximas seções.

3.1.1 PAPÉIS

As tarefas mencionadas anteriormente são executadas pelos papéis que um funcionário pode desempenhar no jogo. Quando um papel for escolhido para um funcionário, este dedicará todo o seu tempo de trabalho exercendo a função desejada. Diferente de algumas empresas onde um funcionário pode exercer diferentes papéis, em SDM foi decidido que cada

funcionário poderá apenas exercer um papel por vez. No entanto, o jogador pode a qualquer momento alterar o papel desempenhado por um funcionário.

Na concepção do SDM, foram escolhidos seis tipos diferentes de papéis que podem ser atribuídas a um funcionário. Estes papéis são de analista, arquiteto, gerente, marketing, programador e testador. O objetivo da utilização destes seis papéis é expandir as possibilidades das funções que podem ser desempenhadas pelos integrantes da equipe. Cada um destes papéis possui uma função especifica que contribui para o desenvolvimento do software, que são descritas a seguir.

O analista de sistemas serve como um tradutor entre as necessidades do usuário e o modelo feito pelo arquiteto que é utilizado pelo programador durante o desenvolvimento. Para isto, o analista deve ter um conhecimento abrangente da área de negócios na qual o sistema será desenvolvido, a fim de que possa implementar corretamente as regras de negócio. Dito isto, o analista é responsável em aumentar as chances de sucesso na validação do software, que será descrito em mais detalhes mais a frente na seção 3.3.1.

O arquiteto de software é o profissional responsável pela concepção, pelo projeto e desenvolvimento da arquitetura do sistema a ser desenvolvido. Ele é o responsável em criar um procedimento de testes para garantir que o software esteja no nível de qualidade inicialmente proposto. Além disso, ele é o responsável por gerar protótipos, que serão detalhados na seção 3.3.3. O arquiteto também trabalha em conjunto com o testador da equipe, gerando os planos de testes.

O gerente é o indivíduo responsável pelo planejamento e controle da execução dos trabalhos dos membros da equipe durante o dia-a-dia. O gerente, dentro da abordagem proposta, também gerencia os recursos humanos da equipe, ou seja, é o responsável pela contratação de novos funcionários. Além desta tarefa, o gerente auxilia indiretamente em quase todas as funções desempenhadas pelos outros papéis durante o desenvolvimento do produto.

O papel do marketing é negociar com o cliente o contrato do projeto. Este perfil é responsável pelas negociações que serão descritas na seção 3.3.2. O funcionário de marketing também consegue influenciar nas requisições feitas pelo cliente aos analistas de sistemas, conseguindo convencer o cliente que uma funcionalidade inicialmente não desejada seja

aceitável. Com esse convencimento, o marketing consegue auxiliar o analista durante as validações do software.

O código não surge sozinho. Para isto, é necessário ter programadores para desenvolver o software. Sendo assim, o programador é responsável por gerar as linhas de código do software, fazendo o desenvolvimento do produto solicitado torna-se realidade. Porém, o programador também é responsável, inadvertidamente, por colocar *bugs* no software durante o desenvolvimento. Estes *bugs* prejudicam a qualidade final do produto e a quantidade de *bugs* introduzidos é influenciada pela linguagem de programação utilizada, a experiência do programador com esta linguagem, a compatibilidade da metodologia de trabalho da equipe, e o desempenho em programação de acordo com as suas características humanas.

Por último, porém não menos importante que os demais, o testador é a pessoa responsável por realizar a verificação do software a fim de fornecer informações sobre sua qualidade em relação ao contexto em que ele deve operar. Isto garante que o software opere da forma devida. Esta responsabilidade inclui o ato de utilizar o produto com o intuito de encontrar seus possíveis *bugs* para submeter os mesmos à equipe de desenvolvimento para correção. Caso eles não sejam corrigidos ou encontrados, a qualidade do produto final fica prejudicada. Como foi mencionado anteriormente, o testador trabalha em conjunto com o arquiteto da equipe, caso tenha.

3.1.2 CARGO

O cargo de um funcionário define o quanto que este funcionário contribuirá para o desenvolvimento do software. Como consequência, cargos afetam o salário do funcionário e sua produtividade. Existem diferentes classes de cargos presentes no SDM onde todos eles seguem um padrão, tais como o desenvolvedor júnior, desenvolvedor pleno e desenvolvedor sênior.

Para facilitar o entendimento dos cargos distintos em SDM, adotam-se os cargos pertencentes à classe Júnior como o cargo padrão de cada funcionário. O cargo da classe Pleno, caso seja atribuído a um funcionário da empresa, proporciona um aumento de 20% na contribuição do funcionário no papel no qual ele exerce e um aumento de 30% no seu salário.

No cargo da classe Sênior, estes valores sobem para um aumento de 40% de contribuição e 50% de salário. Esta alteração na produtividade é reflexo do aumento salarial.

É possível perceber que parece não valer a pena alterar o cargo dos funcionários da empresa devido à desproporção do ajuste de salário. No entanto, alguns projetos disponíveis ao jogador necessitarão que o software esteja pronto em um tempo mais curto do que o ideal, tornando a escolha de cargo mais um fator de decisão para o desenvolvimento do software. Esta decisão aumenta a produtividade de um setor sem ser necessário escalonar outro funcionário. Outro motivo para se aproveitar deste sistema é pelo fato do tamanho da equipe possuir um limite, não permitindo a contratação de mais funcionários.

3.1.3 ATRIBUTOS

Alem de papéis e cargos, os funcionários possuem atributos que influenciam no seu desempenho de trabalho. No entanto, esses atributos não são diretamente relacionados a cada papel, como por exemplo um atributo de gerenciamento que seria utilizado para o papel de gerente. Para tornar o SDM ainda mais realista e ter mais um diferencial dos demais jogos, o sistema adotado para atributos foi à utilização de características humanas de uma pessoa. Esses atributos humanos são os atributos que o jogador pode consultar para formular uma avaliação de desempenho que um funcionário tem em determinado papel. Porem o mapeamento destes atributos humanos para os atributos de desempenho não é informado ao jogador. Com isso, ele precisa utilizar o bom senso e experimentos para conseguir descobrir quais atributos são mais importantes para cada função. No entanto, isto não significa que apenas alguns atributos influenciaram no desempenho. Na realidade, todos eles influenciam no desempenho de cada papel, porém em graus diferentes.

Depois de ter realizado um estudo sobre o perfil (SANTOS, 2005) (RUSSO, 2007) de funcionários que desempenham determinados papéis, foram selecionados nove atributos humanos que são utilizados no SDM para representar as características do funcionário. Os atributos são: adaptabilidade; autodidatismo; detalhista; negociação; objetividade; organização; paciência; raciocínio lógico; e relacionamento humano. Estes atributos humanos são descritos a seguir com mais detalhes e comentados para que tipo de papel é mais adequado. No entanto, vale relembrar que todos os atributos influenciam no desempenho final.

