UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Virginia Farias de Sousa

ITEST LEARNING: UM JOGO PARA O APOIO AO ENSINO DE TESTES DE SOFTWARE

QUIXADÁ
Junho de 2012

Virginia Farias de Sousa

ITEST LEARNING: UM JOGO PARA O APOIO AO ENSINO DE TESTES DE SOFTWARE

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Orientadora Prof[a]. Msc. Carla Ilane Moreira Bezerra

QUIXADÁ
Junho de 2012

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

S696i Sousa, Virginia Farias de.
iTest Learning: um jogo para o apoio ao ensino de testes de software / Virginia Farias de Sousa. – 2012.
65 f. ; il. color., enc. ; 30cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus Quixadá, Curso de Sistemas de Informação, Quixadá, 2012.
Orientação: Msc. Carla Ilane Moreira Bezerra.

1. Software - testes 2. Software Educacional 3. Engenharia de Software I.Título.

CDD 005

Virginia Farias de Sousa

ITEST LEARNING: UM JOGO PARA O APOIO AO ENSINO DE TESTES DE SOFTWARE

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Aprovado em: 19/ novembro / 2012.

BANCA EXAMINADORA

_____
Prof. Msc. Carla Ilane Moreira Bezerra (Orientador)
Universidade Federal do Ceará-UFC

_____
Prof. Msc. Camilo Camilo Almendra
Universidade Federal do Ceará-UFC

_____
Profa. Dra. Paulyne Mattews Jucá
Universidade Federal do Ceará-UFC

Dedico aos meus pais, Maria Vilani e Francisco Ferreira (*in memorian*)
e meus irmãos, Henrique Verlan, Valdelia Souza e Francisco Vilamar.

AGRADECIMENTOS

A Deus, pela força que sempre me deu nos momentos de dificuldade. Pelo seu amor e bondade infinitos. Por me permitir concluir mais uma etapa de minha vida.

Aos meus pais, Francisco Ferreira de Sousa (*in memorian*) e Maria Vilani de Farias Sousa, por seus ensinamentos, dedicação, proteção e amor incondicional. Pessoas honestas e trabalhadoras, que me fizeram a pessoa que sou hoje.

Aos meus irmãos, Henrique Verlan, Valdélia e Francisco Vilamar, pelo carinho e apoio durante toda essa caminhada.

Aos meus parentes, por sempre estarem dispostos a ajudar e apoiar sempre que necessário.

À Ythaianny Menezes, por sua amizade verdadeira e por estar comigo em todos os momentos.

Aos amigos que fiz durante o curso, pela amizade construída e por todos os momentos que passamos durante esses quatro anos.

À minha orientadora, Carla Ilane, pelo ensinamento e dedicação. Pela paciência e confiança em mim.

Aos professores, Paulyne Jucá e Camilo Almendra, por aceitarem contribuir com esse trabalho participando da banca e pelas suas valiosas contribuições na melhoria do mesmo.

Ao grupo PET de Sistemas de Informação, pelo companheirismo, crescimento e amadurecimento que me proporcionou. Em especial, ao Tutor Davi Romero, pelos ensinamentos e ajuda em todos os momentos.

A todo o corpo docente do Campus de Quixadá, que nos deram os subsídios necessários para que pudéssemos nos tornar bons profissionais.

Aos demais funcionários da instituição, por sua presteza e ótimo trabalho.

À empresa Ifactory Solutions, pelo tempo de experiência profissional proporcionado.

A todos os alunos que participaram da pesquisa de avaliação do trabalho, pois sem eles este trabalho não estaria completo.

A todos os demais que direta ou indiretamente contribuíram para a construção deste trabalho.

"Suba o primeiro degrau com fé. Não é necessário que veja toda a escada. Apenas dê o primeiro passo."
(Martin Lutter King)

RESUMO

As exigências por software com maior qualidade têm motivado a definição de métodos e técnicas para o desenvolvimento de softwares que atinjam os padrões de qualidade impostos. Com isso, o interesse pela atividade de teste de software vem aumentando nos últimos anos. No processo de desenvolvimento de software é importante que sejam adotados métodos, técnicas e ferramentas que permitam a realização da atividade de testes de maneira sistematizada e com fundamentação científica, de modo a aumentar a produtividade e a qualidade e diminuir os custos. Para isso, é necessário um bom planejamento de teste de software. No entanto, nos currículos dos cursos de graduação de TI, o ensino de teste de software não recebe muita atenção. Neste contexto, com o objetivo de contribuir para um melhor aprendizado nessa área, este trabalho propõe um jogo para auxiliar no ensino de teste de software focado na fase de planejamento de forma a auxiliar no estímulo e motivação do aluno para o aprendizado do conteúdo ministrado. Para isso, foi elaborado e construído um jogo que auxiliasse no ensino de testes de software, e foi realizada uma avaliação com alunos da disciplina de Verificação e Validação. Os resultados obtidos mostram que o jogo é útil para apoiar o ensino de teste de software.

Palavras chave: Teste de Software, Jogos Educacionais, Planejamento de Testes de Software.

ABSTRACT

The demand for software with higher quality has motivated the definition of methods and techniques for the development of applications which meet the imposed quality standards. The interest in the testing activity has increased in the past years. In the software development process it is important that methods de adopted, techniques and tools that allow the execution of the testing activity in a systematized manner with a scientific basis, in order to increase productivity and quality, and decrease costs. For this, it is necessary to have good software test planning. However, in IT under graduation courses, test teaching is not given much attention. In that context, aiming to contribute for a better learning in this subject, this paper proposes a game to aid in software test teaching, focused at the planning phase in order to help to stimulate and motivate the student to learn the content taught. For this, a game was elaborated and built to aid software test teaching, an evaluation was done with students enrolled in the Verification and Validation course. The results obtained, show that the game is useful to support software test teaching.

Keywords: Software Testing, Educational Games, Software Test Planning.

LISTA DE ILUSTRAÇÕES

Figura 1– IEEE – Defeito, erro e falha. Fonte: IEEE, 1990. 16
Figura 2 – Modelo 3P x 3E do ciclo de vida do processo de testes de software. Fonte: adaptado de Rios e Moreira (2003). 17
Figura 3 – Dimensões do teste. Fonte: adaptado de Bastos et al. (2007). 19
Figura 4 – Modelo V de Teste de Software. Fonte: Craig e Jaskiel (2002). 21
Figura 5 – Modelo FURPS. Fonte: XMind (2011). 26
Figura 6 – Tela inicial do jogo. Fonte: a autora. 36
Figura 7 – Tela do tutorial explicando o funcionamento das fases. Fonte: a autora. 37
Figura 8 – Tela do tutorial explicando o sistema de pontuação do jogo. Fonte: a autora. 37
Figura 9 – Tela do tutorial explicando o sistema de ajuda do jogo. Fonte: a autora. 38
Figura 10 – Tela de escolha do nível de dificuldade e projeto. Fonte: a autora. 39
Figura 11 – Tela de descrição do projeto escolhido. Fonte: a autora. 39
Figura 12 – Tela da Fase 1. Fonte: a autora. 40
Figura 13 – Tela da Fase 2. Fonte: a autora. 40
Figura 14 – Tela da Fase 3. Fonte: a autora. 41
Figura 15 – Tela da Fase 4. Fonte: a autora. 42
Figura 16 – Tela da Fase 5. Fonte: a autora. 42
Figura 17 – Tela da Fase 6. Fonte: a autora. 43
Figura 18– Tela do *feedback* – Planejamento do participante. Fonte: a autora. 44
Figura 19 – Tela do *feedback* – Planejamento sugerido. Fonte: a autora. 45
Figura 20 – Tela de ir para o *ranking*. Fonte: a autora. 45
Figura 21 – Tela de *ranking*. Fonte: a autora. 46
Figura 22 – Diagrama de Atividades. Fonte: a autora. 47
Figura 23 – Diagrama de classes. Fonte: a autora. 48
Figura 24 - Resultado da Pergunta 1. Fonte: a autora. 50
Figura 25 - Resultado da Pergunta 2. Fonte: a autora. 51
Figura 26 - Resultado da Pergunta 3. Fonte: a autora. 51
Figura 27 - Resultado da Pergunta 4. Fonte: a autora. 52
Figura 28 - Resultado da Pergunta 5. Fonte: a autora. 52
Figura 29 - Resultado da Pergunta 6. Fonte: a autora. 53

LISTA DE TABELAS

Tabela 1- Respostas das questões objetivas do Questionário. Fonte: a autora. 50
Tabela 2 - Resultado da questão objetiva referente ao grau de dificuldade do jogo. 53

SUMÁRIO

1 INTRODUÇÃO 12

2 REVISÃO BIBLIOGRÁFICA 14

2.1 Testes de Software 14
2.1.1 O Processo de Testes de Software 17
2.1.2 Níveis de Testes 19
2.1.3 Técnicas de Testes 21
2.1.4 Tipos de Testes 22
2.2 Jogos e Ensino Através de Ferramentas Interativas 26

3 PROCEDIMENTOS METODOLÓGICOS 31

4 PROJETO DO JOGO 33

4.1 *Design* Instrucional 33
4.1.1 Público-Alvo 33
4.1.2 Conhecimento Requerido 33
4.1.3 Objetivos Educacionais 33
4.2 *Design* do Jogo 34
4.2.1 Conceito do Jogo 34
4.2.1.1 Descrição do Jogo 34
4.2.1.2 Gênero do Jogo 35
4.2.1.3 Plataforma do Jogo 35
4.2.2 Mecânica do Jogo 35
4.2.2.1 Gameplay 36
4.3 Modelagem do Jogo 46
4.3.1 Diagrama de Atividades 46
4.3.2 Diagrama de Classes 48
4.3.3 Tecnologia Utilizada 48

5 ESTUDO DE CASO 49

6 CONSIDERAÇÕES FINAIS 55

REFERÊNCIAS 56

APÊNDICES 59

APÊNDICE A – QUESTIONÁRIO DE AVALIAÇÃO DO JOGO 59
APÊNDICE B – TERMO DE CONCORDÂNCIA 61
APÊNDICE C – RESPOSTAS DAS QUESTÕES DISCURSIVAS - AVALIAÇÃO DO JOGO 62

1 INTRODUÇÃO

Segundo Delamaro et al. (2007), tem-se notado um enorme crescimento do interesse, por parte dos desenvolvedores, nas questões relacionadas à qualidade de software. Em particular, a indústria tem despertado para a grande importância da atividade de teste que, apesar de, por um lado, poder contribuir para a melhoria da qualidade de um determinado produto, por outro, pode representar um custo significativo dentro dos orçamentos empresariais. Por isso, é importante que, no processo de desenvolvimento de software, sejam adotados métodos, técnicas e ferramentas que permitam a realização da atividade de testes de maneira sistematizada e com fundamentação científica, de modo a aumentar a produtividade e a qualidade e diminuir os custos.

Testar um software é executar um programa com a intenção de encontrar erros (MYERS, 2004). Testes são necessários porque verificam o atendimento aos requisitos do usuário. Apesar da importância da atividade de testes de software, o ensino da disciplina Testes de Software tem recebido pouco destaque na maioria dos cursos de graduação em Ciência da Computação (SHEPARD et al., 2001). ELBAUM et al. (2007) destacam que técnicas e princípios de testes de software deveriam ser integrados cedo no currículo de tais cursos.

Atualmente existem vários trabalhos relacionados ao ensino de Engenharia de Software (SOUZA et al., 2010; THIRY et al., 2010; MONSALVE et al., 2010; FIGUEIREDO et al., 2010; VON WANGENHEIM et al., 2009b; KIELING E ROSA, 2006). No ensino da disciplina Teste de Software pode-se destacar na literatura dois jogos educacionais, o Jogo das 7 Falhas (DINIZ e DAZZI, 2011) e o U-Test (SILVA, 2010). Os dois jogos têm por objetivo ensinar técnicas de caixa-preta, e para isso o jogador deve utilizar as técnicas de classe de equivalência ou análise de valor-limite.

Assim, os jogos relacionados ao ensino de Testes de Software encontrados não dão suporte ao planejamento, criação do plano de teste e do documento de especificação de testes.

Diante desse contexto, este trabalho visa apoiar o ensino de Testes de Software apresentando um jogo web para este fim. No jogo, o aluno seguirá um tutorial online que lhe ajudará a entender os conceitos do planejamento de testes de software, a partir da prática com simulação de problemas para possíveis sistemas.

Como objetivo principal do trabalho tem-se o desenvolvimento e avaliação de um jogo educacional para apoiar o ensino de testes de software, proporcionando um ambiente de

simulação para a experimentação de práticas relacionadas ao planejamento de testes de software. Tendo como objetivos específicos:

- Fazer um levantamento de conceitos de testes de software e trabalhos relacionados sobre o ensino de Engenharia de Software;
- Desenvolver um jogo educacional para apoiar o ensino de teste de software, com foco no planejamento de testes;
- Avaliar o jogo desenvolvido.

O trabalho está organizado em 7 capítulos, como descritos a seguir: O capítulo 1 apresenta a proposta do trabalho, bem como seus objetivos gerais e específicos. O capítulo 2 apresenta uma revisão bibliográfica sobre os assuntos abordados neste trabalho, conceituando testes de software e jogos educacionais. O capítulo 3 apresenta os procedimentos metodológicos utilizados para a construção deste trabalho. No capítulo 4 é apresentado o projeto do jogo de forma detalhada. O capítulo 5 mostra um estudo de caso realizado e seus resultados e por fim, no capítulo 6 são descritas as conclusões do trabalho, bem como são propostas possibilidades de continuidade do trabalho realizado.

2 REVISÃO BIBLIOGRÁFICA

As exigências por software com maior qualidade têm motivado a definição de métodos e técnicas para o desenvolvimento de softwares que atinjam os padrões de qualidade impostos. Com isso, o interesse pela atividade de teste de software vem aumentando nos últimos anos (BARBOSA et al., 2000).

A Engenharia de Software evoluiu significativamente nas últimas décadas, procurando estabelecer técnicas, critérios, métodos e ferramentas para a produção de software, em consequência da crescente utilização de sistemas baseados em computação em praticamente todas as áreas da atividade humana. Isso provoca uma crescente demanda por qualidade e produtividade, tanto do ponto de vista do processo de produção como do ponto de vista dos produtos gerados. Através de um conjunto de etapas que envolvem o desenvolvimento e aplicação de métodos, técnicas e ferramentas, a Engenharia de Software oferece meios para que tais objetivos possam ser alcançados (BARBOSA et al., 2000).

A qualidade de produtos de software está fortemente relacionada à qualidade do processo de software, ou seja, para obter um produto final com uma boa qualidade, é necessário garantir essa qualidade desde a primeira atividade do ciclo de vida até a fase de manutenção do sistema (ROCHA et al., 2001). Qualidade de software, segundo Pressman (2006), é a conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido. Para ROCHA et al. (2001), qualidade de software pode ser definida como um conjunto de características que devem ser alcançadas em um determinado grau para que o produto atenda às necessidades de seus usuários.

