KELY SANTOS MACÊDO

AS AVENTURAS DE JACK TEST:
JOGO EDUCACIONAL PARA O APOIO AO ENSINO DE TESTE DE SOFTWARE

VITÓRIA DA CONQUISTA

MARÇO – 2014

UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA

DEPARTAMENTO DE CIÊNCIAS EXATAS

KELY SANTOS MACÊDO

AS AVENTURAS DE JACK TEST:
JOGO EDUCACIONAL PARA O APOIO AO ENSINO DE TESTE DE SOFTWARE

Trabalho de Conclusão de Curso apresentado à disciplina Projeto de Computação Supervisionado II do Departamento de Ciência Exatas da UESB, como requisito parcial para a obtenção do título Bacharel em Ciência da Computação.

Orientador: Esp. Fabrício de Sousa Pinto

VITÓRIA DA CONQUISTA

MARÇO - 2014

KELY SANTOS MACÊDO

JOGO AS AVENTURAS DE JACK TEST:
UM JOGO EDUCACIONAL PARA O APOIO AO ENSINO DE TESTE DE SOFTWARE

Trabalho de Conclusão de Curso apresentado com requisito parcial à obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade Estadual do Sudoeste da Bahia.

Banca Examinadora:

Orientador:

_____
Esp. Fabrício de Sousa Pinto
Universidade Estadual do Sudoeste da Bahia

Membro 1:

_____
Dr. Francisco dos Santos Carvalho
Universidade Estadual do Sudoeste da Bahia

Membro 2:

_____
Esp. Crijina Chagas Flores
Universidade Estadual do Sudoeste da Bahia

Vitória da Conquista – BA, 07 de março de 2014

AGRADECIMENTOS

Agradeço a Deus por ter me dado forças para superar as dificuldades e concluir está etapa da minha vida.

Aos meus pais, irmãos e esposo pelo amor, incentivo e paciência que dispensaram a mim.

Ao meu amado filho, que com pouco mais de um ano foi meu maior motivador. Pelo amor, carinho e sorrisos que me fortaleceram para essa jornada.

À querida amiga Roberta, pelo apoio, incentivo e pela amizade construída, que fica para sempre.

Ao orientador Fabrício pelo auxilio, apoio e paciência que teve comigo.

E a todos, que de alguma forma contribuíram para conclusão deste trabalho, o meu muito obrigada.

RESUMO

Uma das atividades que podem garantir a qualidade do software é o Teste de Software e, apesar disso, tem se dado pouca importância a esse assunto nos cursos de graduação, seja pela falta de tempo disponível, já que a disciplina Engenharia de Software é extensa, seja pela ausência de atividades práticas e lúdicas ou pela falta de motivação do aluno. Para contornar os problemas citados, neste trabalho foi proposto o desenvolvimento do jogo educacional “As aventuras de *Jack Test*” para auxiliar no ensino-aprendizagem dos conteúdos de testes de software. Os jogos reforçam os conceitos dados em sala de aula, por meio da prática, além de motivar o aluno, permitindo uma aprendizagem mais profunda. O jogo foi desenvolvido nas linguagens *HTML 5* e *Javascript*. Com o objetivo de avaliar a efetividade do jogo foi elaborado um questionário que foi aplicado aos alunos da Faculdade de Tecnologia e Ciência (FTC) da cidade de Vitória da Conquista - BA que cursam a disciplina de Engenharia de Software e tiveram acesso ao jogo. Por meio da análise das respostas do questionário pode-se validar o jogo, verificando uma contribuição positiva do jogo ao ensino de teste de software.

Palavras-chave: Engenharia de Software. Qualidade de software. Teste de Software. Jogos Educacionais.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

No mundo moderno o software passou a ocupar um espaço importante nas empresas e na vida da população em geral, e em um mercado consumidor cada vez mais exigente, surgiu a necessidade de se preocupar com a qualidade dos produtos e serviços oferecidos.

As empresas passaram a ter a preocupação de desenvolver softwares de alta qualidade, em um curto período de tempo e que atendam às necessidades dos clientes. Um software sem qualidade resulta em prejuízos financeiros para os desenvolvedores e usuários finais, insatisfação dos clientes, gastos desnecessários com correção de erros e com a manutenção do software, desgaste na imagem da empresa desenvolvedora, entre outros problemas.

O teste de software é uma das atividades que garantem a qualidade do software. Testar um software é executar um programa com o objetivo de encontrar erros.

Apesar da importância que o teste de software possui na garantia da qualidade dos softwares, é dedicado pouco tempo a esse assunto nos cursos de graduação da área de computação, além de existirem poucas atividades práticas e lúdicas, o que é essencial para complementar o ensino e auxiliar na aprendizagem.

Alguns jogos podem ser desenvolvidos com o objetivo de solucionar esses problemas, facilitando, dessa forma, o ensino-aprendizagem. Os jogos educacionais são utilizados no auxílio de ensino de diversas áreas, inclusive a computação, tornando o processo de ensino-aprendizagem mais atrativo e mais motivador para o aluno.

Este trabalho visou o desenvolvimento de um jogo educacional, que tem como objetivo auxiliar o ensino-aprendizagem de Teste de Software. Dessa forma, torna o ensino mais atraente, motivando o aluno; ajuda o aluno a entender melhor as fases do Teste de Software e fortalece os conceitos adquiridos em sala de aula.

Para o desenvolvimento desse projeto foi realizada, inicialmente uma ampla pesquisa bibliográfica sobre Engenharia de Software, com ênfase em Qualidade e Teste de Software, e sobre Jogos Educacionais. Além disso, foi feita também, pesquisas em artigos e em material disponível na internet relacionado aos assuntos citados acima.

Posteriormente foi iniciado o desenvolvimento do jogo “As Aventuras de *Jack Test*”. Para isso foram utilizadas as linguagens *HTML 5*, *JavaScript* e *Cascading Style Sheets* (CSS). Com o intuito de desenvolver o jogo foi necessário definir cenário, personagens e enredo; implementar e testar o jogo.

Neste primeiro capítulo apresenta uma visão geral do que foi abordado no trabalho desenvolvido, fornece alguns conceitos principais e o objetivo do trabalho. No segundo é feita uma revisão bibliográfica, onde são apresentados os conceitos de Engenharia de Software, Testes de Softwares e Jogos Educacionais, respectivamente. Já no terceiro capítulo é realizada a demonstração passo a passo do desenvolvimento do jogo “As aventuras de *Jack Test*”. O quarto capítulo apresenta uma relação dos trabalhos relacionados e uma breve comparação entre eles e o jogo desenvolvido. Por fim, o quinto capítulo faz uma análise das conclusões desse trabalho e apresenta sugestões para trabalhos futuros.

2 REVISÃO DA LITERATURA

2.1 Engenharia de Software

Sommerville (p.5, 2010) afirma que “software não é apenas um programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente.”

Segundo Mendes (2002, p.3) “o termo software não se restringe apenas aos programas de computadores associados a uma aplicação, mas também envolve toda a documentação necessária para instalação, uso, documentação e manutenção dos programas”.

Rios e Moreira (2006) relatam que o software conquistou um papel importante nas empresas e que essa importância tende a aumentar, pois as atividades e os produtos tendem a depender cada vez mais do software. Porém os produtos desenvolvidos, na sua maioria apresentam um grande número de defeitos que prejudicam a usabilidade, a segurança, a confiabilidade e a funcionalidade dos mesmos, o que resulta, muitas vezes em prejuízos para os desenvolvedores e usuários finais.

Inicialmente o desenvolvimento do software era visto como uma forma de arte, não era utilizada nenhum tipo de metodologia e não existia nenhuma preocupação com a documentação.