O atributo “Adaptabilidade” é a capacidade de reagir a mudanças. Este atributo, especificamente no SDM, reflete na habilidade do funcionário a se adaptar a alterações no escopo e planejamento do software. Este atributo é bastante utilizado por analistas e arquitetos e é importante para gerentes;

O atributo “Autodidatismo” é responsável pela capacidade do aprendizado sem precisar ter um instrutor. O próprio indivíduo faz buscas e pesquisas do material necessário para a aprendizagem. Este atributo é importante para o programador. Além de ser utilizado para determinar o desempenho nas tarefas, este atributo é aplicado quando um funcionário está em treinamento. Esta aplicação é utilizada para alterar o tempo de duração necessária para a conclusão do treinamento;

O atributo “Detalhista” é utilizado para medir a capacidade do funcionário em prestar a atenção aos mínimos detalhes do problema, verificando ponto a ponto o nível de detalhamento das atividades. No SDM este atributo é fortemente utilizado por testadores e importante para analistas e arquitetos;

O atributo de negociação é usado para determinar a habilidade do funcionário para realizar trocas de convencimentos, ou seja, uma parte persuade a outra apresentando os benefícios mais relevantes em relação ao ponto de vista defendido. Este atributo, como o próprio nome já diz, é essencial durante negociações e é fortemente usado pelo funcionário de marketing. Mas como já foi dito anteriormente, ele sozinho não determina o desempenho final das negociações;

O atributo “Objetividade” representa a capacidade de ser objetivo. Serve para buscar a solução mais simples e funcional, principalmente durante a implementação do software. Sendo assim, é um bom atributo para os programadores;

O atributo responsável pelo estruturamento do trabalho e combinação dos esforços individuais para realizar esforços coletivos é a organização. Ele também é útil para planejamentos e criação de cronogramas, tornando-o assim um bom atributo para gerentes;

Paciência é uma virtude de manter um controle emocional equilibrado, sem perder a calma, ao longo do tempo. Consiste basicamente de tolerância a erros ou fatos indesejados. É a capacidade de suportar incômodos e dificuldades de toda ordem, de qualquer hora ou em

qualquer lugar; De persistir em uma atividade difícil, tendo ação tranquila. Este atributo é importante para os testadores, porem não menos importante para os demais papéis;

O atributo “Raciocínio Lógico” define a capacidade de procurar uma solução de um problema quando são disponíveis dados como ponto de partida, porém não se sabe muito bem como alcançar o objetivo. Serve para identificar erros intuitivamente, compreender a lógica por trás do problema e efetuar contas matemáticas. Este atributo é muito importante para programadores e testadores;

O atributo “Relacionamento Humano” representa a característica responsável pela habilidade de comunicação interpessoal e de persuasão. Ser capaz de expor ideias e conseguir distinguir o vocabulário técnico do de negócios. Este atributo é fortemente utilizado por funcionários que desempenham o papel de analistas, gerentes e marketing.

3.1.4 ESPECIALIZAÇÕES

Outro aspecto que cada funcionário possui são as especializações. Estas especializações podem ser em relação a ferramentas, linguagens de programação ou técnicas de ES. As especializações em linguagens de programação refletem quais as linguagens que o funcionário esta apto a trabalhar. Especializações em técnicas são utilizadas para determinar se o funcionário esta familiarizado com determinadas metodologias de ES que são adotadas pela equipe de desenvolvimento. Em ambos os tipos de especializações, caso o funcionário não esteja apto a trabalhar de acordo com os critérios escolhidos pelo jogador, este funcionário sofrerá penalidades em sua produtividade. Para contornar este problema, o jogador terá a opção de treinar o funcionário na especialização desejada. A questão de treinamento será tratada no item 3.1.5.

Diferente das demais, a especialização de ferramentas é utilizada para auxiliar o funcionário a exercer sua função. Para cada papel existente no SDM, existem especializações relacionadas que são levadas em consideração, caso o funcionário às possua, para aumentar o rendimento produtivo dele. A falta deste tipo de especialização não levará a nenhuma penalidade para o funcionário.

3.1.5 TREINAMENTO

Como foi mencionado anteriormente, é possível treinar um funcionário para que possa adquirir uma especialização nova. O treinamento pode ser escolhido pelo jogador e ocorrer a qualquer momento. Para treinar um funcionário, deve-se escolher uma especialização que se deseja aprender e pagar uma quantia de dinheiro para que se possa iniciar o treinamento. O tempo de duração do treinamento dura vários dias, e é influenciado pelo atributo de autodidatismo. Durante este tempo o funcionário se dedicará apenas ao treinamento, não sendo permitido que ele exerça outra função simultaneamente. Somente quando o treinamento for concluído o funcionário poderá retomar as suas obrigações na empresa.

Com a opção de treinamento, o jogador consegue perceber que é necessário, antes de começar o desenvolvimento, treinar a equipe para que possa alcançar os requisitos estabelecidos do projeto. Caso não seja feito um treinamento nos funcionários e eles não estejam aptos, o desempenho deles vai ser reduzido e com isso a qualidade do software será afetada negativamente. Outro aprendizado que o jogador pode obter com o aspecto de treinamento é que se um funcionário já pertence à equipe do jogador a mais tempo, será muito provável que este já tenha passado por vários treinamentos e com isso esteja apto a realizar as novas tarefas sem precisar de um treinamento prévio.

3.1.6 HORAS DE TRABALHO, MORAL E ESTAMINA

O número de horas de trabalho é outro fator que influencia na produtividade de um funcionário. As horas de trabalho dos funcionários podem ser escolhidas individualmente, permitindo assim que um determinado funcionário trabalhe mais horas que os demais. No entanto, este fator afeta diretamente o salário do funcionário e sua estamina.

O moral e estamina de um funcionário definem o comportamento que este terá durante o trabalho e esta diretamente relacionada com a produtividade. Como padrão, foi adotado uma métrica para moral e estamina de 0 a 100, onde 100 representa que o funcionário esta se dedicando ao máximo no trabalho e 0 que ele não esta trabalhando. A estamina, como mencionada antes, é afetada pelas horas de trabalhos. Caso o funcionário esteja fazendo horas extras, a sua estamina começa a reduzir, causando assim uma queda de rendimento. Caso o

funcionário esteja trabalhando menos horas que o normal, a sua estamina aumenta com o passar dos dias e caso esteja trabalhando o numero normal de horas, a sua estamina estabiliza.

Com isso pode-se concluir que quando um funcionário começar a fazer horas extras, a sua produtividade é maior em curto prazo, pois a sua estamina começa a diminuir. Depois de alguns dias, essa produtividade gerada pelas horas extras tem seu efeito negado por causa da estamina e caso isso continue, mesmo que o funcionário esteja trabalhando mais do que o normal, a sua produtividade vai cair alem do normal. Isso pode ser comparado analogamente com o cansaço do funcionário. A única maneira de fazer a sua produtividade voltar aos níveis normais é pela redução das horas de trabalho do funcionário afetado, ou seja, fornecer uma folga para que este possa recuperar o fôlego.

Moral também afeta a produtividade do funcionário, porem não por cansaço e sim por vontade de trabalhar. O moral é responsável pelo desejo do funcionário de permanecer na empresa. Caso o moral do funcionário esteja muito baixa, este pode pedir demissão voluntariamente. A falta de pagamento ao funcionário afeta negativamente o moral, então caso não seja feito pagamentos, o moral dele é reduzido. Sendo assim, é possível que o funcionário peça demissão por falta de pagamentos. Outro meio que pode reduzir o moral dos funcionários é não concluir o projeto. O moral é recuperado através de promoção de cargo, desenvolvimento de um projeto desafiador ou conseguir concluir o projeto corrente.

Esse sistema de moral, estamina e horas de trabalho fornecem ao jogador mais uma possibilidade estratégica para o desenvolvimento do software. Cabe ao jogador fazer a ponderação dos benefícios e prejuízos da utilização de tais funções e julgar quando for necessário utilizá-lo.

3.2 EQUIPE

A equipe do jogador é composta de funcionários contratados que se dedicam ao desenvolvimento do software. Os integrantes da equipe podem executar tarefas distintas, porém todos obedecem algumas restrições que são impostas pelo jogador ou pelo contrato do projeto. Estas restrições podem ser uma linguagem de programação imposta pelo cliente e uma metodologia de trabalho, que é escolhida pelo jogador.

Restrições de linguagem de programação, caso existam no projeto corrente, devem ser respeitadas. Ou seja, se existir uma restrição deste tipo, o software só será desenvolvido quando a linguagem adotada pela equipe de trabalho do jogador for a mesma que esta no contrato do projeto. Porém, nem todos os contratos vêm com esta restrição, podendo assim ter projetos em que fica a cargo do jogador decidir qual linguagem será utilizada para o desenvolvimento.

A restrição de metodologia é escolhida pelo jogador, obrigatoriamente. Ela só pode ser alterada na primeira semana do projeto. Esta restrição informa qual metodologia a equipe vai utilizar para o desenvolvimento do software. Uma vez escolhida a metodologia de trabalho para a equipe, esta não poderá mais ser alterada e é utilizada para contabilizar penalidades nos funcionários que não estejam qualificados a trabalhar na metodologia escolhido.