2.1 Testes de Software

O processo de desenvolvimento de software envolve uma série de atividades nas quais, apesar das técnicas, métodos e ferramentas empregados, erros no produto ainda podem ocorrer. Atividades agregadas sob o nome de Garantia da Qualidade de Software têm sido introduzidas ao longo de todo o processo de desenvolvimento, entre elas atividades de V&V – Verificação e Validação – com o objetivo de minimizar a ocorrência de erros e riscos associados (BARBOSA et al., 2000). As atividades de Verificação e Validação servem para

assegurar que o software funcione de acordo com o que foi especificado e atenda aos requisitos dos *stakeholders*[1] (KOSCIANSKI; SOARES, 2007).

Segundo Koscianski e Soares (2007), a verificação é o processo de determinar se a saída de uma fase está de acordo com os requisitos especificados na fase anterior. Verificar não é demonstrar que a saída de uma fase do desenvolvimento é correta, mas averiguar se o software está de acordo com as especificações preestabelecidas. Já a validação é definida como o processo de confirmar que a especificação de uma fase ou do sistema completo é apropriada e consistente com os requisitos dos *stakeholders*.

Dentre as técnicas de verificação e validação, a atividade de teste é uma das mais utilizadas, constituindo-se em um dos elementos para fornecer evidências da confiabilidade do software em complemento a outras atividades, como, por exemplo, o uso de revisões e de técnicas formais e rigorosas de especificação e de verificação (MALDONADO, 1991).

Existem várias definições sobre teste de software na literatura. Dentre elas:

> [...] o processo de avaliar um sistema ou um componente de um sistema por meios manuais ou automáticos para verificar se ele satisfaz os requisitos especificados ou identificar diferenças entre resultados esperados e obtidos. (ANSI/IEEE Standart 729, 1983, p. 729).

> [...] o processo de executar um programa com o objetivo de encontrar erros [...]. (MYERS, 2004, p. 11).

> [...] um conjunto de atividades que podem ser antecipadamente planejadas e conduzidas sistematicamente, com o objetivo de encontrar erros, defeitos e/ou falhas no sistema. (PRESSMAN, 2006, p. 289).

O objetivo do teste é encontrar defeitos, revelando que o funcionamento do software em uma determinada situação não está de acordo com o esperado. Um teste bem-sucedido identifica defeitos que ainda não foram descobertos e que podem ser, então, corrigidos pelo programador (KOSCIANSKI e SOARES, 2007).

É importante diferenciar os conceitos ligados a teste de software: defeito, erro e falha. A terminologia padrão para Engenharia de Software do IEEE 610.12-1990 (IEEE

[1] Qualquer organização ou indivíduo ativamente envolvido no projeto cujos interesses podem ser afetados pelo projeto. Seus interesses podem ser financeiros ou não, positivos ou negativos, e podem ser afetados durante o projeto ou após seu encerramento (PMBOK, 2004).

Standard Glossary of Software Engineering Terminology), de acordo com a Figura 1, define os três conceitos abaixo:

- **Erro.** É um defeito cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta;
- **Defeito (ou Falta).** É uma manifestação concreta de um erro num artefato de software. Um erro pode resultar em diversos defeitos;
- **Falha.** É o comportamento operacional do software diferente do esperado pelo usuário. Uma falha pode ter sido causada por diversas faltas e algumas faltas podem nunca causar uma falha.

Figura 1– IEEE – Defeito, erro e falha. Fonte: IEEE, 1990.

Na prática, não se consegue testar completamente um programa nem garantir que ele está livre de defeitos. É quase impossível testar todas as possibilidades de métodos e alternativas de entrada de dados, ou as diversas possibilidades e condições criadas pela lógica do programador (BASTOS et al., 2007). BASTOS et al. (2007) afirmam ainda que é praticamente impossível garantir que um software esteja isento de erros, embora, em alguns casos específicos (softwares de alto risco), essa afirmação possa chegar bem perto da verdade.

Na elaboração do planejamento do teste, uma das etapas é a elaboração da estratégia de teste, que compreende a definição dos seguintes itens: o nível de teste, a técnica de teste a ser utilizada, o tipo de teste a ser aplicado no software (CRESPO et al., 2004). Esses itens serão definidos nas subseções a seguir.

2.1.1 O Processo de Testes de Software

De acordo com (RIOS e MOREIRA, 2003), o ciclo de vida do processo de teste, mostrado na Figura 2, é composto por diversas etapas ou fases, sendo quatro delas sequenciais ou em cascata, e duas paralelas. A cada uma das etapas corresponderão atividades, produtos e documentos.

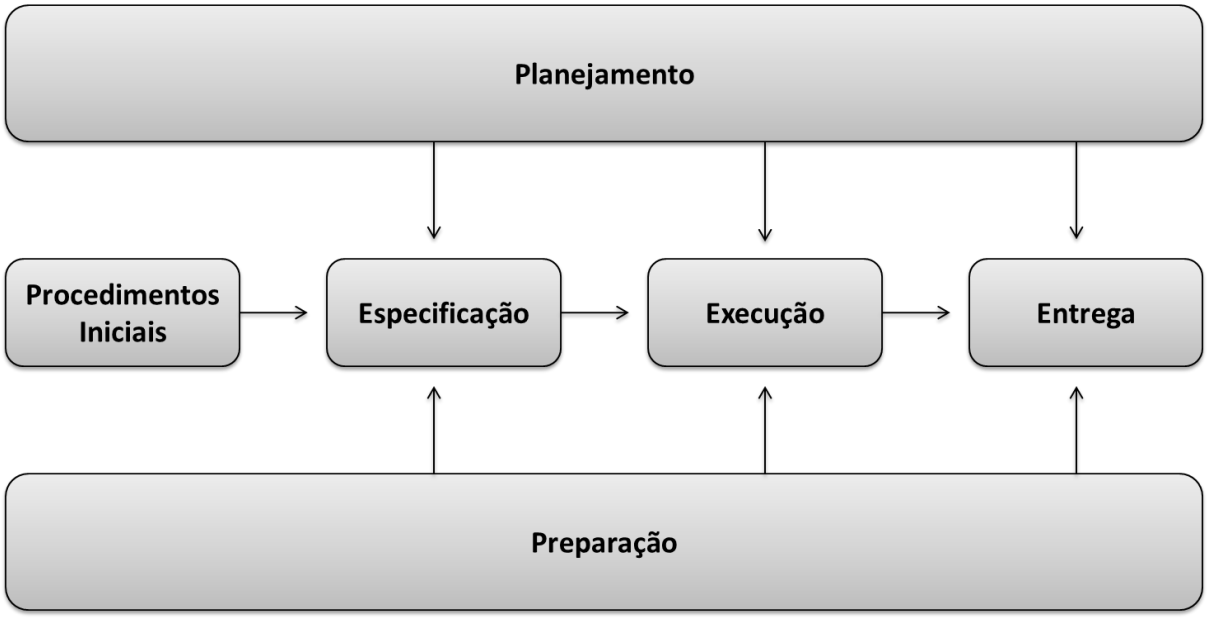

Figura 2 – Modelo 3P x 3E do ciclo de vida do processo de testes de software. Fonte: adaptado de Rios e Moreira (2003).

Esse modelo foi chamado de Modelo 3P x 3E por (BASTOS et al., 2007). O detalhamento das etapas do modelo serão descritas a seguir, de acordo com Bastos et al. (2007).

2.1.1.1 Procedimentos Iniciais

Nesta etapa, deverá ser aprofundado um estudo dos requisitos de negócio que dará origem ao sistema de informação a ser desenvolvido, de modo a garantir que o mesmo esteja completo e sem ambiguidade.

2.1.1.2 Planejamento

Consiste em elaborar a Estratégia de Teste e o Plano de Teste a serem utilizados de modo a minimizar os principais riscos do negócio e fornecer os caminhos para as próximas etapas. A atividade de planejamento tem de permanecer ativa até que o projeto seja concluído, visto que se fará necessário avaliar constantemente se os rumos do projeto estão dentro do que foi previsto e planejado.

2.1.1.3 Preparação

Nesta etapa, o objetivo é preparar o ambiente de teste (equipamentos, pessoas, ferramentas de automação, hardware e software), para que os testes sejam executados corretamente. Trata-se de uma etapa que vai correr em paralelo com outras etapas, como mostra a Figura 2.

2.1.1.4 Especificação

Os objetivos básicos desta etapa são elaborar/revisar casos e roteiros de teste. No cumprimento dos objetivos é preciso que se observe o seguinte: os casos de teste e os roteiros de teste devem ser elaborados dinamicamente durante o decorrer do projeto de teste. Isso equivale a dizer que eles serão elaborados à medida que a equipe de desenvolvimento liberar alguns módulos ou partes dos sistemas para teste.

2.1.1.5 Execução

Executar os testes planejados e registrar os resultados obtidos são tarefas que precisam obedecer as diretrizes a seguir:

- Os testes deverão ser executados de acordo com os casos de teste e os roteiros de teste;
- Devem ser usados os scripts de teste, caso seja empregada alguma ferramenta de automação de testes;
- Os testes deverão ser executados integralmente, por regressão ou parcialmente, sempre que surgir alguma mudança de versão dos programas em teste e nos ambientes de teste preparados (desenvolvimento, testes, homologação, produção), conforme previsto na Estratégia e nos Planos de Teste.

2.1.1.6 Entrega

Nesta etapa, o projeto de teste é finalizado. Será concluída/arquivada toda a sua documentação e serão relatadas todas as ocorrências desse projeto que forem consideradas relevantes à melhoria do processo. Deve ser recolhida toda a documentação de teste preparada e elaborado um relatório gerencial com as conformidades e não conformidades encontradas.

Uma das atividades na elaboração do planejamento dos testes é decidir qual estratégia de teste será usada. Segundo Bastos et al. (2007), uma estratégia de teste deve incluir: os estágios ou níveis de teste a serem abordados (unidade, integração, sistema e

aceitação); fase do desenvolvimento em que se aplica o referido teste; os tipo de teste que devem ser executados (funcional, desempenho, carga, estresse, etc.); as técnicas e ferramentas a serem empregadas no teste (funcionais ou estruturais); e os critérios de conclusão e êxito do teste que serão aplicados. Os critérios de teste podem, por exemplo, permitir que o software evolua para o teste de aceitação quando 95% dos casos de teste tiverem sido executados com êxito.

Resumidamente, é possível traduzir a Estratégia de Teste como as dimensões do teste (Figura 3).

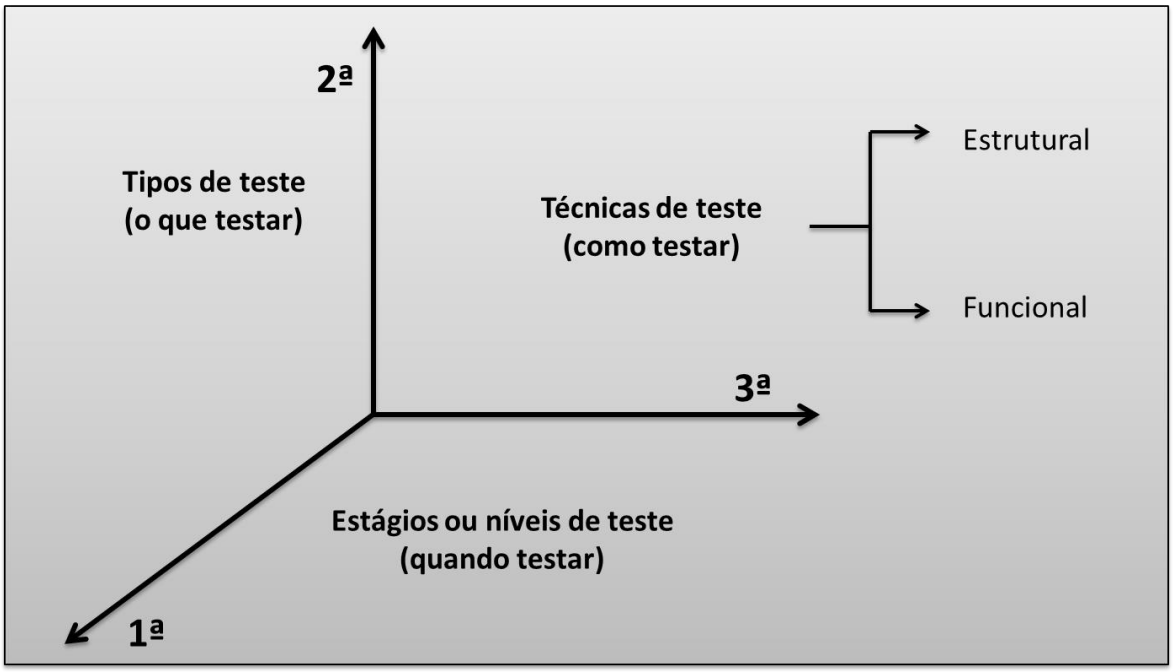

Figura 3 – Dimensões do teste. Fonte: adaptado de Bastos et al. (2007).

2.1.2 Níveis de Testes

Testes são executados em diferentes níveis (ou estágios) do desenvolvimento de um software. O nível (ou estágio) de teste é uma das dimensões do teste que representa "quando", ou melhor, a que fase do projeto se aplica um determinado teste (BASTOS et al., 2007). Esses testes, de acordo com Bastos et al. (2007), são aqueles listados a seguir.

2.1.2.1 Teste de Unidade

Estágio mais baixo da escala de teste, aplicado aos menores componentes de código criados, visando garantir que eles atendam às especificações funcionais e arquiteturais.

Costuma ser feito pelo programador e testa as unidades individuais: funções, objetos e componentes.

2.1.2.2 Teste de Integração

Teste do sistema ao término de cada iteração, dentro de um ambiente operacional controlado, para validar a exatidão e a perfeição na execução de suas funções, referentes aos casos de uso da iteração. Em geral, é realizado pelo analista de sistemas para um módulo ou conjunto de programas.

2.1.2.3 Teste de Sistema

Execução do sistema como um todo, dentro de um ambiente operacional controlado, para validar a exatidão e a perfeição na execução de suas funções, acompanhando cenários sistêmicos elaborados pelo profissional de requisitos do projeto. Costuma ser feito pelo analista de teste em ambiente de teste.

2.1.2.4 Teste de Aceitação

É a última ação de teste antes da implantação do software, sendo sua execução de responsabilidade do cliente. O objetivo do teste de aceitação é verificar se o software está pronto e pode ser usado por usuários finais para executar as funções e tarefas para as quais foi construído. Em geral é feito pelo usuário em ambiente de homologação.

A representação gráfica mais utilizada de um modelo que represente o desenvolvimento versus o teste é o modelo “V” (Figura 4), porque sua simplicidade mostra as atividades em linhas gerais (BASTOS et al., 2007).