Naquela época ainda existia mais um fator que se agregava a esses problemas: programar era uma tarefa muito mais complexa do que é hoje. O desenvolvimento e as correções eram mais trabalhosos e demandavam muito esforço e tempo, as linguagens utilizadas eram linguagens de baixo nível ou até mesmo linguagens de máquina.

Dessa forma surgiam inúmeros problemas com a qualidade do software, o tempo de desenvolvimento e o orçamento na maioria das vezes ultrapassavam aqueles que foram inicialmente estabelecidos. Para solucionar esses problemas surgiu a Engenharia de Software.

O objetivo da Engenharia de Software é fornecer métodos e ferramentas para desenvolver software de qualidade e a um baixo custo.

Segundo Fritz Bauer (1969), citado por Pressman (p.31, 2009) a Engenharia de Software é “a criação e a utilização de sólidos princípios de engenharia a fim de obter softwares econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais”.

De acordo com Falbo (p.1, 2013) “a Engenharia de Software trata de aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de software”.

A Engenharia de Software possibilita o desenvolvimento de software mais confiáveis, com melhor qualidade e a um menor custo. Para isso é necessário um planejamento das ações que serão executadas durante o desenvolvimento do produto, surgiu então o processo de desenvolvimento de software.

2.1.1 Processo de Desenvolvimento de Software

O processo de desenvolvimento de software é um conjunto de atividades que tem como objetivo obter um produto de alta qualidade, que atinjam as necessidades do usuário final, dentro do tempo determinado e a um custo estabelecido no início do processo.

De acordo com Sommerville (p. 42-43, 2010)

> um processo de software é um conjunto de atividades que leva à produção de um produto de software. [...] Os processos de software são complexos e, como todos os processos intelectuais e criativos, dependem de julgamento humano.

Na definição de um processo de desenvolvimento de software devem ser levados em consideração as atividades que serão realizadas, os recursos necessários, os artefatos requeridos, os procedimentos adotados e o modelo de processo escolhido.

Segundo Sommerville (p. 43, 2010) “um modelo de processo de software é uma representação abstrata de um processo de software.” Ele ainda afirma (p. 6, 2010) que “os modelos de processo incluem atividades, que fazem parte do processo de software, os produtos de software e os papéis das pessoas envolvidas na engenharia de software.”

De acordo com Pressman (2010) os modelos de processo foram propostos para organizar a desordem nas etapas de desenvolvimento do software. Esses modelos trazem uma estrutura para as etapas da Engenharia de Software, o que facilita o trabalho das equipes de desenvolvimento.

Existem diversos modelos de processos de software, porém de uma maneira geral o ciclo de vida de um software envolve as seguintes fases:

- **Levantamento de Requisitos.** Os desenvolvedores e usuários, em conjunto, estabelecem os requisitos do produto que se deseja desenvolver, esses requisitos normalmente consistem nos serviços que o produto deve oferecer, suas limitações e os seus objetivos.
- **Análise de Requisitos.** Depois de identificados os requisitos do sistema que será desenvolvido deverão ser modelados, avaliados e documentados.
- **Projeto.** Com base na análise de requisitos, são escolhidas as ferramentas, o tipo de aplicação e a estrutura de dados que serão utilizadas no desenvolvimento do software.
- **Implementação.** O software é codificado em uma linguagem que torna possível a compilação e geração do código-executável.
- **Testes.** Atividades de testes são executadas com o objetivo de validar o software.
- **Implantação.** Depois de testado ocorre a instalação do software no ambiente do usuário. Essa fase inclui o treinamento dos usuários para a utilização correta do sistema, a configuração do ambiente de produção e, quando necessário a conversão da base de dados.
- **Manutenção.** Esta fase consiste em fazer alterações necessárias depois que o software foi entregue ao usuário final.

2.1.2 Qualidade de Software

Houve um aumento pela procura por software de alta qualidade. Por esse motivo, empresas de desenvolvimento de software estão procurando aprimorar o

processo de desenvolvimento através de atividades de garantia de qualidade, para que com isso o usuário final tenha maior confiança no software produzido.

A qualidade é um dos aspectos mais importantes que deve ser levado em consideração no processo de desenvolvimento de software. Antes, porém, é necessário conceituar qualidade de software.

Para Bartié (2002) qualidade de software é um processo que focaliza todas as etapas com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos.

Rios e Moreira (2006) consideram um software com qualidade quando o número de defeitos encontrados no processo de teste é aceitável pelo desenvolvedor e pelo cliente, quando o software é entregue dentro do prazo e os custos no final do processo são os mesmos estabelecidos no inicio do processo.

Já Pressman (p. 724, 2009) define a qualidade de software como

> conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e as características implícitas que são esperadas de todo software profissional desenvolvido.

De acordo com Sommerville (2010) existe uma variedade de atributos de qualidade de software que devem ser considerados durante o processo de planejamento de qualidade. Uma parte importante desse planejamento é selecionar os principais atributos de qualidade para o software em desenvolvimento e planejar como eles podem ser obtidos. O Quadro 1 elenca alguns atributos de qualidade de software.

Quadro 1- Atributos de qualidade de software

Atributos de Qualidade de Software		
Segurança	Facilidade de compreensão	Portabilidade
Proteção	Facilidade de teste	Facilidade de uso
Confiabilidade	Adaptabilidade	Facilidade de reuso
Facilidade de recuperação	Modularidade	Eficiência
Robustez	Complexidade	Facilidade de aprendizado

Fonte: SOMERVILLE (2010).

A qualidade de software é uma combinação de fatores que variarão de acordo com as diferentes aplicações e clientes que as solicitam. Entre esses fatores

que contribuem para a construção de software de qualidade, Bartié (2002) cita profissionais experientes, ferramentas e metodologias adequadas, participação constante dos usuários finais, bom entendimento do problema, entre outros.

A garantia de qualidade de software (SQA) é um conjunto de atividades aplicadas durante todo o processo de desenvolvimento de software e tem como objetivo garantir que o processo de desenvolvimento e o produto final apresentem o nível de qualidade desejado.

De acordo com Pressman (2009) a atividade de SQA compreende as tarefas de aplicações de métodos técnicos, realizações de revisões técnicas formais, atividades de testes de software, aplicações de padrões, controle de mudanças, medição e manutenção de registros e reportagens.

Entre essas atividades destacam-se a de Verificação, Validação e Teste (VV&T). Conforme Falbo:

- Verificação: visa assegurar que o software, ou determinada função do mesmo, está sendo desenvolvido corretamente;
- Validação: visa garantir consistência, completitude e corretitude do produto em cada fase e entre fases consecutivas do ciclo de vida do software.
- Teste: examina o comportamento do produto por meio de sua execução.

2.2 Teste de Software

Devido necessidade de melhorar a qualidade dos softwares desenvolvidos, muitas empresas de desenvolvimento de software procuraram formas de aperfeiçoamento das técnicas de teste de software.

> As organizações estão buscando eficiência para conseguir sobreviver em um ambiente cada vez mais hostil – o de um mercado cada vez mais competitivo. As empresas estão buscando a tecnologia para reduzir custos e ampliar sua forma de atuação. Estão sofisticando seus sistemas para tomar decisões cada vez mais complexas, com a intenção de ganhar eficiência e controle (BARTIÉ, p. 4, 2002).

O teste de software é um indicador da qualidade do produto, mais do que um meio de detecção e correção de erros e grande importância no processo de desenvolvimento do software.

Os testes são executados com a intenção de descobrir erros, dessa forma pode-se concluir que, ao contrário do que possa parecer, um teste bem sucedido é aquele em que erros ainda não descobertos são revelados.