3.2.1 CONTRATAÇÃO

A equipe pode ser alterada ao decorrer do jogo por ações tomadas pelos funcionários ou pela escolha do jogador. A primeira alternativa já foi explicada anteriormente, onde é afetado pelo moral, e a segunda, que é feita pelo do jogador, pode ser de contratação ou demissão de um ou mais funcionários.

O jogador pode demitir funcionários da equipe a qualquer momento. Porém para que se possa fazer uma alteração de forma a adicionar um novo integrante, é necessário que um dos funcionários pertencentes à equipe esteja desempenhando o papel de gerente. Através deste gerente, o jogador pode solicitar adições na equipe, caso ainda possua vaga. Caso não tenha vaga disponível na equipe e o jogador mesmo assim desejar contratar um funcionário, é necessário escolher um membro atual da equipe para que possa ceder a sua vaga. Ou seja, este funcionário selecionado será demitido para ceder o seu lugar.

3.3 DESENVOLVIMENTO

O desenvolvimento é a parte fundamental no SDM. É durante o desenvolvimento que o jogador deve fazer as decisões de gerenciamento e de ES. Em função das decisões tomadas, o produto pode ser bem sucedido ou fracassado. Algumas destas decisões já foram

discutidas em itens anteriores, como a contratação, metodologias de trabalho, escalonamento de funcionários e suas respectivas horas de trabalhos.

Antes de entrar nas outras possibilidades de decisões que podem ser feitas será apresentada uma descrição do produto, na qual a empresa do jogador se compromete a desenvolver. Em seguida, são discutidas as opções que podem ser tomadas pelo jogador que podem afetar no desenvolvimento do produto.

3.3.1 PRODUTO

No SDM o objetivo é criar o produto que o cliente deseja, ou seja, é o desenvolvimento do software que o cliente quer que seja desenvolvido. O software possui características limitadoras, como foi citado anteriormente, que serão impostas pelo cliente. Outras características pertencentes ao software são o tamanho, *deadline*, qualidade e pagamento mensal.

Os projetos possuem *deadlines* que devem ser respeitados, ou seja, a data final de entrega do produto. Caso o dia do deadline chegue e o jogador não consiga terminar o software, o contrato será perdido e penalidades de moral serão aplicadas a equipe. Em função do deadline, o tamanho do software informa, de maneira indireta, o nível de dedicação necessária de programação para completar o software dentro do prazo e o grau de complexidade de validação do software. Ele é informado no momento em que o jogador escolhe, numa lista de possibilidades, o projeto desejado para o desenvolvimento.

Outro aspecto do software é a qualidade, que representa o grau de rigor que é cobrado pelo cliente em relação a qualidade do software. Ele é informado no momento da escolha do projeto. Um cliente que não se preocupa muito com a qualidade vai exigir menos durante a avaliação final do produto. No SDM a qualidade é traduzida no valor que o cliente penaliza o jogador no pagamento final para cada *bug* que o software desenvolvido possua. Outro aspecto também ligado à qualidade, que no entanto não possui uma exigência explicita, é a validação. Esta validação se refere ao modelo criado pelo analista, que é utilizado para o desenvolvimento do software. A validação informa o quão perto este modelo se aproxima do que foi pedido pelo cliente. Ou seja, se o que esta sendo desenvolvido reflete no que o cliente desejava.

A validação é o resultado direto das atividades feitas pelo analista da equipe. Quando o software for concluído, ele será avaliado em relação à quantidade de *bugs* encontrados e deixados no produto, e o seu grau de validação. Em cima deste grau, o valor final que o jogador recebe é ajustado para ficar de acordo com o que foi desenvolvido.

3.3.2 NEGOCIAÇÃO

Durante o desenvolvimento, é permitido que sejam realizadas negociações com o cliente a respeito do produto. Estas negociações afetam alguns aspectos do contrato do software. Para que negociações sejam realizadas, é necessário que a equipe de desenvolvimento tenha um representante de marketing.

Através do funcionário que estiver exercendo o papel de marketing, é possível fazer alterações no projeto. Estas alterações podem ser de deadline, qualidade, escopo e recursos. Quando uma alteração for solicitada, o cliente exige que outro aspecto do contrato seja compensado. Por exemplo, caso o jogador queira que seja feita uma alteração no deadline a fim de estender o prazo, vai ser necessário selecionar outro item, como por exemplo, um aumento da qualidade para poder compensar a alteração. Os resultados das negociações são afetados diretamente pelo desempenho do funcionário de marketing envolvido.

3.3.3 PROTOTIPAÇÃO

Ao decorrer do desenvolvido do software, é possível que sejam construídos protótipos para serem entregues ao cliente. Estes protótipos têm como objetivos auxiliar na compreensão e entendimento do que o cliente deseja que o software seja. Desta forma, os protótipos são usados para alinhar o que foi projetado pela equipe com o que foi requisitado pelo cliente, ou seja, causa uma alteração no nível de aceitação do software.

Protótipos são opcionais, sendo feitos apenas quando forem solicitados explicitamente pelo jogador. Esta solicitação deve ser feita a um funcionário da equipe que esteja executando a função de arquiteto. Uma vez solicitado, o arquiteto vai desenvolver um protótipo e entregar ao cliente para avaliação, obtendo um retorno. Este retorno é aplicado no desenvolvimento do software, que permite detectar alguns pontos de discrepância no que foi desenvolvido e em seguida concertá-los.

3.4 JOGADOR

Ao decorrer das descrições dos demais elementos do SDM, foi mencionado diversas vezes a existência do jogador e a sua intervenção no jogo. Ao combinar as informações previamente discutidas sobre o jogador, é possível ter uma ideia do papel que o jogador desempenha.

O jogador é a entidade responsável pela empresa na qual esta interagindo. Ele é o dono desta empresa. Através dela, ele toma decisões que influenciam diretamente no desenvolvimento do produto. Todas as partes de recursos humanos, planejamento, administração e gerenciamento são controladas pelo jogador. Além disso, ele é o responsável por administrar a situação financeira da empresa, podendo cortar gastos ao reduzir a carga horária de seus funcionários ou demiti-los. Também pode não poupar gastos no desenvolvimento do produto, sendo até mesmo uma estratégia onde é investido fortemente em prototipação, treinamentos e nos cargos dos funcionários da equipe a fim de terminar o projeto mais cedo e com mais qualidade. Esta estratégia pode ser adotada, pois na conclusão de um projeto ocorre um pagamento que se iguala a vários meses de trabalho, caso o produto esteja em excelente qualidade.

3.5 DIFERENCIAL

Os trabalhos que foram citados no Capitulo 2 têm como propósito o ensino de conceitos específicos de ES. Por causa dessa especialização vários outros aspectos não menos importantes são deixados fora do escopo dos mesmos. Tendo isto em vista, o SDM é um jogo que se propõem a ter uma maior área de abrangência em ES. Neste sentido, o jogo foca bastante na parte de recursos humanos, ou seja, nas pessoas que desenvolvem o software para o jogador. Com essa adição de complexidade aos funcionários, o jogo apresenta um novo aspecto onde o jogador precisa administrar a equipe de trabalho. Este gerenciamento é uma das dificuldades encontrada no desenvolvimento de projetos (TERRIBILI, 2010)**.**

Outro diferencial da abordagem proposta em relação aos demais trabalhos é o fato da passagem de tempo no jogo SDM ser em tempo real, tornando assim o jogo mais interativo. Os demais jogos citados, com exceção do TIM, contam com uma estratégia de

passagem de tempo que é baseada em turnos. Nos jogos de cartas, a passagem de tempo precisa ser inevitavelmente por turnos, porém o SimSE, mesmo não sendo um jogo de cartas e sim um jogo de computador, decidiu adotar este modelo.