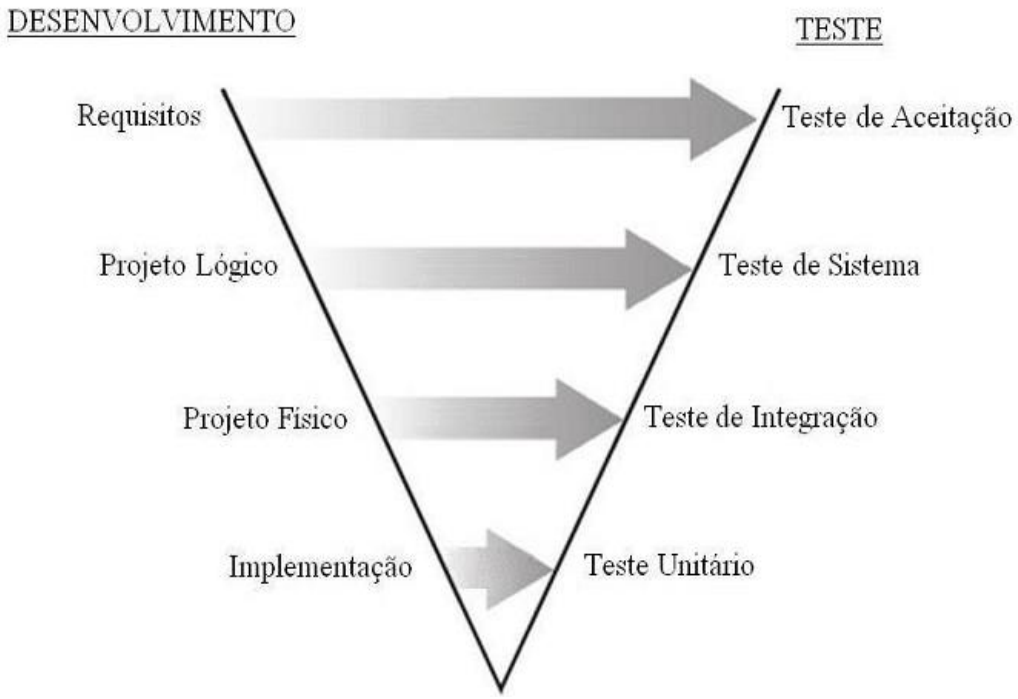

Figura 4 – Modelo V de Teste de Software. Fonte: Craig e Jaskiel (2002).

O Modelo V de Teste de Software ilustra a relação entre atividades do ciclo de desenvolvimento (Requisitos, Projeto Lógico, Projeto Físico e Implementação) e níveis de teste (Teste Unitário, teste de Integração, Teste de Sistema e Teste de Aceitação).

O Teste de Aceitação, apesar de ser normalmente o último realizado, deve ser o primeiro a ser planejado, desde que os requisitos estejam devidamente documentados. Da mesma forma, os testes de Sistema serão planejados com base nas etapas de Projeto Lógico e Requisitos; os testes de Integração serão planejados com base nas etapas de Projeto Físico, Projeto Lógico e Requisitos; e os testes de Unidade serão planejados com base nas etapas de Implementação, Projeto Físico, Projeto Lógico e Requisitos (CRAIG e JASKIEL, 2002).

Os testes deverão ser realizados na ordem inversa à qual eles foram planejados, respeitando, então, a seguinte sequência: Teste Unitário, Teste de Integração, Teste de Sistema e Teste de Aceitação. Em geral, esse modelo reforça a ideia de que o teste não é uma fase, mas uma parte integrante do ciclo de desenvolvimento do software.

2.1.3 Técnicas de Testes

A realização dos testes deve refletir as ações tomadas para descobrir erros introduzidos na codificação das funcionalidades definidas nas especificações dos programas e outros erros inseridos durante a codificação do programa (BASTOS et al., 2007).

As técnicas de teste de software dividem-se basicamente em teste de caixa branca (estrutural), técnica de teste baseado no conhecimento da estrutura lógica do programa; teste de caixa preta (funcional), técnica de teste onde é necessário ter apenas o conhecimento das entradas e saídas possíveis do sistema; e teste de caixa cinza, uma combinação entre os o teste de caixa preta e o teste de caixa branca. Essas técnicas são descritas a seguir:

2.1.3.1 Teste de Caixa Branca

Também chamado de teste estrutural, consiste em uma abordagem de testes que são derivados do conhecimento da estrutura e da implementação do software (SOMMERVILLE, 2007). O testador busca testar e conhecer todo o código do sistema, examinando o caminho lógico para verificar seu funcionamento. Nesta técnica não há preocupação com os requisitos do sistema, ou seja, se o software está de acordo com os requisitos do cliente, mas sim com seu funcionamento (LEWIS, 2000).

2.1.3.2 Teste de Caixa Preta

Conhecido também como teste funcional, é um método de teste realizado na interface do programa. Um teste de caixa preta examina alguns aspectos de um sistema sem se preocupar muito com a estrutura interna do software (PRESSMAN, 2006). Esta técnica verifica a funcionalidade do software, preocupando-se em executar todas as funções exigidas pelo usuário corretamente, sem se preocupar muito com o que foi implementado (LEWIS, 2000).

2.1.3.3 Teste de Caixa Cinza

Os testes de caixa cinza consistem na combinação entre os testes de caixa preta e caixa branca. Esta técnica analisa a parte lógica mais a funcionalidade do sistema, fazendo uma comparação do que foi especificado com o que está sendo realizado. Usando esse método, o testador comunica-se com o desenvolvedor para entender melhor o sistema e otimizar os casos de testes que serão realizados (LEWIS, 2000).

2.1.4 Tipos de Testes

Os tipos de testes referem-se às características do software que podem ser testadas (CRESPO et al., 2004). Serão apresentados os tipos de testes descritos em RUP (2002). O RUP classifica testes sob cinco fatores de qualidade do modelo FURPS (Functionality, Usability, Reliability, Performance, Supportability), uma metodologia de qualidade muito

usada no mercado. Nessa metodologia, as categorias de qualidade e seus tipos de teste (Figura 5) associados, conforme Bastos et al. (2007), são aquelas apresentadas a seguir.

2.1.4.1 Funcionalidade

Para atender à categoria de funcionalidade, o software em teste deve estar de acordo com a especificação funcional, que não deve ser confundida com as técnicas de teste funcional.

- **Teste de função.** Concentra-se na validação das funções do objetivo do teste conforme o esperado. Esse trabalho é realizado em diferentes estágios, como unidade, integração ou sistema.
- **Teste de regressão.** Verifica se as partes do software não afetadas por uma alteração continuam operando conforme o que foi especificado.
- **Teste de volume.** Submete a aplicação a ser testada a grandes quantidades de dados para determinar se os limites que causam a falha foram alcançados. Enfoca parâmetros de sobrecarga.
- **Teste de segurança.** Destinado a garantir que o objetivo do teste e os dados (ou sistemas) possam ser acessados apenas por determinados atores. Esse teste é implementado e executado em vários objetos de teste.

2.1.4.2 Usabilidade

Representa a facilidade de uso do sistema pelos usuários. Alguns testes que podem garantir a usabilidade:

- **Teste de interface.** Verifica a interação do usuário em relação ao aplicativo, garantindo acesso e navegação apropriados através das funções do aplicativo. Neste teste também se examina se os objetos na interface funcionam de acordo com o especificado.
  - **Teste de usabilidade.** Enfatiza a facilidade de uso da aplicação por seus clientes ou usuários. As aplicações devem ser suficientemente amigáveis para atingir os objetivos do negócio.

2.1.4.3 Confiabilidade

Garante a confiabilidade do sistema, a permanência de operação, a integridade dos dados, a confiabilidade da estrutura de dados e também da aplicação. Alguns testes podem ser utilizados para garantira confiabilidade:

- **Teste de integridade.** Teste destinado a avaliar a robustez do objetivo do teste (resistência a falhas) e a compatibilidade técnica em relação à linguagem, sintaxe e utilização de recursos. Esse teste é implementado e executado em vários objetivos do teste, como unidades e unidades integradas;
- **Teste de estrutura.** Teste destinado a avaliar a adequação do objetivo do teste em relação a seu *design* e sua formação. Em geral, esse teste é realizado em aplicativos habilitados para a Web, garantindo que todos os links estejam conectados, que o conteúdo apropriado seja exibido e que não haja conteúdo órfão;
- **Teste de estresse.** Tipo de teste de confiabilidade destinado a avaliar como o sistema responde em condições anormais. O *stress* no sistema pode abranger cargas de trabalho extremas, memória insuficiente, hardware e serviços indisponíveis ou recursos compartilhados limitados. Normalmente, esses testes são executados para compreender melhor como e em quais áreas o sistema será dividido, para que os planos de contingência e a manutenção de atualização possam ser planejados e orçados com bastante antecedência;
- ***Smoke test.*** *O smoke test* exercita o sistema numa única passagem, em geral, utilizando *script* de execução automática, e não deve ser exaustivo, mas capaz de expor os maiores problemas. Esse teste é executado após a construção de cada nova versão.

2.1.4.4 Desempenho

Garante a velocidade de processamento da informação. Alguns testes que garantem o desempenho:

- **Teste de avaliação de desempenho ou *benchmark*.** Compara o desempenho de um objetivo do teste (novo ou desconhecido) a um sistema e uma carga de trabalho de referência conhecidos;
- **Teste de contenção.** Verifica se a aplicação pode lidar de forma aceitável com as demandas de vários atores no mesmo recurso (registros de dados, memória, etc.);
- **Teste de carga.** Usado para validar e avaliar a aceitabilidade dos limites operacionais de um sistema de acordo com cargas de trabalho variáveis, enquanto o sistema em teste permanece constante. As variações na carga de trabalho normalmente incluem a emulação das cargas de trabalho médias e máximas que ocorrem dentro de tolerâncias operacionais normais;
- **Perfil de desempenho.** Teste em que o perfil de andamento do objetivo do teste é monitorado (inclusive fluxo de execução, acesso a dados e chamadas de função e de sistema), a fim de identificar e lidar com gargalos de desempenho e processos ineficientes.

2.1.4.5 Suportabilidade

Representa a capacidade do programa de funcionar em diversos ambientes diferentes. Alguns testes que garantem a suportabilidade:

- **Teste de configuração.** Garante que o objetivo do teste funcione conforme o esperado em diferentes configurações de hardware e/ou software. Esse teste também pode ser implementado como um teste de desempenho do sistema;
- **Teste de instalação.** Garante que o objetivo do teste seja instalado conforme o esperado em diferentes configurações de hardware e/ou software e sob diferentes condições (como no caso de espaço insuficiente em disco ou interrupção de energia). Esse teste é implementado e executado em aplicativos e sistemas.

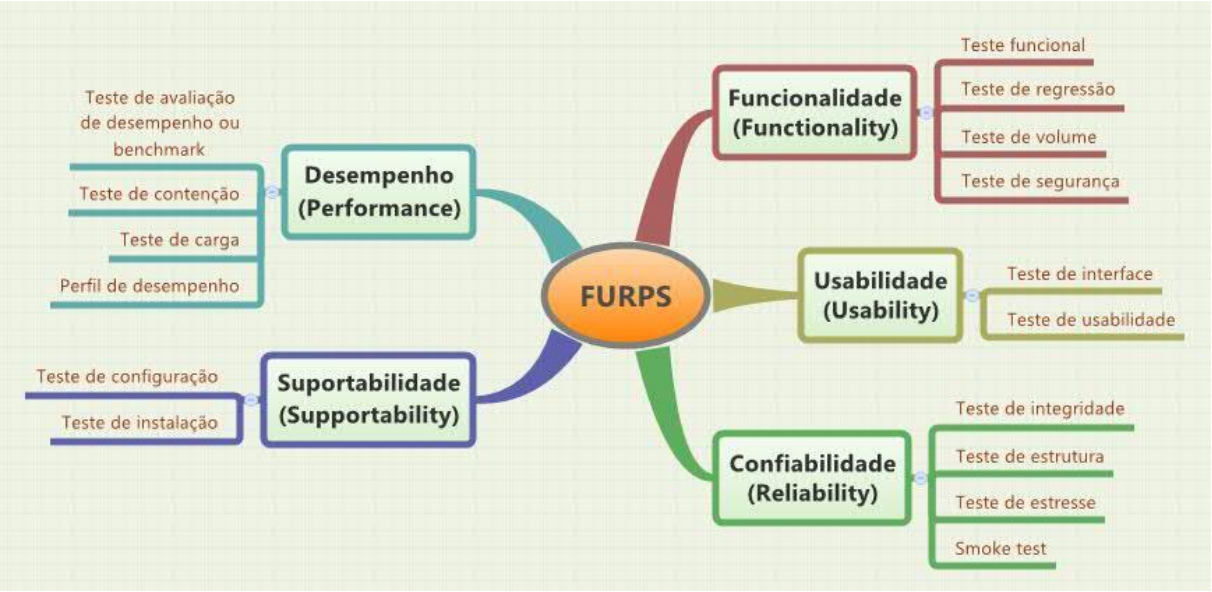

Figura 5 – Modelo FURPS. Fonte: XMind (2011).

2.2 Jogos e Ensino Através de Ferramentas Interativas

Jogos têm sido utilizados para ajudar no ensino de diversas áreas do conhecimento. Estes, quando utilizados como ferramentas educacionais, podem permitir a experimentação de situações que seriam vivenciadas fora do contexto educacional, como, por exemplo, situações diretamente ligadas ao ambiente profissional (SILVA, 2010). Os jogos, de maneira geral, compreendem um conjunto de atividades que podem envolver um ou mais participantes, que comumente desafiam as habilidades físicas ou mentais dos jogadores, exigindo que estes sigam um conjunto de regras para atingir um determinado objetivo, previamente estabelecido (SILVA, 2010).

Jogos educacionais são projetados para ensinar determinado assunto, expandir conceitos, reforçar o desenvolvimento, compreender um acontecimento histórico ou cultural ou auxiliar na aprendizagem de uma habilidade (YEE, 2006). Particularmente, os jogos educacionais podem permitir o “aprender fazendo” em situações realistas, com fornecimento de *feedback* imediato. Além disso, a natureza envolvente dos jogos pode tornar o aprendizado mais divertido (KAFAI, 2001). Pesquisas na área de treinamento e educação sugerem o uso de jogos, pois podem engajar o estudante, reforçando conceitos através da prática, e aprofundando os conhecimentos (EL-SHAMY, 2001).

Existem diversos tipos de jogos educacionais: jogo de simulação, aventura, quebra-cabeças, experimentais e motivacionais (DEMPSEY et al., 1994). Este trabalho tem

como foco um jogo educacional de computador do tipo simulação, que são jogos que produzem diversas situações da vida real com objetivo de formação, análise ou previsão, combinando as características de um jogo (competição, regras, jogadores) com as da simulação (incorporação de recursos do mundo real) (JONES, 1995). Porém, um fator importante apresentado por Poole (2000), refere-se ao fato de que alguns jogos não podem ser facilmente classificados em um único gênero ou estilo de jogo, apresentando características presentes em mais de uma categoria. Esse fato pode explicar as diferentes classificações existentes para os jogos.

Os jogos proporcionam um ambiente simulado onde o aluno pode realmente experimentar, de forma segura, as alternativas à solução de um problema e verificar suas consequências. Mesmo que ruins, estas consequências se limitam ao ambiente simulado (ALDRICH, 2005), permitindo que o aluno possa rever as estratégias adotadas e aprender também a partir de seus erros (SILVA, 2010).