Rios e Moreira (p. 8, 2006) definem teste de software como “um processo que visa a sua execução de forma controlada com o objetivo de avaliar seu comportamento baseado no que foi especificado”.

2.2.1 Processo de Teste de Software

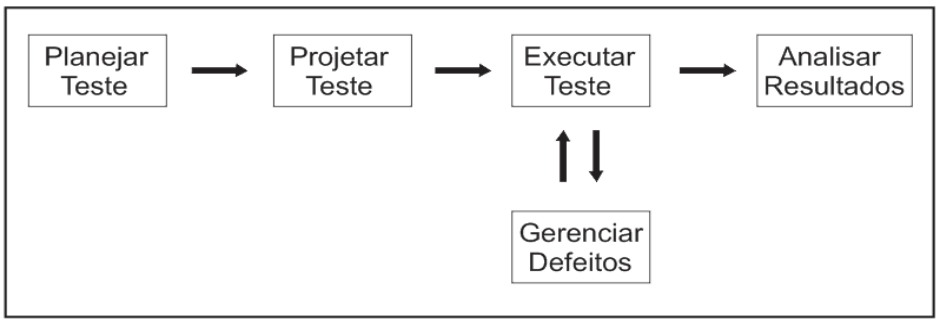
Segundo Rios e Moreira (p. 8, 2006) “o processo de teste deve basear-se em uma metodologia aderente ao processo de desenvolvimento, em pessoal técnico qualificado, em ambiente e ferramentas adequadas”.

De acordo Sommerville (2010), o processo de teste de software possui dois objetivos:

1. Demonstrar que o software atende aos requisitos que foram especificados no início do processo;
2. Descobrir falhas e defeitos no software onde seu comportamento não é o esperado.

A Figura 1 ilustra as atividades do processo de teste de software. Essas atividades são:

Figura 1 - Processo de desenvolvimento de software



Fonte: Rios (2003).

1. **Planejamento.** Nesta etapa é produzido um plano de teste. Ocorre a elaboração da proposta de testes baseada em prazos, custos e qualidade, dimensiona recursos e estabelece estimativas de acordo com as necessidades do cliente.
2. **Projeto de Casos de Teste.** Aqui ocorre a identificação dos casos de testes que serão construídos e modificados em função de mudanças que o cliente desejar.
3. **Execução dos Casos de Teste.** Ocorre a execução dos testes que foram planejados, garantindo que o comportamento do software continue em conformidade com os requisitos estabelecidos inicialmente.
4. **Análise dos Resultados Obtidos.** É realizada uma análise e confirmação dos resultados da fase de execução dos casos de teste.
5. **Gerenciar Defeitos ou Incidentes.** Nesta etapa são registradas e acompanhadas as correções dos defeitos encontrados na fase de execução dos casos de testes.

2.2.2 Metodologias de Teste de Software

Existem duas maneiras de construírem testes:

1. **Método de Caixa Preta.** Verifica-se o funcionamento do software por meio de suas interfaces, o que permite, na maioria das vezes, verificar se todas as funções são operacionais.
2. **Método de Caixa Branca.** Realiza-se um minucioso exame na estrutura interna e detalhes procedimentais, para verificar se existem ou não defeitos. Os caminhos lógicos são testados exaustivamente.

Pressman (2010) afirma que apesar de parecer lógico pensar que um bem efetuado teste de caixa branca seria o suficiente para oferecer 100% de softwares corretos, na prática isso não é o que ocorre, pois mesmo em programas pequenos o número de caminhos lógicos pode ser muito grande e testes exaustivos apresentam problemas lógicos.

Porém o teste de caixa branca não perde sua importância, pois um número pequeno de caminhos lógicos pode ser escolhido e executado e, então, estruturas de dados importantes podem ser testadas.

Dessa forma o teste de caixa preta não substitui o teste de caixa branca, eles são complementares, pois descobre erros que não seriam descobertos somente com o teste de caixa branca.

> Os atributos, tanto do teste de caixa branca como do de caixa preta, podem ser combinados para oferecer uma abordagem que valide a interface com o software e garanta seletivamente que o funcionamento interno do software esteja correto (PRESSMAN, p. 816, 2009).

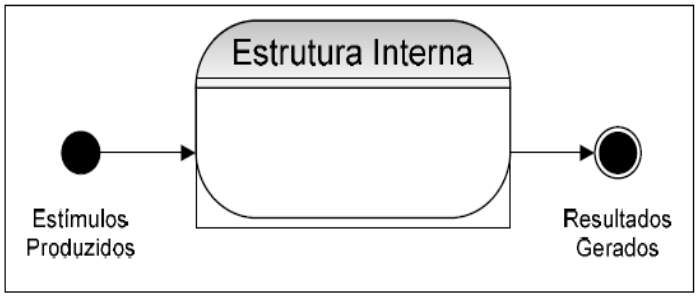
2.2.2.1 Teste de Caixa Preta

O teste de caixa preta, também conhecido como teste funcional, concentra-se nos requisitos funcionais do software. São utilizados para garantir que os requisitos estabelecidos sejam atendidos pelo software desenvolvido.

O teste funcional não está preocupado em verificar internamente o processamento do software, nem mesmo como ele foi desenvolvido, mas se o algoritmo aplicado no software produz os resultados esperados, analisando se este compreende os requisitos do sistema de forma adequada (BARTIÉ, 2002).

A Figura 2 ilustra a definição de teste de caixa preta. Nos testes de caixa preta são fornecidos dados de entradas (estímulos produzidos) e posteriormente faz a comparação entre os resultados gerados e os resultados esperados.

Figura 2 – Visão de teste de caixa preta



Fonte: Bartié (2002).

Rios e Moreira (p. 13, 2006) afirmam que o objetivo do teste de caixa preta é “verificar a funcionalidade e a aderência aos requisitos, em uma ótica externa ou do usuário, sem se basear em qualquer conhecimento do código e da lógica interna do componente testado”.

De acordo com Sommerville (2007) o teste de caixa preta envolve dois passos que são identificar as funções que o software deve realizar e criar casos de testes que checam se essas funções estão sendo realizadas pelo software.

Segundo Pressman (2009) os testes de caixa preta têm o objetivo de descobrir erros tais como:

1. Funções incorretas ou ausentes;
2. Erros de interfaces;
3. Erros nas estruturas de dados;
4. Erros de desempenho;
5. Erros de inicialização e término.

Os testes de caixa preta englobam as técnicas abaixo (PRESSMAN, 2009):

1. **Particionamento de Equivalência.** Divide as entradas do programa em classes de dados e os casos de testes podem, então, ser derivados dessa classe de dados.
2. **Análise de Valor limite.** Concentra nos casos de teste que colocam à prova os valores que estão nas fronteiras do domínio de entrada, onde erros ocorrem com maior frequência.
3. **Técnicas de Grafos de Causa-Efeito.** São levantadas possíveis condições de entrada (causas) e ações do programa (efeitos). Então essas causas e efeitos são relacionados em um grafo, esse grafo é convertido em uma tabela de decisão, de onde são derivados os casos de testes.
4. **Testes de Comparação.** É utilizada quando houver redundância de hardware e software. Essa redundância ocorre quando dois ou mais sistemas semelhantes trabalham simultaneamente. É realizada uma comparação entre as saídas que foram geradas nos sistemas para as mesmas entradas.

As vantagens do teste funcional são a alta capacidade de detectar erros, eles são mais simples de serem implementados e para sua realização não necessita que se tenha conhecimento da tecnologia que foi empregada. E apresentam como desvantagem o fato de depender de uma boa especificação de requisitos o que, muitas vezes não é bem feito.