Por ultimo, os demais jogos tem ou um ambiente em 2D (SimSE e TIM), ou fazem uso de cartas (PnP, JEEES e SimulES). Já o SDM foi desenvolvido em um ambiente de desenvolvimento de jogos que abriu a possibilidade de criar o jogo totalmente 3D.

4 DESENVOLVIMENTO

4.1 INTRODUÇÃO

O SDM foi desenvolvido utilizando a ferramenta de desenvolvimento de jogos Unity3D (HIGGINS, 2010). O Unity3D é um ambiente de produção que facilita a criação de jogos, abstraindo elementos técnicos durante o desenvolvimento. Com esta abstração, o Unity3D permite que o desenvolvedor se concentre apenas na criação do jogo sem precisar se preocupar com elementos do motor. A Figura 10 exibe a interface de desenvolvimento do Unity3D. A imagem superior no canto esquerdo exibe a visão de desenvolvedor e a imagem inferior mostra como o jogo é visualizado enquanto está sendo jogado. As três colunas a direita são utilizadas para o desenvolvimento, onde a primeira, da esquerda para direita, exibe todos os objetos contidos no cenário. A segunda informa todos os elementos disponíveis para o desenvolvimento como, por exemplo, figuras, texturas e *scripts*. A última coluna mostra a visão de inspetor, que contem as características do objeto selecionado, permitindo que alterações sejam feitas.

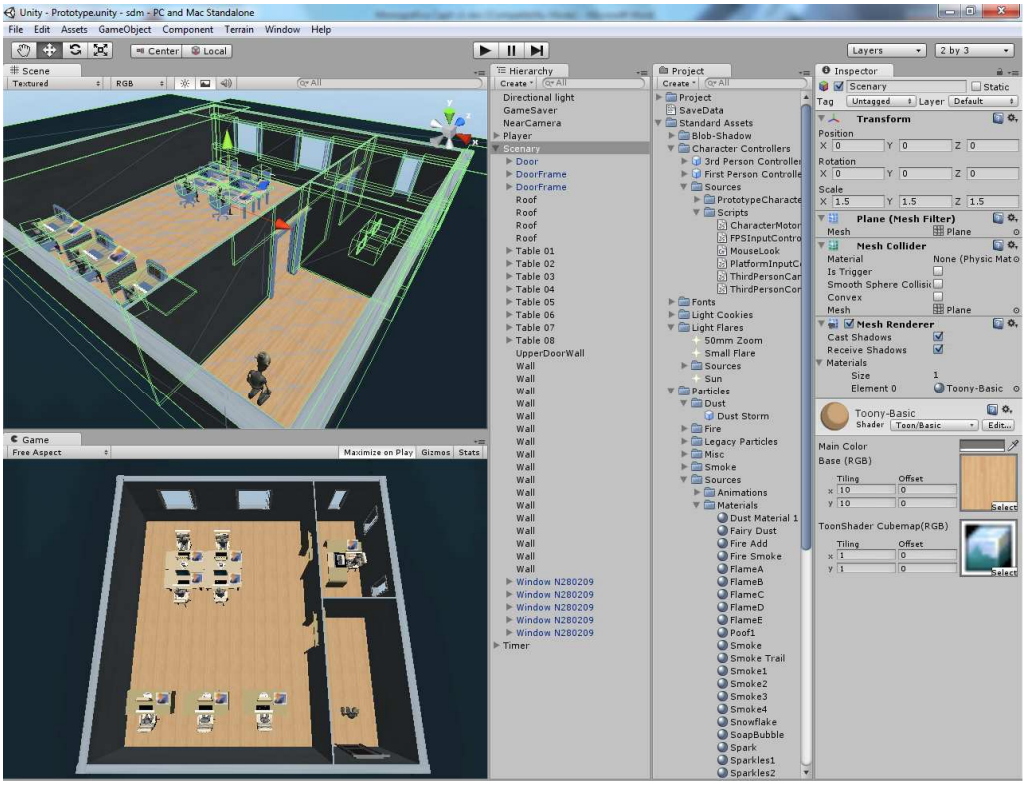

Figura 10- Área de Trabalho do Unity3D

Vale ressaltar que o jogo Game Dev Story (GAMES, 2010) teve uma influencia no desenvolvimento do SDM, mesmo não sendo um jogo de ES. O Game Dev Story é um jogo que simula uma empresa desenvolvedora de jogos controlada pelo jogador. Os elementos que foram influenciados foram o *gameplay* do SDM e a interface do usuário. A Figura 11 mostra duas imagens da interface do Game Dev Story.

O elemento que se destacou mais no Game Dev Story que influenciou no desenvolvimento da interface do SDM foi o *feedback* fornecido ao jogador. Conforme os funcionários trabalham, ícones são exibidos, informando ao jogador o desempenho do funcionário. Estes ícones podem ser de balões contendo uma mensagem ou de figuras ilustrativas que representam a área de atuação do funcionário. Na Figura 11 é possível ver estes dois tipos de ícones.

Figura 11- Interface do Game Dev Story

Na próxima seção será apresentado como o SDM foi implementado. Começando pela interface, são apresentados todos os componentes pertencentes a interface do SDM. Em seguida é apresentado o dialogo do jogador com os funcionários, mostrando as janelas de interação que informam os dados dos funcionários e permitem fazer alterações no funcionário relacionadas ao trabalho. Depois é mostrado como é realizada a contratação de novos integrantes para a equipe, a negociação com o cliente e a criação de protótipos. Por fim as janelas de finalização do projeto corrente e de novas possibilidades de contratos são apresentadas. Terminando o capitulo, colocamos os resultados obtidos por um experimento realizado com alunos da UFF, que foram submetidos a uma sessão de jogo do SDM e em seguida responderam um questionário relacionado ao jogo.

4.2 IMPLEMENTAÇÃO

Nesta seção é explicada à implementação do SDM. Como já foi discutido na seção 4.1, o SDM foi desenvolvido utilizando a ferramenta Unity3D para facilitar o desenvolvimento do jogo. A Figura 12 exibe a interface da versão atual do SDM, feita no Unity3D.

A Figura 12, além de mostrar a interface do jogo, exibe o momento em que o jogador inicia o jogo. É possível perceber que existe uma tela de introdução onde são apresentados ao jogador algumas explicações básicas sobre o jogo. Uma vez que o jogador terminar de ler estas informações e fechar a janela inicial, outra janela aparece mostrando os dados do projeto que deve ser desenvolvido. Após serem lidas estas informações o jogador está pronto para iniciar o desenvolvimento do software. A Figura 13 mostra a janela que contém as informações do projeto, que neste caso é o projeto utilizado como tutorial.

