BRUNO CÉSAR DE OLIVEIRA

TestEG - UM SOFTWARE EDUCACIONAL PARA O ENSINO DE TESTE DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS - BRASIL
2013

BRUNO CÉSAR DE OLIVEIRA

TestEG - UM SOFTWARE EDUCACIONAL PARA O ENSINO DE TESTE DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:
Engenharia de Software

Orientador:
Prof. Dr. Heitor Augustus Xavier Costa

LAVRAS
MINAS GERAIS - BRASIL
2013

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca
Central da UFLA

Oliveira, Bruno César de

TestEG - Um Software Educacional para o Ensino de Teste de Software / Bruno César de Oliveira. Lavras - Minas Gerais, 2013. 80 p.

Monografia de Graduação - Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Software Educacional. 2. Teste de Software. 3. Engenharia de Software. I. OLIVEIRA, B. C. de. II. Universidade Federal de Lavras. III. TestEG - Um Software Educacional para o Ensino de Teste de Software.

BRUNO CÉSAR DE OLIVEIRA

TESTEG - UM SOFTWARE EDUCACIONAL PARA O ENSINO DE TESTE DE SOFTWARE

Monografia apresentada ao Colegiado do Curso de Ciência da Computação, para a obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 16 de Abril de 2013.

Prof. Dr. Rêmulo Maia Alves	UFLA
Profa. Dra. Ana Paula Piovesan Melchiori	UFLA

Prof. Dr. Heitor Augustus Xavier Costa

(Orientador)

LAVRAS – MG

2013

Agradecimentos

À Deus em primeiro lugar, por ser meu guia e mantenedor, à Universidade Federal de Lavras (UFLA) e o Departamento de Ciência da Computação (DCC), pela oportunidade de realização da graduação.

Aos Professores do Departamento de Ciência da Computação da UFLA, pelos ensinamentos transmitidos.

Ao professor Dr. Heitor Augustus Xavier Costa ela orientação, paciência, amizade, dedicação e seus ensinamentos que foram de grande relevância para a realização deste trabalho e crescimento profissional.

Aos amigos graduandos Sérgio Henrique Bento Mira, Paulo Roberto de Almeida e Alan Henrique Gabriel, pela preciosa ajuda no desenvolvimendo do projeto.

SUMÁRIO

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

TestEG - Test Educational Game

RESUMO

O processo de ensino-aprendizagem em Engenharia de Software tem passado por questionamentos. Um dos grandes desafios enfrentados no ensino de Engenharia de Software é suprir a necessidade de utilizar métodos de ensino que permitam tornar esse processo mais efetivo. As exigências por sistemas de software com alto padrão qualidade têm motivado a definição de métodos e técnicas para o seu desenvolvimento a fim de atingir esses padrões. Não há garantias que um sistema funcionará isento de erros, o seu projeto varia em nível de complexidade aumentando ou diminuindo a probabilidade de erros. Neste trabalho, o objetivo é apresentar o desenvolvimento de um jogo educacional computacional na área de Teste de Software para minimizar os problemas encontrados no processo de ensino-aprendizagem de alguns assuntos abordados nessa área. Algumas análises sobre engines para jogos foram realizadas e, ao final, optou-se pela utilização da Unity3D. Para obter informações sobre a relevância da utilização do TestEG no processo de ensino-aprendizagem em teste de software, foi realizada uma avaliação por meio de aplicação de um questionário com o propósito de verificar características como aceitação, jogabilidade e satisfação. Com resultados obtidos, tem-se indícios que o TestEG adequa-se às exigências de um software educacional, além de mostrar-se uma ferramenta efetiva no apoio ao ensino em Teste de Software, contribuindo com o processo de ensino-aprendizagem.

Palavras-chave: Teste de software, Jogos Educacionais, Engenharia de Software

TestEG - Test Educational Game

ABSTRACT

The teaching and learning process in software engineering has undergone questioning. One challenge in the teaching software engineering is to fulfill the need to use teaching methods that will make this process more effective. The requirements for software with high quality have

motivated the development of methods and techniques for its development. There is no guarantee that a system will operate error-free, its design varies in level of complexity by increasing or decreasing the likelihood of errors. In this work, the goal is to present the development of an educational game in the area of Software Testing to minimize the problems encountered in the teaching and learning of certain subjects covered in this area. Some analyzes of game engines were held and we opted to use the Unity3D. An evaluation was made using a questionnaire for verifying acceptance, playability, and enjoyment. With obtained results, there are evidences that the TestEG suits the requirements of an educational software, besides showing an effective tool in supporting education in Software Testing, contributing to the process of teaching and learning.

Keywords: Software Testing, Educational Games, Software Engineering.

1. INTRODUÇÃO

O processo de ensino-aprendizagem em Engenharia de Software tem passado por questionamentos [Hilburn; Towhidnejad, 2007]. Em geral, o treinamento tradicional para capacitação de gerentes de projetos de software nas universidades e em cursos de formação específica é realizado por meio de aulas expositivas, centradas no professor [Paludo; Raabe, 2010]. Quando expostos às situações encontradas na indústria, os recém-formados deparam-se com cenários nos quais técnicas e métodos aprendidos são pouco aplicados [Nauman; Uzair, 2007]. A utilização da tecnologia acelerou, modificou e acentuou o ritmo e a forma como o mundo pode ser visto e como as pessoas comunicam-se. Seguindo esse ritmo, a maneira como o conhecimento é transmitido tem sofrido importantes modificações.

Um dos grandes desafios enfrentados no ensino de Engenharia de Software é suprir a necessidade de utilizar métodos de ensino que permitam tornar o processo de ensino-aprendizagem mais efetivo [Santos et al., 2008]. Uma alternativa para suplantar essa necessidade nos métodos tradicionais de ensino é proporcionada pela utilização de jogos educativos [Baker et al., 2005; Prensky, 2001]. Apesar de esforços em pesquisas e em desenvolvimento desses jogos, sua utilização para o ensino de Engenharia de Software ainda não é um recurso comum no ambiente acadêmico. Esses jogos são desenvolvidos para terem fundamentos educacionais e elementos pertinentes, tais como, contexto, objetivos, regras, feedback, competição e interação [Prensky, 2001].

As exigências por sistemas de software com alto padrão qualidade têm motivado a definição de métodos e técnicas para o seu

desenvolvimento a fim de atingir esses padrões. Com isso, o interesse pela atividade de teste de software vem aumentando nos últimos anos. Essa atividade consiste em uma análise dinâmica do sistema de software e relevante para a identificação e eliminação de erros de implementação, que persistem ao final do projeto [Delamaro et al., 2007]. O teste de software, realizado após a sua integração, visa à identificação de erros nas funções e de características de desempenho em desacordo com as especificações [Pressman, 2009]. Os custos das atividades de produção de um sistema de software estão fortemente ligados às atividades de teste, daí vê-se a importância de ter bons métodos de ensino para essas atividades.

O presente trabalho tem como principal propósito apresentar o desenvolvimento de um jogo educacional computacional, cuja finalidade é auxiliar alunos no processo de ensino-aprendizagem em assuntos pertinentes a Teste de Software. Além disso, nesse jogo, visa-se permitir a familiarização com um ambiente real de trabalho, por meio de tarefas providas no decorrer do jogo e, quando executadas, agregarem conhecimento aos jogadores.

1.1. Motivação

Não há garantias que um sistema de software, ao final de sua construção, funcionará isento de erros, o seu projeto varia em nível de complexidade aumentando ou diminuindo a probabilidade de erros [Myers, 2004]. Essa complexidade pode ser elevada ainda mais de acordo com o tamanho do projeto, a quantidade de pessoas envolvidas e os tipos de sistemas de software. Ao realizar um teste, as suas diretrizes deveriam ser testadas, no entanto, na prática, isso pode não ser viável, pois as possibilidades de execução de software são imensas. Assim, deve-se ter profissionais capacitados para os testes realizados serem filtrados pelas permutações mais relevantes [Myers, 2004].

A falha em um sistema de software custa caro, traz prejuízos financeiros às empresas e as consequências podem ser incalculáveis. Além disso, ela pode afetar a confiabilidade dos clientes, que, após sofrerem a frustração de utilizar um sistema com falhas, optarão por empresas que se mostrem de maior confiança. O SEI[1] (Software Engineering Institute) informa que: i) 30% dos projetos são cancelados antes de serem finalizados; ii) 70% dos projetos falham nas entregas das funções esperadas; iii) Custos dos projetos extrapolam mais de 180% dos valores previstos; e iv) Prazos excedem mais de 220%.

O SEI reitera que esses percentuais variam de acordo com os medidas de qualidade adotadas, evidenciando que modelos de qualidade e técnicas de testes são importantes para o sucesso do projeto. No contexto de teste de software, deve haver aplicação dos conceitos aprendidos em sala de aula da melhor maneira possível, para que os testes possam ser realizados com êxito no mercado de trabalho. Por isso, entende-se que o estudo de teste de software deve abranger conceitos e sua aplicação para que o aluno enfrente problemas/situações reais e adquira experiência para realizar a tarefa da forma mais adequada. Dessa forma, o processo de ensino-aprendizagem deve ser apoiado por uma ferramenta, por exemplo, um software educacional para simular situações encontradas na indústria, para complementar a formação do aluno.

1.2. Objetivo

Neste trabalho, o objetivo é apresentar o desenvolvimento de um jogo educacional computacional na área de Teste de Software a fim de minimizar os problemas encontrados no processo de ensino-aprendizagem de alguns assuntos abordados nessa área. Dessa forma,

[1] http://www.sei.cmu.edu/

espera-se que o aluno (jogador) tenha consolidado/fixado os conceitos abordados de maneira lúdica. Os objetivos específicos deste trabalho são:

- Aprofundar conhecimento em conceito, técnicas e processos de teste de software;
- Estudar a engine de jogos Unity3D[2];
- Entender a importância de utilizar jogos para auxiliar o processo de ensino-aprendizagem.

1.3. Metodologia de Desenvolvimento

A metodologia de pesquisa é um conjunto de métodos, técnicas e procedimentos cuja finalidade é viabilizar a execução da pesquisa, cujo resultado é um novo produto, processo ou conhecimento [Jung, 2009].

1.3.1. Tipos de Pesquisa

Uma pesquisa pode ser classificada em (Figura 1-1) [Jung, 2009]:

- Quanto a Natureza: i) Pesquisa Básica (gerar conhecimento sem finalidade de aplicação) e ii) Pesquisa Aplicada (gerar conhecimento com finalidade de aplicação);
- Quanto aos Objetivos: i) Exploratória (descobrir/inovar); ii) descritiva (como?); e iii) explicativa (por quê?);
- Quanto as Abordagens: i) Quantitativa e ii) Qualitativa;
- Quanto aos Procedimentos: i) Survey; ii) Pesquisa-Ação; iii) Estudo de Caso Único ou Múltiplos; iv) Operacional; e v) Experimental.

Além disso, os métodos para a coleta dos dados podem ser por meio de (i) observação do participante, (ii) grupos focados, (iii) entrevistas, (iv) questionário, (v) experimentação e (vi) observação.

[2] http://unity3d.com/

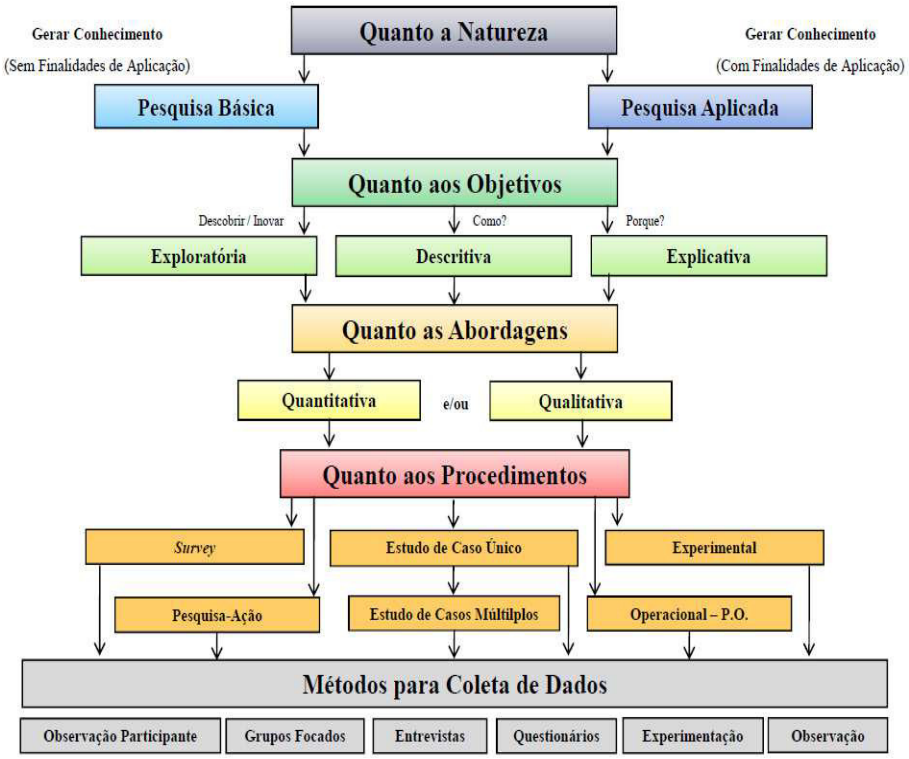

Figura 1-1 - Tipos de Pesquisa Científica [Jung, 2009]

A natureza deste trabalho pode ser classificada como **pesquisa aplicada**, pois, em seu desenvolvimento e em sua utilização, são utilizados conceitos de Teste de Software. Com relação aos objetivos, este trabalho pode ser classificado como **pesquisa descritiva**, pois observam, registram e analisam fenômenos conceituais de Teste de Software, no qual o aluno (jogador) desempenha o papel de um Gerente de Teste. No tocante a abordagem, este trabalho pode ser classificado como **pesquisa qualitativa e quantitativa**, pois tem como preocupação aperfeiçoar, melhorar o ensino em Teste de Software, além de apresentar inúmeros resultados, na forma de gráficos. Por fim, os procedimentos deste trabalho podem ser caracterizados como **pesquisa experimental**, visto que houve a implementação de um jogo e testes foram feitos para

realizar melhorias. A coleta de dados foi por meio de **experimentação** e resposta a um **questionário** pelos participantes na utilização do jogo.

1.3.2. Procedimentos Metodológicos

Para concretizar a realização deste trabalho, houve empenho para o entendimento de métodos envolvidos no processo de ensino-aprendizagem, mais especificamente em Engenharia de Software. Informações importantes quanto à formação dos alunos e sua inserção no mercado de trabalho foram analisadas, o que permitiu melhor identificação das necessidades dos futuros profissionais (egressos das Instituições de Ensino Superior na área de Computação atuantes em Engenharia de Software/Teste de Software).

A fundamentação teórica foi consolidada por meio de estudo de artigos obtidos dos repositórios de eventos/revistas científicos fornecidos pelo Periódicos/CAPES e de livros que abordam o assunto. Além disso, reuniões periódicas ocorreram entre o orientado e o orientador, durante o levantamento do estado da arte de jogos nas área de Engenharia de Software, mais especificamente sobre o assunto Teste de Software. Isso ocorreu, visando às correções e às trocas de informações a respeito do andamento do trabalho e ao auxílio nas tomadas de decisão. Tais reuniões foram imprescindíveis para a realização deste trabalho, pois, nas decisões, eram muitas as possibilidades, por exemplo, (i) qual área da Engenharia de Software escolher, (ii) dentro da área escolhida, qual trabalho desenvolver e (iii) quais ferramentas utilizar.

Algumas análises sobre engines para jogos foram realizadas e, ao final, optou-se pela utilização da Unity3D[3] para a implementação do jogo

[3] http://unity3d.com/

apresentado neste trabalho. Essa escolha foi em decorrência dessa engine ser uma ferramenta robusta e poderosa, contendo requisitos para a implementação do jogo, tais como, modelagem visual interativo e possibilidade de integração entre linguagens de alto nível e banco de dados. No decorrer do desenvolvimento do jogo, houve constante preocupação com sua manutenção e sua evolução, portanto a modelagem e a programação foram feitas de forma a facilitar futuros aperfeiçoamentos, alterações ou correções.

1.4. Estrutura do Trabalho

Este trabalho está organizado da seguinte forma.

Alguns trabalhos relacionados são brevemente descritos no Capítulo 2.

A utilização de jogos na educação, destacando a importância de métodos inovadores no processo de ensino-aprendizagem, descrevendo tipos de jogos utilizados no ensino e discorrendo sobre jogos computacionais no ensino de Engenharia de Software, é apresentada no Capítulo 3.

A importância de teste de software, técnicas existentes e sua aplicação são relatadas no Capítulo 4.

O funcionamento, a dinâmica e a modelagem do jogo são descritos no Capítulo 5.

A avaliação do jogo é apresentada no Capítulo 6.

Conclusões, contribuições, limitações e sugestões de trabalhos futuros são discutidas no Capítulo 7.

2. TRABALHOS RELACIONADOS

Neste capítulo, são apresentados alguns trabalhos relacionados, com o tema jogos para ensino de Engenharia de Software. Jogos educacionais são apontados como um tipo promissor de material instrucional, que pode trazer benefícios para a educação na área de Engenharia de Software. Esse jogos tendem a ser atrativos e motivadores aos alunos, propiciando aumento de seu desempenho ao despertar o interesse, curiosidade e contribuição positiva no processo de ensino-aprendizagem.

O Planeger [Kieling et al., 2006] foi desenvolvido para apoiar o ensino de conceitos de gerência de projetos, cujo objetivo é a simulação de alguns processos utilizados no gerência de escopo e de tempo. Esse jogo possui os usuários (i) administrador e (ii) jogador. O jogador pode utilizar o módulo tutorial para aprender sobre gerência de projetos e jogar em diversos cenários cadastrados na base de dados do jogo. O administrador é capaz de adicionar, modificar e remover cenários da base de dados. Cenários são representações de projetos e compostos por uma descrição e cinco fases: i) Escopo; ii) EAP (Estrutura Analítica do Projeto); iii) Definição de Atividades; iv) Sequenciamento de Atividades; e v) Caminho Crítico. Essas fases são a mesma utilizada pelos gerentes de projetos para planejar o escopo e o tempo de um projeto. O jogador deve utilizar as informações contidas em fases anteriores para conseguir resolver corretamente as fases seguintes do jogo.