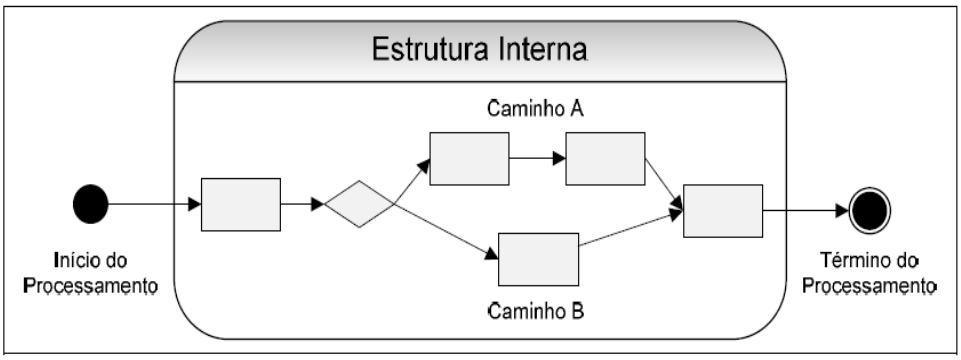
2.2.2.2 Teste de Caixa Branca

O teste de caixa branca, também conhecido como teste estrutural, analisa e avalia o comportamento interno do software, ou seja, o testador tem acesso ao código fonte e tem como objetivo garantir que a estrutura do software funcione corretamente.

O teste de caixa branca é baseado na arquitetura interna do software, são empregados métodos para identificar erros na estruturas internas dos programas através da simulação de situações que exercitem adequadamente todas as estruturas usadas na codificação (BARTIÉ, 2002).

Como mostra a Figura 3 o teste de caixa branca deve observar os possíveis caminhos na estrutura interna do software e verificá-los.

Figura 3 – Visão de teste de caixa branca



Fonte: Bartié (2002).

Segundo Pressman (2009) usando métodos de teste de caixa branca, pode-se criar casos de testes com os propósitos elencados abaixo:

1. Garantir que os caminhos independentes dentro de um módulo tenham sido exercitados ao menos uma vez;

2. Exercitar todas as decisões lógicas para valores falso ou verdadeiro;
3. Executar todos os laços em suas fronteiras e dentro de seus limites;
4. Exercitar as estruturas de dados internas para garantir a sua validade.

Os testes de caixa branca englobam as técnicas citadas abaixo:

1. **Teste de Caminho Básico.** São derivados casos de testes com o objetivo de executar cada instrução ao menos uma vez;
2. **Teste de Condição.** Coloca à prova as condições lógicas no código fonte do programa;
3. **Teste de Fluxo de Dados.** Seleciona caminho de testes de acordo com o uso de variáveis;
4. **Teste de Laços (*loop*).** Concentra-se exclusivamente na validade de construção de laços.

As vantagens dos testes estruturais são a alta eficiência na detecção de erros e a característica de que podem ser modelados pelo próprio desenvolvedor e possuem as desvantagens de serem difíceis de implementar, os resultados podem ser demorados, concentram-se apenas na lógica interna e não verifica a conformidade da lógica com a especificação.

2.2.3 Fases de Teste de Software

Quando o tamanho e a complexidade do processo de teste de software aumentam é necessário a divisão das atividades de testes em fases, podendo, dessa forma, focar em diversos tipos de erros. As próximas seções apresentam essas fases.

2.2.3.1 Teste Unitário

Os testes unitários, também conhecidos como teste de unidade, são aplicados na menor unidade do software (que são chamados de módulos).

Esse tipo de teste tem o objetivo de verificar cada módulo que compõe o software desenvolvido isoladamente, para assegurar que cada um desses módulos realiza a função especificada.

Rios e Moreira (p. 13, 2006) afirmam que “os testes unitários verificam o funcionamento de um pedaço do sistema ou software isoladamente ou que possam ser testados separadamente, podendo, inclusive, ser um programa ou um componente”.

Segundo Pressman (2009) o teste de unidade utiliza a descrição detalhada do projeto como um guia, caminhos de controle são testados para descobrirem erros dentro do módulo testado. Ele ainda afirma que o teste de unidade baseia-se no método de caixa branca e pode ser realizado em paralelo com outros módulos.

2.2.3.2 Teste de Integração

O fato de terem sido realizados exaustivos testes unitários, não significa que não poderão ocorrer erros quando os módulos testados forem colocados para funcionar em conjunto.

O teste de integração tem como objetivo descobrir erros associados às interfaces que poderão surgir em decorrência da integração desses módulos.

> O teste de integração é uma técnica sistemática para a construção da estrutura de programa, realizando-se, ao mesmo tempo, testes para descobrir erros associados a interfaces. O objetivo é, a partir dos módulos testados no nível de unidade, construir a estrutura de programa que foi determinada pelo projeto (PRESSMAN, p.849, 2009).

Nesse tipo de testes as abordagens mais utilizadas são:

1. ***Big Bang*.** É um processo não incremental, onde todos os componentes são integrados de uma só vez e o programa é testado como um todo. Esse tipo de abordagem deve ser evitada: a correção dos erros é uma tarefa complexa, pois fica muito difícil isolar a causa do erro.

2. **Bottom-up (integração ascendente).** É uma estratégia de integração incremental. Os componentes de níveis mais básicos da arquitetura são testados em conjunto por controladores, que foram criados para simular as interações dinâmicas e suas interfaces. À medida que se desenvolve a interação, os componentes substituem esses controladores e novos são criados para simular as interações e as interfaces um nível maior da arquitetura. Esse processo ocorre até que não existam níveis a serem alcançados, ou seja, até que seja atingido o nível máximo de interação (BARTIÉ, 2002)
3. ***Top-down* (integração descendente).** É uma abordagem incremental, em que as unidades de maior nível hierárquico são criadas, essas unidades sofrem um processo de refinamento e decomposição sucessivos, até que seja alcançado o menor nível estrutural do projeto e que todas as unidades tenham sido criadas. A cada vez que uma unidade é criada e alterada testes de integração avaliam se as interfaces com as outras unidades continuam compatíveis (BARTIÉ, 2002).

A escolha de um método de integração depende das características do software e do cronograma do projeto. Uma forma ideal seria o *teste sanduíche*, uma abordagem combinada, onde é usado o método *top-down* para os níveis superiores da estrutura do programa e o método *bottom-up* para os níveis subordinados (PRESSMAN, 2009).

2.2.3.3 Teste de Validação

O objetivo do teste de validação (ou teste de aceitação) é verificar se o software, como um todo, cumpre a função para a qual ele foi especificado.

Teste de aceitação “são os testes finais de execução do sistema, realizados pelos usuários, visando verificar se a solução atende aos objetivos do negócio e a seus requisitos, no que diz respeito à funcionalidade e usabilidade, antes da utilização no ambiente de produção” (Rios; Moreira, p.15, 2006).

Esse teste é realizado pelo cliente através de um conjunto de testes de caixa preta com o objetivo de determinar se o software está em conformidade com os requisitos que foram estabelecidos inicialmente. Apesar de o teste de aceitação ser realizado pelo cliente, as equipes de testes e do projeto dão o suporte necessário para que esse teste seja conduzido.

Existem duas técnicas utilizadas nos teste de validação, são elas:

1. **Teste Alfa.** É conduzido no ambiente do desenvolvedor do produto com os usuários finais. O software é testado pelo usuário final e o desenvolvedor, que está presente, registra os erros e problemas ocorridos e realiza os reparos necessários.
2. **Teste Beta.** É realizado no ambiente do usuário final e o desenvolvedor normalmente não está presente. O usuário registra todos os erros e problemas encontrados durante o teste e informa ao desenvolvedor, que realiza as alterações necessárias.