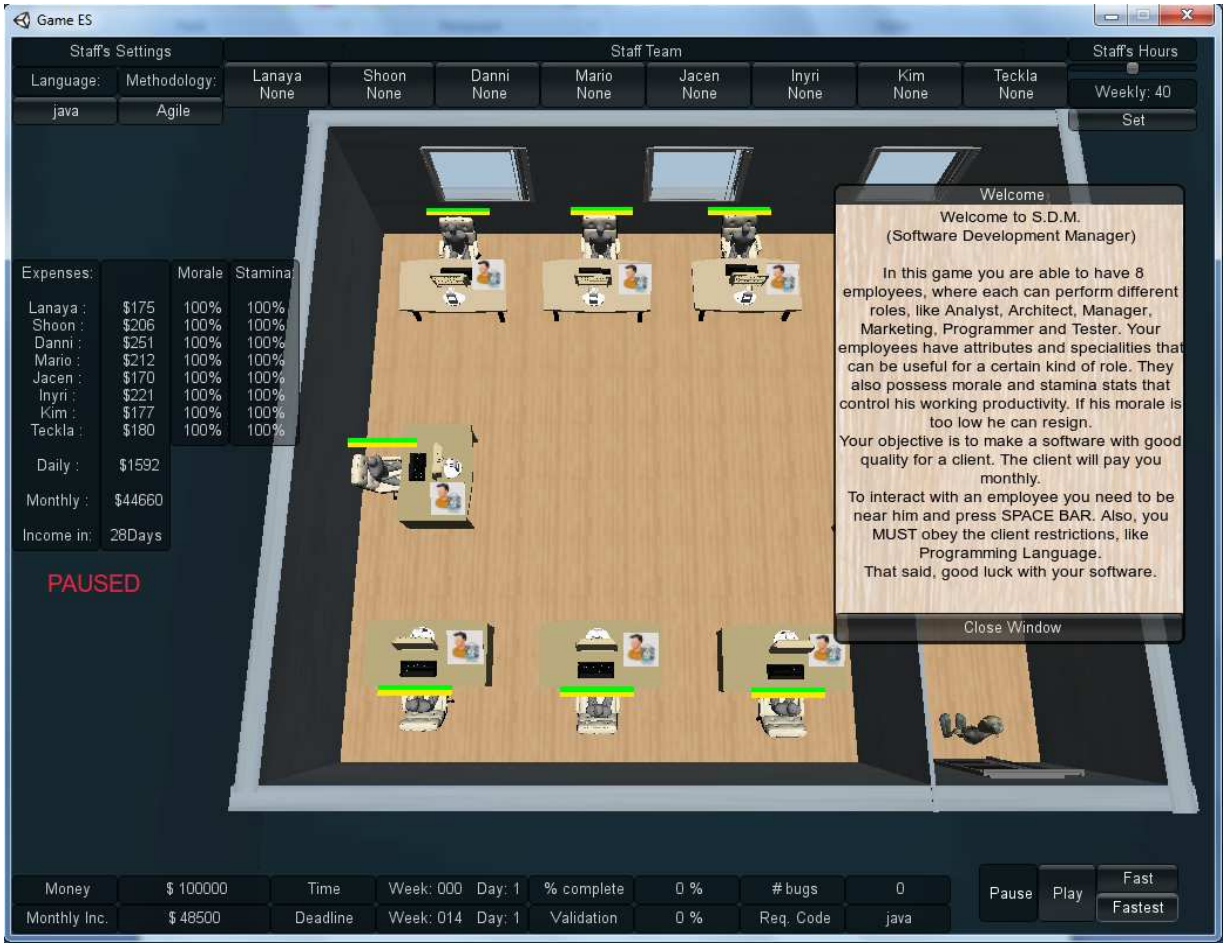

Figura 12- Interface do SDM

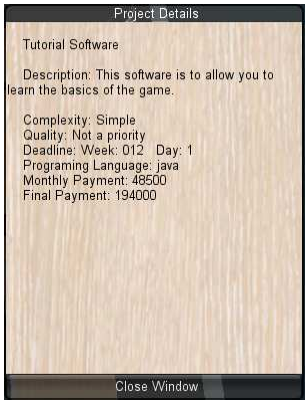

Figura 13- Informações do Projeto

É essencial que o jogador observe os requisitos do produto que vai ser desenvolvido. Estes requisitos são exibidos na janela de informações do projeto, mas caso o jogador se esqueça, ele pode consultar estes requisitos utilizando a interface. Estes requisitos são informados na parte inferior da interface, como é ilustrado na Figura 14.

Figura 14- Interface inferior do SDM

É possível perceber que além dos requisitos, existem outros elementos contidos nesta seção da interface. No canto direito, estão os botões que permitem alterar a velocidade da passagem de tempo. A escala utilizada no jogo para passagem de tempo no modo normal consiste em mapear seis segundos no tempo real a um dia no jogo. No modo rápido e mais rápido a passagem do tempo é mapeada em quatro e dois segundos, respectivamente. Além do controle da passagem do tempo, no canto esquerdo é informada a situação financeira do jogador e a sua renda mensal gerada pelo projeto. Ao lado das informações financeiras, encontram-se os dados do software corrente que está sendo desenvolvido. Dentre eles estão: o dia atual no jogo; o *deadline* de entrega; quanto já foi desenvolvido; a porcentagem de validação do modelo do software em relação ao que foi solicitado pelo cliente; o número de *bugs* no software; e a linguagem de programação solicitada pelo cliente.

Ainda na interface, na parte superior encontram-se os elementos configuráveis pelo o jogador, como é mostrado na Figura 15. Começando pelo lado esquerdo, tem-se as opções de linguagem de programação que é adotada pela a equipe para o desenvolvimento do software, bem como a metodologia de trabalho adotada pela equipe. A linguagem deve ser

compatível com a que foi requisitada pelo cliente. A metodologia de trabalho e a linguagem são alteradas conforme mostrado na Figura 16. Seguindo adiante, são informados os funcionários pertencentes à equipe do jogador, exibindo os nomes e seus respectivos papéis. Nesta parte da interface é possível fazer alterações no papel do funcionário, na quantidade de horas de trabalho, exibir a ficha do funcionário e a possibilidade de colocar o funcionário em treinamento. O canto direito apresenta a opção de alterar a hora de trabalho de todos os funcionários da equipe.

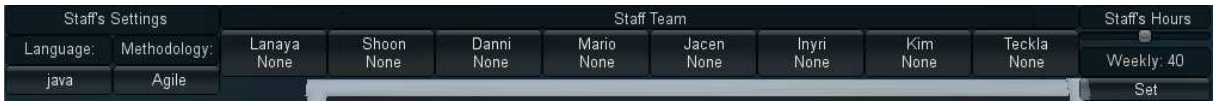

Figura 15- Interface Superior do SDM

Figura 16-Listas de opções na interface do usuário

No canto esquerdo da tela encontra-se o último elemento da GUI do SDM. Esta seção é responsável em informar ao jogador dados de cada funcionário pertencente à equipe. Estes dados se referem ao salário diário, o moral e a estamina dos funcionários. Abaixo destas informações e de acordo com estes dados, é exibido o gasto total com os funcionários diariamente e mensalmente. Esta janela também exibe a quantidade de dias restantes até o jogador receber o dinheiro mensal do cliente. A Figura 17 ilustra este elemento da interface.

Figura 17- Janela de Resumo sobre a equipe

Como foi mencionado, é permitido fazer alterações nos funcionários através da interface do usuário. Há outra maneira alternativa de se fazer isto, mediante a interação com o funcionário que deseja fazer as modificações. Esta interação é realizada através de uma janela de dialogo que contém opções das ações que o jogador pode realizar com o funcionário com

quem esta interagindo. Diálogos também são utilizados como uma forma de *feedback*. O funcionário, através de diálogos, pode manifestar a insatisfação no trabalho, de acordo com o estado do moral. A Figura 18 ilustra a forma alternativa de manipulação do funcionário, através do dialogo de interação entre o jogador e um funcionário.

Figura 18- Janela de dialogo

Como é possível perceber, a caixa de dialogo possui as mesmas opções que são exibidas na interface, representada na Figura 16. Porém, nos diálogos existe uma quinta opção. Esta opção só é visível na interação com o funcionário e varia de acordo com o papel que o mesmo estiver executando. Na Figura 18, o funcionário ilustrado é um gerente, que como consequência exibe a opção de contratação de novos integrantes para a equipe. Além do gerente, o arquiteto e o marketing possuem opções específicas, que são de criação de protótipo e de negociação, respectivamente.

Continuando na apresentação das interfaces e seguindo a ordem dos itens que aparecem na Figura 18, o próximo item a ser apresentado é a ficha do funcionário. A janela que exibe esta ficha, conforme pode ser visto na Figura 19, apresenta todas as informações, que são visíveis ao jogador referentes ao funcionário. No lado superior esquerdo são apresentadas as informações de nome, moral, estamina, papel, cargo, horas que ele trabalha semanalmente e quanto ele recebe por mês e por dia de trabalho. Do lado direito, são mostrados os valores de cada atributo do funcionário. Finalizando esta janela, na parte inferior encontram-se as especializações do funcionário.