No jogo X-Med [Wangenheim et al., 2009], o objetivo principal é exercitar a aplicação de medição de software voltada para a gerência de projetos. O público alvo do jogo é formado por alunos de pós-graduação

em cursos de Computação e/ou profissionais de Engenharia de Software. O jogo é projetado como um complemento às aulas tradicionais ou cursos de e-learning provendo ambiente para exercitar os conceitos apresentados. Presume-se que os alunos tenham conhecimento básico em Engenharia de Software, medição de software, gerência de projetos e dos modelos de qualidade CMMI (Capability and Maturity Model Integrated) e MPS.BR (Modelo de Produção de Software Brasileiro). O jogador assume o papel de um analista de software e, por meio de uma sequência de tarefas, avalia e faz medições. Além disso, ele pode desempenhar o papel de analista de medição e definir e aplicar um processo de medição. Cada tarefa apresenta seis alternativas como solução e os pontos são ganhos de acordo com seus acertos. Ao final, um relatório é gerado e pontuação final é atribuída.

No jogo Elicit@ção [Bernardi et al., 2008], o aluno desempenha o papel de analista no processo de elicitação de requisitos. A modelagem é realizada considerando o cenário para um Sistema para Gestão de Espetáculos Teatrais. Nesse cenário, são inseridos agentes (Clientes) que desempenham papéis de acordo com suas funções e suas responsabilidades. No estudo em questão, podem ser destacados os papéis de Gerente de Teatros e Cinemas, de Vendedores de Bilhetes e de Gerente de Programação Artística. As entrevistas acontecem por meio de cenas, onde são inseridos os agentes participantes. O aluno pode escolher caminhos, ou seja, quem entrevista ao longo do processo. Dessa forma, uma estrutura performativa foi desenvolvida com diferentes possibilidades, permitindo ao aluno tomar decisões (por exemplo, ordem das entrevistas e clientes a serem entrevistados) de forma a deixar a seu critério a melhor abordagem a ser seguida. Por fim, um conjunto de regras normativas é estabelecido segundo as diretrizes sugeridas para o

planejamento e a execução de uma entrevista, de forma a verificar se o aluno segue as orientações transmitidas a ele em sala de aula.

O Honey [Souza et al., 2008] é um jogo computacional que pode ser utilizada para fins didáticos, possibilitando aos professores e aos estudantes das diversas áreas da Computação, mais especificamente da área de Engenharia de Software, implementar e testar diversos temas de ensino com facilidade. Essa ferramenta funciona da seguinte forma: após a autenticar-se no Honey, o aluno tem acesso à tela principal, em que os recursos de aprendizado estão organizados em abas: i) Presentation: realização de apresentações; ii) Enviroment: execução e exibição do browser; iii) Files: disponibilização de arquivos úteis aos alunos durante o manuseio da aplicação na aula; iv) Home: informações sobre o módulo de e-learning executado no momento; e v) VNC Support: suporte ao aluno durante a utilização. A fim de avaliar e consolidar os conceitos ensinados, um ambiente com diversas portas é apresentado ao aluno. Em cada uma dessas portas, o acesso a uma prática da metodologia apresentada é garantido e o jogador é levado a uma sala composta por objetos e atores. Os atores realizam atividades com o intuito de fornecer ao aluno, o entendimento de alguma prática da metodologia XP (eXtreme programming), enquanto os objetos representam algum conceito sobre a metodologia.

O The Incredible Manager [Dantas, 2003] é um jogo de simulação para treinamento de gerentes de projetos de software. O jogador atua como gerente no planejamento e no controle de projetos de desenvolvimento de software com demandas específicas de orçamento, de cronograma e de qualidade. O jogo tem três componentes principais: i) modelos da dinâmica de sistemas para representar projetos de

desenvolvimento de software; ii) simulador de modelos que permitem simulação com estrutura dinâmica; e iii) máquina de jogo visual que apresenta o contexto do jogo e as respostas multimídia da simulação que permite a interação com o jogador. Ao utilizar o jogo, o jogador é requisitado a atuar como um gerente de projetos, planejando e controlando projetos de software para que sejam executados com sucesso, ou seja, dentro do cronograma e orçamento previstos. No início, o jogador que atua como o gerente de projetos deve construir um plano do projeto, formar a equipe e alocar recursos nas atividades, que deverão permanecer dentro do cronograma e do orçamento estipulado. O jogador interage com o jogo, efetuando alterações em seu planejamento para alcançar os seus objetivos.

O SE•RPG [Benitti; Molléri, 2008] é um jogo que simula o ambiente de desenvolvimento de software utilizando um cenário de uma empresa de desenvolvimento fictícia com diversas personagens com as quais o jogador interage. Ao iniciar o jogo, é apresentada breve descrição do projeto selecionado contendo o orçamento e o prazo para o seu desenvolvimento. Com essa descrição, o jogador deve definir em que modelo de processo será o desenvolvimento (apenas duas opções: Cascata e Iterativo) e qual será linguagem de programação a ser utilizada. Em seguida, cabe ao jogador definir uma equipe para o desenvolvimento com base em breve resumo das habilidades de cada personagem. O jogador pode iniciar o desenvolvimento do software, atribuindo tarefas para a sua equipe de maneira sequencial a partir da elicitação de requisitos e pode controlar o desenvolvimento do projeto, verificando o tempo gasto no projeto e o orçamento restante. Além disso, ele acompanha o andamento do processo de desenvolvimento. Durante o desenvolvimento, o jogador pode contratar ou demitir funcionários,

sendo possível verificar o progresso de cada etapa e a produtividade de cada membro da equipe. Ao final da implementação, o projeto é finalizado e o software é entregue ao cliente. Breve comentário é apresentado a respeito das metas do jogador e resultados obtidos, com pontuação determinada pelas metas de prazo, custo e escopo.

O Scrumming [Isotton, 2008] é um jogo educacional que simula a utilização de algumas práticas do Scrum [Schwaber, 2004] e busca suprir as necessidades encontradas no ensino de métodos ágeis para gerenciamento de projetos. Nesse jogo, o objetivo não é simular os processos utilizados no Scrum, mas focar na definição e na simulação de sprints. Existem dois tipos de usuários: i) Administrador; e ii) Scrum Master e dois módulos: i) administrativo, utilizado pelo usuário Administrador para realizar atividades que precedem a simulação (adicionar funcionários, adicionar ou remover atividades iniciais ao projeto, etc.); e ii) simulação: utilizado pelo usuário Scrum Master para simular sprints de um projeto. Para a simulação, cinco tipos de funcionários podem ser criados: i) gerente de projetos; ii) líder técnico; iii) desenvolvedor; iv) engenheiro de teste; e v) testador. Além disso, para cada tipo de funcionário, há três níveis de experiência que variam na competência e na produtividade: i) sênior; ii) intermediário; iii) iniciante.

O RE-O-Poly [Smith; Gotel, 2008] foi desenvolvido para explicar e explorar boas práticas de Engenharia de Requisitos, uma vez que os requisitos são, muitas vezes, conflitantes e os jogadores têm de aprender a resolver conflitos e determinar prioridades. O ambiente do jogo "obriga" a responder questões gerais e específicas do projeto, fazer constantes decisões pró-ativas sobre projetos responsáveis e responder apropriadamente às situações imprevistas que têm impacto em seus

projetos. O jogador experimenta a consequência das decisões de maneira que simula a real experiência do projeto, por meio de (i) perda de um projeto, (ii) perda de credibilidade ou (iii) retrocesso no progresso geral. O Conselho de Jogo abrange quatro etapas básicas em um processo geral: i) Análise de Elicitação; ii) Documentação; iii) Validação; e iv) Gerenciamento de Mudanças.

No SimVBE [Jain; Boehm, 2006], o foco é no ensino de Valor Crítico de Sucesso em um processo de Engenharia de Software. O jogador deve identificar os interessados no sistema com o que eles entendem como Valores Críticos de Sucesso e/ou Valores de Preferência dentro de uma configuração simulada. Do mesmo modo, eles devem determinar uma estratégia para equilibrar esses valores definidos com os interessados identificados. Representações utilizadas para modelar o jogo e seu desenvolvimento são baseadas em protótipos, as informações são extraídas de projetos, especificações e design de produto e cenários são criados no ambiente. O jogador no SimVBSE encarrega-se de evoluir cada um dos protótipos criados.

O SimulES [Monsalve et al., 2010] pode ser jogado de 4 a 8 jogadores. O seu objetivo é os participantes terminarem um projeto de software e o vencedor é aquele que implantar o projeto primeiro. Com esse jogo, os jogadores aprendem importantes conceitos de computação e, especialmente, de Engenharia de Software. Os recursos do jogo são: i) um tabuleiro; ii) cartões de projeto; iii) cartas; e iv) um dado. O tabuleiro é uma área na qual cada jogador coloca seus engenheiros de software em colunas e os artefatos em linhas. Os artefatos podem ser do tipo: i) requisitos; ii) desenhos; iii) códigos; iv) rastros; e v) ajuda aos usuários. Antes do início do jogo, um cartão de projeto é escolhido aleatoriamente

de uma série de projetos disponíveis. As informações desse cartão devem ficar visíveis aos jogadores. A cada jogada, o jogador lança o dado e, de acordo com o número tirado, retira cartas nos montes.

O SimSE [Qing et al., 2007] é um jogo computacional educacional em um ambiente de simulação, cujo objetivo é reduzir a distância entre a quantidade de conhecimentos de Engenharia de Software dada aos estudantes em palestras e a quantidade de prática. Nesse jogo, os jogadores podem praticar um processo "virtual" de Engenharia de Software de forma gráfica, interativa, divertida e em um cenário em que o feedback direto permite aprender a causa e o efeito das relações dos processos de Engenharia de Software. Além disso, o jogador atua como um gerente de projetos em uma empresa de porte médio, atribuindo tarefas aos seis engenheiros de acordo com seus pontos fortes. Em seguida, ele ajusta a tarefa atribuída de acordo com vários eventos que acontecem no jogo. O jogador pode motivar os engenheiros de software com aumento do salário ou bônus para "entregar" o software. Se o jogador não puder entregar dentro do prazo e/ou o custo do projeto excede o limite, o jogador falha no jogo; caso contrário, uma pontuação é dada ao entregar o software com base em seu desempenho. Os modelos são criados utilizando um construtor de modelo que permite a especificação de: i) entidades importantes na simulação, ou seja, empregados, artefatos, projetos, ferramentas e clientes; ii) ações/atividades que essas entidades podem participar; iii) regras que especificam os efeitos dessas ações sobre o resto da simulação; iv) gráfico representação de cada uma das entidades importantes na simulação; e v) entidades começam com o estado inicial.

O MO-SEProcess [Tao; Qing, 2009] é um jogo para multiplos jogadores on-line, de processos de Engenharia de Software, com base no SimSE, concebido, para simular a gestão de um projeto de desenvolvimento de software, e desenvolvido no Second Life (SL). Os jogadores colaboram para desenvolver um sistema de software, formando uma equipe. Nesse jogo, seis funções são fornecidas e os jogadores podem escolher qualquer papel para desempenhar, mas apenas pode ser encarregado da função escolhida. Durante o jogo, o jogador pode interagir com outros jogadores utilizando vários meios de comunicação previsto no SL. A pontuação da equipe é dada no final do jogo, considerando o tempo de entrega do produto, o trabalho realizado e a colaboração entre os jogadores.

Quando os futuros pilotos de aeronaves aprendem a sua profissão, há um problema: eles precisam saber como se comportar em situações perigosas, mas não há necessidade de expô-los a esse risco apenas por uma questão de treinamento. As companhias aéreas providenciaram simuladores para realizar esse treinamento sem qualquer risco (embora em custo considerável). A mesma abordagem foi aplicada para a educação de gerentes de projeto de software. O projeto e o seu ambiente são simulados, mas o gerente de projeto de software (o jogador) é real. Essa abordagem foi chamada SESAM (Software Engineering Simulation by Animated Models) [SESAM, 2009; Drappa; Ludewig, 1999; Drappa; Ludewig, 2000]. A simulação pode melhorar os cursos de Engenharia de Software, mas não pode substituir o ensino tradicional, pois pode afetar a motivação do jogador, mas não a sua habilidade. O jogador pode contratar/demitir funcionários e pedir-lhes para realizar uma tarefa para o desenvolvimento de sistemas de software. O simulador gerencia muitas variáveis internas, por exemplo, o grau em que a especificação reflete o

requisito, mas apenas para uso interno, o jogador não pode vê-la. "Ele é tão mal informado como qualquer projeto real gerente". Quando o jogo acaba, o jogador recebe sua pontuação e pode analisar seu desempenho utilizando uma ferramenta de análise disponível no jogo.

O potencial dos jogos no contexto educacional é considerável, pois apresentam diversos benefícios aos usuários, dentre eles [Silva; Kirner, 2010]: i) desenvolvimento de raciocínio; ii) desenvolvimento e ampliação cultural; iii) apoio na alfabetização; iv) compreensão de regras e estratégias; e v) aperfeiçoamento da memorização. Desde o início da civilização, "o brincar" não é uma atividade exclusiva das crianças, mas se estende aos adultos, afinal "o brincar" não se restringe ao período da infância, apenas é predominante nesse período [Velasco, 1996]. Em jogos, pode-se preparar o indivíduo para o trabalho e inseri-lo em um grupo social. Ao jogar, o indivíduo sente-se pertencente a uma equipe e cumpre diversas funções. Além disso, o contato com outras pessoas é proporcionado pelo jogo, fazendo com que se habitue a considerar pontos de vista e opiniões diferentes.

A construção de um jogo para apoiar o processo de ensino-aprendizagem em Teste de Software pode contribuir para a sedimentação/fundamentação de conceitos pertinentes ao tema aos alunos, de maneira que eles possam aprender a colocar em prática, o conteúdo teórico aprendido em sala. Com relação aos jogos apresentados, o projeto apresentado neste trabalho possui o diferencial de construir um jogo computacional educacional (TestEG) envolvendo Teste de Software.

3. USO DE JOGOS NA EDUCAÇÂO

3.1. Considerações Iniciais

Ao longo dos anos, a maneira de como jogar um jogo modificou-se. De certa forma, os jogos acompanham a modernidade, utilizando recursos disponíveis ao seu tempo. Os jogos estão presentes no cotidiano das pessoas, não se restringindo a apenas uma fase da vida. Os jogos podem servir como ferramentais educativas, têm o poder de divertir o jogador, proporcionam prazer ao mesmo tempo em que ensinam, aumentam o poder de absorção da informação e exercitam de maneira eficiente funções intelectuais do jogador [Tarouco et al., 2004]. Sob essas perspectivas, entende-se que jogos educacionais aceleram o aprendizado, contribuindo para um despertar maior do aluno em prol do conhecimento e melhorando seu vínculo afetivo com a aprendizagem [Barbosa, 2000].

Existem diferentes tipos de jogos classificados de acordo com seus objetivos, por exemplo, jogos de ação, de aventura, lógicos, estratégicos, esportivos e role-playing games (RPG). Alguns deles podem ser utilizados com propósitos educacionais [Tarouco et al., 2004], sendo que há vários formatos, tais como, cartas, tabuleiros e os computacionais. Neste capítulo, são apresentadas informações a respeito da utilização de jogos na educação para auxiliar no processo de ensino-aprendizagem.

A importância dos jogos no contexto educacional é abordada na Seção 3.2. Os tipos de jogos existentes para auxiliar no processo de ensino-aprendizagem são mostrados na Seção 3.3. Informações acerca da utilização de jogos no ensino da Engenharia de Software são apresentadas na Seção 3.4.

3.2. Importância

Houve evolução dos jogos com relação aos estímulos dos aspectos cognitivos, o aluno tem vários sentidos estimulados durante o desenrolar de um jogo. A realidade dos jogos não pertence apenas à realidade infantil, mas ganhou conotação e aspectos educativos, sendo aproveitado nos vários níveis de ensino [Kahl et al., 2007]. Com o surgimento de novas práticas de ensino aliadas ao crescente desenvolvimento tecnológico, foi permitida a introdução dos jogos educacionais nas escolas, cuja função é apoiar professores e alunos no processo de ensino-aprendizagem [Moraes et al., 2008]. Oportunizar formas de construção do conhecimento pelo aluno é um dos principais objetivos dos jogos educacionais [Aranha, 2006].

Jogar permite aos alunos entenderem as regras, terem-nas bem definidas e fazerem com que identifiquem em quais contextos elas devem ser aplicadas, modificando-as quando houver novos contextos. No decorrer do jogo, fatores como autonomia, criatividade, situações adversas e poder criativo são revelados [Tarouco et al., 2004]. A inserção de jogos no contexto do processo de ensino-aprendizagem implica em vantagens (Tabela 3-1) e desvantagens (Tabela 3-2) [Grando, 2001]. Por exemplo, (i) o jogo requer a participação ativa do aluno na construção do seu conhecimento e (ii) a utilização dos jogos como fator de motivacional aos alunos são duas vantagens; por outro lado, (i) a constante interferência do professor faz com que o jogo perca a "ludicidade" e (ii) a coerção do professor, exigindo que o aluno jogue, são duas desvantagens.

Tabela 3-1 - Vantagens dos Jogos [Grando, 2001]

Vantagens
Fixar conceitos aprendidos de forma motivadora ao aluno.
Introduzir e desenvolver conceitos de difícil compreensão.

Desenvolver estratégias de resolução de problemas (desafio dos jogos).
Capacitar em tomar decisões e saber avaliá-las.
Por significado para conceitos aparentemente incompreensíveis.
Propiciar o relacionamento de diferentes disciplinas (interdisciplinaridade).
Requer a participação ativa do aluno na construção do seu próprio conhecimento.
Favorecer a socialização entre alunos e a conscientização do trabalho em equipe.
Utilizar jogos é fator de motivação para os alunos.
Favorecer o desenvolvimento da criatividade, de senso crítico, da participação, da competição sadia, da observação, das várias formas de uso da linguagem e do resgate do prazer em aprender.
Utilizar as atividades com jogos para reforçar/recuperar habilidades de que os alunos necessitem, sendo útil no trabalho com alunos de diferentes níveis.
Permitir ao professor identificar e diagnosticar erros de aprendizagem e de atitudes e dificuldades dos alunos.