Diante desse contexto, diversos jogos têm sido propostos para apoiar o ensino de Engenharia de Software nas suas mais diversas áreas de conhecimento. Entre os jogos educacionais existentes nessa área, podemos citar, por exemplo, a Ilha dos Requisitos, o Planeger, o X-MED, o SimulES, entre outros. Esses jogos são brevemente descritos nesta seção.

A Ilha dos Requisitos (GONÇALVES et al., 2011) é um jogo educativo que auxilia o ensino dos principais conceitos envolvidos na área de conhecimento Engenharia de Requisitos (ER), como o processo de ER, o papel e responsabilidades do analista de requisitos e os tipos de requisitos. Durante uma partida, o jogador realiza desafios, onde cada desafio está relacionado a um conteúdo de ER. O objetivo do jogo é ajudar o personagem Jack Reqs a fugir da "Ilha dos Requisitos" antes que o vulcão a exploda. O jogo é em 3ª pessoa e deve ser terminado em 45 minutos. O mapa do jogo aponta diferentes locais, onde cada um representa um desafio para o jogador, que coleta pontos durante esses desafios. Quando o jogador não consegue resolver o desafio, sofre uma penalidade no tempo do jogo. É possível ainda obter dicas sobre os desafios através do notebook de Jack, que não está com a bateria totalmente carregada, fazendo com que se tenha uma quantidade limitada de acessos a essas dicas durante o jogo. Ele possui um *ranking* de pontuações que exibe os resultados de todos os jogadores que já o jogaram. O jogo é web e foi desenvolvido na plataforma *flash*[2].

[2] http://www.adobe.com/flashplatform/

Outro jogo é o Planager (KIELING e ROSA, 2006), para a área de conhecimento de Gerência de Projetos. O jogo abrange as áreas de conhecimento de gerenciamento de escopo e de tempo do PMBOK (*Project Management Body of Knowledge*, corpo de conhecimento em gerência de projeto do PMI – *Project Management Institute*). O objetivo do jogo é fazer com que o jogador passe por várias fases, sendo avaliado no final de cada uma delas. Ele possui dois módulos: tutorial, onde o aluno pode revisar os conceitos de gerência de projetos aprendidos em aula e ter uma visão de como é o jogo; e o módulo jogo, onde o aluno pode praticar seus conhecimentos. O jogo possui diversos cenários, que são representações de projetos, e são compostos por uma descrição e cinco fases: escopo, Estrutura Analítica do Projeto (EAP), definição de atividades, sequenciamento de atividades e caminho crítico. O jogador deverá utilizar as informações contidas nas fases anteriores para conseguir resolver corretamente as próximas fases do jogo. Ainda é possível criar cenários para o jogo, pois a ferramenta possui um módulo administrador que permite a realização dessa tarefa. O jogo foi desenvolvido utilizando arquitetura *desktop* e possui sistema de pontuação e *ranking*.

Na área de conhecimento de Medição de Software existe o X-MED (VON WANGENHEIM et al., 2009b). Esse jogo tem como objetivo exercitar a aplicação de medição de software voltada para a gerência de projetos alinhada ao nível 2 de maturidade do CMMI-DEV v1.2 (CMMI, 2006) ou nível F do MPS.Br (MPS.BR, 20011). O jogador joga de forma individual, sem interação com outros jogadores e/ou um instrutor, onde assume o papel de analista de medição e segue sequencialmente todas as tarefas de um programa de medição. Para apoiar a decisão do aluno, o jogo apresenta instruções, materiais, etc. O aluno deve selecionar uma solução para cada tarefa, que possui seis possíveis soluções como alternativas. O jogo possui ainda tempo para realização das tarefas, *feedback* e uma pontuação pré-definida. Ele segue para o passo seguinte, independente da alternativa selecionada elo jogador, utilizando como entrada a alternativa correta do passo anterior. No final, há uma pontuação total e um relatório final. O jogo é *desktop* e integra elementos de multimídia, como áudio, animações, etc.

Os jogos citados acima são todos digitais. Porém, há também jogos não digitais, como de tabuleiros, de cartas, etc. Dentre esses, pode-se destacar o SimulES ou Simulador de Engenharia de Software (FIGUEIREDO et al., 2010), que se trata de um jogo de cartas. A dinâmica do jogo inicia com a escolha de uma carta de projeto a qual contém as informações sobre o projeto e suas restrições, como qualidade e orçamento. Em seguida, os participantes do jogo devem utilizar as cartas de engenheiro de software para conceber, construir e testar a

solução. Durante o jogo podem surgir cartas de problemas, que simulam obstáculos comuns em projetos de desenvolvimento de software. O jogo se encerra no momento em que o primeiro jogador concluir o projeto escolhido no início do jogo, respeitando as restrições impostas por este. Ele possui características, como competição, pois cada jogador compete diretamente com os demais; e *feedback* imediato, permitindo ao jogador identificar seu desempenho em comparação com os demais. Posteriormente, foi desenvolvida uma versão digital desse jogo, o SimulES-W.

Durante a revisão da literatura, foram encontrados também jogos na área de conhecimento de testes de software, como destacado na seção 1. Estes são descritos a seguir, destacando suas características.

O Jogo das 7 Falhas (DINIZ e DAZZI, 2011) é um jogo *single-player*, no qual o jogador assume o papel de testador de uma equipe de teste de software de uma empresa fictícia, com a finalidade de descobrir as sete falhas existentes em cada funcionalidade testada, correlacionando as falhas com uma classe de equivalência ou um valor-limite. O jogo consiste em descobrir as falhas existentes nas funcionalidades de um software a ser testado no menor tempo possível. Ele possui dois níveis de complexidade, baixa e média, que são compostos por uma funcionalidade onde existem sete falhas a serem descobertas. O jogador só passará para o nível 2 caso descubra as sete falhas existentes no nível 1 dentro do tempo estimado (25 minutos para o nível 1; 40 minutos para o nível 2). Caso o tempo se esgote antes do jogador identificar as sete falhas em cada nível, o jogo se encerra, e o jogador é eliminado. Caso o jogador descubra as sete falhas do nível 1 e as sete falhas do nível 2 dentro do tempo, ele é vencedor, tendo alcançado o objetivo proposto pelo jogo. Existem 33 falhas cadastradas no jogo, dentre as quais são sorteadas 7 de forma aleatória para cada novo jogo. O jogo é web e possui ainda uma listagem dos requisitos da funcionalidade de cada nível, bem como as regras disponíveis para serem acessadas ao longo do jogo.

Outro jogo na área de conhecimento de testes de software é o U-Test (SILVA, 2010). Neste, o jogador assume o papel de um testador responsável por escrever teste de unidade para funções já escritas de um sistema hipotético. O objetivo geral do jogo é aplicar técnicas para seleção de dados de entrada para o teste de unidade. Durante o jogo, o jogador deve aplicar técnicas de teste de software, como particionamento em classes de equivalência e análise de valor limite. No final, o jogo apresenta um *feedback* sobre o desempenho do jogador efetuando críticas para possíveis problemas e parabenizando o jogador por completar

todos os desafios. O jogo é web, com uma interface desenvolvida em *flash,* e possui um *ranking* de pontuação com os melhores resultados.

Uma característica comum a todos estes jogos é o reconhecimento da distância entre a teoria aprendida em sala de aula e sua aplicação em uma situação real. Desta forma, o principal motivador na construção destes jogos é promover o ganho de experiência pelos alunos através da simulação de problemas reais de desenvolvimento de projetos de software.

3 PROCEDIMENTOS METODOLÓGICOS

Como descrito na seção 2.2, já existem aplicações para o ensino de teste de software. Entretanto, elas abordam o ensino de teste de caixa-preta ou teste unitário, focando no ensino de classes de equivalência e análise de valor limite. Não foi encontrada nenhuma aplicação que focasse no ensino do planejamento de testes de software. Neste contexto, este trabalho visa o desenvolvimento de um jogo para apoiar o ensino de Testes de Software, focando na fase de planejamento de testes.

O planejamento de testes consiste em definir um plano de testes que determine os critérios a serem avaliados no produto a ser testado, os recursos necessários e o cronograma para execução dos testes (ROCHA et al., 2001). A atividade de planejamento permite definir previamente as ações a serem realizadas na execução de uma determinada tarefa. O planejamento dos testes deve fazer parte do planejamento global do software e deve resultar em um plano de teste que estabelece recursos, define estratégias, critérios e técnicas de teste a serem aplicadas na definição do conjunto de casos e procedimentos de teste necessários para avaliar o software (SILVA, 2011).

O jogo não cobrirá todos os itens que um planejamento de testes pode conter. Os itens cobertos são definidos com base em (SILVA, 2011), que apresenta uma definição de uma metodologia de teste de software para micro e pequenas empresas com base nos modelos de documentos disponibilizados pelo IEEE Standart 829 (1998), e são descritos a seguir:

- **Escopo:** descreve uma visão geral sobre o projeto que está sendo testado. O escopo é fornecido pelo próprio jogo, de maneira que seja dada uma breve descrição do projeto para o qual será feito o planejamento de teste de software;
- **Itens de Teste:** descreve os itens que serão objetos de teste. No jogo, é representado por uma fase onde o jogador deverá escolher quais os itens a serem testados de acordo com o projeto escolhido;
- **Tipos de Teste:** identifica os tipos de testes a serem realizados. Também está representado no jogo em uma fase, em que o jogador define os tipos de teste que farão parte do projeto;
- **Níveis de Teste:** é uma das dimensões de teste que define "quando", ou melhor, a que fase do projeto se aplica um determinado teste. Este item não é contemplado no modelo de (SILVA, 2011), mas foi introduzido no

escopo do jogo para reforçar o conhecimento no processo de testes dentro do processo de desenvolvimento de software. O jogador terá de conhecer quais os níveis de teste de software bem o que eles representam;

- **Ferramentas:** representado em uma fase do jogo, este item corresponde à escolha das ferramentas que serão utilizadas para a realização dos testes. As ferramentas estão contidas na seção de recursos físicos descrita em (SILVA, 2011);
- **Artefatos:** descreve os documentos que serão deverão ser desenvolvidos ao longo das atividades de teste. É uma fase do jogo que tem como objetivo fazer com que o jogador saiba quais artefatos podem ser gerados durante o processo de testes.

4 PROJETO DO JOGO

Um dos objetivos deste trabalho é desenvolver um jogo educacional de simulação para apoiar o ensino de teste de software, de forma a simular a realização do planejamento de testes de software. De forma a proporcionar um melhor entendimento e definição do jogo, foi utilizado o game design document, que define o "processo de criação do conteúdo e das regras de um jogo" (BRATHWAITE e SCHREIBER, 2009). Um documento de design de jogo deve conter os objetivos que um jodagor deve alcançar e as regras que o jogador deve seguir para alcançar os objetivos. Para isso, pode-se dividir a concepção do jogo em duas partes: Design Intrucional e Design do Jogo. As mesmas são descritas a seguir.

4.1 *Design* Instrucional

O design intrucional contém a definição do público alvo, conhecimento prévio necessário, objetivos educacionais, itens de avaliação.

4.1.1 Público-Alvo

O público alvo deste jogo são alunos de graduação de cursos da área de computação/informática, que estejam cursando a disciplina de Engenharia de Software ou Teste de Software.

4.1.2 Conhecimento Requerido

Apesar de o jogo ser voltado para iniciantes na área de teste de software, é necessário que o jogador possua conhecimentos prévios de Engenharia de Software, de forma a permitir que o aluno compreenda o processo de desenvolvimento de software, a importância do teste de software e seus diferentes momentos de aplicação durante esse processo de desenvolvimento; e de Teste de Software, de maneira a ter uma base de conhecimento relativa aos conceitos de teste de software que serão utilizados no decorrer do jogo.

4.1.3 Objetivos Educacionais

Espera-se que o jogador, ao final do jogo, tenha adquirido as habilidades necessárias para o desenvolvimento de um planejamento de teste de software.

4.2 *Design* do Jogo

Compreende a descrição do jogo, sua classificação quanto ao gênero, a plataforma para utilização, a descrição da mecânica do jogo (jogabilidade), a descrição das funcionalidades da interface com o jogador, sistema de pontuação e as características técnicas do jogo, que se referem às tecnologias utilizadas para o desenvolvimento do mesmo.

4.2.1 Conceito do Jogo

Define a descrição do jogo, o gênero e a plataforma de execução do jogo.

4.2.1.1 Descrição do Jogo

O jogo consiste em realizar um planejamento de testes a partir de cenários propostos que representam especificações de sistemas hipotéticos. Essas especificações contêm características que determinarão os itens de teste, tipo de teste, níveis de teste e critérios de aceitação a serem adotados no desenvolvimento do sistema. O jogador poderá escolher o nível de dificuldade do jogo (fácil, médio, difícil), a partir do qual poderá selecionar o projeto para iniciar o jogo. O jogo possui 6 fases:

- Fase 1: Itens de Teste – nesta fase os jogador deve escolher os itens do projeto a serem testados. Essa etapa pode ser considerada como uma das mais importantes, pois ela indica o que realmente vai fazer parte da execução do teste, ou seja, quais itens do software serão testados;
- Fase 2: Tipos de Teste – nesta fase, o jogador define quais tipos de teste serão realizados durante o processo de teste. A escolha dos tipos de teste permite definir como serão realizados os testes para cada item do projeto que será testado. Permite também traçar uma estratégia de teste, através da definição desses tipos de teste com a técnica de teste a ser utilizada;
- Fase 3: Níveis de Teste – para esta fase, o jogador definirá por quais níveis de teste o projeto passará. Os níveis de teste definem quando os testes devem ser executados. Isso é importante para saber o que deve ser feito e em que momento deve ser feito;
- Fase 4: Critérios de Aceitação – nesta fase, serão definidos os critérios de aceitação que farão com que um teste executado seja aprovado ou não. Aqui o jogador definirá os critérios que dirão quando o software pode ser considerado correto ou funcional;

- Fase 5: Ferramentas – para esta fase, o jogador deve escolher quais ferramentas serão utilizadas no processo de testes. As ferramentas são importantes para melhorar a produtividade através da automatização e gerenciamento do processo de testes;
- Fase 6: Artefatos: - nesta fase, o jogador indica quais artefatos podem ser gerados no processo de teste de software. O jogador deverá ter conhecimento do que pode ser produzido no processo de teste de software bem como do que pode ser necessário.

O jogador só poderá passar para a fase seguinte quando concluir a atual. Ele ganhará pontos por cada acerto, e será penalizado em seus pontos a cada erro. A qualquer momento, o jogador poderá tirar dúvidas em relação aos conceitos abordados através da funcionalidade de ajuda, que estará disponível no decorrer do jogo. Quando o planejamento for concluído, a aplicação dará um *feedback* para o jogador, mostrando seus erros e/ou acertos e explicando o porquê dos mesmos. Nesse *feedback* também será dada uma proposta de como poderia ser o planejamento dos testes, já que não há como definir uma resposta absoluta.

4.2.1.2 Gênero do Jogo

O iTest Learning pertence à categoria de jogos educacionais de simulação, que provê um ambiente para a realização do planejamento de teste de software através de uma breve descrição de um projeto hipotético.