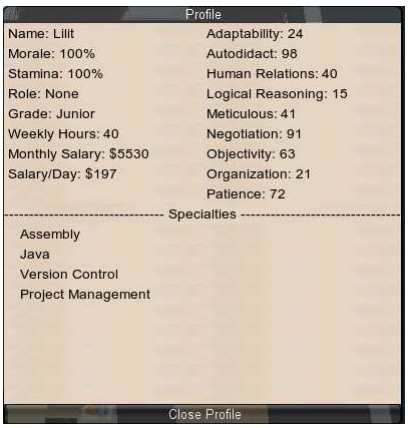

Figura 19- Ficha do funcionário

O próximo item a ser apresentado é de treinamento. Nesta janela, encontram todas as opções de treinamento que podem ser escolhidas para o funcionário. Estas opções são organizadas por tipo, que são: linguagens de programação, métodos de trabalho e ferramentas. Nesta mesma janela, também é possível ver quais são as especializações que o funcionário já possui. A Figura 20 exibe a janela de treinamento de funcionários.

Figura 20- Janela de treinamento dos funcionários

A próxima janela é a de tarefas. Nesta janela, que aparece no canto superior direto, é possível alterar e escolher o papel desempenhado pelo funcionário, dentre as seis escolhas possíveis: analista, arquiteto, gerente, marketing, programador e testador. O canto esquerdo possui as opções de alterar o cargo do funcionário para junior, pleno ou sênior, e a opção de demitir o funcionário selecionado. A Figura 21 exibe esta janela.

Figura 21- Janela de alteração de tarefas dos funcionários

O próximo item é a janela de alteração de horas de trabalho do funcionário. Nesta janela é possível alterar a quantidade de horas que o funcionário trabalha por semana através de uma barra de movimento, como mostrado na Figura 22. Apesar de não ser exibido nesta janela, quando a quantidade de horas de trabalho do funcionário é alterada, o jogador esta alterando indiretamente a estamina do funcionário.

Figura 22- Janela de configuração das horas de trabalho

Além destas interfaces, o jogo possui janelas para configuração de papéis. Estas janelas são apresentadas na seguinte ordem: contratação, negociação e prototipação.

A opção de contratação, e sua respectiva janela, só estão disponíveis caso o funcionário que o jogador esteja interagindo desempenhe o papel de gerente. A janela de contratação é utilizada para alterar a equipe de funcionários do jogador, podendo acrescentar ou substituir integrantes. No canto esquerdo da janela é apresentada uma listagem dos possíveis candidatos a serem contratados, com seus respectivos custos de contratação. Quando um candidato é selecionado, ao lado da janela é exibida uma janela contento as informações do funcionário. O canto direto da janela de contratação exibe a equipe atual do jogador. Para contratar algum candidato, o jogar deve selecionar uma posição da lista do canto direito para o novo integrante. Caso esta posição já esteja sendo utilizada por algum funcionário, este funcionário é substituído pelo novo integrante. A Figura 23 ilustra esta janela de contratação e a janela de informações do candidato.

Figura 23-Janela de Contratação

A próxima opção é a de negociação, onde só é permitido ser realizada por um funcionário que esteja executando o papel de marketing. Esta opção abre uma janela de negociação, que contém os quatro aspectos negociáveis: escopo, tempo, recurso e qualidade. A janela de negociação é dividida em duas partes: a superior, que permite ao jogador escolher

qual aspecto ele deseja alterar para se beneficiar, e a parte inferior, onde o jogador escolhe qual aspecto é compensado pela alteração. Na atual versão do jogo, a alteração só pode ser feita de um para um, ou seja, o jogador escolhe o aspecto desejado e outro que é compensado. A Figura 24 exibe a janela de negociação.

Figura 24- Janela de negociação

Finalizando as opções de diálogo, o ultimo item é a prototipação. Esta opção, assim como as demais, é uma opção especializada onde se requer um funcionário que esteja exercendo o papel de arquiteto. Esta janela permite a criação de protótipos. Para isto, o jogador precisa escolher entre os tipos de protótipos que o arquiteto pode fazer. Estes tipos são referentes apenas à complexidade, podendo ser protótipos simples, regulares ou complexos. Cada tipo possui um custo diferente e um resultado diferenciado que afeta a validação do modelo. A Figura 25 ilustra a janela de prototipação com os três diferentes tipos de protótipos existentes no SDM.

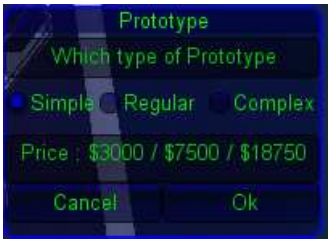

Figura 25- Janela de Prototipação

Todas as opções de dialogo possíveis já foram explicadas. Agora é apresentado como a conclusão de um projeto é realizada e avaliada, seguida da janela de escolha de um novo projeto. Quando um projeto é finalizado, existem duas possibilidades: conclusão (o jogador conseguiu terminar o projeto a tempo) e fracasso (o jogador não conseguiu finalizar o projeto). Para ambos os casos, uma janela é exibida contendo as estatísticas do projeto. A Figura 26 mostra o caso de conclusão do projeto.

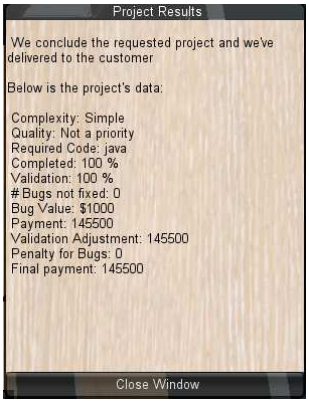

Figura 26- Janela de conclusão do projeto

Logo depois da janela de finalização do projeto, é exibida ao jogador a janela de seleção de um novo projeto. Nesta janela o jogador escolhe, a partir de uma lista de opções, um novo projeto a ser desenvolvido. Quando um projeto é selecionado nesta janela, outra janela será aberta contendo as informações do projeto selecionado. Uma vez que um projeto estiver selecionado, este novo projeto se torna o próximo produto a ser desenvolvido pelo jogador, sendo a equipe mantida para trabalhar nesse novo projeto. Os mesmos conceitos que foram explicados para o primeiro projeto valem para os demais projetos. A Figura 27 exibe a janela de escolha de projetos e ao lado dela um trecho da janela que contem os detalhes.

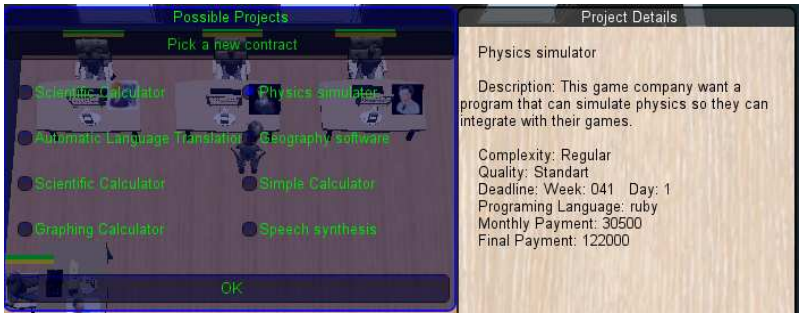

Figura 27- Janela de escolha de projetos

A Figura 28 mostra os funcionários da equipe trabalhando no software. Como é possível perceber, números aparecem sobre a mesa dos funcionários, representando o progresso das tarefas que estão desempenhando. Além do progresso, também são exibidas alterações nos estados do moral e estamina de cada funcionário, por números e por barras. A barra verde representa o moral e a barra amarela a estamina. Estas barras reduzem, aumentam e alteram de cor dependendo dos estados do moral e estamina. Na Figura 28 é possível ver um ganho de estamina devido à alteração das horas de trabalho da equipe.