Tabela 3-2 - Desvantagens dos Jogos [Grando, 2001]

Desvantagens
Dar ao jogo caráter puramente aleatório, tornando-se "apêndice" em sala de aula, quando mal utilizados. Os alunos jogam e sentem-se motivados apenas pelo jogo, sem saber o porquê jogam.
Gastar mais tempo com as atividades de jogo em sala de aula. Se o professor não estiver preparado, pode existir "sacrifício" de outros conteúdos pela falta de tempo.
Ter falsas concepções de que devem ensinar os conceitos utilizando jogos. Em geral, as aulas transformam-se em verdadeiros cassinos, sem sentido algum para o aluno.
Perder a “ludicidade” do jogo pela interferência constante do professor, destruindo a essência do jogo.
Existir a coerção do professor, exigindo que o aluno jogue, mesmo que ele não queira, destruindo a voluntariedade pertencente à natureza do jogo.
Ter dificuldade de acesso e de disponibilidade de materiais e de recursos sobre a utilização de jogos no ensino que possam vir a subsidiar o trabalho docente.

O aluno tem sua atenção prendida pelo contexto do jogo, caso esse jogo receba estímulos variados. Isso acontece quando um aluno é colocado sob circunstâncias desafiadoras para resolver problemas, fazendo com que ele seja capazes de auto avaliar-se e de medir o seu desempenho [Moratori, 2003]. Os jogos desenvolvem a antecipação e a estratégia e ampliam o raciocínio lógico que podem ser utilizados para desenvolverem habilidades utilizadas na vida real [Braga et al., 2007]. As pessoas apresentam formas variadas de aprendizado, não havendo

necessariamente melhor ou pior maneira. Portanto, resta aos gestores descobrirem alternativas que colaborem para o desenvolvimento das competências dos alunos e levem-nos a conhecimento de seus estados cognitivo [Moratori, 2003].

3.3. Tipos de Jogos

Os jogos educacionais caracterizam-se como ferramentas importantes nas formas de ensino atual, a possibilidade de interação e de integração maior é um estímulo para a criatividade e a consciência de certas dinâmicas [Aranha, 2006]. Quanto aos tipos de jogos, tem-se [Tarouco et a1., 2004]:

- **Ação**. Esses jogos refletem de forma direta no aperfeiçoamento das características psicomotoras, principalmente das crianças. O ganho é com relação à coordenação dos olhos, da mão e dos reflexos. O poder de raciocínio torna-se mais rápido, por causa da velocidade com que tem que responder aos estímulos do jogo. O ideal é o jogo variar entre períodos de estímulos cognitivos mais intensos com períodos de estímulos de maior utilização de habilidades motoras;
- **Aventura**. Esses jogos oferecem mais liberdade ao usuário para descobrir e trabalhar no ambiente. Algumas atividades tornam-se completamente inviáveis em sala de aula, principalmente por questões físicas. Em um ambiente bem modelado, o aluno pode vivenciar e experimentar uma realidade antes impossível;
- **Lógico**. Esses jogos são mais desafiadores, mas não trabalham os reflexos. De certa forma, o desafio e os reflexos acabam integrados, pois temporizadores em jogos de raciocínio instigam o aluno ao desenvolvimento de reflexos mais ágeis;

- **Role-playing game (RPG)**. Esses jogos permitem ao usuário controlar um personagem com o qual interage com outros personagens. A alteração dos atributos do personagem decorre da ação da escolha feita pelos usuários;
- **Estratégicos**. Esses jogos são voltados para o desenvolvimento de habilidades e de sabedoria de seus usuários, sendo necessário administrar, construir e planejar. Esse tipo de jogo pode proporcionar uma simulação em que o usuário utiliza conhecimentos adquiridos em sala de aula, percebendo uma forma prática de aplicá-los.

Além disso, há variações quanto às formas de implementação desses jogos, podendo utilizar ambiente tradicional (cartas, tabuleiros...etc) ou ambiente computacional.

3.4. Jogos Computacionais

O computador está presente na realização das tarefas cotidianas e corriqueiras das pessoas. Sobre a ótica educacional, há discussão a respeito da utilização do computador como um instrumento no processo de ensino-aprendizagem, havendo muitas questões que ainda precisam ser respondidas. O computador é um dos principais motivadores para a continuidade do avanço tecnológico, não pode mais ser ignorado pela escola, mas o desafio é fazer com que esse potencial seja utilizado no processo de educação e, principalmente, de uma forma efetiva e eficiente. Deve-se aliar projetos escolares com o preparo dos futuros cidadãos [Smole; Diniz, 2001]. O processo de ensino-aprendizagem passou e está passando por um processo de revolução por causa da introdução do computador no âmbito escolar [Valente, 1993]. A utilização da tecnologia em atividades educacionais tem se tornado comum a cada dia, principalmente no ensino básico [Krüger et al., 2001]. Para a efetiva

implantação dessa utilização, são necessários quatro "ingredientes" [Valente, 1993]: i) computador; ii) software educativo; iii) professor capacitado para utilizar o computador como meio educacional; e iv) aluno.

É extremamente ampla a área de atuação do computador, pois ele exerce influência (in) direta nas camadas sociais e em diversas atividades humanas. Na educação, o computador tem sido utilizado para (i) ensinar sobre computação (ensino de computação) e (ii) ensinar qualquer assunto (ensino com a utilização do computador) [Valente, 1993]. Um software educacional deve ser avaliado para descobrir se ele atende ou não as necessidades e as propostas de ensino adotadas [Gomes et al., 2002]. Sendo o professor o "condutor" do ensino, cabe a ele efetuar a escolha do software educacional a ser utilizado. Ele deve definir quais parâmetros de qualidade devem ser adequados às necessidades do alunos [Gomes et al., 2002]. Se o professor falha nessa escolha, o processo de ensino-aprendizagem pode ser seriamente comprometido. Em consequência dessa falha e de outros fatores (por exemplo, sistema, funcionamento e estrutura física da escola), a prática da informática na escola, muitas vezes, distancia-se de seu caráter pedagógico.

A máquina deve ser capaz de transmitir ao aluno conceitos sobre diversos domínios [Valente, 1993]. Em um lado, o computador "ensina" o aluno com o software; por outro lado, o aluno "ensina" o computador com o software, o que torna o ensino uma espécie de "via de mão dupla". A utilização de software educacional permite ao aluno aprender com o computador, sendo jogos educacionais e simuladores exemplos que exerce essa função. Deve-se atentar quanto aos reais objetivos na utilização do computador na educação, pois sua introdução na educação

não deve ser por causa do modismo ou de estar atualizado com relação às inovações tecnológicas [Valente, 1993].

Os jogos são excelentes ferramentas no tocante a instruir os alunos, a ideia de motivar enquanto se diverte é uma combinação perfeita para estimulá-los e aumentar sua capacidade de retenção do que foi ensinado, pois o jogo passa a exercitar mais funções cognitivas e intelectuais do jogador [Tarouco et al., 2004]. O software educativo não substitui outras fontes de consulta e não consegue suprir as lacunas que o professor pode, por ventura, deixar. Assim, ele deve ser utilizado como ferramenta complementar no processo de ensino-aprendizagem, mas sua dinâmica provê ao aluno mais agilidade na busca de informações que sirvam ao seu aprendizado [Lopes et al., 2004]. Jogos educacionais são apontados como um promissor material de ensino, que pode trazer benefícios para a educação em Engenharia de Software. Os pesquisadores devem empenhar mais esforços para constatar/evidenciar as vantagens pedagógicas que o computador possa oferecer para a aprendizagem dos alunos [Wangenheim et al., 2009].

3.5. Considerações Finais

Com o advento da computação e o surgimento de novas tecnologias, muitos recursos estão disponíveis para auxiliar no processo de ensino-aprendizagem. A partir disso, a visão tradicional e os métodos de ensino utilizados anteriormente, com o passar do tempo, tornaram-se obsoleto, compelindo ao professor a missão de mesclar novos e antigos métodos de ensino de acordo com suas necessidades.

Os jogos educacionais podem ser considerados ferramentas complementares no processo educativo, fixando os conceitos

desenvolvidos em sala de aula. Eles não devem ser utilizado por simples modismo ou estar atualizado com as inovações, mas por colaborar de forma significativa no método do processo de ensino-aprendizagem. Os jogos têm por características despertar nos jogadores interesse, motivando-os e auxiliando-os na absorção das ideias. Por isso, esse ambiente, onde são utilizados jogos computacionais educacionais, costuma ser rico e prazeroso, tornando fácil e eficiente o processo de ensino-aprendizagem.

4. TESTE DE SOFTWARE

4.1. Considerações Iniciais

Nos dias atuais, a demanda por sistemas de software com qualidade é cada vez maior, porém a busca por menores custos e tempo de produção tendem a fazer com que a produção desses sistemas diminua. Assim, a atividade de teste de software assume papel importante, pois consiste em executar o software de forma controlada e monitorada a fim de avaliar seu comportamento [Crespo et al., 2004]. Com mercado mais competitivo, profissionais da área de qualidade têm sido motivados para desenvolver técnicas no sentido de propiciar a construção de sistemas de software com os padrões de qualidades impostos [Delamaro et al., 2007], o que explica o aumento expressivo das atividades que envolvem testes de software. A atividade de testar sistemas de software é fundamental para avaliá-los, mas não é uma atividade trivial e exige base de conhecimentos, de habilidades e de infraestrutura específicos [Crespo et al., 2004]. Neste capítulo, são apresentados conceitos, técnicas, aplicações e finalidades de teste de software.

Teste de software é contextualizado na Seção 4.2. SWEBoK, importante documento para a Engenharia de Software, é apresentado resumidamente na Seção 4.3. A área de conhecimento Teste de Software é detalhada na Seção 4.4.

4.2. Contextualização

Muitos esforços são feitos na tentativa de definir a atividade de teste, da visão intuitiva a uma definição formal [Beizer, 1995]. As afirmações generalizam um conceito sobre teste em ambas as visões, que caminham para um mesmo conceito [Crespo et al., 2004]: com o objetivo

de avaliar como se dá o comportamento do software, diante das especificações, executa-se o software controlando o processo. O processo de software, apesar das técnicas e dos métodos envolvidos, não garante que eventuais erros possam ocorrer. A atividade de teste é extremamente utilizada em técnicas de verificação e de validação, constituindo um dos elementos que fornecem confiabilidade ao software [Delamaro et al., 2007].

Com o teste, o que se faz é uma análise detalhada do software. Assim, o testador é capaz de identificar erros e, posteriormente, eliminá-los. O conjunto de informações provenientes da atividade de teste é importante às atividades de depuração, de manutenção e de estimativa de confiabilidade de software. O teste de software envolve basicamente quatro etapas [Pressman, 2009]: i) planejamento dos testes; ii) projeto de casos de teste; iii) execução dos testes; e iv) avaliação dos resultados dos testes. Embora as técnicas de testes não sejam recentes, pois são utilizadas desde a década de 70, muitas empresas passam por dificuldade quando vão utilizá-la [Crespo et al., 2004]. Em teste de software, há carência de profissionais especializados e dificuldade de implantar um processo de teste com as técnicas apresentadas na literatura.

4.3. SWEBoK

Patrocinado pela IEEE (Institute of Electrical and Electronics Engineers), o SWEBoK (Guide to the Software Engineering Body of Knowledge) é utilizado como referência para estudos em Engenharia de Software [SWEBoK, 2013]. O surgimento desse guia se deu em decorrência da necessidade de formalizar a Engenharia de Software, visando à definição das fronteiras que a delimitam em relação às demais e servir de base para caracterizar os profissionais dá área. Nesse guia, há

critérios e conteúdos objetivos, claros e bem definidos, sendo uma ferramenta de excelência em aplicabilidade na prática, além de fornecer formalizações direcionadas para certificações profissionais e acadêmicas.

Os objetivos do SWEBoK são (i) caracterizar o conteúdo da Engenharia de Software, (ii) estabelecer um conjunto apropriado de critérios e normas para a prática profissional da Engenharia de Software, (iii) marcar as fronteiras entre a Engenharia de Software e as demais disciplinas relacionadas e (iv) prover uma fundação para certificação individual e para licenciamento de profissionais. No SWEBoK, a Engenharia de Software está organizada em 10 Áreas de Conhecimento [SWEBoK, 2013]:

- **Requisitos de Software**. O conteúdo referente à análise, à especialização e à validação de requisitos de software é apresentado;
- **Design de Software**. Há duas maneiras de definir design de software [IEEE, 1990]: i) é considerado uma atividade do ciclo de vida, em que os requisitos de software são analisados para descreverem de forma detalhada a estrutura interna do software; e ii) é visto como um documento contendo descrição detalhada da arquitetura do software, além da organização dos seus diversos componentes;
- **Construção de Software**. Há a codificação, a verificação e os testes unitários, de integração e de depuração realizados de maneira combinada;
- **Testes de Software**. Atividades de testes são realizadas para melhorar o software, identificando possíveis falhas ou defeitos. Durante os testes, vários documentos são gerados para ter controle do que está sendo feito;
- **Manutenção de Software**. As atividades necessárias para proporcionar o custo efetivo para apoio do software são realizadas. Após o software estar em uso, anomalias podem ser descobertas, modificação no

ambientes operacional podem ocorrer e novos requisitos dos usuários podem ser descobertos/solicitados;

- **Gerenciamento de Configuração do Software**. Há (i) a identificação da configuração do software em pontos distintos do tempo, (ii) o controle das modificações, (iii) a manutenção da integridade e (iv) a auditoria da configuração;
- **Gerenciamento da Engenharia de Software**. São aplicadas atividades de gestão, de planejamento, de coordenação, de medição, de controle e de monitoramento e relatório que asseguram ou tendem a assegurar o desenvolvimento e a manutenção do software;
- **Processo de Engenharia de Software**. O processo de desenvolvimento de software é definido, implementado, mensurado, gerenciado, modificado e aperfeiçoado;
- **Ferramentas e Métodos de Engenharia de Software**. Ferramentas computacionais para o desenvolvimento de software são utilizadas para automatizar o processo de Engenharia de Software;
- **Qualidade de Software**. São abordadas questões e atividades relativas à garantia da qualidade do software.

4.4. Área de Conhecimento: Teste de Software

Executar testes é verificar dinamicamente como é o comportamento do software quando são fornecidas determinadas sequências de entrada [SWEBoK, 2013]. Não se pode garantir que um software funcionará livre de erros [Myers, 2004], mas, se a fase de teste for bem executada, a maioria deles pode ser descoberta e corrigida e o software ser entregue com nível de confiabilidade maior. Essa área de conhecimento está organizada em subáreas: i) Fundamentos; ii) Níveis; iii) Técnicas; iv)

Medidas; e v) Processo. As três primeiras são detalhadas na próximas seções.

4.4.1. Fundamentos de Teste de Software

Teste de software deve levar em consideração o contexto do software, sendo que há variação quanto à forma de utilização de cada um, alterando a maneira como o teste deve ser executado. Em um sistema de software bancário, por exemplo, necessita-se de planejamento e de execução rigorosa de teste para não se tornar um caos (aumento de custo); nesse caso, o dano poderia ser incalculável (monetariamente). Da mesma forma, um sistema de software hospitalar, que "lida" com vidas, em hipótese alguma pode falhar (tragédia). Muitos das causas de falhas em sistemas de software são advindos de erros humanos. A nomenclatura erro, defeito e falha é comumentemente confundida, mas há sutil diferença [IEEE, 1990]:

- **Erro**. Discrepância entre o valor computado e o valor esperado (saída gerada vs. esperada);
- **Falta**. Condição que leva o sistema a não realizar uma função esperada, chamado bug;
- **Falha**. Incapacidade do software em realizar uma função de acordo com sua especificação.

Para entender as questões relacionadas a essa subárea, deve-se ter em mente alguns conceitos [Myers, 2004]:

- **Critérios de Seleção de Testes**. Forma de decidir qual(is) o(s) conjunto(s) de casos de teste deve(m) existir e qual(is) deve(m) ser satisfeito(s). Pode ser utilizado para o processo de seleção dos casos de teste, decidindo se o teste pode ou não ser parado;

- **Efetividade do Teste/Objetivos de Teste**. Testar é a observação de uma amostra de execuções do sistema de software. Seleções de amostras podem ser guiadas por diferentes objetivos;
- **Teste para Identificação de Defeito**. Um teste de sucesso causa a falha do sistema de software;
- **Problema do Oráculo**. Oráculo que faça a verificação quanto ao comportamento do sistema de software e determine o veredito de falha ou sucesso;
- **Limitações Teóricas e Práticas de Teste**. Teste de software pode ser utilizado para provar a presença de bugs, mas não para mostrar sua ausência. Teste precisa ser dirigido baseado no risco e pode ser visto como estratégia de administração de risco;
- **Testabilidade**. Diferentes probabilidades e características comportamentais, que levam o código a falhar se algo estiver incorreto, são examinadas.

4.4.2. Técnicas de Testes

Na norma IEEE 610.12-1990 [IEEE, 1990], as técnicas são definidas como procedimentos técnicos e gerenciais que otimizam o processo e contribuem para a avaliação. São muitas as maneiras possíveis de testar um software e, independente da tecnologia da linguagem de programação, as técnicas continuam tendo grande valia. Isso é consequência da finalidade do teste que continua o mesmo: encontrar falhas no software.

4.4.2.1. Teste Funcional

Teste funcional é realizado no sistema de software como se ele fosse uma caixa em que apenas é possível visualizar o lado externo. Por

esse motivo, teste funcional é chamado Teste de Caixa Preta, pelo qual é mais conhecido [Myers, 1979]. Teste de caixa preta é orientado a dados, pois são fornecidas entradas e apenas se visualiza a resposta produzida como saída. Os detalhes de implementação não são investigados, somente as funções do sistema são relevantes. O teste tende a se tornar mais rico, com o aumento da quantidade de entradas, mas, em uma situação real para uma quantidade de entradas extremamente grande, seria impossível testar todas [Myers, 2004].