2.2.3.4 Teste de Sistema

O teste de sistema é uma composição de diferentes testes e tem como finalidade testar por completo o sistema baseado em computador. Cada um tem uma finalidade distinta, mas juntos estão voltados para validar a exatidão na execução das funções do sistema (PRESSMAN, 2009).

Planejar os testes de sistema não é fácil, exige um perfeito entendimento dos requisitos não funcionais do software desenvolvido, o que muitas vezes não são claramente descritos ou representam (BARTIÉ, 2002).

“Nesse estágio de testes deve ser simulada a operação normal do sistema, sendo testadas todas as suas funções de forma mais próxima possível do que irá ocorrer no ambiente de produção” (RIOS; MOREIRA, p. 14, 2006).

Os testes que compõem o teste de sistema são:

1. **Teste de Recuperação.** Nesse tipo de teste o software é forçado a falhar com o objetivo de verificar se a recuperação é executa de forma eficiente.

Para Rios e Moreira (p. 16, 2006) os testes de recuperação "validam a capacidade e qualidade de recuperação do software após *crashes*, falhas de hardware ou outros problemas catastróficos".

2. **Teste de Segurança.** Nesse tipo de teste verifica se os mecanismos de proteção que foram projetados protegerão o sistema de acessos não permitidos.

   Os testes de segurança "validam a capacidade de proteção do software contra acesso interno ou externo não autorizado" (RIOS; MOREIRA, p. 16, 2006).

3. **Teste de Estresse.** Nesse tipo de teste o software é submetido a situações anormais.

   O sistema é executado de forma que exija recursos em quantidade, frequência e/ou volumes anormais, é então avaliado o seu comportamento e verificado até quando pode ser exigido e quais são as falhas decorrentes do teste (PRESSMAN, 2009).

4. **Teste de Desempenho.** Também chamado de teste de performance, verifica o desempenho e o tempo de resposta do software.

   "Visam a garantir que o sistema atende aos níveis de desempenho e tempo de resposta acordados com os usuários e definidos nos requisitos" (RIOS; MOREIRA, p.16, 2006).

2.2.4 Depuração

Muitas vezes teste de software é confundido com a depuração, porém são conceitos diferentes.

Depuração (*debugging*) é o processo de analisar e localizar *bugs* de software quando não se comporta da forma esperada.

Ocorre em consequência de testes bem-sucedidos. Quando é revelado um erro, a depuração é o processo que resulta na remoção do erro (PRESSMAN, 2009).

Portanto, a depuração é uma atividade de apoio aos testes de software, mas não pode substituí-los.

2.3 Jogos Educacionais

Um jogo é caracterizado como educacional quando é elaborado com o objetivo de auxiliar no ensino-aprendizagem.

Wangenheim (2012) define jogo educacional como uma competição entre jogadores que agem sob regras com o objetivo de vencer e que foi projetado especificamente para o ensino de um determinado assunto.

> O jogo pode ser considerado como um importante meio educacional, pois propicia um desenvolvimento integral e dinâmico nas áreas cognitiva, afetiva, linguística, social, moral e motora, além de contribuir para a construção da autonomia, criticidade, criatividade, responsabilidade e cooperação das crianças e adolescentes (MORATORI, p.9, 2003).

Para Michael e Chen (2005), citados por Silva (2010), os jogos educacionais referem-se aos jogos ligados a educação e treinamento. Eles afirmam que “para um jogo ser considerado educacional, este deve conter, ainda em seu projeto, objetivos educacionais explicitamente definidos, sem o propósito de ser jogado por simples prazer” (SILVA, p.41, 2005).

Um dos principais motivos para a utilização dos jogos educacionais é o seu fator motivacional, tornando o processo de ensino-aprendizagem mais atrativo, permitindo, dessa forma uma melhor assimilação do assunto através da prática.

De acordo com Gibson, Aldrich e Prensky (2007), citado por Silveira (2012), existem quatro características que contribuem para aumentar a motivação de aprendizagem. Essas características são: desafio, fantasia, curiosidade e controle.

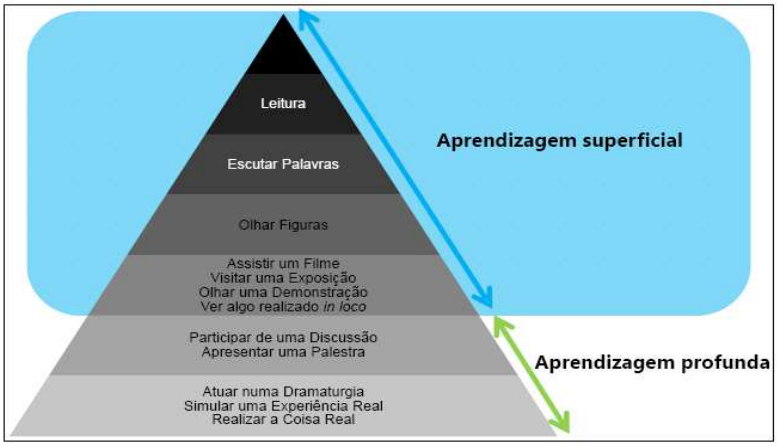
> O desafio mantém os estudantes comprometidos com a atividade, a fantasia aumenta o entusiasmo dos estudantes providenciando um cenário imaginário e atraente para os mesmos, a curiosidade despertada pelo novo cenário imaginário e atraente para os mesmos, a curiosidade despertada pelo novo cenário estimula os estudantes a explorar o desconhecido, o controle fornece a sensação de alto-determinação aos aprendizes (SILVEIRA, p.54, 2012).

Conforme Wangenheim (2012), em sua palestra “Como ensinar com jogos?”, em aulas expositivas os alunos perdem a concentração e o interesse depois de certo tempo. Ela afirma que os métodos utilizados nesse tipo de aula provocam apenas

uma aprendizagem superficial. Para alcançar uma aprendizagem mais profunda é necessária a utilização de outros métodos que estimulem o aluno.

A Figura 4 ilustra os níveis de aprendizagem. Para Wangenheim (2012), as atividades de leitura, olhar figuras, visitar uma exposição, assistir um filme, entre outras atividades, só provoca uma aprendizagem superficial. Para uma aprendizagem profunda os alunos e professores deverão, por exemplo, simular uma experiência real, realizar discussões e palestras.

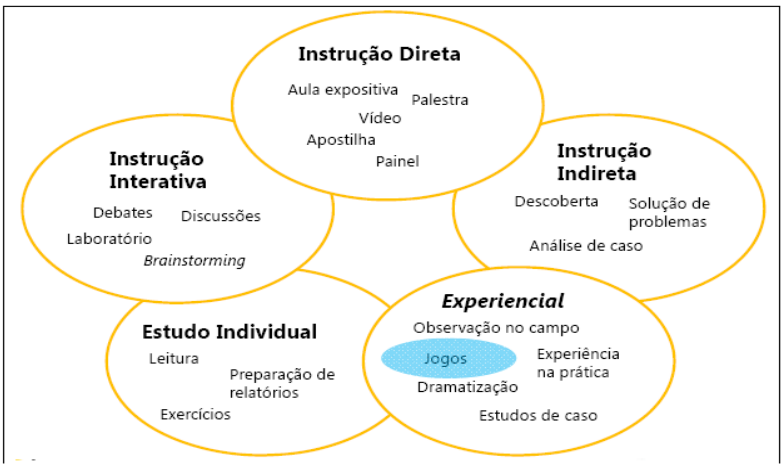
Figura 4 - Níveis de aprendizagem



Fonte: Wangenheim (2012).