4.2.1.3 Plataforma do Jogo

O jogo é executado em computador, através de um navegador com acesso à World Wide Web.

4.2.2 Mecânica do Jogo

Nesta seção, é apresentado o *gameplay* do jogo. Ainda não há uma definição "universalmente aceita" esse termo. Em uma tradução livre, o *gameplay* pode ser considerado como a jogabilidade do jogo. Para (ROLLINGS e ADAMS, 2003), *gameplay* pode ser considerado como o núcleo do jogo.

4.2.2.1 Gameplay

O iTest Learning é um jogo *single-player*, onde o jogador deve realizar um planejamento de teste de software a partir da especificação de projeto.

A tela inicial (Figura 6) apresenta ao jogador uma breve descrição do que é o jogo, definindo seu objetivo e introduzindo a importância do planejamento do teste de software. Há ainda uma explanação de como deve ser iniciado o jogo.

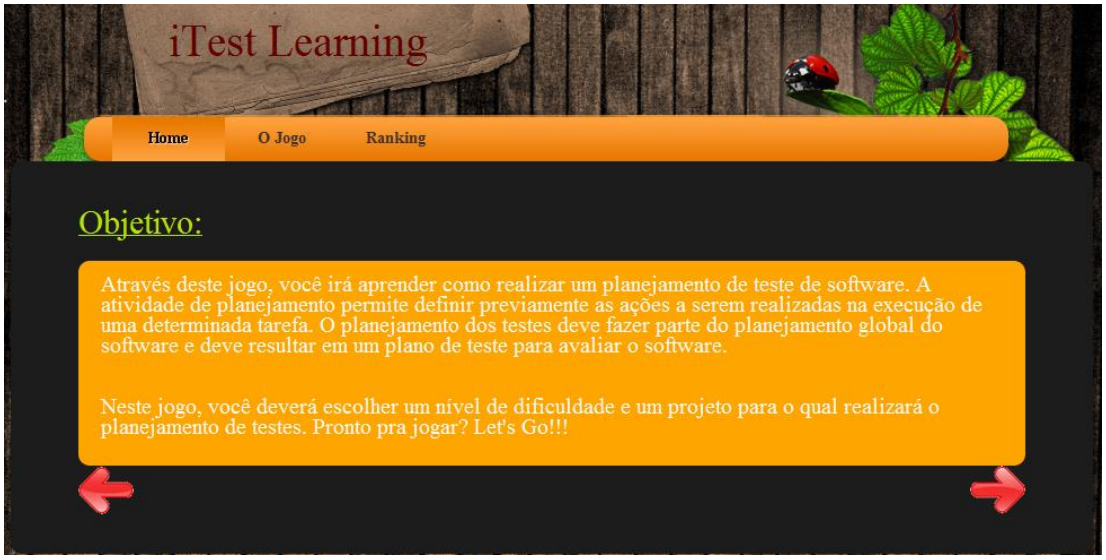

Figura 6 – Tela inicial do jogo. Fonte: a autora.

Para dar sequência ao jogo, o jogador deve clicar nas imagens de setas direcionais para seguir no jogo ou voltar. Na tela seguinte, há um pequeno tutorial mostrando as funcionalidades de cada item do jogo. A primeira parte do tutorial mostra como está dividido o jogo, ou seja, descreve cada fase do jogo e o que cada uma representa, como ilustrado na Figura 7.

Figura 7 – Tela do tutorial explicando o funcionamento das fases. Fonte: a autora.

A segunda parte do tutorial explica como funciona o sistema de pontuação do jogo. É possível acompanhar a evolução da pontuação durante todo o jogo. A cada fase, são atribuídos pontos positivos para cada acerto e o jogador é penalizado com pontos negativos caso cometa erros. Isso é feito de modo a incentivar que o jogador tenha mais cuidado com suas ações. A Figura 8 ilustra o que foi descrito.

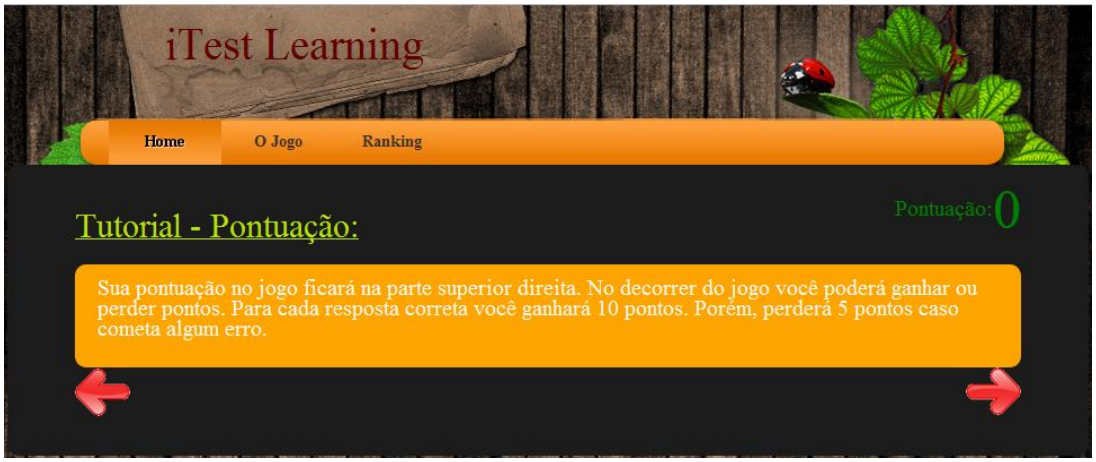

Figura 8 – Tela do tutorial explicando o sistema de pontuação do jogo. Fonte: a autora.

Durante todo o jogo é possível tirar dúvidas em relação aos conceitos que são abordados em cada fase. Por exemplo: na fase de escolha dos tipos de teste, são fornecidas definições dos tipos de testes que apresentados na fase, de forma que o jogador possa lembrar-se desses conceitos. Para isso, basta clicar no ícone de ajuda no menu do jogo. Há ainda, durante todo o projeto, em todas as fases, a opção de visualização da descrição do projeto.

Isso permite que o jogador possa ter acesso a essa descrição sem necessitar sair da fase na qual está. Para tal, basta clicar no ícone correspondente, representado por uma caderneta de anotações. Por último, há o ícone de visualização das respostas propostas pelo jogo, representado por um *check*. Ele permite visualizar as respostas propostas para a fase corrente. Isso permite que o jogador possa ter um *feedback* imediato de suas ações, mesmo que ao final do jogo haja um *feedback* geral em relação ao planejamento realizado pelo jogador. Porém, uma vez utilizada essa opção, não será mais possível jogar na fase, por motivos óbvios. Também não é permitido voltar para a fase anterior, uma vez que já passou por ela. A Figura 9 ilustra o que foi descrito.

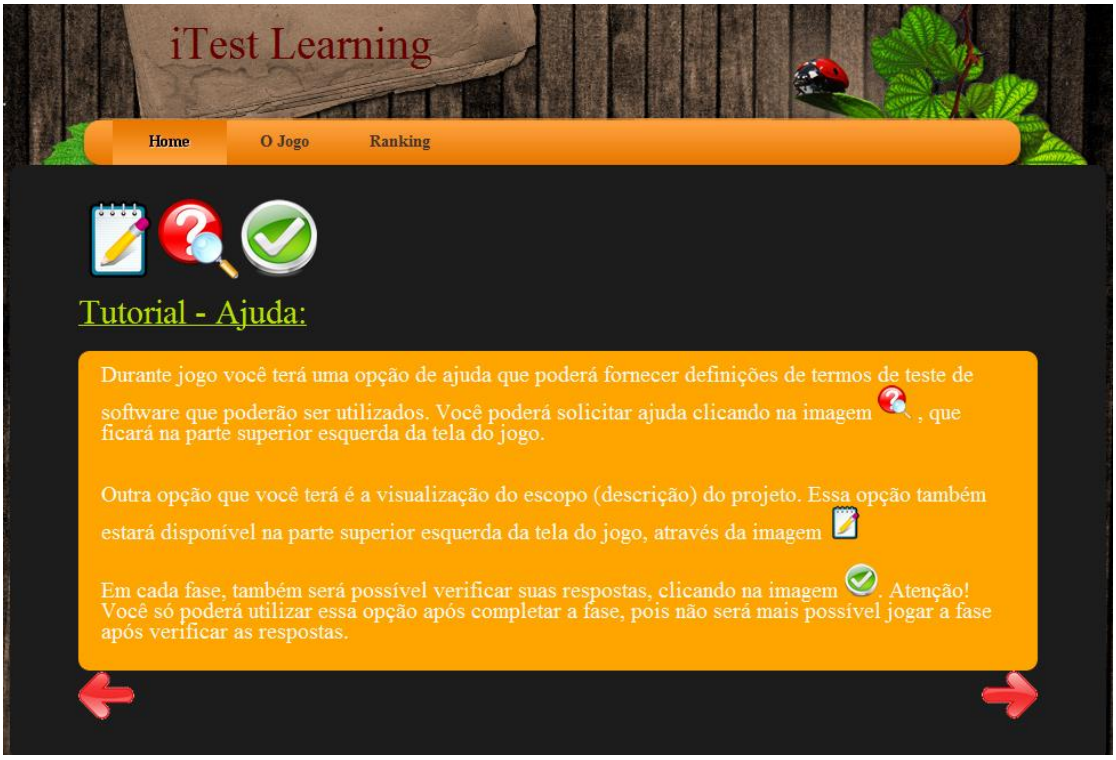

Figura 9 – Tela do tutorial explicando o sistema de ajuda do jogo. Fonte: a autora.

Sabendo como funciona o jogo, suas fases e suas funções, é possível dar início ao jogo. O jogo possui três níveis de dificuldade: fácil, médio, difícil. Podem existir diversos projetos que são relacionados a algum nível de dificuldade. Para iniciar o jogo, é preciso escolher um desses níveis de dificuldade e depois escolher um projeto referente ao nível de dificuldade escolhido (Figura 10).

Figura 10 – Tela de escolha do nível de dificuldade e projeto. Fonte: a autora.

Ao escolher o projeto, o jogo apresenta a descrição do projeto escolhido ao jogador (Figura 11). A partir daí, ele deverá realizar o planejamento de testes de software para esse projeto baseado na sua descrição.

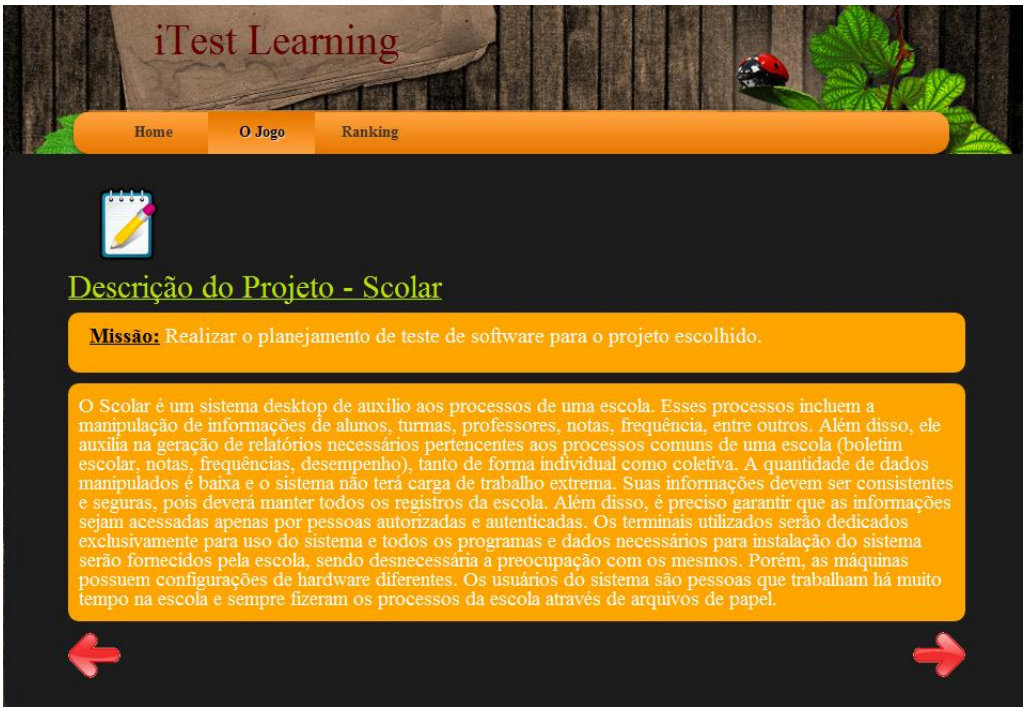

Figura 11 – Tela de descrição do projeto escolhido. Fonte: a autora.

Na primeira fase do jogo (Figura 12), o jogador deve escolher quais itens de teste farão parte do planejamento, ou seja, quais os itens dentre os apresentados estão de acordo com o projeto e serão os itens a serem testados durante a fase de execução.

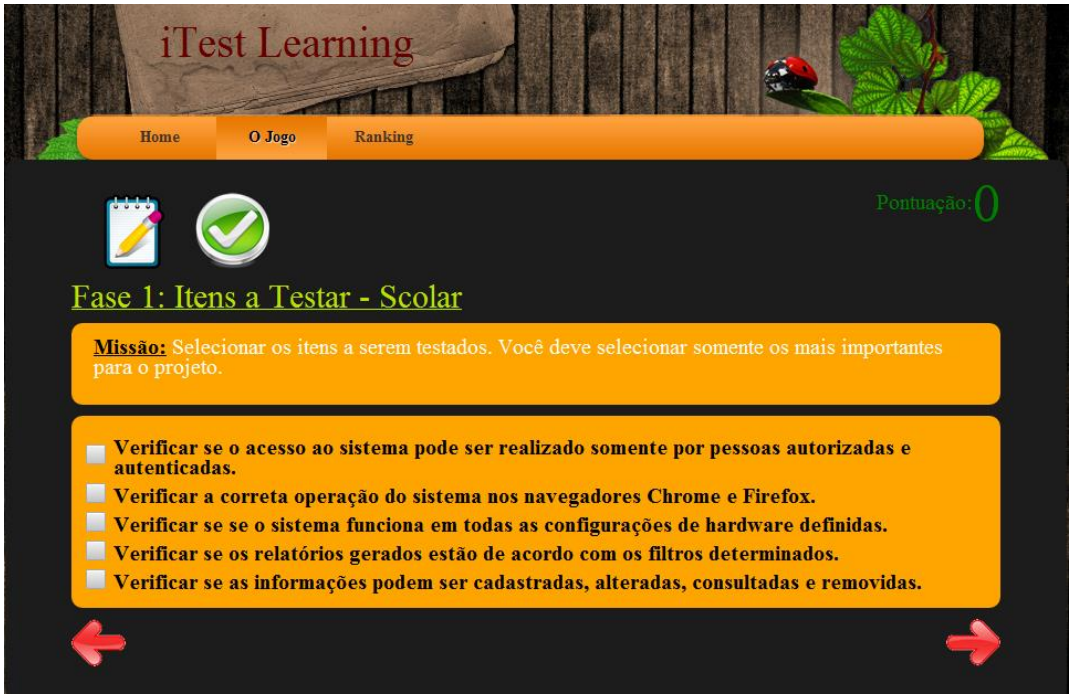

Figura 12 – Tela da Fase 1. Fonte: a autora.