O teste funcional pode ser organizado em dois processos principais [Pressman, 2009]: i) Fazer a identificação quanto às funções realizadas pelo sistema de software; e ii) Crias casos de testes que verifiquem se o sistema de software executa tais funções. Por isso, é importante ter especificação apropriada e que represente os requisitos de usuário adequadamente, pois as funções são identificadas considerando as suas especificações. Especificações ambíguas, em relação ao sistema de software, podem fazer com que as entradas não sejam as mesmas aceitas por esse sistema [Jiantao, 1999]. Há algumas técnicas de Teste de Caixa Preta, por exemplo [Myers, 2004; Dustin, 2003; Pressman, 2009; Burnstein, 2003]:

- **Particionamento em Classes de Equivalência**. Para a geração de casos de teste, é proposto um modelo de combinação de condições. Utilizando um caso de uso, divide-se a entrada do sistema de software em classes a serem testadas. O relacionamento lógico entre as causas e os efeitos são representados por meio de grafos;
- **Análise do Valor Limite**. Particionamento de classes de equivalência não delimitam valores de entrada e os erros estão mais propensos a ocorrer quando valores do limite da fronteira são utilizados. Nessa análise, uma das prioridades é delimitar o valor limite;

- **Grafo de Causa e Efeito**. Particionamento de classes de equivalência não permite a combinação de condições, sendo uma deficiência. Grafo de causa e efeito complementa as anteriores ao fazer uma proposição combinando valores de entrada para gerar casos de teste.

4.4.2.2. Teste Estrutural

Ao contrário do teste funcional, o teste estrutural (Teste de Caixa Branca) realiza a avaliação interna dos componentes do sistema de software. Ele apresenta algumas limitações relacionadas às atividades de teste como estratégia de validação [Rapps; Weyuker, 1985], o que torna um problema quando se trata da automação do processo de validação. No teste estrutural, atua-se diretamente sobre o código fonte do componente do sistema de software, complementa-se a técnica funcional e realizam-se aspectos como [Pressman, 2009]: i) Teste de Condição; ii) Teste de Fluxo de Dados; iii) Testes de Ciclos; iv) Testes de Caminhos Lógicos; e v) Códigos Nunca Executados. O propósito é avaliar o código fonte para encontrar erros em sua estrutura interna, deve-se garantir [Pressman, 2009] que sejam "exercitados" (i) cada caminho do código, pelo menos uma vez, (ii) os estados verdadeiro e falso das decisões lógicas e (iii) as estruturas de dados utilizadas internamente.

Teste estrutural possui alguns critérios que se baseiam em diferentes conceitos e componentes para poder determinar os requisitos de teste e são classificados em [Pressman, 2009]: i) Critério Baseado em Fluxo de Controle; ii) Critério Baseado em Fluxo de Dados; e iii) Critério Baseado na Complexidade. Há algumas técnica de teste de caixa branca, por exemplo [Pressman, 2009]:

- **Teste do Caminho Básico**. São criados casos de testes para cada caminho possível no fluxo de controle do sistema de software.

Primeiramente, cria-se o fluxo representando-o por meio de grafo, os nós são os comandos e as arestas são o sentido do controle lógico. Os nós que possuem condições associadas são chamados de nós predicados e têm por caraterística possuírem duas ou mais arestas saindo;

- **Complexidade Ciclomática**. É fornecida uma medida qualitativa da complexidade do sistema de software. Ao utilizar essa técnica no contexto de Teste do Caminho Básico, o valor computado da complexidade ciclomatica define a quantidade de caminhos independentes. Nessa técnica, é fornecida a quantidade máxima de testes a serem executados para garantir que cada instrução seja executada pelo menos uma vez.

4.4.3. Níveis de Testes

Ao final do desenvolvimento de uma função, ela deve ser testada por uma equipe que não a desenvolveu. Em outros casos, as funções são testadas simultaneamente com o projeto, reproduzindo o processo até ao final do projeto. Práticas como essas podem impactar positivamente no tempo de conclusão do projeto. Basicamente, há cinco níveis de testes [Myers, 2004]:

- **Teste de Unidade**. São testadas as menores unidades desenvolvidas do sistema de software. O objetivo é detalhar essas unidades, na tentativa de identificar alguma falha decorrente de defeitos de lógica, no momento da implementação;
- **Teste de Integração**. Após as unidades serem testadas, elas devem ser integradas para constituir a funcionalidade do sistema de software, o que pode ocasionar uma falha. Essas falhas são detectadas na realização dos testes de integração;

- **Teste de Sistema**. O objetivo é executar o sistema sob a perspectiva do usuário final;
- **Teste de Aceitação**. Um grupo de usuários finais selecionados simula operações de rotina para encontrar anomalias de execução;
- **Teste de Operação**. Há a preocupação com a quantidade de tempo que o sistema permanecerá em funcionamento sem que ocorrer falhas. Compete aos administradores desse sistema realizar os testes. Em determinados casos, quando um sistema é produzido para substituir outro existente e em funcionamento, deve-se garantir o suporte ao negócio.

4.5. Considerações Finais

Este capítulo abordou conceitos e definições de teste de software. Foram apresentados a contextualização dos termos e o detalhamento com os fundamentos, as técnicas e os níveis de testes abordados no SWEBoK. Esses conceitos são necessários para entendimento teórico do funcionamento do jogo educacional, visando à ambientação do usuário/jogador no universo que implementa as atividades de um testador de software.

5. TestEG - *TEST EDUCATIONAL GAME*

5.1. Considerações Iniciais

Neste capítulo, é apresentado o jogo computacional educacional TestEG (Test Educacional Game), cujo propósito é apoiar o processo de ensino-aprendizagem de assuntos envolvendo Teste de Software. Para isso, são detalhadas a funcionalidade e a dinâmica do jogo para melhor entendimento de como foi sua implementação e conhecimento das suas especificações.

Especificações do TestEG são descritas na Seção 5.2. A modelagem do TestEG com o Diagrama de Casos de Uso e o Diagrama Navegacional é apresentada na Seção 5.3. A modelagem de dados do TestEG é mostrada na Seção 5.4. O funcionamento do TestEG e as informações necessárias para jogá-lo são discutidos na Seção 5.5.

5.2. Especificação do Software

No jogo TestEG, o jogador assume o papel de Gerente de Teste. Ao iniciar o jogo, o Gerente de Teste recebe um orçamento inicial e deve contratar uma equipe de três funcionários para compor o seu time de testadores de software. O cenário do jogo é um ambiente empresarial, com a equipe selecionada realizando seus trabalhos. No decorrer do jogo, o Gerente de Teste deve auxiliar a sua equipe solucionado suas dúvidas e dando as informações necessárias para a execução do trabalho. O Gerente de Teste poderá realizar os treinamentos dos funcionários contratados inicialmente, verificar o desempenho desses funcionários e ler conteúdos (teoria) a cerca de Testes de Software.

Quando se realiza um treinamento, um valor predefinido é descontado do orçamento, sendo proporcional ao tipo de treinamento escolhido. Esse desconto ocorre a cada 2 minutos de jogo, tempo fictício equivalente a um mês de trabalho. Há um ranking pelo qual os jogadores podem acompanhar o desempenho de outros jogadores. No total, 10 perguntas devem ser respondidas dentro do tempo de 10 minutos, sem que ocorra o esgotamento do orçamento.

A funcionalidade do TestEG é apresentada na Figura 5-1, utilizando o Diagrama de Casos de Uso da UML. O ator Jogador pode realizar os casos de uso: i) Fazer Login; ii) Realizar Estudo; iii) Contratar Funcionários; iv) Treinar Funcionários; v) Responder Perguntas; vi) Consultar Ranking dos 3 Primeiros Colocados; vii) Consultar Ranking dos 10 Primeiros Colocados; viii) Desligar Som; ix) Ligar Som; x) Atualizar Dados; e xi) Ver Dinâmica/Regras do Jogo. O ator Administrador pode realizar os casos de uso: i) Fazer Login; ii) Incluir Administrador; iii) Excluir Administrador; iv) Alterar Administrador; v) Incluir Jogador; vi) Excluir Jogador; vii) Alterar Jogador; viii) Incluir Perguntas; ix) Excluir Perguntas; e x) Incluir Respostas. Além desse diagrama, a modelagem dos dados do TestEG é apresentada na Figura 5-2, utilizando o Modelo Relacional para representar as tabelas. Nessas tabelas, contém as informações necessárias para a execução do TestEG, tais como, perguntas, dados dos jogadores e dos administradores, características de cada personagem do jogo e ranking.

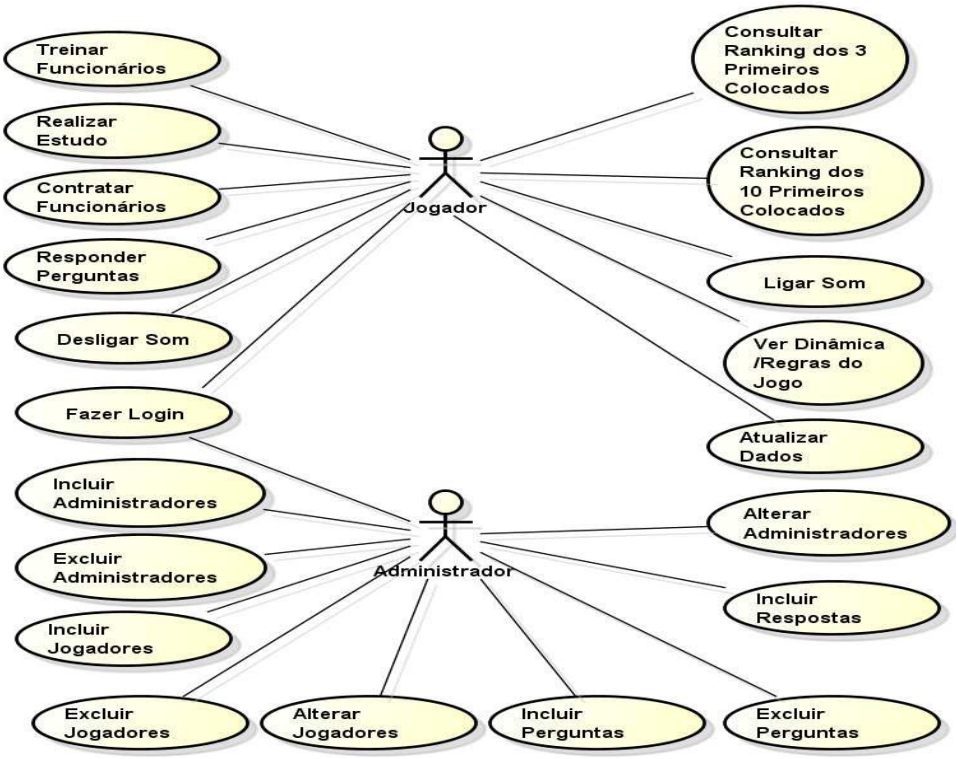

Figura 5-1 - Diagrama de Casos de Uso

5.3. Funcionamento

Nesta seção, breve descrição da funcionalidade do TestEG é apresentada por meio de suas telas (interface com o jogador) e suas especificações.

5.3.1. Organização Modular

O TestEG é um jogo single-player no qual o jogador assume o papel de Gerente de Teste em uma empresa de desenvolvimento de software. No decorrer do jogo, o jogador deve tomar decisões e efetuar ações típicas de um gerente de teste para alcançar os objetivos do jogo. No TestEG, há dois módulos, os quais, para ter acesso, o usuário deve ter login e senha previamente cadastrada:

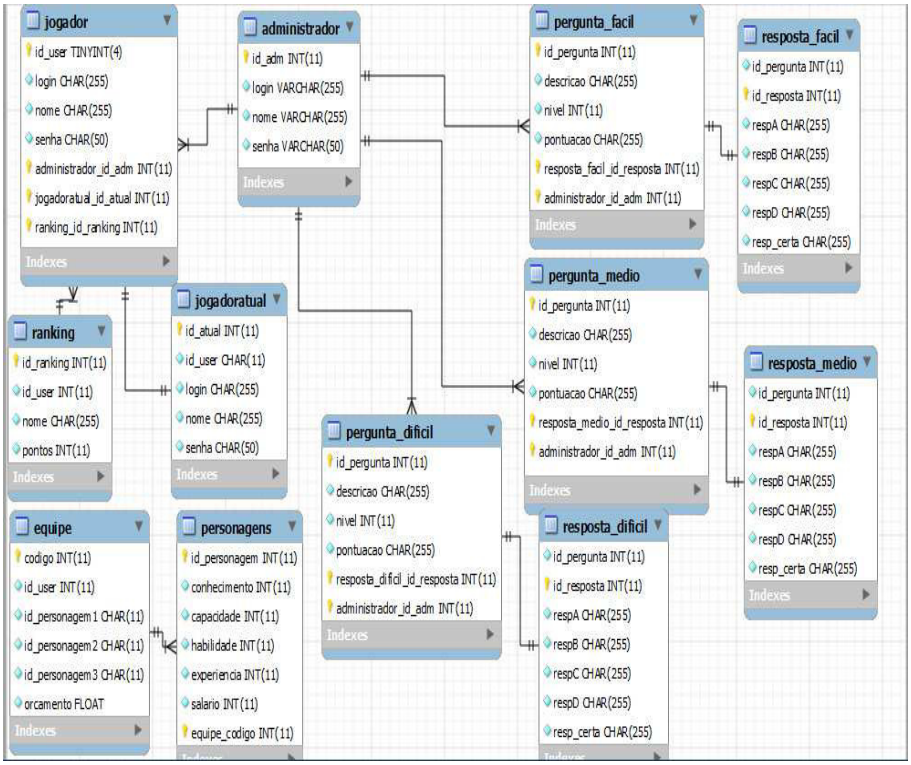

Figura 5-2 - Modelo Relacional

- **Módulo Administrador**. Permite ao usuário realizar ações de Administrador por realizar a manutenção de dados cadastrais de jogadores, de outros administradores e de perguntas;
- **Módulo Jogador**. Permite ao usuário realizar ações de Jogador por atualizar seus dados, ver o ranking de pontuação, consultar as regras do jogo e jogar.

5.3.2. Regras

A dinâmica do jogo é baseada em algumas regras que, de acordo com o desempenho do Jogador, conduzirá-lo a vitória (sucesso) ou a derrota (falha). O Jogador deve conhecer essas regras para tomar as melhores decisões, com relação às circunstâncias apresentadas. Tais regras estão disponíveis para consulta no próprio jogo (Figura 5-3):

- **Regra #1**. No decorrer do jogo, serão sorteadas 10 perguntas. Você vence, caso consiga respondê-las no prazo de 10 minutos;
- **Regra #2**. O nível de dificuldade é inversamente proporcional ao nível de habilidade dos funcionários. Quanto mais habilidosos forem os funcionários, mais fáceis são as perguntas;
- **Regra #3**. Você perde o jogo caso seu o orçamento ou o tempo acabe sem que você tenha respondido as 10 perguntas;
- **Regra #4**. A cada 2 minutos, os gastos com funcionários são descontados do orçamento;

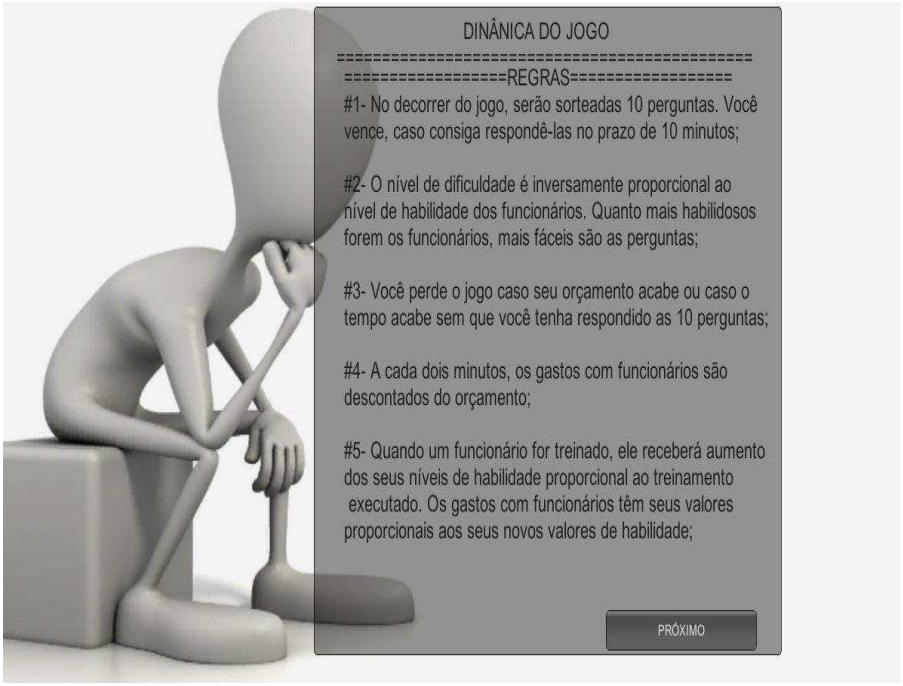

Figura 5-3 - Regras do Jogo

- **Regra #5**. Quando um funcionário for treinado, ele receberá aumento dos seus níveis de habilidade proporcional ao treinamento executado. Os gastos com funcionários têm seus valores proporcionais aos seus novos valores de habilidade;

- **Regra #6**. Cada pergunta tem pontuação correspondente ao seu nível de dificuldade, que varia:

✓ Perguntas de Nível Fácil: 0-100;

✓ Perguntas de Nível Médio: 101-300;

✓ Perguntas de Nível Difícil: 301-500.