Wangenheim (2012) ainda afirma que para que se obter uma aprendizagem mais profunda é necessário utilizar outras estratégicas instrucionais. A Figura 5 apresenta quais são as estratégias utilizadas e quais as atividades empregadas em cada tipo de estratégia instrucional.

Figura 5 - Estratégias instrucionais



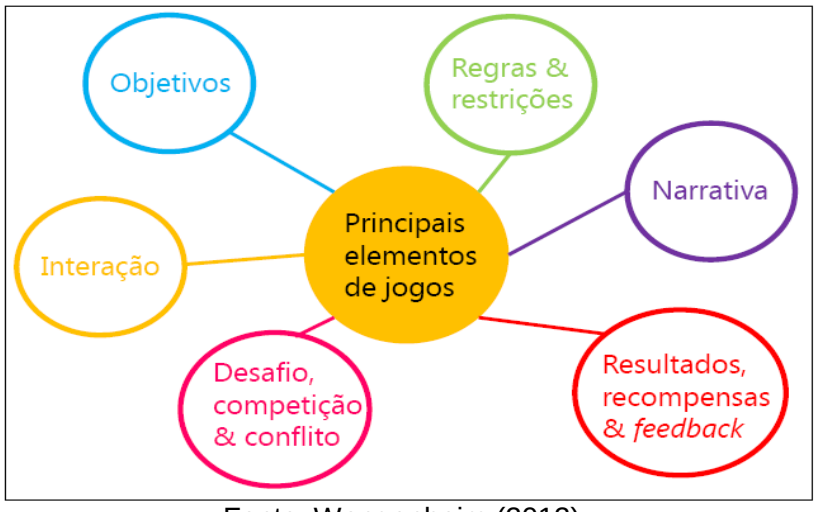
Fonte: Wangenheim (2012).

Wangenheim (2012) afirma que os jogos são definidos por vários elementos principais que trazem diversas vantagens quando o jogo é utilizado como um meio de ensino-aprendizagem.

Como mostrado na Figura 6 esses elementos são:

- **Objetivos.** Vencer, ser o melhor entre os competidores, alcançar um maior nível de aprendizagem;
- **Regras.** Definem o que pode ou não pode ser feito pelo jogador;
- **Restrições.** Podem ser ganhadas ou perdidas (como por exemplo dinheiro ou vidas);
- **Narrativa.** Fornece um fundo de ficção e motivam as ações, porém também existem jogos que não possuem nenhum tipo de narrativa.
- **Interação.** Um dos elementos mais importantes do jogo;
- **Desafio, competição e conflito.** O desafio previne que o jogo não se torne monótono, a competição é mais um elemento motivador;
- ***Feedback.*** Indica para o aluno onde ele acertou e errou no jogo.

Figura 6 - Principais elementos de um jogo



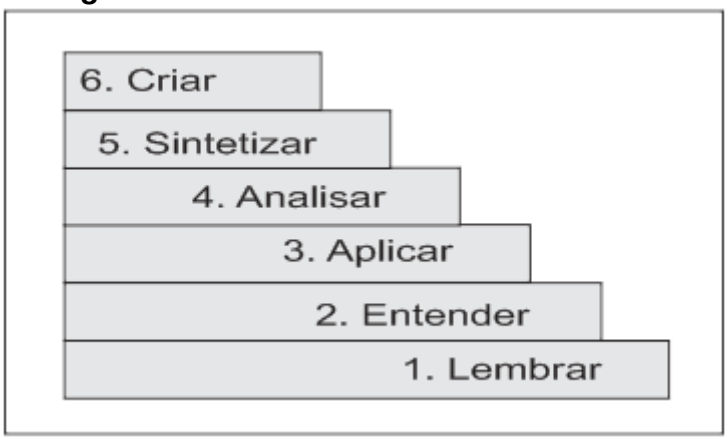
Fonte: Wangenheim (2012).

O nível de aprendizagem de um jogo educacional pode ser avaliado de acordo com a Taxonomia Revisada de Bloom.

A Taxonomia Revisada de Bloom é uma versão atualizada da Taxonomia de Bloom realizada por Dr. Lorin Anderson, em 2001.

De acordo com Silva F. (2013) a versão de Anderson possui seis capacitações. A Figura 7 ilustra essas capacitações:

Figura 7 – Taxonomia revisada de Bloom



Fonte: adaptado de Ferraz (2010).

- **Lembrar.** Reconhecer e recordar informações importantes da memória de longa duração.
- **Entender.** Capacidade de fazer sua própria interpretação do material educacional, como leituras e explicações do professor.
- **Aplicar.** Refere-se a usar o procedimento aprendido em uma situação familiar ou nova.
- **Analisar.** Dividir o conhecimento em partes e pensar como essas partes se relacionam com a estrutura geral.
- **Avaliar.** Engloba verificação e crítica.
- **Criar.** Não fazia parte da primeira taxonomia, é o principal componente da nova versão. Essa capacitação envolve reunir elementos para dar origem a algo novo.

O jogo contribui para o aprendizado quando comparadas a outras estratégias instrucionais de ensino. Porém pode existir uma série de desvantagens quando é utilizado esse tipo de estratégia instrucional.

De acordo com Grando (2001), a inserção de jogos no contexto de ensino-aprendizagem apresentam vantagens e desvantagens, que são listadas no Quadro 2:

Quadro 2 - Vantagens e desvantagens dos jogos

Vantagens	Desvantagem
▪ **Fixação de conceitos** já aprendidos de uma forma motivadora para os alunos; ▪ **Introdução e desenvolvimento de conceitos** de difícil compreensão; ▪ Desenvolvimento de **estratégias de resolução de problemas** (desafios dos jogos); ▪ Aprender a **tomar decisão** e saber **avaliá-las**; ▪ **Significação** para conceitos aparentemente incompreensíveis; ▪ Propicia o relacionamento das diferentes disciplinas (**interdisciplinaridade**); ▪ O jogo requer a **participação ativa do aluno** na construção do seu próprio **conhecimento;** ▪ O jogo favorece a **socialização** entre os alunos e a conscientização do **trabalho em equipe**; ▪ A utilização dos jogos é um fator de **motivação** para os alunos; ▪ Dentre outras coisas, o jogo favorece 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**; ▪ As atividades com jogos podem ser utilizados para reforçar ou recuperar habilidades de que os alunos necessitem. Útil no trabalho com alunos de diferentes níveis; ▪ As atividades com jogo permitem ao professor identificar, diagnosticar alguns erros de aprendizagem, as atitudes e as dificuldades dos alunos;	▪ Quando os jogos são mal utilizados, existe o perigo de dar ao jogo um **caráter puramente aleatório**, tornando-se um **"apêndice" em sala de aula**. Aos alunos jogam e se sentem motivados apenas pelo jogo sem saber porque jogam; ▪ **O tempo gasto** com as atividades de jogo em sala de aula **é maior**, se o professor não estiver preparado, pode existir um sacrifício de outros conteúdos pela falta de tempo; ▪ As **falsas concepções** de que devem **ensinar todos os conceitos através dos jogos**. Então, as aulas, em geral, transformam-se em verdadeiros cassinos também sem sentido algum para o aluno; ▪ A **perda de "ludicidade"** do jogo pela interferência constante do professor destruindo a essência do jogo; ▪ A coerção do professor, exigindo que o aluno jogue, mesmo que ele não queira, destruindo a voluntariedade pertencente a natureza do jogo; ▪ A dificuldade de acesso e disponibilidade de materiais e recursos sobre o uso de jogos no ensino, que possam vir a subsidiar o trabalho docente.

Fonte: Granado (2001)

3 DESENVOLVIMENTO DO JOGO AS AVENTURAS DE *JACK TEST*

3.1 Projeto de Jogo

Este trabalho propõe o desenvolvimento do jogo educacional “As aventuras de *Jack Test*”, que tem como função auxiliar no ensino-aprendizagem dos principais conceitos envolvidos em Teste de Software.