A segunda fase (Figura 13) permite ao jogador definir quais os tipos de teste que serão realizados durante o desenvolvimento do software. Para isso, o jogador deve ter conhecimento sobre os tipos de testes existentes de forma a definir de maneira correta quais deles são apropriados ao projeto.

Figura 13 – Tela da Fase 2. Fonte: a autora.

Utilizada de forma a medir o conhecimento do jogador no processo de teste de software, a terceira fase (Figura 14) - níveis de teste – serve para o jogador identificar em quais níveis de teste são realizados os teste propostos.

Figura 14 – Tela da Fase 3. Fonte: a autora.

O critérios de aceitação são definidos na fase 4 (Figura 15). Aqui o jogador deve escolher quais critérios são relevantes para fazer com que os testes possam ser aceitos ou rejeitados. O jogador deve perceber que os critérios de aceitação devem estar de acordo com os itens de teste escolhidos na primeira fase.

Figura 15 – Tela da Fase 4. Fonte: a autora.

A fase seguinte, ferramentas, irá definir quais ferramentas poderão ser utilizadas no projeto para a realização dos testes ou para auxílio no processo de teste. O jogador deverá escolher as ferramentas que estão de acordo com o projeto e os testes que serão realizados no projeto. Esta fase é ilustrada na Figura 16.

Figura 16 – Tela da Fase 5. Fonte: a autora.

A última fase (Figura 17) define quais artefatos podem fazer parte do processo de teste. Esses artefatos podem ser tanto de saída quanto de entrada. É importante que o jogador

saiba o que é produzido no processo de testes e também o que pode ser necessário como estrada para a realização de determinadas tarefas.

Figura 17 – Tela da Fase 6. Fonte: a autora.

Ao final do jogo, uma tela com um *feedback* contendo todo o planejamento feito pelo jogador é exibida (Figura 18 e Figura 19). Logo abaixo, há também um planejamento que seria recomendado para o projeto escolhido. O *feedback* contém uma breve descrição do porquê de cada escolha, de forma a fazer o jogador refletir e também discutir as respostas com outros jogadores. Assim, o jogador poderá fazer uma comparação do seu planejamento com o planejamento recomendado. É preciso lembrar que o planejamento recomendado não necessariamente quer dizer que é o correto, pois é difícil definir o que é certo e o que é errado absolutamente.

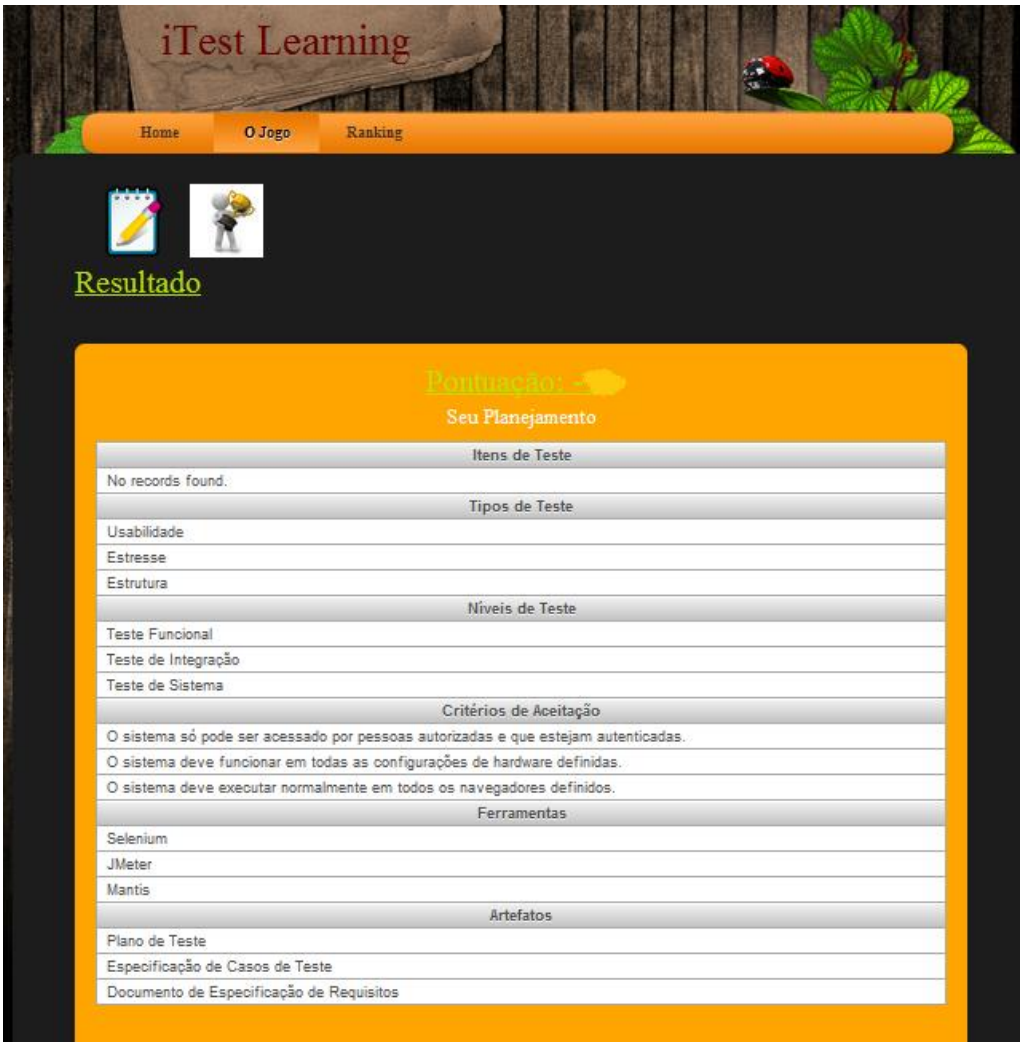

Figura 18– Tela do *feedback* – Planejamento do participante. Fonte: a autora.

Figura 19 – Tela do *feedback* – Planejamento sugerido. Fonte: a autora.

Depois do término o jogo, o jogador tem a opção de ir para o *ranking* do iTest Learning (Figura 20). Para isso, basta clicar no ícone que corresponde ao *ranking* e colocar seu nome.

Figura 20 – Tela de ir para o *ranking*. Fonte: a autora.

O ranking serve como uma forma indireta de interação entre os jogadores, através da competição que é gerada para estar no topo do ranking. Existe um ranking para cada

projeto (Figura 21). Para visualizar, basta ir ao menu ranking e escolher o projeto do qual deseja ver o ranking.

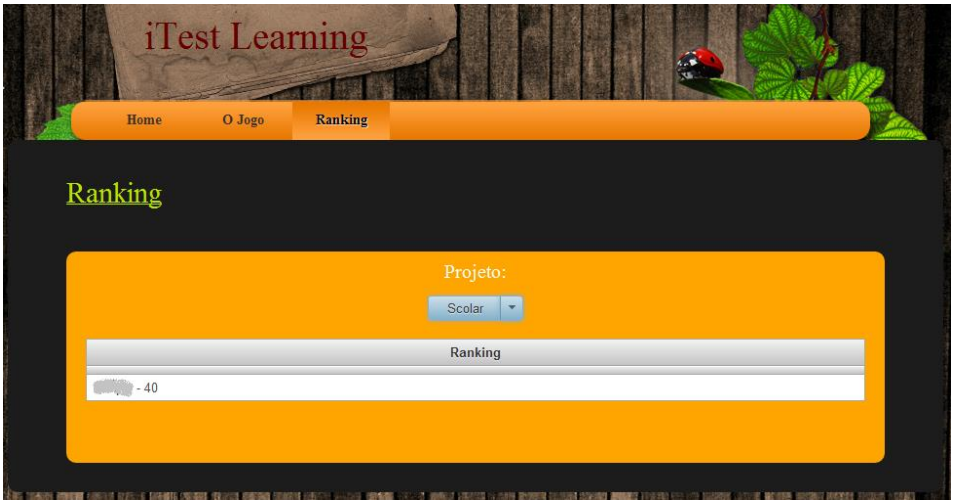

Figura 21 – Tela de *ranking*. Fonte: a autora.

4.3 Modelagem do Jogo

Nesta seção, são apresentados o diagrama de atividades do jogo, o diagrama de classes, o modelo do banco de dados e as tecnologias utilizadas durante o desenvolvimento do jogo.

4.3.1 Diagrama de Atividades

Representa a execução de ações ou atividades e os fluxos que são disparados pela conclusão de outras ações ou atividades (MELO, 2004). Abaixo é mostrado o diagrama de atividades do iTest Learning (Figura 22).

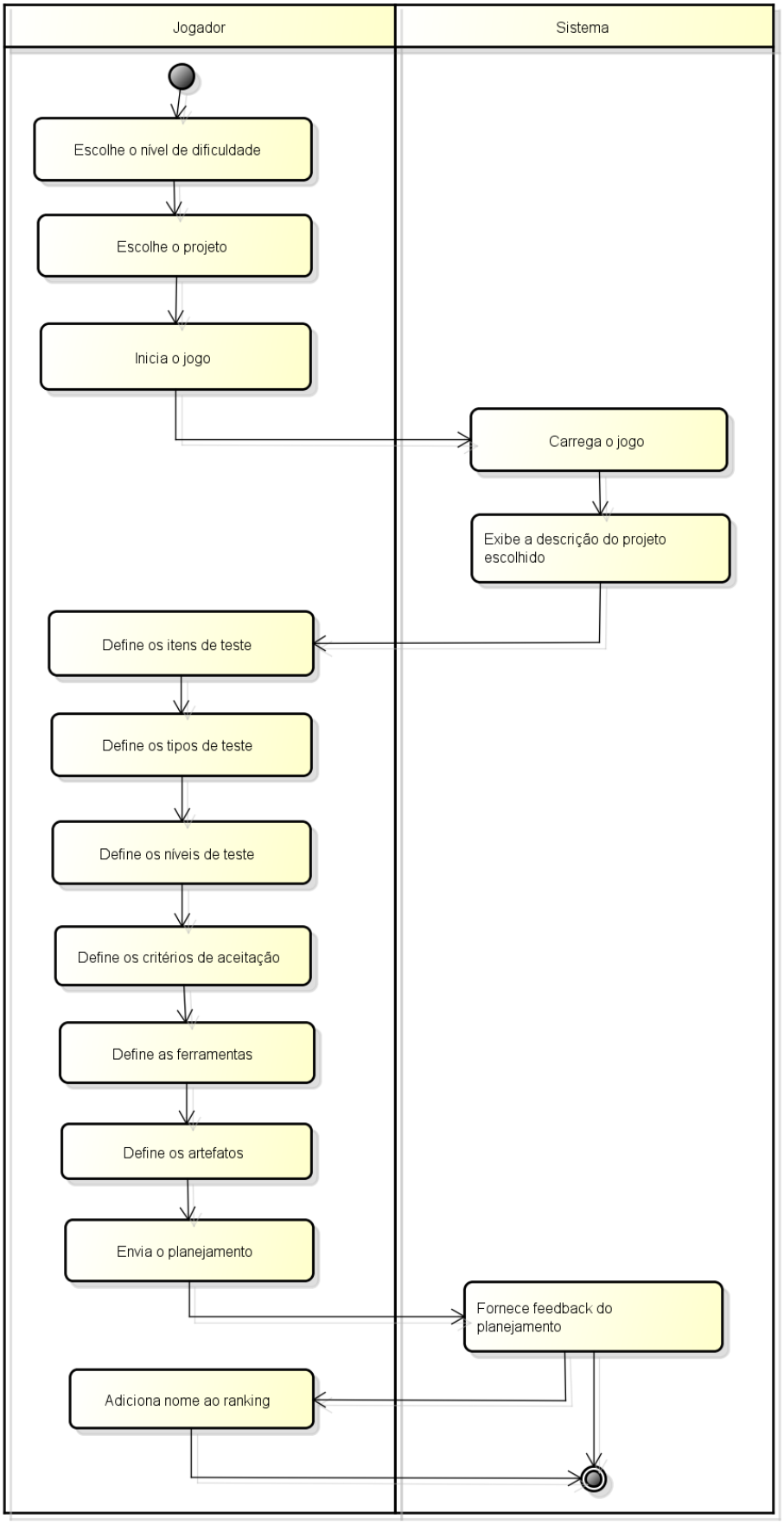

Figura 22 – Diagrama de Atividades. Fonte: a autora.

4.3.2 Diagrama de Classes

Apresenta elementos conectados por relacionamentos e é usado para exibir entidades do mundo real, além de elementos de análise e projeto (MELO, 2004). Na Figura 23 é apresentado o diagrama de classes do iTest Learning.

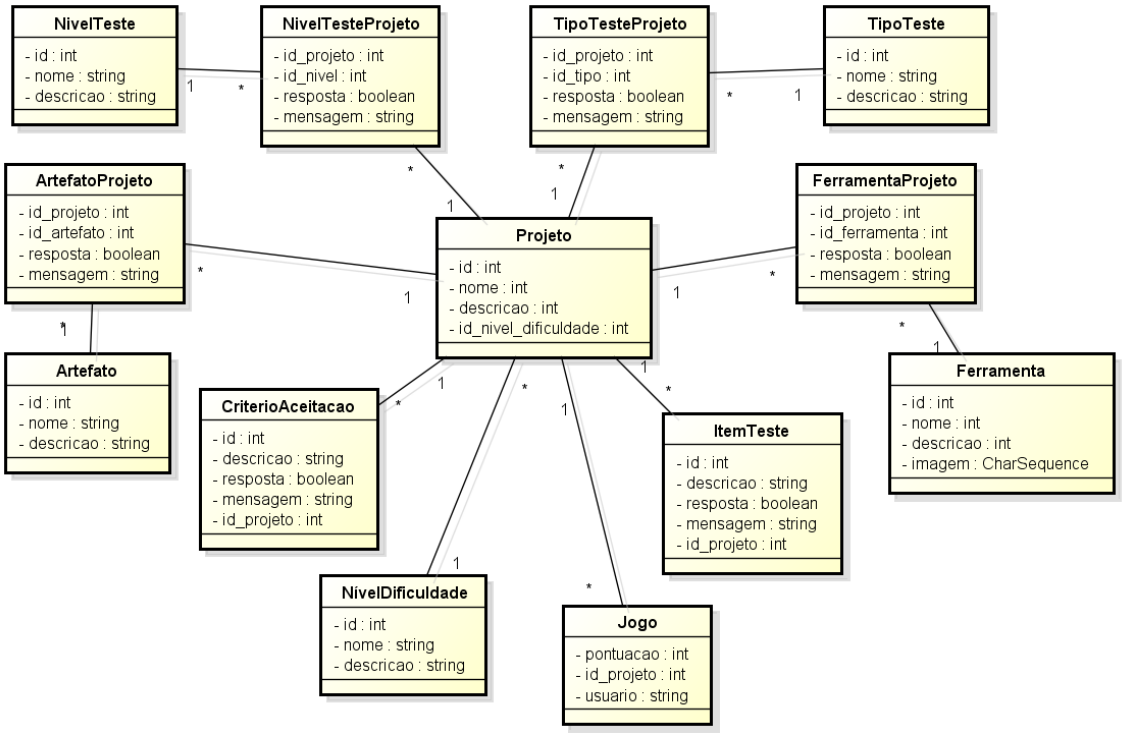

Figura 23 – Diagrama de classes. Fonte: a autora.