- **Regra #7**. Para cada resposta certa, os pontos correspondentes à pergunta são somados na pontuação total;
- **Regra #8**. Para cada resposta errada, 10% da pontuação referente à pergunta são subtraídos do seu total de pontos;
- **Regra #9**. A cada três respostas erradas consecutivas, 20% do valor da última pergunta respondida errada são subtraídos do seu total de pontos;
- **Regra #10**. A cada resposta errada, o tempo é acrescido em um minuto;
- **Regra #11**. Ao responder uma pergunta certa, o funcionário que teve sua dúvida esclarecida tem suas habilidades aumentadas. O aumento de suas habilidades é proporcional ao nível de dificuldade da pergunta, sendo o aumento de 5% para perguntas fáceis, 8% para perguntas de nível médio e 10% para perguntas difíceis;
- **Regra #12**. Ao responder uma pergunta errada, o funcionário que fez a solicitação tem suas habilidades diminuídas. A diminuição de suas habilidades é proporcional ao nível de dificuldade da pergunta, sendo a diminuição de 5% para perguntas fáceis, 8% para perguntas de nível médio e 10% para perguntas difíceis;
- **Regra #13**. A pontuação final é dada pela soma de três valores:

✓ Total de pontos obtidos com respostas certas e erradas;

✓ Média final das habilidades dos funcionários, sendo x a porcentagem final da habilidade:

- Se $x \leq 15\%$, soma 300 pontos;
- Se $15\% < x \leq 30\%$ soma 600 pontos;

- Se x > 30%, soma 1000 pontos;

✓ Valor final do orçamento. Sendo x a porcentagem final do orçamento, com relação ao orçamento inicial:

- Se x ≥ 50%, soma 1000 pontos;
- Se 20%< x < 50%, soma 600 pontos;
- Se 0% < x ≤ 20%, soma 300 pontos;

5.3.3. Módulo Administrador

O Administrador no TestEG deve realizar autenticação no jogo, fornecendo login e senha nos campos apropriados (Figura 5-4). Em seguida, aparece a tela apresentada na Figura 5-5, contendo as ações possíveis que ele pode realizar: i) Cadastro de Administrador; ii) Cadastro de Usuário; e iii) Cadastro de Perguntas. Caso o Administrador opte por Cadastro de Jogador, ele é direcionado para a tela apresentada na Figura 5-6. Nessa tela, ele pode escolher uma das alternativas, em relação a jogadores: i) Cadastrar (Figura 5-7); ii) Remover (Figura 5-8); e iii) Alterar (Figura 5-9).

Na opção Cadastrar, o Administrador deve fornecer o login e o nome do Jogador e ter atenção aos dados digitados para não ocorrer em ação indesejada. O não fornecimento de algum dos campos retorna uma mensagem de erro. Na opção Remover, o Administrador pode fazer a remoção de um Jogador. Na opção Alterar, o Administrador deve fornecer o atual e o novo login e o atual e o novo nome do Jogador, mesmo que a intenção seja alterar apenas um dos campos. Os campos devem ser preenchidos para a alteração ser confirmada.

Figura 5-4 - Tela de Autenticação

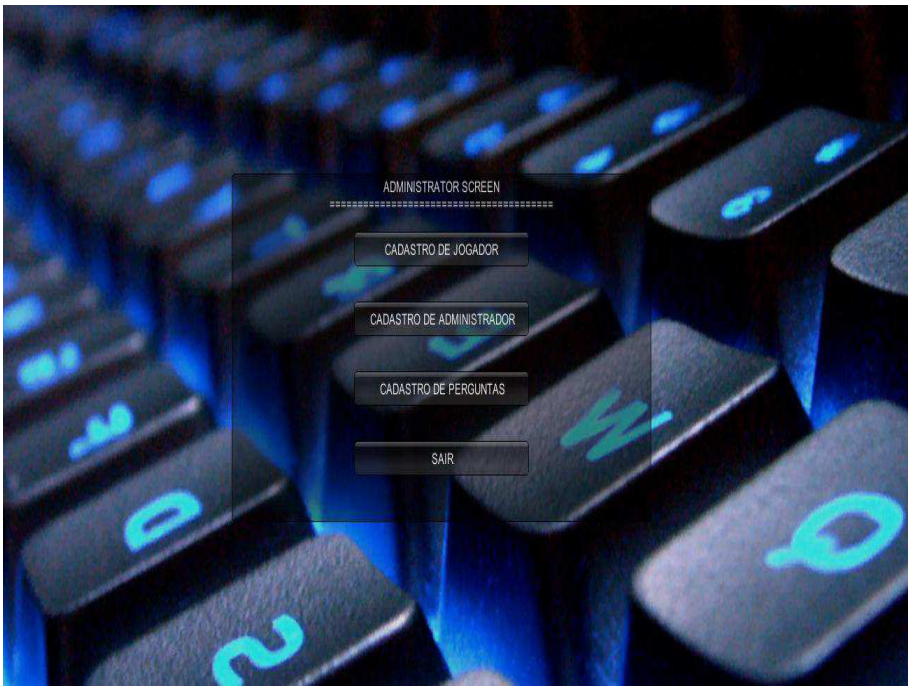

Figura 5-5 - Ações de Administrador

Caso o Administrador opte por Cadastro de Administrador, as opções são similares as de Cadastrar de Jogador: i) Cadastrar (Figura 5-10); ii) Remover (Figura 5-11); e iii) Alterar (Figura 5-12). Na opção Cadastrar, o Administrador pode efetuar o cadastro de outros administradores. Na opção Remover, o Administrador deve fornecer o login e o nome do administrador a ser removido. Na opção Alterar, o Administrador deve fornecer o atual e o novo login e o atual e o novo nome do administrador, mesmo que a intenção seja alterar apenas um dos campos. Os campos devem ser preenchidos para que a alteração seja executada com sucesso.

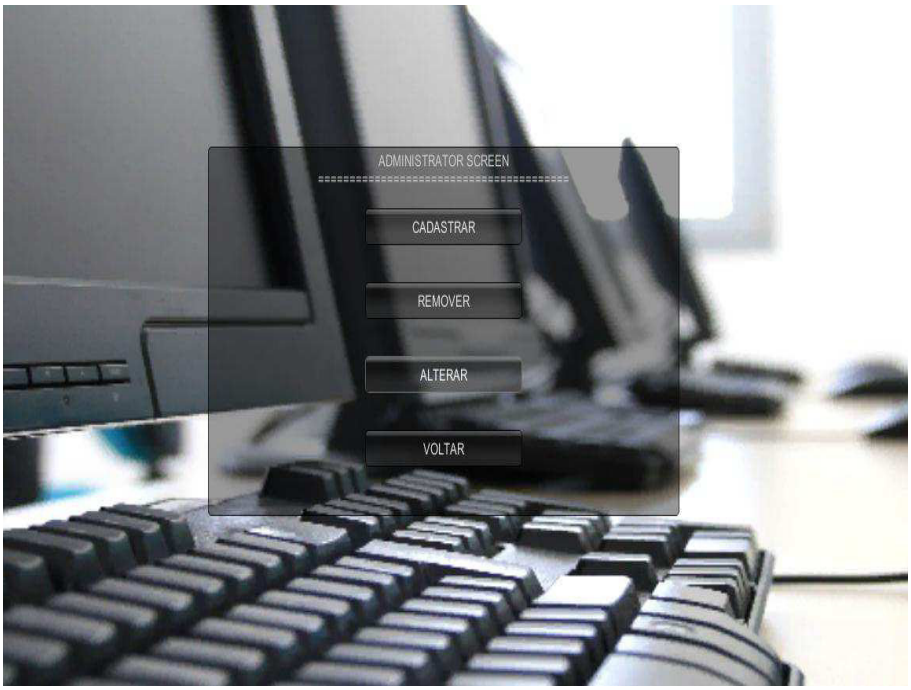

Figura 5-6 - Manutenção do Cadastro de Jogadores

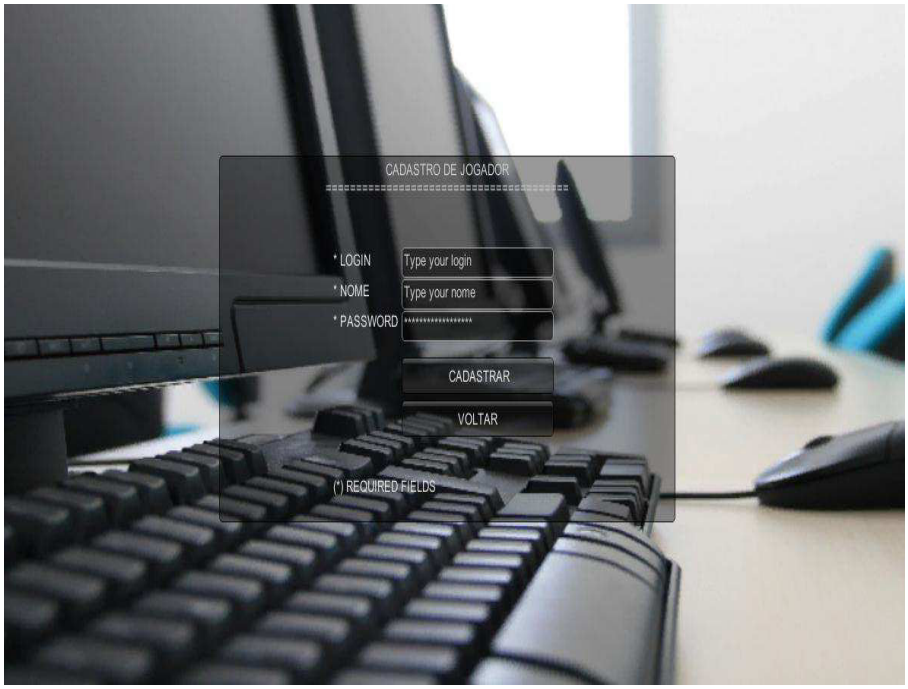

Figura 5-7 - Cadastro de Dados de Jogadores

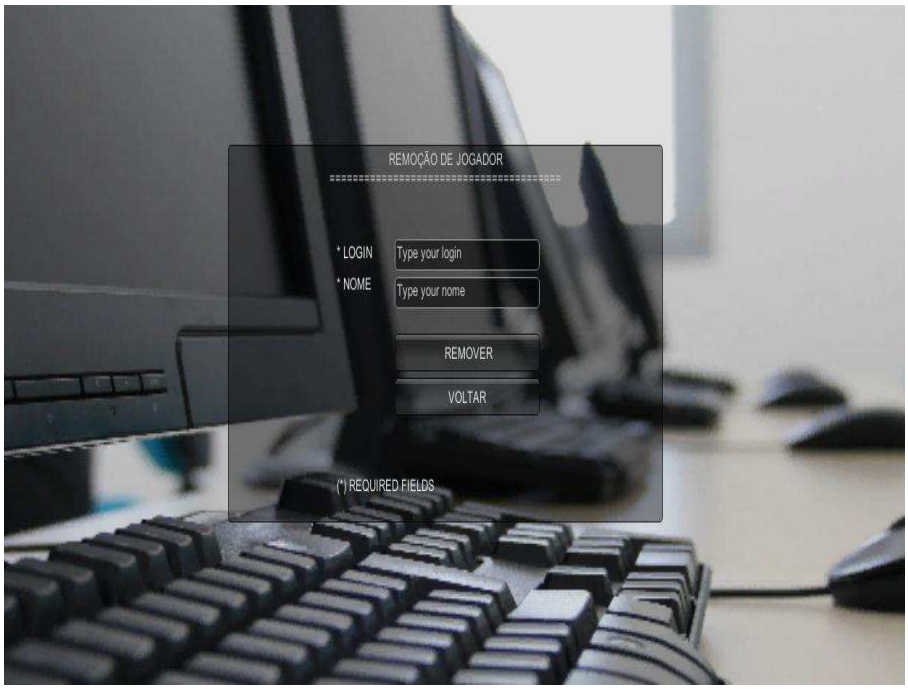

Figura 5-8 - Remoção de Jogadores

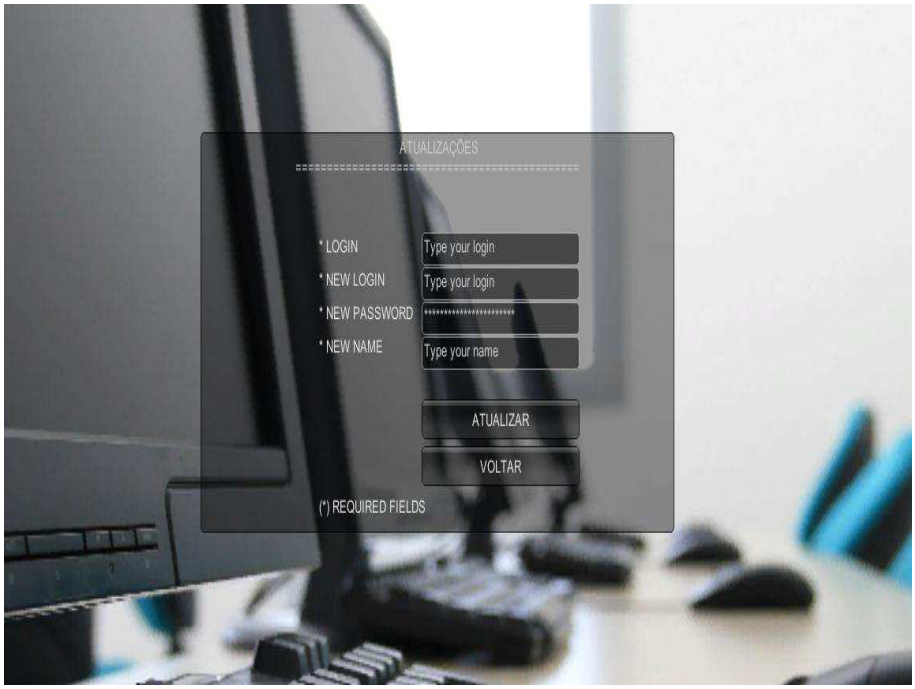

Figura 5-9 - Alteração de Dados de Jogadores

No TestEG, um dos objetivos é medir o conhecimento dos Jogadores sobre teste de software. Para tanto, ao longo do jogo, eles têm de responder diversas perguntas, cujo nível de dificuldade varia. Ao Administrador do jogo, cabe a tarefa de realizar a manutenção do cadastro de perguntas para abranger os variados conceitos acerca do assunto para "medir" o conhecimento do Jogador, assim como transmitir e aperfeiçoar os seus conhecimentos.

Figura 5-10 - Cadastro de Dados de Administradores

Figura 5-11 - Remoção de Administradores

A tela de manutenção do cadastro de perguntas é apresentada na Figura 5-13. O Administrador deve conhecer as regras do jogo que definem a pontuação das perguntas segundo o seu nível de dificuldade. Não é permitido o incluir perguntas com pontuação fora da faixa de valores definidos. Caso isso ocorra, uma mensagem de erro é enviada ao Administrador alertando-o sobre essa faixa. Após executar a inclusão correta da pergunta, o Administrador é direcionado para informar as respostas da pergunta incluída (Figura 5-14). Além disso, o Administrador tem a opção de excluir perguntas (Figura 5-15) de nível fácil, de nível médio e de nível difícil, separadamente. Ao excluir as perguntas, as respectivas respostas são excluídas.