Figura 28- Funcionarios trabalhando

4.3 AVALIAÇÃO PRELIMINAR

A fim de avaliar o jogo desenvolvido, foram selecionados arbitrariamente seis alunos da UFF para jogarem o SDM. O objetivo deste experimento foi investigar qual seria a percepção dos alunos com o jogo e avaliar se conseguiram apreender os conteúdos de ES apresentados. Para isto, foi elaborado um questionário de avaliação que é divido em duas partes: a primeira parte, referente à caracterização do participante, e a segunda, sobre a avaliação do jogo. As perguntas do questionário são divididas em duas categorias: objetivas e discursivas. A Tabela 1 apresenta o resumo das perguntas objetivas do questionário.

Tabela 1- Questionario objetivo

Pergunta	Opção 1	Opção 2	Opção 3	Opção 4	Opção 5	Opção 6
Formação acadêmica?	Doutorado	Doutorando	Mestrado	Mestrando	Graduação	Graduando
Experiência em ES?	Nunca aprendeu	Já leu sobre	Cursando	Fez disciplina	Participei de um curso	-
Qualidade do Jogo?	Detestou	Não gostou	Indiferente	Gostou	Gostou muito	-
Quanto à jogabilidade?	Incompreensível	Complicado	Normal	Fácil	Muito fácil	-
Aprendeu algo novo?	Não	Sim	-	-	-	-
Entre jogo e aula?	Aula	Jogo	Aula e jogo	-	-	-
Gostaria de jogar novamente?	Não	Sim	-	-	-	-
Despertou interesse?	Não	Sim	-	-	-	-

Dentre as pessoas selecionadas, 67% estão na graduação, 17% fazendo mestrado e 16% fazendo doutorado. Neste grupo, 83% já fizeram uma disciplina de ES e 17% estão cursando. Da amostra avaliada, todos os participantes gostaram de ter jogado o jogo. Em relação à jogabilidade, 34% acharam fácil, 17% acharam que o jogo é complicado e os demais que o jogo está normal em termos de jogabilidade. Referente à aprendizagem sobre os conteúdos de ES, 50% dos participantes afirmaram ter aprendido algo novo com o jogo, enquanto os demais não aprenderam nenhum conteúdo novo. Dos participantes, quando perguntados se gostariam de jogar novamente, 83% responderam positivamente, justificando ser possível aplicar os conhecimentos de ES e pela maneira que o gerenciamento no jogo é realizado. Os demais, 17%, afirmaram que não têm interesse de jogar novamente, porém indicariam o jogo para alunos que estivesse cursando uma disciplina de ES. Após terem experimentado o jogo, 50% afirmaram que o SDM despertou interesse em ES. Estes dados são apresentados na Tabela 2.

Tabela 2- Avaliação do jogo

Pergunta	Opção 1	Opção 2	Opção 3	Opção 4	Opção 5	Opção 6
Formação acadêmica?	0%	16%	0%	17%	0%	67%
Experiência em ES?	0%	0%	17%	83%	0%	-
Qualidade do Jogo?	0%	0%	0%	67%	33%	-
Quanto à jogabilidade?	0%	17%	49%	34%	0%	-
Aprendeu algo novo?	50%	50%	-	-	-	-
Entre jogo e aula?	17%	0%	83%	-	-	-
Gostaria de jogar novamente?	17%	83%	-	-	-	-
Despertou interesse?	50%	50%	-	-	-	-

Quanto ao conteúdo, os participantes conseguiram identificar conceitos de ES referentes a planejamento, monitoramento, codificação, validação, testes, gestão de pessoas,

negociação e metodologias de desenvolvimento. Além destes conceitos, a função e importância desempenhada por cada papel também foi identificada. De acordo com os participantes, o aspecto mais marcante durante a experiência com o jogo foi a gestão de pessoas. Quando perguntados se os conceitos apresentados ajudaram no entendimento, todos os participantes responderam positivamente.

5 CONCLUSÃO

Esta monografia apresentou uma nova abordagem de ensino de ES, que visa auxiliar no aprendizado e fixação do conteúdo apresentado durante as aulas teóricas. Como discutido anteriormente, este trabalho envolveu, além desta monografia, a implementação do jogo SDM e sua avaliação preliminar. As principais contribuições deste trabalho como um todo estão no entendimento e aprendizado do conteúdo apresentado.

Durante interações com o jogo, foi identificado que ele auxilia no entendimento do conteúdo que é apresentado no SDM. Os conteúdos de ES vistos em aulas teóricas, quando aplicados em situações reais ou simulação ajudam no entendimento e esclarecimento. Além disto, situações reais ou simuladas demonstram a importância destas técnicas e suas vantagens e desvantagens em sua utilização em situações reais.

Diferente dos demais jogos de ES citados nesta monografia, o SDM demonstra a importância de cada papel desempenhado no processo de desenvolvimento do software. Estes papéis exercem tarefas distintas que são importantes na etapa de desenvolvimento. Através de elementos da interface, o jogador recebe informações destas tarefas e consegue aprender a importância de cada papel e suas respectivas responsabilidades.

A partir da análise critica sobre a abordagem proposta e da implementação do jogo, podem ser identificadas limitações. Algumas destas limitações, que se baseiam nas decisões tomadas durante a implementação do SDM, são detalhadas a seguir.

Na atual versão do SDM não é possível definir uma iteração. Diariamente é fornecido ao jogador um *feedback* do rendimento de cada funcionário. Pelo fato de ser informado diariamente, o jogador pode ficar desorientado em relação à avaliação do ritmo de desenvolvimento do projeto. Para contornar este problema, poderia ser implementado uma opção que permita o jogador escolher o tamanho das iterações, que informariam o rendimento da equipe e de cada funcionário individual dentro do período escolhido.

A forma em que as metodologias são apresentadas no SDM só demonstra que existem dois diferentes grupos: clássicos e ágeis. Não é transmitida nenhuma informação sobre as características de cada metodologia. Uma alternativa para solucionar este problema seria a implementação de janelas de informações, onde conteriam a listagem dos aspectos característicos de cada metodologia.

A concepção desta nova abordagem levou a construção de um jogo de ensino de ES. Apesar da versão atual do SDM ser funcional, ainda existem diversos aspectos que podem ser melhorados. Alguns desses trabalhos futuros são discutidos a seguir.

Uma ideia proposta durante a concepção do jogo era de incluir um sistema de afinidade entre os integrantes da equipe do jogador. Este sistema afetaria a integração de novos funcionários na equipe, fazendo com que o jogador refletisse na decisão de substituir um funcionário da equipe por outro no meio do desenvolvimento do software. O novo integrante da equipe precisaria, além de passar por treinamentos, interagir com os demais integrantes para saber o modelo de trabalho em que a equipe se baseia. Este modelo não se refere à metodologia, mas sim aos padrões de trabalho adotados pela equipe.

Outra ideia é de aprofundar ainda mais nas metodologias de trabalho, os métodos clássicos e ágeis. Na atual versão do jogo, as metodologias são apenas utilizadas para penalizar os integrantes que não sabem trabalhar na metodologia adotada. Isto pode ser mais bem trabalhado, expandindo estas metodologias para os modelos de desenvolvimento que cada um possui, como, por exemplo, o modelo cascata e *scrum*. Utilizando esta expansão nas metodologias, poderiam ser incluidos detalhes das características destes modelos com o intuito de afetar no desenvolvimento do software e informar ao jogador as diferenças entre os modelos de desenvolvimento.

Na atual versão do SDM, o moral é afetado por alguns fatores que já foram discutidos. Uma proposta para ser utilizada em trabalhos futuros é fazer uma ligação do cargo do funcionário com o moral. Esta ligação seria do tipo de insatisfação que ocorreria quando um funcionário passasse um grande período de tempo sem ter uma promoção. Ou seja, depois de certo tempo exercendo o cargo, o funcionário começaria a exibir insatisfação, pois ele não estaria evoluindo hierarquicamente dentro da empresa.

Outra proposta que foi incluída durante a concepção inicial do jogo, porém não foi implementada, é a inserção de mais um fator de dificuldade no jogo. Este fator seria referente à saúde do funcionário, podendo fazer com que seja possível o funcionário ficar doente. O aspecto de saúde introduziria um elemento de incerteza na etapa de desenvolvimento e forçaria o jogador a pensar em planos de contingência para tais situações.