4.3.3 Tecnologia Utilizada

O jogo foi implementado na linguagem Java, utilizando o *framework* Java Server Faces, que se destina a simplificar o desenvolvimento de interfaces de usuário baseadas em web. Foram utilizados ainda os frameworks Hibernate, para persistência dos dados, e o Primefaces, suíte de componentes JSF customizados, a maioria com suporte Ajax, que ajuda no desenvolvimento de aplicações web dando mais facilidade e robustez no desenvolvimento. Para isso, a arquitetura foi baseada no Model-View-Controller (MVC), um padrão ou arquitetura que tem por objetivo dividir o sistema em camadas de modo a lógica de negócio da lógica de apresentação. Como servidor de aplicação foi utilizado o Tomcat, o banco de dados utilizado foi o *PostgreSql* e o XHTML na camada de visão.

5 ESTUDO DE CASO

Neste capítulo são apresentados sobre a aplicação e os resultados obtidos da avaliação realizada. O processo de avaliação tem como objetivo verificar a efetividade do jogo proposto, através de uma avaliação qualitativa.

A avaliação foi realizada no dia 12/06/2012 com 7 alunos da turma de Verificação e Validação do quinto semestre do curso de Engenharia de Software da Universidade Federal do Ceará, Campus Quixadá. Antes, foi realizada uma apresentação e esclarecimento dos objetivos da avaliação e sobre a livre participação. Logo após, os participantes assinaram um termo de concordância e foram orientados quanto ao preenchimento do questionário da pesquisa. O questionário deveria ser preenchido após a utilização da ferramenta por parte do participante. O Questionário de Avaliação e o Termo de Concordância encontram-se disponíveis nos Apêndices A e B, respectivamente.

O questionário apresenta questões objetivas (múltipla escolha) e discursivas. Para cada questão objetiva, o participante deveria apontar um valor de 1 a 4, correspondendo às respostas “não”, “mais para não”, “mais para sim” e “sim”, respectivamente. Porém, a pergunta “Em relação ao grau de dificuldade do jogo, você o considera:” é uma exceção, pois suas alternativas de resposta são “muito fácil (precisa aumentar o grau de dificuldade)”, “fácil”, “adequado (não precisa ser alterado)”, “difícil” e “muito difícil (precisa diminuir o grau de dificuldade)” e seus resultados serão avaliados à parte das demais questões objetivas. Já as questões discursivas foram definidas como campos de texto onde o participante poderia expressar sua opinião sobre o jogo de forma escrita.

A Tabela 1 resume as respostas dadas pelos participantes em relação às questões objetivas. A coluna **Pergunta** corresponde à pergunta contida no questionário. As respostas dos participantes estão contidas na coluna **Respostas**.

Tabela 1- Respostas das questões objetivas do Questionário. Fonte: a autora.

Perguntas	**Respostas**							**Média**
	1	**2**	**3**	**4**	**5**	**6**	**7**	
Você gostou do jogo?	4	4	4	4	4	4	4	4
Os objetivos de aprendizagem foram atingidos com o jogo?	3	4	4	3	3	3	3	3,29
O conteúdo do jogo foi relevante para o aprendizado?	4	3	4	3	3	3	2	3,15
O jogo ajuda a desenvolver a habilidade de planejar teste de software?	4	4	4	2	3	2	3	3,15
O método de ensino do jogo foi adequado?	4	4	4	3	3	4	3	3,58

Em relação à primeira pergunta da Tabela 1 – "Você gostou do jogo?" – é possível observar que 7 participantes (100%) responderam "sim", conforme a Figura 24.

Figura 24 - Resultado da Pergunta 1. Fonte: a autora.

Em relação à segunda pergunta – "Os objetivos de aprendizagem foram atingidos com o jogo?" – é possível observar que 2 participantes (29%) responderam "sim" e 5 (71%) responderam "mais para sim", conforme a Figura 25.

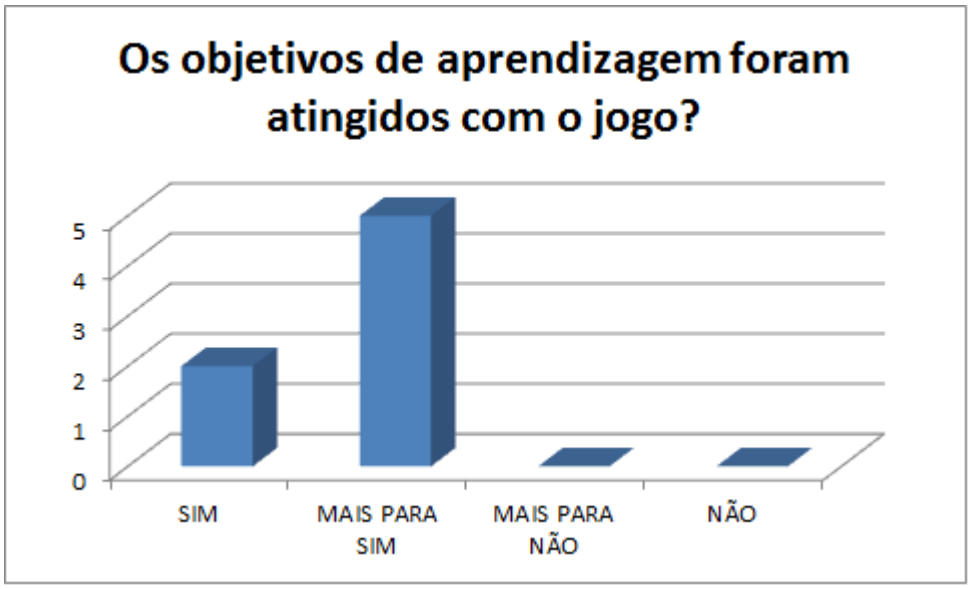

Figura 25 - Resultado da Pergunta 2. Fonte: a autora.

Em relação à terceira pergunta – "O conteúdo do jogo foi relevante para o aprendizado?" – é possível observar que 2 participantes (28%) responderam "sim", 4 (68%) responderam "mais para sim" e 1 (14%) respondeu "mais para não", conforme a Figura 26.

Figura 26 - Resultado da Pergunta 3. Fonte: a autora.

Em relação à quarta pergunta – "O jogo ajuda a desenvolver a habilidade de planejar teste de software?" – é possível observar que 3 participantes (42%) responderam "sim", 2 (29%) responderam "mais para sim" e 2 (29%) responderam "mais para não", conforme a Figura 27.

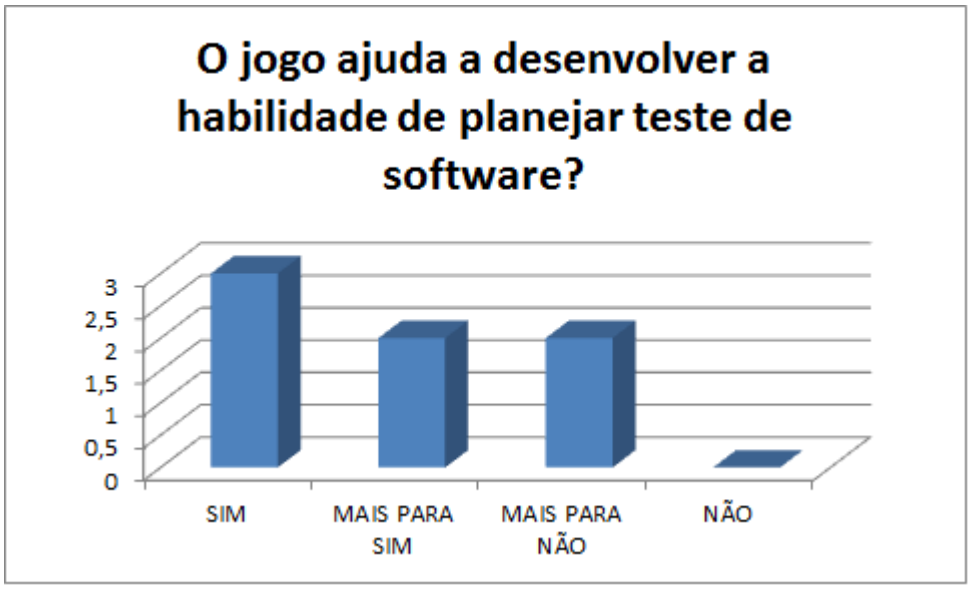

Figura 27 - Resultado da Pergunta 4. Fonte: a autora.

Em relação à quinta pergunta – "O método de ensino do jogo foi adequado?" – é possível observar que 4 participantes (57%) responderam "sim" e 3 (43%) responderam "mais para sim", conforme a Figura 28.

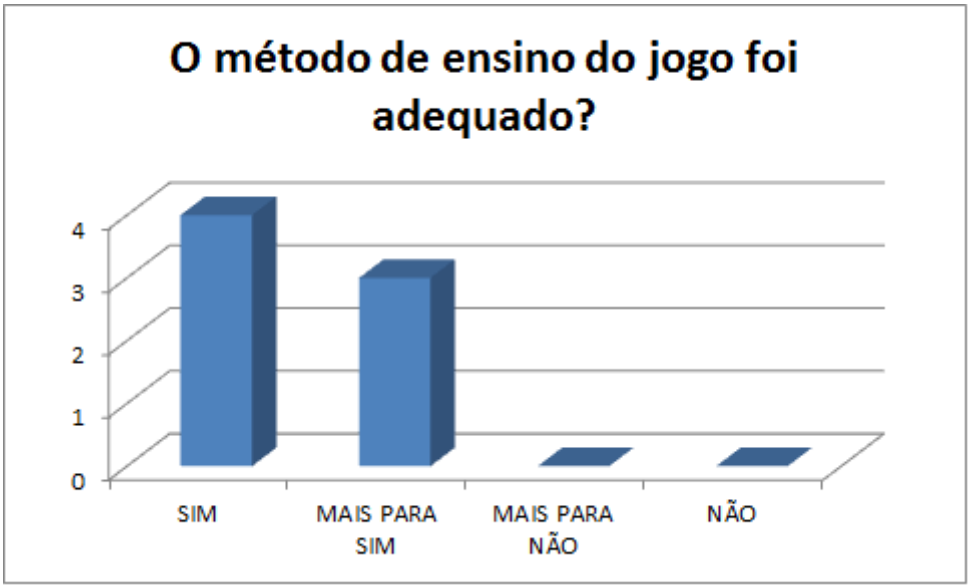

Figura 28 - Resultado da Pergunta 5. Fonte: a autora.

Com base nesses resultados, é possível verificar que o iTest Learning agradou aos participantes e sua metodologia de ensino é adequada. Ele ainda provê um aprendizado em relação ao conteúdo proposto, faltando apenas direcioná-lo um pouco mais de maneira a estar mais focado no planejamento de testes de software.

Ainda com relação às perguntas objetivas, a Tabela 2 mostra as respostas para a questão “Em relação ao grau de dificuldade do jogo, você o considera:”, onde MF respresenta a alternativa de resposta “muito fácil”, F representa “fácil”, A respresenta “adequado”, D representa “difícil” e MD representa “muito difícil”.

Tabela 2 - Resultado da questão objetiva referente ao grau de dificuldade do jogo.

Pergunta	**Respostas**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Em relação ao grau de dificuldade do jogo, você o considera:	F	A	F	A	F	F	F

Em relação à sexta pergunta – “Em relação ao grau de dificuldade do jogo, você o considera:” – é possível observar que 2 (29%) participantes responderam “adequado” e 5 (71%) responderam “fácil”, como ilustra a [figura].

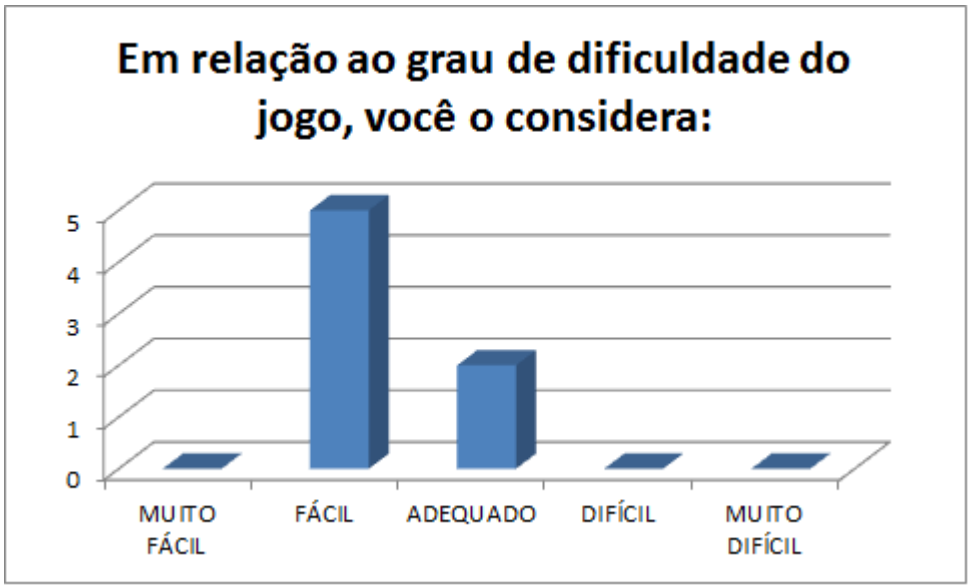

Figura 29 - Resultado da Pergunta 6. Fonte: a autora.

Com base no resultado apresentado, é possível concluir que o nível de dificuldade do jogo está adequado, não estando nos limites de fácil ou difícil. Deve-se levar em consideração que a avaliação foi feita em relação a uma instância de jogo que pertencia ao nível de dificuldade fácil, já que havia apenas essa instância no iTest Learning. Isso pode ter influenciado na avaliação da dificuldade geral do jogo, já que deveria se avaliar se o nível fácil está adequado a um nível fácil.

Entretanto, não se pode assumir uma generalização do resultado pelo fato de a amostra ser relativamente pequena.

O APÊNDICE C apresenta as respostas das questões discursivas dos participantes de forma detalhada. As respostas obtidas nessas questões indicam que os participantes gostaram mais do fato de o jogo poder reforçar o conteúdo ministrado na disciplina, das opções de ajuda que são dadas no decorrer do jogo e do poder de tomada de decisão que é proporcionado ao jogador. Em relação àquilo de que menos gostaram no jogo, têm-se as perguntas com pegadinhas, complexidade e especificidade de algumas perguntas e levantamento de dúvidas pela ajuda dada com relação ao problema apresentado. Dentre as sugestões de melhoria, foram citadas inclusão de outras instâncias de jogo para os outros níveis de dificuldade (já que havia apenas para o nível fácil); adição de novas funcionalidades características de jogos, como pular a fase, pedir ajuda, jogar com outras pessoas *online*; inserir alerta para a verificação de respostas (já que essa opção torna desabilita a jogabilidade da fase atual); melhoria do tutorial para definir melhor os itens contidos no jogo.