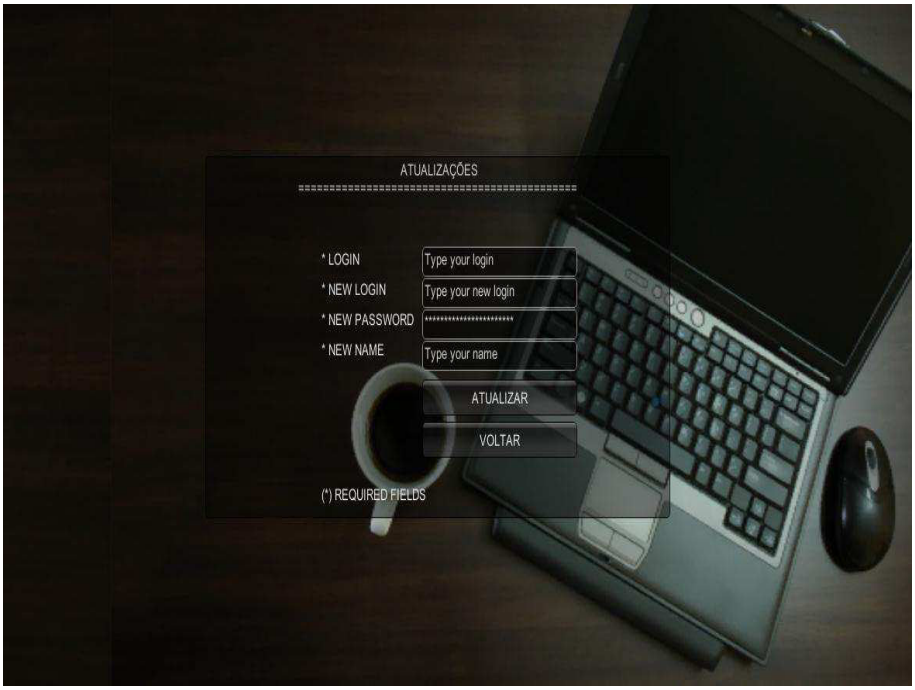

Figura 5-12 - Alteração de Dados de Administradores

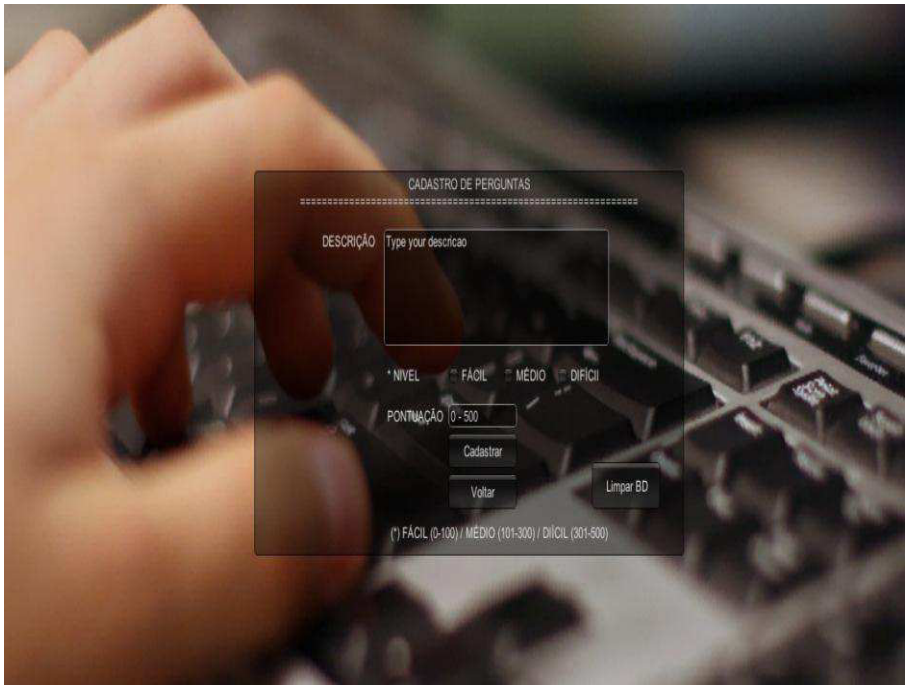

Figura 5-13 - Cadastro de Perguntas

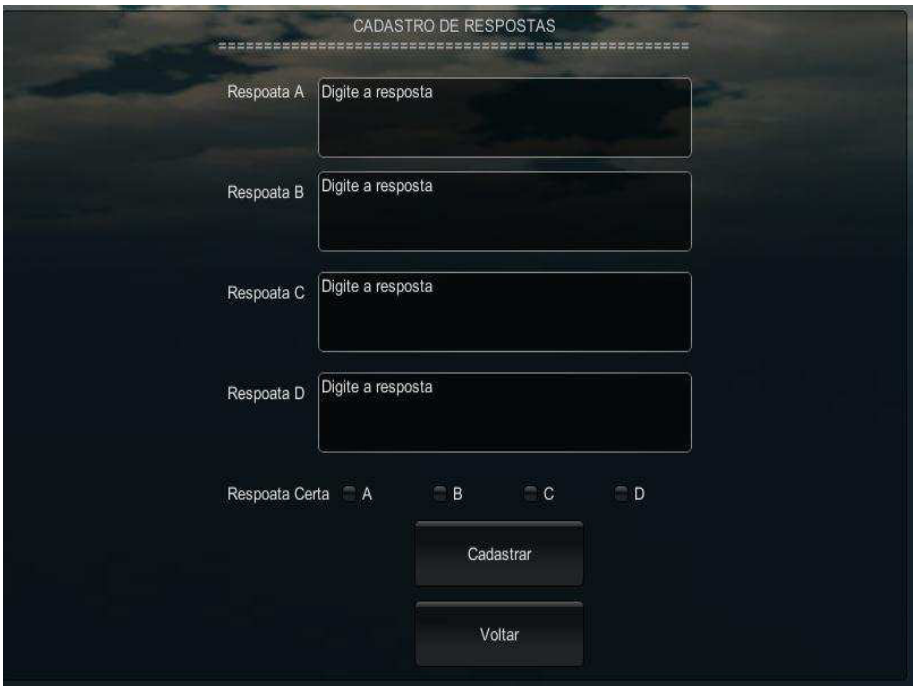

Figura 5-14 - Cadastro de Respostas

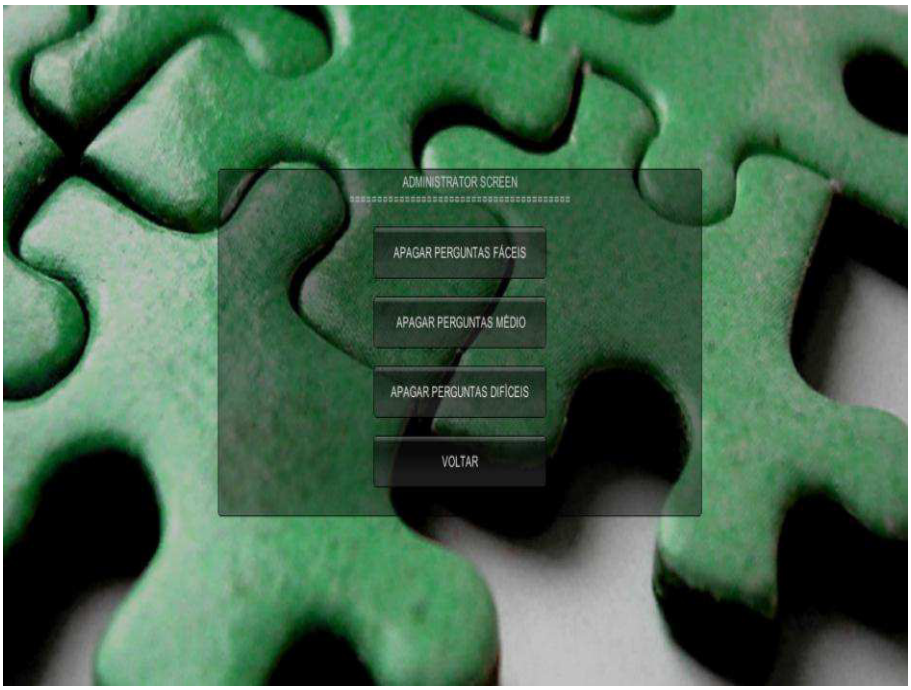

Figura 5-15 - Exclusão de Perguntas

5.3.4. Módulo Usuário

O Módulo Usuário é utilizado pelos alunos (Jogadores) para efetivamente jogar o TestEG e poder aprimorar seus conhecimentos em Teste de Software. O Jogador deve possuir login e senha cadastrada previamente pelo Administrador do jogo. Após autenticar-se escolhendo a opção Player (Figura 5-4), o Jogador é direcionado para tela apresentada na Figura 5-16, cujas ações são: i) Atualizar Dados (Figura 5-17); ii) Ver Ranking (Figura 5-18); iii) Jogar (Figura 5-19); e iv) Dinâmica do Jogo (conhecer as regras descritas na Figura 5-3). Para atualizar os dados, o Jogador deve preencher os campos e, se tudo estiver correto, a atualização é feita com sucesso. O jogador pode consultar o ranking com os três primeiros colocados (Figura 5-18) e a classificação geral com os 10 primeiros colocados (Figura 5-19). No canto superior

direito, o Jogador pode escolher um dos idiomas disponíveis no jogo: Português e Inglês.

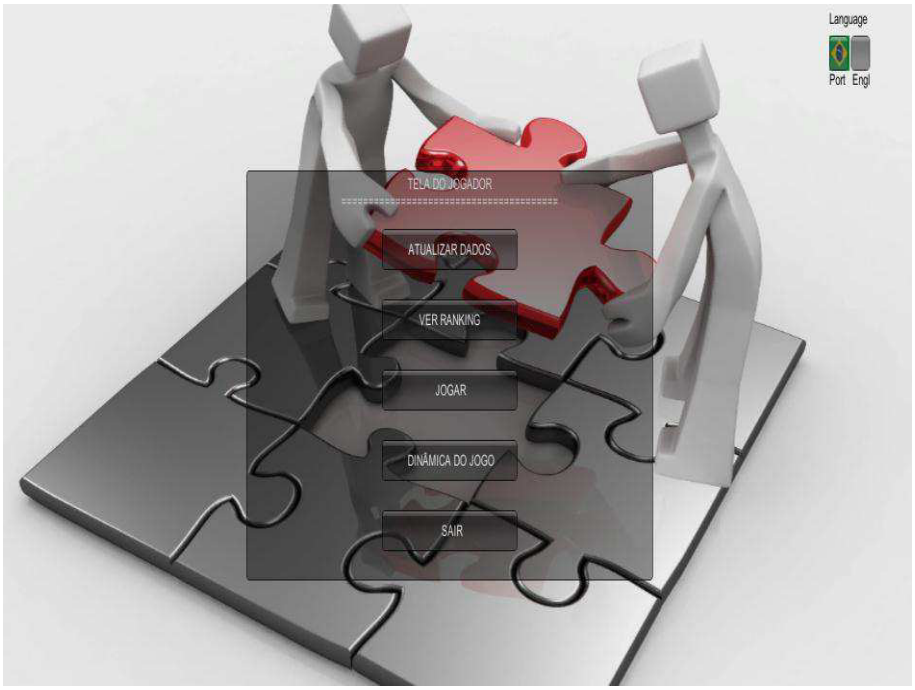

Figura 5-16 - Opções do Usuário

Essas habilidades são intrínsecas ao profissional de teste de software, sendo: i) Conhecimento do Projeto; ii) Capacidade de Diagnosticar Resolver Problemas; iii) Habilidades de Programação; e iv) Experiência. Quanto maior o nível de habilidade do funcionário, maior o seu salário e melhor o seu desempenho na execução de seu trabalho. Cabe ao Jogador decidir pelas melhores opções conforme o orçamento recebido. Ao clicar no botão Adicionar a Equipe, o funcionário torna-se automaticamente membro da equipe de teste.

Após concluir suas escolhas, o Jogador é direcionado para segunda tela com instruções (Figura 5-22) sobre o jogo e com o valor total a ser gasto com despesas de funcionário. Em seguida, o jogador é direcionado ao cenário principal do jogo, um ambiente corporativo, onde se localiza a

equipe de trabalho em plena execução de testes de software (Figura 5-23). Nesse cenário, há alguns elementos importantes que o Jogador deve estar atento:

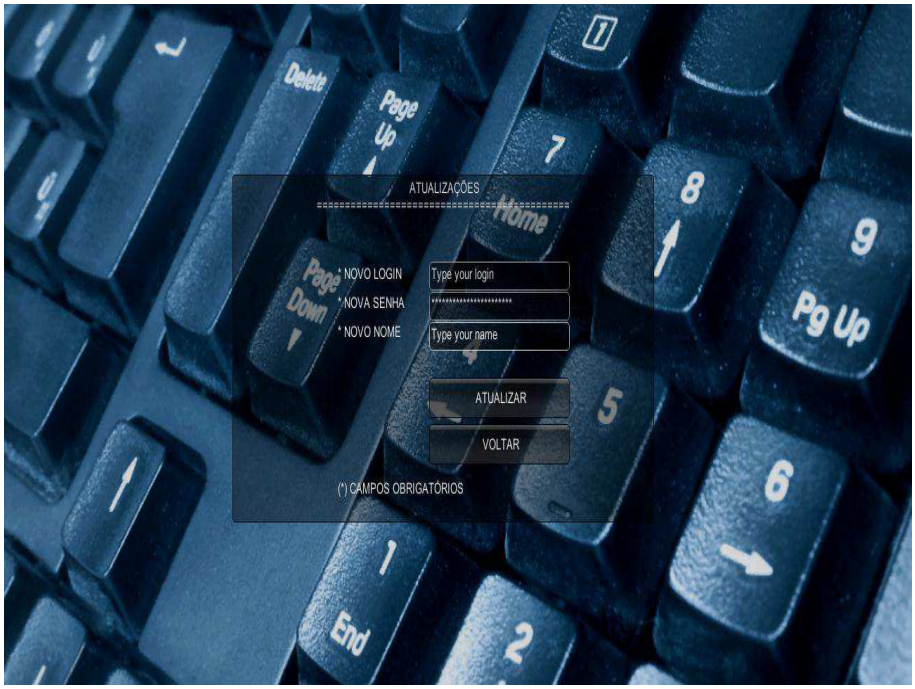

Figura 5-17 - Atualização dos Dados de Jogadores

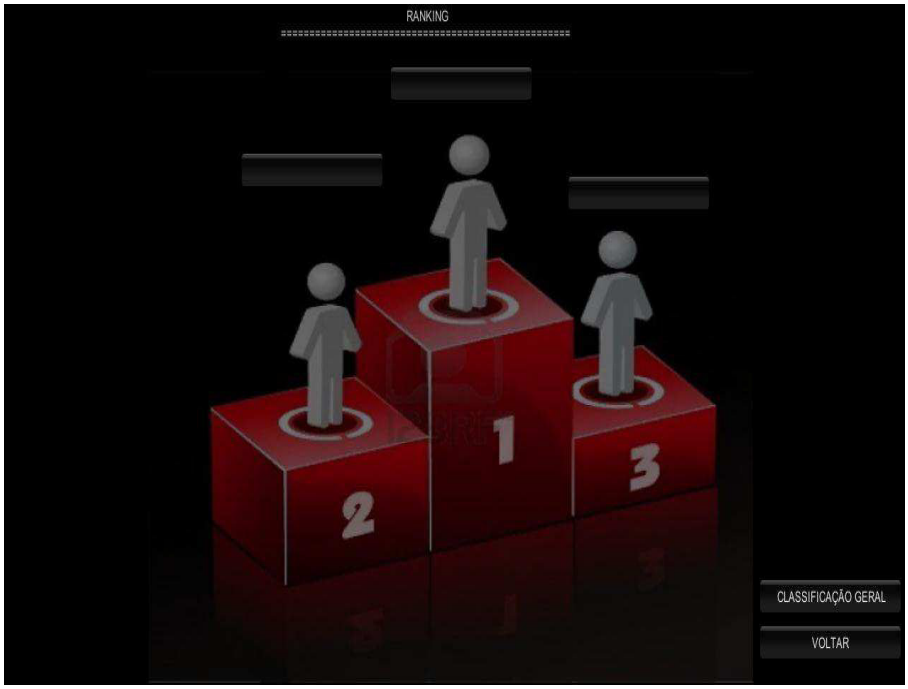

Figura 5-18 - Ranking

- **Tela de Placar**. Nessa tela estará dispostos (i) o placar com a pontuação atual do Jogador, (ii) a quantidade de respostas certas e erradas e (iii) o tempo do jogo.
- **Habilita Som**. O jogo possui efeitos de áudio, os quais podem ser desabilitados;

Figura 5-19 - Classificação Geral

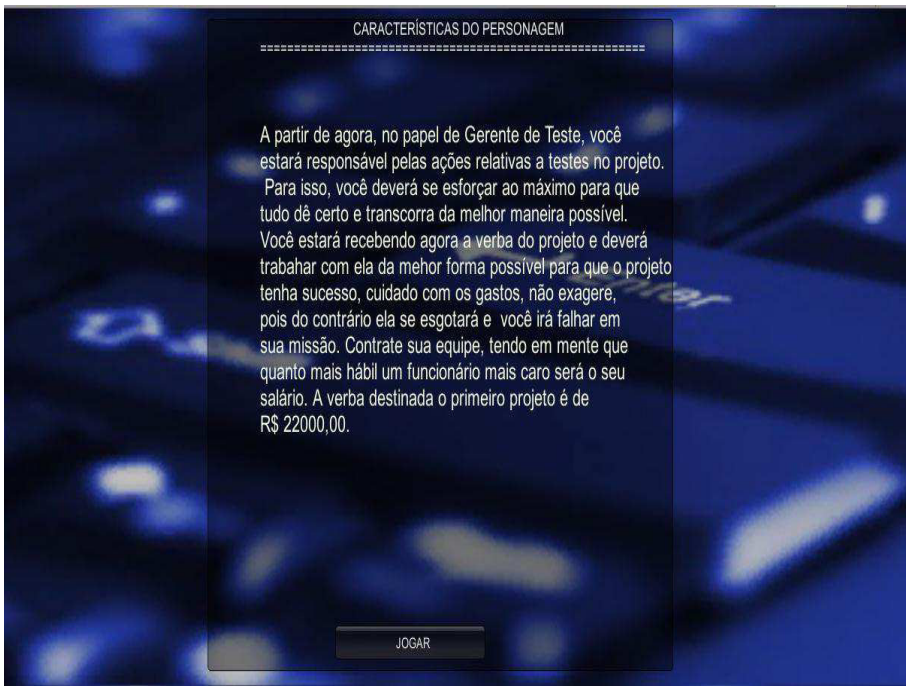

Figura 5-20 - Tela de Instruções

- **Barra de Recursos**. Essa barra contém a porcentagem de recurso disponível. A cada dois minutos, o valor gasto com funcionário é subtraído do orçamento. O valor presente na barra sofrer mudanças conforme as ações do Jogador.

Quando um funcionário possui alguma dificuldade com o seu trabalho, ele solicita ao Jogador (Gerente de Teste) que o ajude. A sinalização do pedido de ajuda é por meio de uma imagem (cubo escrito 'HELP') (Figura 5-24). Esse pedido sugere ao Jogador que se encaminhe até a mesa do funcionário. Ao encostar-se à mesa, uma tela com uma pergunta do funcionário é apresentada (Figura 5-25). Ao respondê-la, o jogador pode aumentar, caso a resposta do jogador esteja correta, ou diminuir, caso a resposta do jogador esteja errada, as habilidades do funcionário. Resposta errada é aplicada penalidade de 1 minuto.

Figura 5-21 - Escolha do Personagem

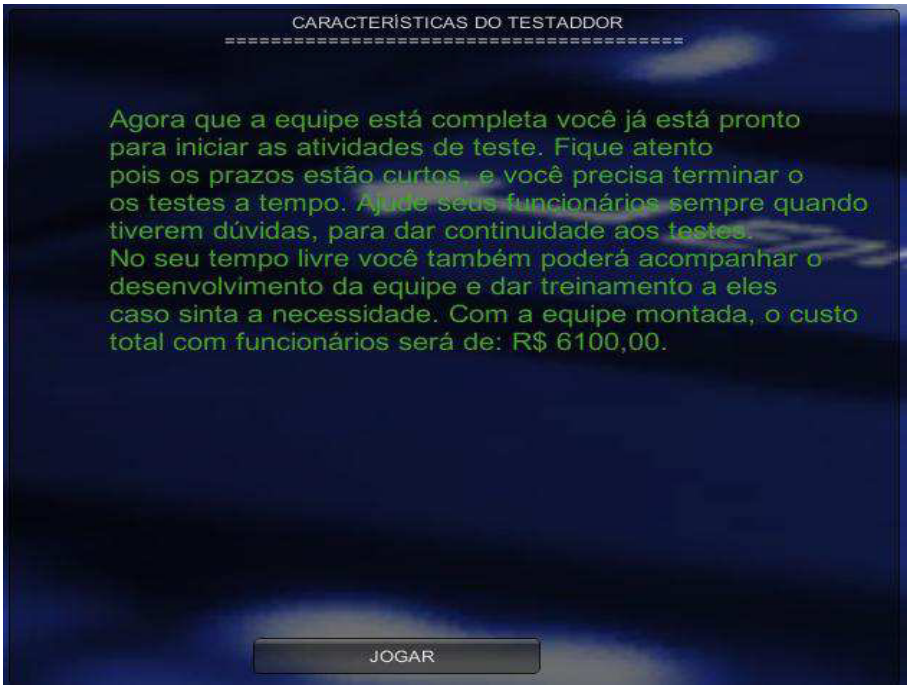

Figura 5-22 - Instruções para o Jogo

Figura 5-23 - Cenário do Jogo

Figura 5-24 - Pedido de Ajuda Pelo Funcionário

No decorrer do jogo, o Jogador pode executar algumas ações para melhorar os seus conhecimentos sobre Teste de Software e capacitar a sua equipe. Para isso, ele deve caminhar em direção à mesa do gerente e, ao encostar, aparece à tela apresentada na Figura 5-26. Nessa tela, o Jogador tem acesso ao conteúdo teórico de Teste de Software. Outra ação importante que ele pode executar é verificar o perfil dos funcionários contratados. Nessa verificação, podem-se observar os níveis de habilidade do funcionário e treiná-lo (Figura 5-27).