Com o objetivo de contribuir para a construção de uma boa documentação do projeto de jogo foi utilizado o *game design document* (GDD).

De acordo com Kreimeier (2012), citado por Motta e Trigueiro (2013), o *game design document* é um documento que descreve diversos elementos e características de um jogo com o intuito de comunicar e guiar os envolvidos no desenvolvimento do jogo.

Diniz (2011) cita Brathwaite e Schreiber quando afirma que o *game design* é o “processo de criação do conteúdo e das regras de um jogo” (Brathwaite e Schreiber, p. 2, 2009).

Para colaborar com a definição e construção do jogo foi elaborado o *design* instrucional e o *design* de jogo, que são apresentados nas seções 5.1.1 e 5.1.2, respectivamente.

3.1.1 *Design* Instrucional

No *design* ou projeto instrucional é definido o público-alvo, o conhecimento prévio necessário, os objetivos instrucionais, os artefatos de entrada e os itens de avaliação do desempenho do aluno.

3.1.1.1 Público-alvo

O público-alvo do jogo são os alunos do curso de graduação em Ciência da Computação ou Sistema de Informação, que cursam a disciplina de Engenharia de Software.

3.1.1.2 Conhecimentos Necessário

Para que possa jogar o aluno deverá ter uma base sobre o conteúdo de Teste de Software. É necessário que o jogador saiba alguns conceitos básicos sobre os testes de unidade, integração, validação e sistema, além de depuração, saiba também definir os termos verificação, validação e teste, diferenciar testes de caixa preta e testes de caixa branca e identificar as etapas do processo de teste de software.

3.1.1.3 Objetivos Instrucionais

Deseja-se que ao término do jogo o aluno tenha reforçado os conceitos sobre teste de software que foram apresentados em sala de aula e que possa diferenciar as fases de testes.

3.1.1.4 Artefatos de Entrada

O aluno deverá ler as regras e as informações sobre o jogo como pré-requisito para jogar.

O jogo é baseado nas seguintes regras:

1. O jogo é formado por oito etapas: quatro fases e quatro desafios;
2. O mapa do jogo representa a Ilha dos Desafios, para acessar uma etapa específica basta clicar no botão vermelho próximo a Jack;

3. O jogador terá três minutos para finalizar cada etapa, se no final deste tempo não conseguir solucionar o problema corretamente perderá o jogo;
4. No decorrer desses três minutos o jogador terá três tentativas para solucionar a etapa;
5. O jogador iniciará cada etapa com a pontuação máxima de 30 pontos. Quanto menor for o tempo em que o jogador solucionar os problemas, maior será sua pontuação;
6. Durante cada etapa o jogador terá o auxilio de uma dica, porém quando usada terá um acréscimo de vinte segundos no tempo decorrido.

3.1.1.5 Avaliação de Desempenho

Os itens de avaliação do desempenho do aluno são o tempo decorrido para o jogador solucionar os problemas propostos e, consequentemente, a pontuação do jogador em cada etapa.

3.1.2 *Design* do Jogo

No design do jogo é apresentado o conceito do jogo, seguindo com a narrativa do jogo e finalizando com a descrição da mecânica do jogo (jogabilidade).

3.1.2.1 Conceito do Jogo

Para criar o conceito do jogo é necessário fazer a descrição do jogo, definir o gênero e a plataforma em que o jogo será executado.

"As aventuras de *Jack Test*" é um jogo educacional, onde o jogador deverá assumir o papel de um engenheiro de teste.

O jogo desenvolvido é constituído por fases e desafios intercalados. Nas fases o jogador terá que responder questões que abordam conceitos das fases de Testes de Software. Já nos desafios são apresentados problemas, com um maior grau de dificuldade, que o jogador deverá solucionar, também relacionados à área de Teste de Software.

"As Aventuras de *Jack Test*" é um jogo de aventura e estratégia. É apresentado ao jogador um mundo inexplorado onde problemas e desafios devem ser solucionados.

O jogo foi desenvolvido para a plataforma *web* e não necessitará de nenhuma pré-configuração do computador.

3.1.2.2 Narrativa

O jogo é apresentado de forma lúdica. Nele os personagens *Jack Test*, um engenheiro de teste, e *James Dep*, piloto de avião, sofrem um acidente de avião e ficam perdidos em uma ilha cheia de mistérios e desafios, a "Ilha dos Desafios". Para sair dessa ilha *Jack Test* construirá um barco e deverá realizar testes para verificar o seu correto funcionamento. O jogador deverá assumir o papel de *Jack Test* e terá que verificar os testes realizados no barco e enfrentar os desafios propostos.

3.1.2.3 Mecânica do Jogo

"As aventuras de *Jack Test*" é um jogo *single-play*, onde o jogador deverá assumir o papel de Jack, um engenheiro de teste, que sofre um acidente de avião durante uma viagem e terá que realizar testes no barco construído e solucionar os desafios propostos para que possa avançar no jogo.

Ao entrar no jogo o jogador se depara com a tela inicial, como mostra a Figura 8. Nessa tela o jogador terá três opções de acesso:

- *Regras*: onde o jogador terá acesso às regras do jogo;
- *Sobre o Jogo*: o jogador obterá informação sobre o jogo;

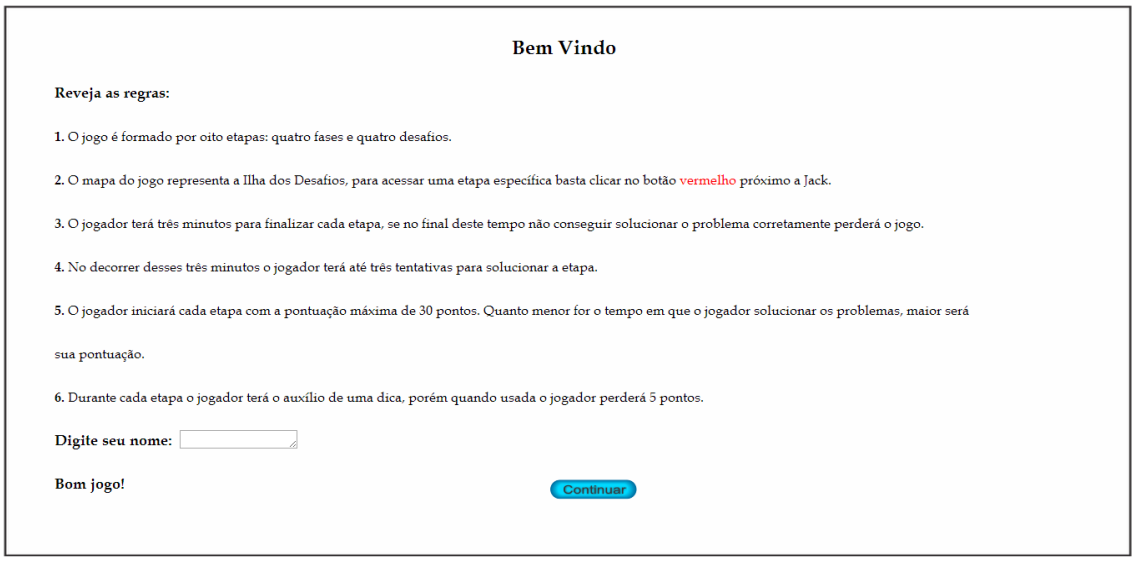
- *Iniciar*: o jogo iniciará.

Figura 8- Tela inicial do jogo

Fonte: O autor.