6 CONSIDERAÇÕES FINAIS

Apesar de já fazer parte de todo o processo de desenvolvimento de software, o teste de software ainda tem tido pouca atenção nos currículos de graduação. E quando há a oportunidade do estudo dessa área, o tema é abordado de maneira muito teórica, faltando muitas vezes se trabalhar a parte prática, como é feito, por exemplo, com disciplinas de programação. Isso pode acontecer por motivos como a falta de tempo hábil para a realização de práticas relacionadas ao conteúdo ministrado. A partir dessa motivação, propôs-se a concepção, construção e avaliação de um jogo educacional para apoiar o ensino de teste de software focado no planejamento de testes.

Para alcançar os objetivos propostos, foi realizada uma revisão bibliográfica relacionada aos conceitos de teste de software e ferramentas voltadas para o ensino de Engenharia de Software. Baseando-se nessa revisão, foi realizada a concepção e construção do jogo proposto. Após a construção do jogo, foi realizada uma avaliação qualitativa do mesmo, de forma a verificar se o jogo construído atingiu seus objetivos. Com os resultados obtidos, foi possível observar que o jogo atingiu os objetivos propostos, sendo uma ferramenta útil para apoiar o ensino de testes de software. A principal contribuição deste trabalho foi a concepção e implementação do jogo iTest Learning.

Atualmente, o iTest Learning possui apenas uma instância de jogo que pertence ao nível de dificuldade fácil. Como trabalhos futuros é previsto a adição de novos projetos para os demais níveis de dificuldade. Além disso, podem-se adicionar novos itens (fases) de maneira a tornar o planejamento de testes mais completo. Outras avaliações também poderiam ser feitas, com um maior número de participantes, de maneira a obter resultados mais consistentes. Também se prever a extensão do jogo para as outras fases dos testes de software como a fase de projeto e execução dos testes.

REFERÊNCIAS

ALDRICH, C. **Learning by Doing: A Comprehensive Guide to Simulations, Computer Games, and Pedagogy in E-learning and Other Educational Experiences**. Hoboken: Wiley, 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: Minicurso apresentado no XIV Simpósio Brasileiro de Engenharia de Software (SBES), João Pessoa, 2000.

BASTOS, A.; RIOS, E.; CRISTALLI, R.; MOREIRA, T. **Base de Conhecimento em Teste de Software**. 2.ed. São Paulo: Martins Fontes, 2007.

BRATHWAITE, B; SCHREIBER, I. **Challenges for game designers**. Boston: Charles River Media, 2009

CMMI Product Team. **CMMI for Development, Version 1.2. Technical Report CMU/SEI-2006-TR-008**, Software Engineering Institute/ Carnegie Mellon University, Pitsburgh, Pennsylvania, 2006.

CRAIG, R.; JASKIEL, S. **Systematic Software Testing**. Norwood: Artech House Publishers, 2002.

CRESPO, A. N.; SILVA, O. J.; BORGES, C. A.; SALVIANO, C. F.; JINO, M. **Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo**. In: III Simpósio Brasileiro de Qualidade de Software (SBQS), 2004.

DELAMARO, M.; MALDONADO, J.; JINO, M. **Introdução ao Teste de Software**. 1.ed. Rio de Janeiro: Elsevier, 2007.

DEMPSEY, J.; RASMUSSEN, K; LUCASSEN, B. **Intructional Gaming: implication for instructional technology**. Annual Meeting of Association for Educacional Communications and Technology, Nashville, 1994.

DINIZ, L. L., DAZZI, R. L. S. **Jogo Digital para o Apoio ao Ensino do Teste de Caixa-Preta**. In: X Simpósio Brasileiro de Qualidade de Software (SBQS), Curitiba, 2011.

ELBAUM, S., PERSON, S. e DOKULIL, J. **Bug Hunt: Making Early Software Testing Lessons Engaging and Affordable**. In: 29th International Conference on Software Engineering (ICSE), Washington, 2007.

EL-SHAMY, S. **Training Games: Everything You Need to Know about Using Games to Reinforce Learning**. Stylus Publishing, 2001.

FIGUEIREDO, K.; FERREIRA, J., MURTA, L.; CLUA, E. **Um Jogo de Estratégia de Gerência de Configuração**. In: III Fórum em Educação de Engenharia de Software (FEES), XXIV Simpósio Brasileiro de Engenharia de Software (SBES), Salvador, 2010.

GONÇALVES, R. Q.; THIRY, M.; ZOUCAS, A. **Avaliação da Aprendizagem em Experimentos com Jogo Educativo de Engenharia de Requisitos**. In: X Simpósio Brasileiro de Qualidade de Software (SBQS), Curitiba, 2011.

IEEE Standard 829-1998: **Standard for Software Test Documentation**, IEEE Press.

JONES, K. **Simulations**: a handbook for teachers and trainers. London: Third Edition Routledge, 1995.

KAFAI, Y. B. **The Educational Potential of Electronic Games: From Games-To-Teach to Games-To-Learn**. In: Conference on Playing by the Rules: The Cultural Policy Challenges of Video Games, Chicago, 2001.

KIELING, E., ROSA, R. **Planeger – Um Jogo para Apoio ao Ensino de Conceitos de Gerência de Projetos de Software**. Monografia de Trabalho de Conclusão de Curso de Graduação, Ciência da Computação, FACIN, PUCRS, Porto Alegre, 2006.

KOSCIANSKI, A., SOARES, M. S. **Qualidade de Software**. 2.ed. São Paulo: Editora Novatec, 2007.

LEWIS, W. E. **Software Testing Continuous Quality Improvement**. Boca Raton: Auerbach, 2000.

MALDONADO, J. C. **Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software**. PhD *thesis*, DCA/FEE/UNICAMP, Campinas, 1991.

MELO, A. C. **Desenvolvendo Aplicações com UML 2.0: do conceitual à implementação**. 2. ed. Rio de Janeiro: Basport, 2004.

MONSALVE, E. S., WERNECK, V. M. B., LEITE, J. C. S. P. **SimulES-W: Um Jogo para o Ensino de Engenharia de Software**. In: III Fórum em Educação de Engenharia de Software (FEES), XXIV Simpósio Brasileiro de Engenharia de Software (SBES), Salvador, 2010.

MPS.BR. **Melhoria de Processo do Software Brasileiro**, Guia Geral, 2011.

MYERS, G. J. **The Art of Software Testing**. 2.ed. New Jersey: John Wiley & Sons, Inc., 2004.

PMBOK. A Guide to the Project Management Body of Knowledge, PMI, Pensylvania, U.S, 2004.

POOLE, S. **Trigger Happy: Videogames and the Entertainment Revolution**. Arcade Publishing, 2000.

PRESSMAN, R. S. **Engenharia de Software**. 6.ed. Rio de Janeiro: McGraw Hill, 2006.

RIOS, E.; MOREIRA, T. **Teste de Software**. Rio de Janeiro: Alta Books, 2003.

ROCHA, A. R. C. da; MALDONADO, J. C.; WEBER, K. C. **Qualidade de Software: teoria e prática**. São Paulo: Prentice Hall, 2001.

ROLLINGS, A.; ADAMS, E. **Andrew Rollings and Ernest Adams on game design**. USA: New Riders Group, 2003.

RUP. **Rational Software Corporation. Rational Unified Process®, RUP,** 2002. Disponível em http://www.wthreex.com/rup/. Acesso em outubro de 2011.

SHEPARD, T., LAMB, M., KELLY, D. **More Testing Should be Taught**. Communications of the ACM, 2001.

SILVA, A. R. **Uma Metodologia de Testes em Software para Micro e Pequenas Empresas Estruturada em Multicritério**. Dissertação de Curso de Mestrado, Informática Aplicada, UNIFOR, Fortaleza, 2011.

SILVA, A. C.; **Jogo Educacional para Apoiar o Ensino de Técnicas para Elaboração de Testes de Unidade**. Dissertação de Curso de Mestrado, Computação Aplicada, UNIVALI, São José, 2010.

SOMMERVILLE, I. **Engenharia de Software**. 8.ed. São Paulo: Pearson Education, 2007.

SOUZA, M. M., RESENDE, R. F., PRADO, L. S., FONSECA, E. F., CARVALHO, F. A., RODRIGUES, A. D. **SPARSE: Um ambiente de Ensino e Aprendizado de Engenharia de Software Baseado em Jogos e Simulação**. In: XXI Simpósio Brasileiro em Informática na Educação (SBIE), João Pessoa, 2010.

THIRY, M., ZOUCAS, A., GONÇALVEZ, R. Q. **Promovendo a Aprendizagem de Engenharia de Requisitos de Software através de um Jogo Educativo**. In: XXI Simpósio Brasileiro em Informática na Educação (SBIE), João Pessoa, 2010.

VON WANGENHEIM, C.; KOCHANSKI, D.; SAVI, R. **Revisão Sistemática sobre Avaliação de Jogos Voltados para Aprendizagem de Engenharia de Software no Brasil**. In: II Fórum em Educação de Engenharia de Software (FEES), XXIII Simpósio Brasileiro de Engenharia de Software (SBES), Fortaleza, 2009a.

VON WANGENHEIM, C; THIRY, M; KOCHANSKI, D; STEIL, L; SILVA, D; LINO, J. **Desenvolvimento de um Jogo para Ensino de Medição de Software**. In: VIII Simpósio Brasileiro de Qualidade de Software (SBQS), Ouro Preto, 2009b.

XMIND COLLABORATIVE MINDS. [site corporativo]. Disponível em: <http://www.xmind.net/share/antoniopassos/furps/>. Acesso em: 01 set. 2011.

YEE, N. The Labor of Fun: how video games blur the boundaries of work and play. **Games and Culture**, v. 1, p. 68-71, 2006.

APÊNDICES

APÊNDICE A – QUESTIONÁRIO DE AVALIAÇÃO DO JOGO

Este questionário tem como objetivo avaliar um jogo para o ensino de planejamento de testes de software que foi desenvolvido para o âmbito de um trabalho de monografia do curso de Sistemas de Informação do Campus UFC Quixadá, pela aluna Virgínia Farias. Contamos com sua colaboração para avaliar a eficiência e eficácia do jogo em relação ao ensino/aprendizado do Planejamento de Testes de Software.

1) Você gostou do jogo?

   ( ) Não

   ( ) Mais para não

   ( ) Mais para sim

   ( ) Sim

2) Os objetivos de aprendizagem foram atingidos com o jogo?

   ( ) Não

   ( ) Mais para não

   ( ) Mais para sim

   ( ) Sim

3) O conteúdo do jogo foi relevante para o aprendizado?

   ( ) Não

   ( ) Mais para não

   ( ) Mais para sim

   ( ) Sim

4) O jogo ajuda a desenvolver a habilidade de planejar teste de software?

   ( ) Não

   ( ) Mais para não

   ( ) Mais para sim

( ) Sim

5) O método de ensino do jogo foi adequado?

( ) Não

( ) Mais para não

( ) Mais para sim

( ) Sim

6) Em relação ao grau de dificuldade do jogo, você o considera:

( ) Muito fácil (precisa aumentar o grau de dificuldade)

( ) Fácil

( ) Adequado (não precisa ser alterado)

( ) Difícil

( ) Muito Difícil (precisa diminuir o grau de dificuldade)

7) O que você mais gostou no jogo?

_____

_____

8) O que você menos gostou no jogo?

_____

_____

9) Você tem alguma sugestão de melhoria para o jogo? Caso sim, descreva-a.

_____

_____

APÊNDICE B – TERMO DE CONCORDÂNCIA

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Você está sendo convidado para participar da pesquisa Avaliação do Jogo iTest Learning.

Você foi selecionado por estar matriculado na disciplina de Verificação e Validação. Sua participação não é obrigatória, a qualquer momento você pode desistir de participar e retirar seu consentimento. Sua recusa não trará qualquer prejuízo em sua relação com o pesquisador ou com a instituição.

O objetivo deste estudo é avaliar o efeito de aprendizagem no uso do Jogo iTest Learning.

Sua participação nesta pesquisa consistirá em utilizar o jogo educacional e responder ao questionário.

Não foram identificados quaisquer riscos na participação desta pesquisa.

Os benefícios relacionados com a sua participação são aprendizagem de conceitos relacionados a teste de software, principalmente com relação ao planejamento de testes.

As informações obtidas através desta pesquisa serão confidenciais e asseguramos o sigilo absoluto sobre sua participação. Os dados não serão divulgados de forma a possibilitar sua identificação.

Virginia Farias de Sousa _____

Nome e assinatura da pesquisadora

Declaro que entendi os objetivos, riscos e benefícios de minha participação na pesquisa e concordo em participar.

_____

Sujeito da pesquisa

APÊNDICE C – RESPOSTAS DAS QUESTÕES DISCURSIVAS - AVALIAÇÃO DO JOGO

Pergunta 1: O que você mais gostou no jogo?	
Participante	**Resposta**
1	-
2	Revisão dos conteúdos vistos na cadeira de Verificação e Validação e esclarecimento de algumas dúvidas.
3	O fato de relembrar conceitos, conseguindo assim melhorar os conceitos vistos em Teste e a questão de verificar as respostas, onde tem-se um *feedback* dos erros fazendo o jogador aprender com os erros.
4	As ajudas, pois servem como um guia para o aprendizado do planejamento de teste.
5	A flexibilidade de tomar decisões proporcionada aos jogadores.
6	-
7	A opção de ajuda, que auxilia no aprendizado de um determinado conceito caso ele tenha dúvidas.

Pergunta 2: O que você menos gostou no jogo?	
Participante	**Resposta**
1	-
2	Algumas perguntas muito complexas. Mas muito bom para o aprendizado.
3	-
4	Em relação às ajudas, as definições expostas as vezes geram dúvida em relação à escolha correta.
5	-
6	As questões com pegadinhas.
7	Algumas perguntas eram muito específicas.

Pergunta 3: Você tem alguma sugestão de melhoria para o jogo? Caso sim, descreva-a.	
Participante	**Resposta**
1	No momento do click em “Verificar Resposta”, exibir uma tela de alerta perguntando se realmente deseja saber a resposta, pois isso irá fazer com que a opção de seleção seja perdida.
2	Poderia ter opções de pular, jogar online com outra pessoa, e pedir ajuda. Tipo numa pergunta sobre caixa branca, se fosse pedir ajuda, teria a explicação do que era caixa branca e tudo mais.
3	Adicionar outros níveis de dificuldade.
4	Em relação às ajudas, as definições expostas as vezes geram dúvida em relação à escolha correta.
5	Não.
6	Questões pegadinhas servem pra tirar dúvidas muito específicas, que às vezes as pessoas nem têm.
7	A ajuda poderia ficar mais explícita, para mostrar que o usuário deve consultá-la para responder às perguntas. A opção de conferir as respostas também deveria ficar mais explícita que deve ser consultada para as fases seguintes. Ou seja, um tutorial inicial explicando melhor esses botões e como utilizá-los para jogar.