Figura 5-25 - Tela de Perguntas

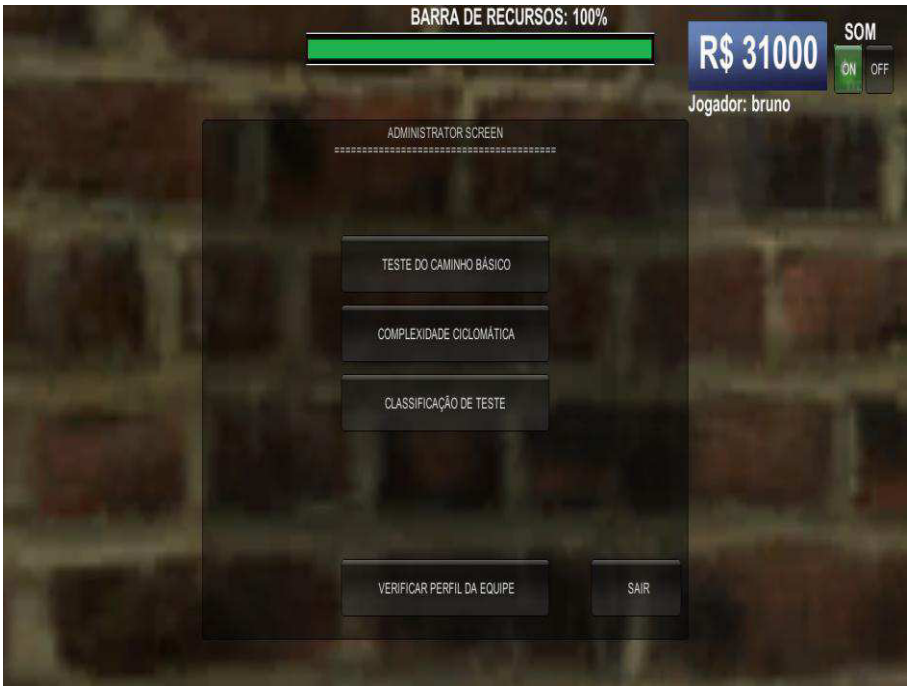

Figura 5-26 - Ações do Gerente de Teste

Ao analisar essas habilidades, o Jogador pode optar por treiná-lo, aperfeiçoando suas habilidades. A realização do treinamento requer atenção do Jogador quanto ao orçamento e ao tempo disponível. O treinamento varia na quantidade de tempo e no preço gastos, sendo essa variação proporcional ao tipo de treinamento escolhido (Figura 5-28). Ao treinar o funcionário (Figura 5-29), o Jogador fica incapacitado, momentaneamente, de executar qualquer ação e deve aguardar a conclusão do treinamento.

Figura 5-27 - Perfil da Equipe

Figura 5-28 - Tipos de Treinamento

Figura 5-29 - Realizar Treinamento da Equipe

O Jogador tem êxito no jogo se responder 10 perguntas, dentro do prazo estipulado e sem esgotar o orçamento, e é direcionado para a tela (Figura 5-30) em tem as opções: i) Reiniciar; ii) Ver Ranking; e iii) Sair. A pontuação final considera fatores, tais como, o nível de habilidade final de cada funcionário, a quantidade de respostas certas e o orçamento final. Caso contrário, quando o Jogador "estoura" (i) o orçamento e/ou (ii) o tempo do projeto, ele é direcionado para a tela apresentada na Figura 5-31.

Figura 5-30 - Vitória do Jogador

Figura 5-31 - Derrota do Jogador

5.4. Considerações Finais

O objetivo do jogo é tornar mais lúdico o processo de ensino-aprendizagem aos assuntos de Teste de Software, mais atrativo ao aluno e mais próximo do ambiente de trabalho real. Ao jogar o TestEG, o aluno pode entender melhor os conceitos de teste e sentir a responsabilidade de criar e comandar uma equipe de trabalho. Houve a preocupação com o controle do professor com relação ao que é "cobrado" do aluno. Para isso, o jogo foi dividido em dois módulos: i) Modulo Administrador e ii) Módulo Jogador.

No decorrer do jogo, o aluno toma decisões intrínsecas a um Gerente de Teste e exerce atividades de um testador de software ao interagir com os funcionários. No jogo, o propósito é contribuir com o ensino de Teste de Software; para tanto, é necessário que o professor utilize o Módulo Administrador para controlar o conteúdo a ser

transmitido ao aluno, para que possa avaliá-lo e sentir quais são realmente as suas maiores necessidades.

6. AVALIAÇÃO

6.1. Considerações Iniciais

Com o objetivo de obter informações sobre a relevância da utilização do TestEG no processo de ensino-aprendizagem em teste de software, foi realizada uma avaliação por meio de aplicação de um questionário, enviado a 127 participantes (jogadores), com o propósito de verificar características como aceitação, jogabilidade e satisfação.

Informações sobre os meios utilizados para realizar a avaliação são apresentadas na Seção 6.2. Resultados obtidos após as avaliações são discutidos na Seção 6.3.

6.2. Avaliação

De maneira geral, graduandos e graduados na área de Computação, estão acostumados a utilizar jogos computacionais, visto que eles estão em constante contato com esses jogos. Como o foco do TestEG é apoiar o processo de ensino-aprendizagem relacionado a teste de software, alunos da Universidade Federal de Lavras cursando ou que cursaram disciplinas que abordam o assunto e profissionais atuantes na área foram convidados por e-mail para serem avaliadores. Além disso, um e-mail convite foi enviado à lista de discussão da SBC (Sociedade Brasileira de Computação - sbc-l). Várias pessoas entraram em contato para utilizar o jogo e contribuíram com a avaliação.

Um questionário foi elaborado e disponibilizado na ferramenta SurveyMonkey[4], cujas perguntas são referentes à utilização e à importância do TestEG no processo de ensino-aprendizagem. O link para

[4] http://pt.surveymonkey.com/

esse questionário foi enviado por e-mail aos convidados. No questionário, além das perguntas objetivas, há espaço destinado a comentários para os avaliadores expressarem suas impressões após a utilização do TestEG. Esse questionário seguiu um modelo proposto [Savi et al., 2010] para avaliação de jogos educacionais. As perguntas foram organizadas em (Figura 6-1): i) caracterização do respondente; ii) motivação; iii) experiência; iv) conhecimento; e v) validação.

6.3. Resultados

O convite para responder o questionário foi enviado para 127 pessoas, sendo que 52 (41%) responderam. Quanto à titulação dos avaliadores, 8 avaliadores (15,4%) possuem doutorado; 10 avaliadores (19,2%) possuem mestrado, 10 avaliadores (19,2%) são graduação e 24 avaliadores (46,2%) cursam graduação. A primeira seção de perguntas foi destinada a caracterizar os respondentes quanto à escolaridade dos avaliadores (Figura 6-2). Pode-se perceber que a maioria dos avaliadores é graduando. A segunda seção de perguntas foi destinada a caracterizar a experiência dos avaliadores em Engenharia de Software (Figura 6-3). O resultado foi 11 avaliadores (21,2%) têm mais de 3 anos de experiência, 16 avaliadores (30,8%) têm até 3 anos de experiência e 25 avaliadores (48,1%) não têm experiência.

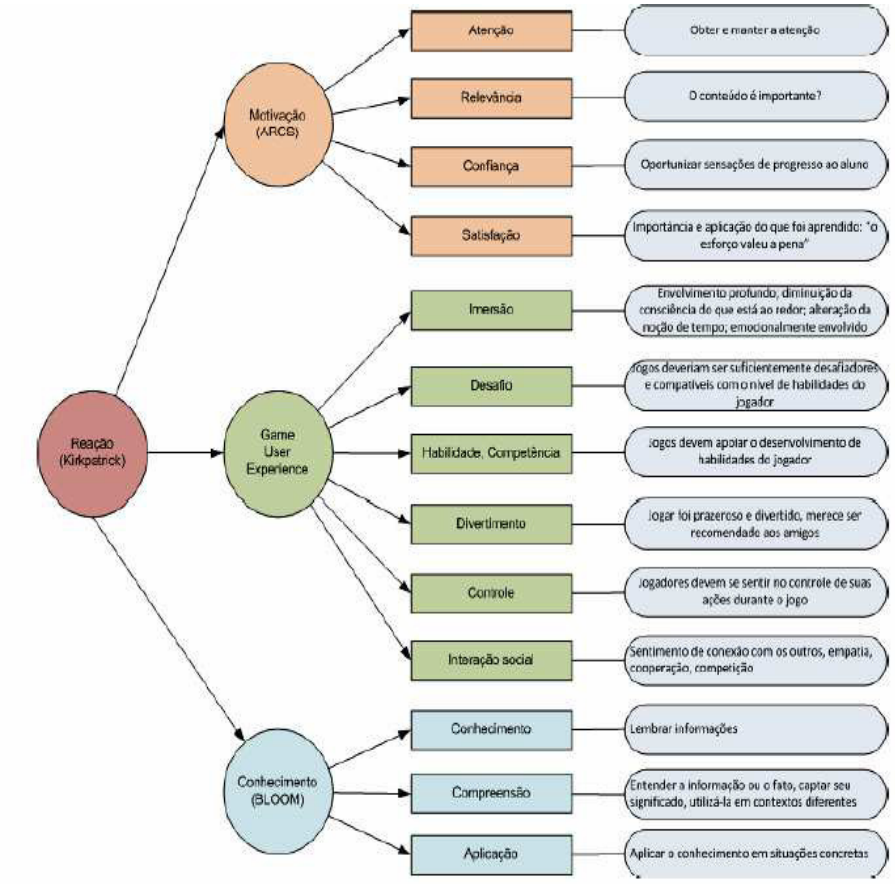

Figura 6-1 - Modelo de Avaliação de Jogos Educacionais

Para verificar o nível de motivação do avaliador com relação ao TestEG, foram elaboradas 7 afirmações, paras quais os avaliadores deveriam escolher uma das seguintes alternativas: i) "Discordo Plenamente"; ii) "Discordo"; iii) "Concordo"; e iv) "Concordo Plenamente". O resultado é apresentado na Figura 6-4.

Analisando a avaliação, 40 avaliadores (83,3%) acharam o design da interface adequado, porém a dificuldade de entender dividiu a opinião dos avaliadores, pois 25 (53,2%) responderam não foi difícil enquanto 22 (46,8%) responderam o contrário. Isso é um indicador que o TestEG ainda precisa ser melhorado para reduzir/eliminar essa dificuldade.

Quanto ao TestEG ser relevante aos seus interesses e possuir conteúdo útil, 40 avaliadores (83,4%) e 37 avaliadores (84,4%) responderam que concordam ou concordam plenamente com essas afirmações, respectivamente. Quanto à quantidade de informações disponíveis e surpreendentes/inesperadas fornecida pelo TestEG, 31 avaliadores (64,6%) responderam que discordam ou discordam plenamente que o TestEG passa muitas informações ao jogador e 32 avaliadores (68,1%) responderam apreenderam informações surpreendentes/inesperadas. Porém cabe realizar avaliação para melhorar o TestEG, tendo em vista que há 17 avaliadores (35,4%) responderam que há muita informação e 15 avaliadores (31,9%) responderam que não apreenderam informações surpreendentes/inesperadas. Dos avaliadores, 33 (68,7%) responderam que se sentiram bem ao concluir o jogo, pois tiveram experiência positiva com o jogo visto, julgando-o estimulante e não se sentindo ansiosos ou entediados. Cabe ressaltar que há diferença em alguns percentuais, pois há variação na quantidade de avaliadores, sendo 48 avaliadores, 48 avaliadores, 45 avaliadores, 47 avaliadores, 48 avaliadores, 47 avaliadores e 48 respondentes, respectivamente.

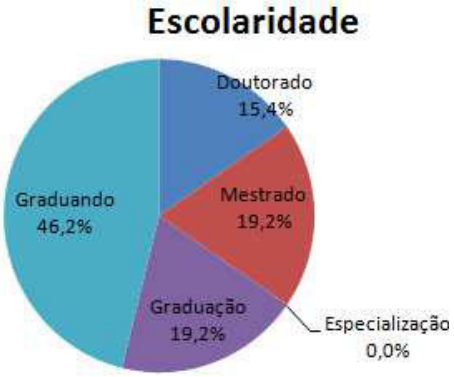

Figura 6-2 - Grau de Escolaridade dos Respondentes

Figura 6-3 - Experiência dos Respondentes

Para verificar a experiência do avaliador após utilizar o TestEG, foram elaboradas 3 afirmações, paras quais os avaliadores deveriam escolher uma das seguintes alternativas: i) "Discordo Plenamente"; ii) "Discordo"; iii) "Concordo"; e iv) "Concordo Plenamente". O resultado é apresentado na Figura 6-5.

Analisando a avaliação, foi um consenso entre os avaliadores quanto facilidade de (i) lembrar-se de mais informações relacionadas ao tema apresentado, (ii) compreender melhor os temas apresentados no jogo e (iii) aplicar melhor os temas relacionados com o jogo. A maioria respondeu concordam ou concordam plenamente, sendo 39 avaliadores (81,3%), 37 avaliadores (77,1%) e 37 avaliadores (77,1%), respectivamente. Com o resultado, pode-se inferir que o TestEG alcançou seu objetivo como ferramenta educacional, pois ampla maioria dos avaliadores sentiu que obtiveram um bom aprendizado com o TestEG. Cabe ressaltar que 48 avaliadores responderam as afirmações.

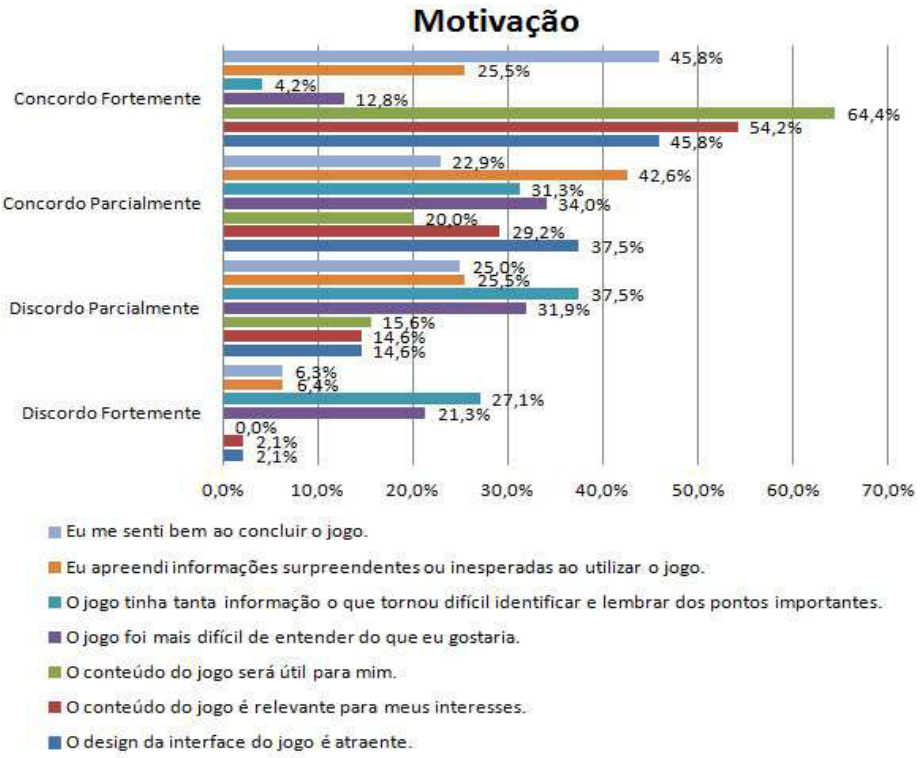

Figura 6-4 - Nível de Motivação com Relação ao TestEG

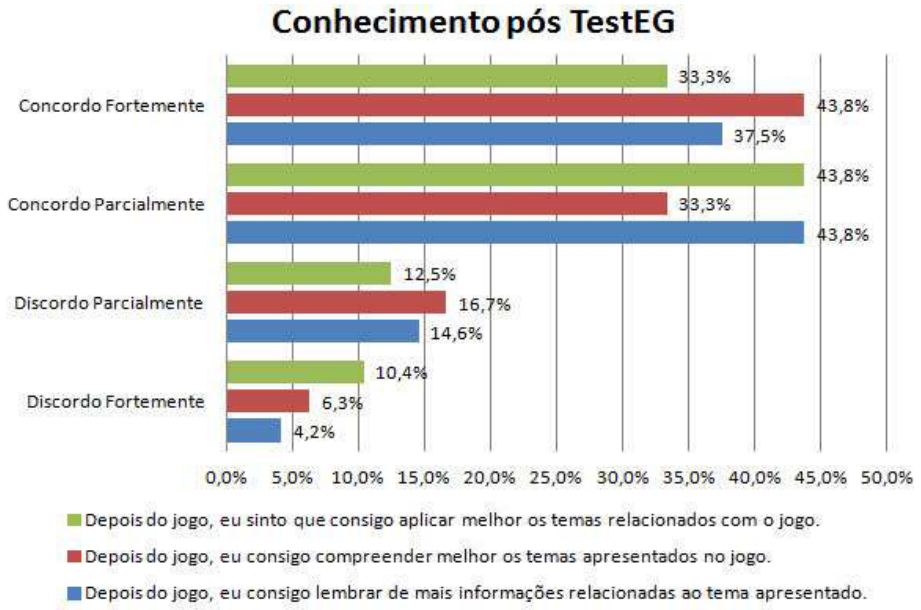

Figura 6-5 - Conhecimento dos Avaliadores após Utilizarem o TestEG

Para verificar se o TestEG contribui para a aprendizagem e desempenho profissional, os avaliadores deveriam escolher uma das seguintes alternativas: i) "Discordo Plenamente"; ii) "Discordo"; iii) "Concordo"; e iv) "Concordo Plenamente". O resultado é apresentado na Figura 6-6 e na Figura 6-7, respectivamente.