O item de negociação presente no jogo também poderia ser alterado, tornando-o mais complexo. Esta alteração permitiria ao jogador escolher diversos itens a sofrerem

alterações no contrato, com o intuito de distribuir o peso de mudança, transmitindo a ideia que todos os aspectos são afetados com a modificação.

Um sistema de reputação para a empresa do jogador foi pensado, mas não foi incluído no jogo. Este sistema afetaria nas decisões de escolha de novos contratos de projetos de acordo com a reputação. Sendo assim, caso a empresa do jogador possuísse uma reputação positiva, projetos de empresas multinacionais estariam disponíveis para o jogador, que forneceriam mais recursos financeiros. E caso a reputação fosse negativa, apenas pequenas empresas desejariam contratar o serviço, fornecendo recursos mais limitados e projetos mais simples e menores.

Juntamente com o sistema de reputação, tem-se a ideia de incluir a possibilidade do jogador possuir vários ambientes de trabalho, ou seja, escritórios. Com essa possibilidade, o jogador poderia ter varias equipes trabalhando em projetos distintos ou até mesmo contribuindo para o mesmo projeto. Estes escritórios poderiam ser comprados pelo próprio jogador, alugados ou até mesmo fornecidos ao jogador enquanto estiver desenvolvendo um produto para um determinado cliente.

6 REFERENCIAS BIBLIOGRAFICAS

BAKER, A.; NAVARRO, E. O.; HOEK, A. VAN DER. Problems and Programmers: An Educational Software Engineering Card Game. In: ICSE, 2003. p. 614-621.

DANTAS, A.; BARROS, M. DE O.; WERNER, C. M. L. Treinamento Experimental com Jogos de Simulação para Gerentes deProjeto de Software. In: SBES, 2004.

ESTUBLIER, J. Software Configuration Management: a roadmap. In: ICSE, 2000.

FIGUEIREDO, E.; LOBATO, C.; DIAS, K.; LEITE, J.; LUCENA, C. Um Jogo para o Ensino de Engenharia de Software Centrado na Perspectiva de Evolução. In: SBC, 2007.

FIGUEIREDO, K.; FERREIRA, J.; MURTA, L.; CLUA, E. Jogo de Estratégia de Gerência de Configuração. In: III Fórum de Educação em Engenharia de Software, 2010.

GAMES, F. **Game Dev Story**. Disponível em: <http://itunes.apple.com/us/app/game-dev-story/id396085661?mt=8>. Acesso em: 5 mai. 2011.

HIGGINS, T. **UNITY: Game Development Tool**. Disponível em: <http://unity3d.com/>. Acesso em: 5 mai. 2011.

KOHWALTER, T. **SDM**. Disponível em: <http://gems.ic.uff.br/sdm/>. Acesso em: 13 jun. 2011.

NAVARRO, E. O. SimSE: A Software Engineering Simulation Environment for Software Process Education. In: ICS, 2002.

PRENSKY, M. **Fun, Play and Games: What Makes Games Engaging**. In: Digital Game-Based Learning, 2001a.

PRENSKY, M. Digital Natives Digital Immigrants. In: On the Horizon, 2001b. v. 9.

PRENSKY, M. The Motivation of Gameplay. In: On The Horizon, 2002. v. 10.

QUINN, C. N. **Engaging Learning**. In: Pfeiffer, 2005.

RUSSO, R. DE F. S. M. Tendência empreendedora do gerente. In: Gest. Prod., 2007. v. 14, p. 581-593.

SANTOS, S. C. G. Psicologia em Estudo - The information technology professionals and their personality analyzed by Rorschach technique. In: Psicol. estud., 2005. v. 10.

TERRIBILI, A. Projetos com equipes remotas: mais uma dificuldade para o gerente. Artigonal, 2010.

ZYDA, M. From Visual Simulation to Virtual Reality to Games. In: IEEE Computer, 2005. v. 38, p. 25-32.

APENDICE A

Formulário de Consentimento

Estudo

Este estudo visa avaliar o quanto as técnicas de Engenharia de Software, informadas através do jogo SDM, são consistentes e beneficentes para o aprendizado, tanto para um aluno experiente quanto para um aluno novo no assunto.

Idade

Eu declaro ter mais de 18 anos de idade e concordar em participar de um estudo conduzido por Troy Costa Kohwalter na Universidade Federal Fluminense.

Procedimento

Este estudo acontecerá em uma única sessão, que incluirá uma sessão do jogo SDM. Eu entendo que, uma vez que o experimento tenha terminado, os trabalhos que desenvolvi serão estudados visando entender a eficácia do jogo proposto.

Confidencialidade

Toda informação coletada neste estudo é confidencial, e meu nome não será divulgado. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do experimento.

Benefícios e liberdade de desistência

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e apresentado. Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada à minha pessoa não seja incluída no estudo. Eu entendo que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas de ensino para a Engenharia de Software.

Pesquisador responsável

Troy Costa Kohwalter

Instituto de Computação – Universidade Federal Fluminense (UFF)

Professores responsáveis (Orientadores)

Prof Leonardo Paulino Murta

Instituto de Computação – Universidade Federal Fluminense (UFF)

Prof Esteban W. Gonzalez Clua

Instituto de Computação – Universidade Federal Fluminense (UFF)

Nome:_____

Assinatura: _____

Data:___/___/___

Este formulário está dividido Formulário de Consentimento e um Questionário que possui 14 questões. Após reponde-las entregue ao responsável presente.

Desde já, agradecemos a sua disponibilidade.

APENDICE B

Questionário de Caracterização do participante

1) Formação Acadêmica

( ) Doutorado

( ) Doutorando

( ) Mestrado

( ) Mestrando

( ) Graduação

( ) Graduando

Ano de ingresso: _____ Ano de conclusão (ou previsão de conclusão): _____

2) Formação Geral

a. Qual é sua experiência em Engenharia de Software? (marque aqueles itens que melhor se aplicam)

( ) Nunca aprendi Engenharia de Software.

( ) Já li material sobre Engenharia de Software.

( ) Estou fazendo uma disciplina sobre Engenharia de Software.

( ) Já participei de um curso sobre Engenharia de Software.

( ) Já fiz uma disciplina sobre Engenharia de Software.

3) Qualidade do jogo

a. Após ter jogado SDM, você diria que:

( ) Detestou.

( ) Não gostou.

( ) Ficou indiferente.

( ) Gostou.

( ) Gostou muito.

b. Quanto à jogabilidade, você diria que o SDM é:

( ) Incompreensível.

( ) Complicado.

( ) Normal.

( ) Fácil.

( ) Muito fácil.

c. Você aprendeu algo de ES com o jogo?

( ) Não.

( ) Sim.

d. Entre o jogo e uma aula, você prefere:

( ) Aula.

( ) Jogo.

( ) Aula e jogo.

e. Você gostaria de jogar SDM novamente?

( ) Não.

( ) Sim.

Por quê? _____

_____

f. O jogo despertou seu interesse por Engenharia de Software?

( ) Não.

( ) Sim.

4) Avaliação do conteúdo

a. Enquanto jogava o SDM, cite os conceitos de Engenharia de Software que conseguiu identificar dentro do contexto do jogo.

_____

_____

b. Dentre os conceitos apresentados, teve algum que se destacou mais? Por quê?

_____

_____

c. Da maneira que foram apresentados, estes conceitos que foram identificados te ajudaram no entendimento? Conseguiu perceber sua utilização e suas vantagens e desvantagens?

_____

_____

d. Caso algum conceito apresentado não lhe ajudou no entendimento, por favor, indique e justifique.

_____

_____

e. Você achou que, no SDM, faltou algum conceito de Engenharia de Software que seria útil no contexto? Caso positivo, qual?

_____

_____

f. Após ter jogado o SDM, que sugestão você daria para o melhoramento do jogo, em relação à abordagem dos conteúdos?

_____

_____