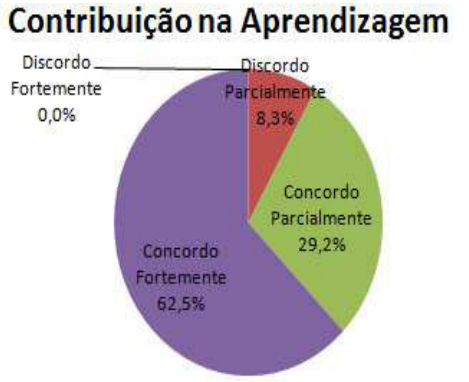

Figura 6-6 - Contribuição na Aprendizagem

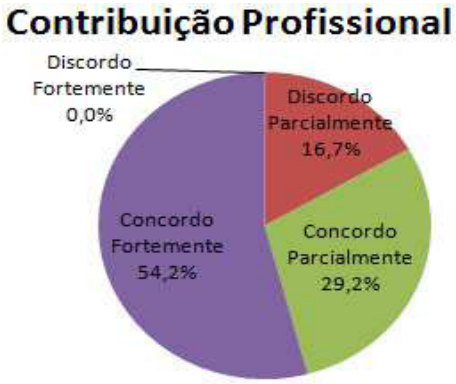

Figura 6-7 - Contribuição Profissional

Analisando a avaliação, a maioria dos avaliadores acredita que o TestEG pode contribuir no processo de ensino-aprendizagem e na sua atuação profissional, pois 44 avaliadores (91,7%) e 40 avaliadores (83,3%) concordam ou concordam plenamente, respectivamente. Cabe ressaltar que 48 avaliadores responderam ambas as afirmações.

6.4. Considerações Finais

Com resultados obtidos por meio da análise das respostas dos avaliadores do jogo educacional computacional TestEG, utilizando um questionário, pode-se caracterizar esse jogo como uma opção para apoiar o processo de ensino-aprendizagem em Teste de Software. Assim, o TestEG adequa-se às exigências de um software educacional e pode ser útil nas salas de aulas, fazendo com que os alunos sintam-se em um

ambiente de trabalho real e exerçam as funções pertinentes a um Gerente de Teste.

O questionário de avaliação contribuiu com críticas e sugestões que posteriormente serão utilizadas para a atualização do TestEG e, consequentemente, a implementação da próxima versão. Nesse questionário, foi disponibilizada uma questão dissertativa aos avaliadores para fazerem comentários/críticas/sugestões. Nesta questão, muitos elogios foram feitos a iniciativa de desenvolvimento do jogo e a maneira como foi proposto e críticas também ocorreram. Tais críticas são pertinentes e podem ser utilizadas para melhorar o TestEG. Dos 52 avaliadores, 29 (55,8%) responderam essa questão. Fazendo uma análise das respostas, podem ser destacados os seguintes pontos:

- Melhorar a forma de conduzir o Gerente de Testes pelo cenário;
- Falta de diversificação de projetos a serem desenvolvidos;
- Melhoria na forma de apresentar as instruções;
- Proporcionar mais interatividade, com desafios que levem a próximas fases.

7. CONSIDERAÇÕES FINAIS

Vários jogos educacionais foram desenvolvidos com o propósito de aperfeiçoar e tornar mais efetivo o processo de ensino-aprendizagem [Santos et al., 2008]. Entretanto, pouco ou nenhum software contempla a área de Teste de Software [Wangenheim et al., 2009], constatando a real necessidade de criação do jogo desenvolvido neste trabalho visando ao apoio do professor. Nesse contexto, este trabalho apresentou o TestEG, um software educacional para apoiar o processo de ensino-aprendizagem de Teste de Software. Nesse jogo, busca-se facilitar o aprendizado dos alunos que passam a aprender testes de software de forma lúdica e próximo ao que se encontra em um ambiente real de trabalho.

Neste capítulo, são apresentadas algumas observações resultantes deste trabalho. Breve conclusão do trabalho é apresentada na Seção 8.1. Algumas contribuições obtidas com a realização deste trabalho são citadas na Seção 8.2. Limitações são explicadas na Seção 8.3. Alguns trabalhos futuros a serem desenvolvidos como desdobramento deste são sugeridos na Seção 8.4.

7.1. Conclusões

O desenvolvimento deste projeto foi possível por causa de estudos e análises quanto aos jogos educacionais e seu funcionamento. Houve preocupação quanto ao entendimento do efeito dos jogos no aprendizado do aluno e quais as principais causas de falha nesse processo. No jogo desenvolvido, o objetivo foi fazer com que o aluno sinta-se mais familiarizado com um ambiente real de trabalho e entenda o seu funcionamento. Ao jogar, o aluno exerce o papel de testador de software e de "tomador" de decisões realizando atividades de um Gerente de Teste

ao contratar e capacitar os funcionários. Dessa forma, ele pode entender um pouco mais sobre as responsabilidades intrínsecas de um Gerente de Teste.

A avalição do jogo possibilitou concluir que é grande a aceitação dos alunos pela utilização de jogos. Cabe aos professores estudarem uma maneira de incluir a utilização de jogos educacionais no processo de ensino-apredizagem e dominar a condução desse processo.

7.2. Contribuições

Neste trabalho, a principal contribuição é o desenvolvimento de um software educacional - TestEG - para apoiar o processo de ensino-aprendizagem. Esse jogo reúne práticas de teste de software e a aplicação delas em uma simulação de ambiente real de trabalho. Espera-se que, com esse jogo, haja evolução de conhecimento aos alunos, tornando o aprendizado fácil, atrativo e lúdico, fazendo com que o aluno divirta-se enquanto aprende, e a lacuna entre teoria e prática torne-se mais estreita.

7.3. Limitações

Esta seção é destinada a apresentar algumas limitações da TestEG identificados durante o desenvolvimento deste trabalho:

- Versão web disponível apenas para navegadores com sistemas operacionais Windows;
- Requer instalação do servidor banco de dados MySql;
- Não permite que vários usuários joguem ao mesmo tempo;

7.4. Trabalhos Futuros

A seguir, são apresentados alguns tópicos que podem ser considerados para dar continuidade deste trabalho:

- Possibilitar ao administrador incluir imagens nas perguntas para facilitar a visualização de gráficos, de diagramas, de tabelas e etc.;
- Tornar o jogo multiplayer, fazendo com que cada jogador assuma um papel dentro da equipe, seja na forma do membro da equipe de teste ou de Gerente de Teste;
- Utilizar o mouse para navegação do personagem principal (Gerente de Teste);
- Criar diversificação de projetos para serem desenvolvidos pelo personagem no jogo;
- Inserir navegabilidade para o conjunto de regras;
- Criar desafios que levem a próximas fases, tornando o jogo mais interativo.

REFERÊNCIAS BIBLIOGRÁFICAS

Aranha, G. Jogos Eletrônicos Como um Conceito Chave para o Desenvolvimento de Aplicações Imersivas e Interativas para o Aprendizado. In: Ciências e Cognição. v. 7. pp. 105-110. 2006.

Baker, A.; Navarro, E. O.; Hoek, V. A. An Experimental Card Game for Teaching Software Engineering Processes. In: Journal of Systems and Software. v. 75. pp. 3-16. 2005.

Barbosa, E. F.; Maldonado, J. C.; Vincenzi, A. M. R.; Delamaro, M. E.; Souza, S. R. S.; Jino, M. Introdução ao Teste de Software. In: II Escola de Informática Norte. pp. 330-378. 2000.

Beizer, B. Black-Box Testing: Techniques for Functional Testing of Software and System. Wiley. 320p. 1995.

Benitti, F. B. V.; Molléri, J. S. Utilização de um RPG no Ensino de Gerenciamento e Processo de Desenvolvimento de Software. In: Workshop sobre Educação em Computação. pp. 258-267. 2008.

Bernardi, G.; Fontoura, L. M.; Cordenonsi, A. Z. Elicit@ção: Ferramenta de Apoio ao Ensino de Elicitação de Requisitos de Software baseada em Instituições Eletrônicas. In: II Workshop Escola de Sistemas de Agentes para Ambientes Colaborativos. 2008.

Braga, A. J.; Araújo, M. M. de; Vargas, S. R. S.; Lemes, A. Uso dos Jogos Didáticos em Sala de Aula. 2007. Disponível em <http://guaiba.ulbra.tche.br/pesquisas/2007/artigos/letras/242.pdf>. Acessado em: 15/02/2009.

Burnstein, I. Practical Software Testing: A Process-oriented Approach. Springer-Verlag.709p. 2003.

Crespo, A. N.; Silva, O. J.; Borges, C. A.; Salviano, C. F.; Teive, M.; Jino, M. Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo. In: III Simpósio Brasileiro de Qualidade de Software. pp. 271-285. 2004.

Dantas, A. R. Jogos de Simulação no Treinamento de Gerentes de Projetos de Software. Dissertação de Mestrado em Engenharia de Sistemas e

Computação. Programa de Pós-Graduação em Engenharia de Sistemas e Computação. UFRJ. 98p. 2003.

Delamaro, M.; Maldonado, J. C.; Jino, M. Introdução ao Teste de Software. Campus. 408p. 2007.

Drappa, A.; Ludewig, J. Quantitative Modelling for the Interactive Simulation of Software Projects. In: Journal of Systems and Software. v. 46. pp. 113-122. 1999.

Drappa, A.; Ludewig, J. Simulation in Software Engineering Training. In: International Conference on Software Engineering. pp. 199-208. 2000.

Dustin, E. Effective Software Testing: 50 Specific Ways to Improve Your Testing. Addison-Wesley Professional. 304p. 2003.

Fernando, L.; Werner, C. M. L. Sobre o Uso de Jogos Digitais para o Ensino de Engenharia de Software. In: II Fórum de Educação em Engenharia de Software. pp. 17-24. 2009.

Gomes, A. S.; Castro Filho, J. A. de ; Gitirana, V.; Spinillo, A.; Alves, M.; Melo, M.; Ximenes, J. Avaliação de Software Educativo para o Ensino de Matemática. In: Workshop de Informática na Educação. 2002. v. 5. Disponível em: <http://www.cin.ufpe.br/~asg/home.php?p=publications>. Acessado em: 25 de julho de 2011.

Velasco, C. G. Brincar, O Despertar Psicomotor. Sprint. 107p. 1996.

Grando, R. C. O Jogo na Educação: Aspectos Didático-Metodológicos do Jogo na Educação Matemática. Disponível em: <www.cempem.fae.unicamp.br/lapemmec/cursos/el654/2001/jessica_e_paula/JOGO.doc>. Acessado em: 13/5/2012.

Hilburn, T. B.; Towhidnejad, M. A Case for Software Engineering. In: Conference on Software Engineering Education & Training. pp. 107-114. 2007.

IEEE. IEEE Computer Society. IEEE Std 829: Standard for Software Test Documentation. September. 1990.

Isotton, E. Scrumming - Ferramenta Educacional para Ensino de Práticas de SCRUM. Trabalho de Conclusão de Curso em Bacharelado em Sistemas

de Informação. Pontifícia Universidade Católica do Rio Grande do Sul. 80p. 2008.

Jain, A.; Boehm, B. SimVBSE: Developing a Game for Value-Based Software Engineering. In: 19th Conference on Software Engineering Education and Training. pp. 103-114. 2006.

Jiantao, P. Software Testing. 1999. Disponível em: <http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/>. Acessado em: 1/12/2008.

Jung, C. F. Metodologia Aplicada a Projetos de Pesquisa: Sistemas de Informação & Ciência da Computação. Taquara, 2009. Disponível em: <http://www.jung.pro.br/moodle/mod/resource/view.php?id=102>. Acessado em: 13/5/2012.

Kahl, K.; Lima, M. E de O.; Gomes, I. Alfabetização: Construindo Alternativas com Jogos Pedagógicos. In: Revista Eletrônica de Extensão. v. 4. n. 5. 2007.

Kieling, E.; Rosa, R. Planager - Um Jogo para Apoio ao Ensino de Conceitos de Gerência de Projetos de Software. Trabalho de Conclusão de Curso de Ciência da Computação. Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Norte. 68p. 2006.

Krüger, S. E.; Fritsh, E. F.; Viccari, R. M. Avaliação Pedagógica do Software STR. In: Revista Brasileira de Informática na Educação. v. 8. pp. 21-33. 2001.

Lopes, M. V. O.; Silva, V. M.; Araujo, T. L. Desenvolvimento Lógico-Matemático do Software ND. In: Revista Latino Americana de Enfermagem. v. 12. n. 1. pp. 92-100. 2004.

Myers, G. J. The Art of Software Testing. Wiley. 192p. 1979.

Monsalve, E.; Werneck, V.; Leite, J. C. S. do P. Evolución de un Juego Educacional de Ingeniería de Software através de Técnicas de Elicitación de Requisitos. In: XIII Workshop on Requirements Engineering. pp. 63-74. 2010.

Moraes, M. B. S.; Dutra, D. L.; Anjos, U. U.; Rego, R. G.; Moraes, R. M.; Machado, L. S. Geoplano: Um Jogo Educacional Inteligente para o

Ensino de Geometria Plana. In: International Conference on Engineering and Technology Education. pp. 559-563. 2008.

Moratori, P. B. Por que Utilizar Jogos Educativos no Processo de Ensino Aprendizagem? 2003. Disponível em: <http://www.nce.ufrj.br/ginape/publicacoes/patrickmaterial/trabfinalpatrick2003.pdf>. Acessado em: 26/03/2012.

Myers, G. J. The Art of Software Testing. John Wiley & Sons. 192p. 1979.

Nauman, M.; Uzair, M. SE and CS Collaboration: Training Students for Engineering Large, Complex Systems. In: Conference on Software Engineering Education and Training. pp.167-174. 2007.

Oh, E.; Hoek, A. Adapting Game Technology to Support Individual and Organizational Learning. In: International Conference on Software Engineering & Knowledge Engineering. pp. 347-354. 2001.

Paludo, L.; Raabe, A. L. A. Análise de Jogos Educativos de Computador para Gerência de Projetos de Software. In: XVIII Workshop sobre Educação em Computação. v. 1. p. 867-876. 2010.

Pfleeger, S. L. Software Engineering: Theory and Practice. Prentice Hall. 576 p. 2001.

Prensky, M. Digital Game-Based Learning. McGraw-Hill. 442p. 2001.

Pressman, R. S. Software Engineering: A Practitioner's Approach. McGraw Hill. 880p. 2009.

Qing, Z.; Tao, W.; Shenglong, T. Adapting Game Technology to Support Software Engineering Process Teaching: From SimSE to MO-SEProcess. In: 3rd International Conference on Natural Computation. pp.777-780. 2007.

Rapps, S.; Weyuker, E. J. Selecting Software Test Data Using Data Fow Information. In: IEEE Transactions on Software Engineering. v. 11. n. 4. pp. 367-375. 1985.

Santos, R. P.; Santos, P. S. M.; Werner, C. M. L.; Travassos, G. H. Utilizando Experimentação para Apoiar a Pesquisa em Educação em Engenharia de Software no Brasil. In: I Fórum de Educação em Engenharia de Software. pp.55-64. 2008.

Savi, R., Wangenheim, C. G. V. e Ulbricht, V. (2010) Proposta de um Modelo de Avaliação de Jogos Educacionais. Revista Novas Tecnologias na Educação. V. 8, n. 3

Schwaber, K. Agile Project Management with Scrum. Microsoft Press. 188p. 2004.

SESAM. Software Engineering Simulation by Animated Models (SESAM). Disponível em: <http://www.iste.uni-stuttgart.de/se/research/sesam/overview/index_e.html>. Acessado em: 07/04/2009.

Silva, K. A. C.; Kirner, C. Vantagens Educacionais no Uso de Jogos em Realidade Aumentada. In: Revista Novas Tecnologias na Educação. v. 2. n. 2. 2010.

Smith, R.; Gotel, O. Gameplay to Introduce and Reinforce Requirements Engineering Practices. In: 16th IEEE International Requirements Engineering Conference. pp. 95-104. 2008.

Smole, K. C. S.; Diniz, M. I. Ler, Escrever e Resolver Problemas: Habilidades Básicas para Aprender Matemática. Artmed. 204p. 2001.

Souza, D. A. C. M.; Vasconcelos, C. R.; Azevedo, R.; Fujioka, R. C.; Almeida, M. J. S. C.; Freitas, F. Honey: Um Ambiente Virtual Baseado em Agentes para Apoiar o Ensino de Engenharia de Software. In: XIX Simpósio Brasileiro de Informática na Educação. pp. 55-64. 2008.

SWEBoK. A Guide to the Project Management Body of Knowledge. Fifth Edition. PMI. 616p. 2013.

Tao, W.; Qing, Z. A Software Engineering Education Game in a 3-D Online Virtual Environment. In: 1st International Workshop on Education Technology and Computer Science. V. 2. pp.708-710. 2009.

Tarouco, L. M. R.; Roland, L. C.; Fabre, M. C. J. M.; Konrath, M. L. P. Jogos Educacionais - Novas Tecnologias na Educação. CINTED-UFRGS. 6 p. 2004.

Valente, A. B. A Intransigência da Transferência de Conhecimento. FDE. 1993.

Virvou, M.; Katsionis, G.; Manos, K. Combining Software Games with Education: Evaluation of its Educational Effectiveness. In: Educational Technology & Society. pp. 54-65. 2005.

Wangenheim, C. G. von; Kochanski, D.; Savi, R. Revisão Sistemática sobre Avaliação de Jogos Voltados para Aprendizagem de Engenharia de Software no Brasil. In: II Fórum de Educação em Engenharia de Software. pp. 41-48. 2009.

Wangenheim, C. G. von; Thiry, M.; Kochanski, D.; Steil, L.; Silva, D.; Lino, J. Desenvolvimento de um Jogo para Ensino de Medição de Software. In: Simpósio Brasileiro de Qualidade de Software. 2009.