Ao clicar em *Iniciar* o jogador será encaminhado para a tela de boas vindas, onde terá acesso novamente às regras do jogo e depois que indicar seu nome o poderá iniciar o jogo, como mostra a Figura 9.

Figura 9- Tela de Boas vindas do jogo



Fonte: O autor.

O jogador terá acesso às etapas do jogo através do mapa da "Ilha dos Desafios", que é mostrado na Figura 10.

Figura 10 - Tela do mapa do jogo

Fonte: O autor.

3.2 Etapas do Jogo

Como já foi dito o jogo “As aventuras de *Jack Test*” foi elaborado em oito etapas, onde cada etapa está relacionada a um conteúdo de Teste de Software.

Os Quadros 3 e 4 elencam os conteúdos que são abordados em cada fase e desafio, respectivamente, e apresentam uma descrição do que deverá ser realizado para que o jogador avance no jogo.

Quadro 3 – Fases do jogo

Fase	Conteúdo abordado	Descrição da fase
1	Teste unitário	O jogador deve escolher as alternativas corretas referentes ao Teste unitário.
2	Teste de integração	O jogador deve escolher as alternativas corretas referentes ao Teste de integração.
3	Teste de validação	O jogador deve marcar qual o teste que será realizado durante a situação descrita.
4	Teste de sistema	O jogador deve marcar qual o teste que será realizado durante a situação descrita.

Fonte: O autor.

Quadro 4 – Desafios do jogo

Fase	Conteúdo abordado	Descrição da fase
1	Garantia de Qualidade de Software	O jogador deve completar as frases com palavras que são conceitos da Garantia de Qualidade de Software.
2	Métodos de Teste de Software	O jogador deve classificar as características apresentadas em Testes de Caixa Preta e Caixa Branca.
3	Processo de Teste de Software	O jogador deve ordenar as fases do Processo de Teste de Software.
4	Depuração	O jogador deve desvendar qual é a palavra secreta.

Fonte: O autor.

As telas das fases e dos desafios seguem o mesmo padrão. No lado esquerdo da tela é apresentado ao jogador informações sobre o tempo decorrido, o número de tentativa e a sua pontuação na etapa. O jogador poderá ter acesso ao botão *Dica*, localizado no canto superior da tela, que quando clicado dará uma dica ao jogador. Existem também os botões *Corrigir* e *Tentar novamente*, que o jogador deverá clicar para verificar se sua resposta está correta e, caso não esteja, poderá tentar novamente caso sua resposta esteja incorreta.

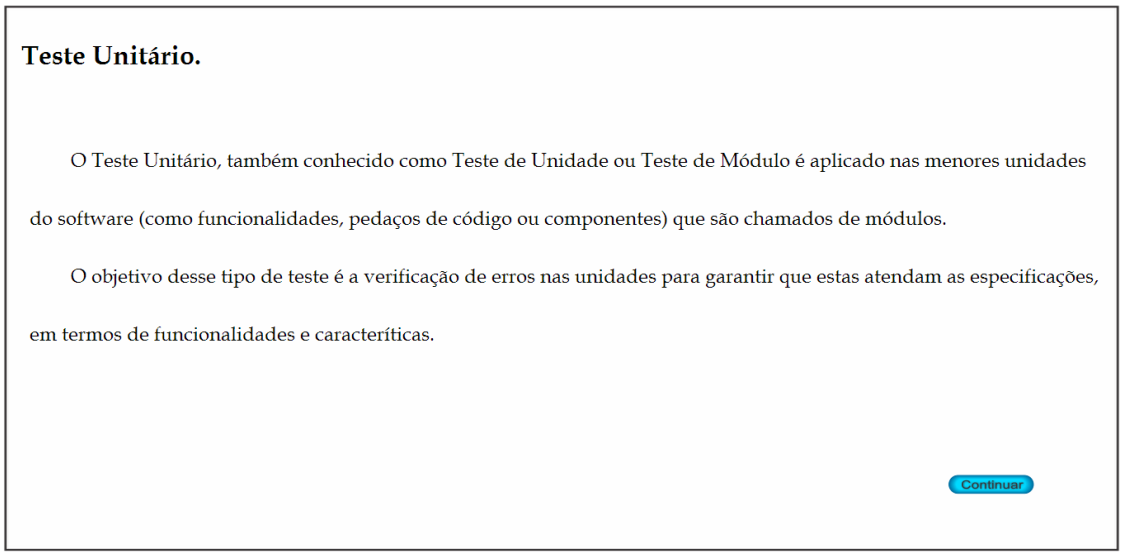
Conforme as ações do jogador mensagens de feedback aparecerão na tela. Essas mensagens são:

- *Resposta correta!*: caso a resposta do jogador esteja correta;
- *Resposta incorreta! Tente novamente!*: caso a resposta do esteja incorreta;
- *Acabaram suas tentativas!*: se o número de tentativas ultrapassou 3;
- *Seu tempo acabou!*: se o tempo ultrapassou os 3 minutos.

As fases e os desafios são descritos detalhadamente nas duas próximas seções.

No final de cada etapa jogador tem acesso a uma rápida explicação sobre o assunto referenciado. A Figura 11 mostra uma breve explicação sobre Teste de integração, apresentada após o jogador responder corretamente à segunda fase.

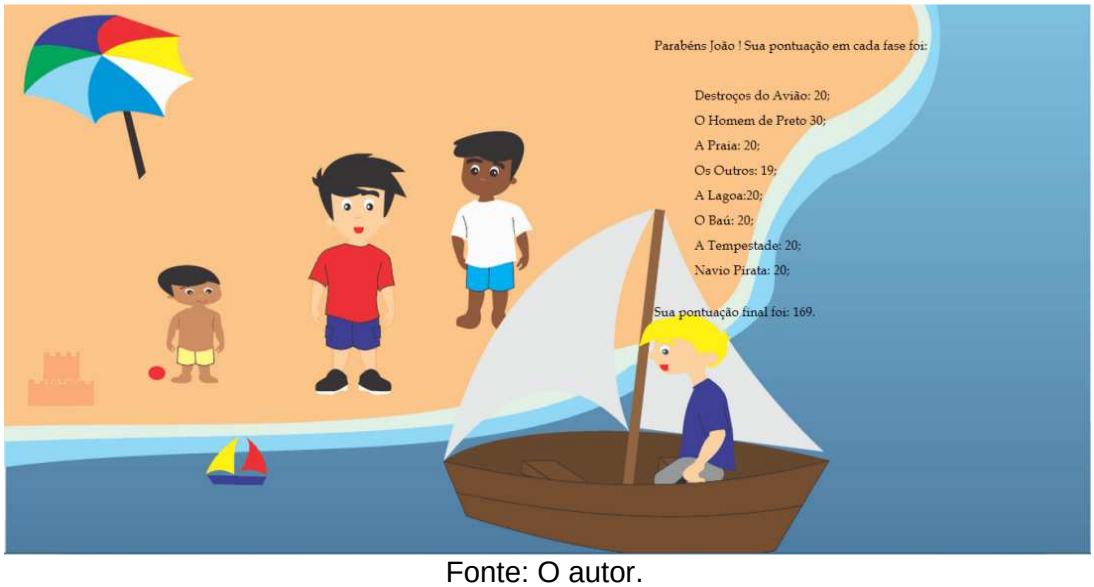
Figura 11 - Tela de explicação sobre teste de unitário



Fonte: O autor.

Existem duas possibilidades para o final do jogo. A primeira é o jogador ter respondido corretamente todas as oito etapas do jogo e, então, ele terá acesso a sua pontuação em cada etapa e sua pontuação final. A Figura 12 demonstra como é apresentado esse *feedback* para o jogador.

Figura 12 - Tela final: Pontuação



Fonte: O autor.

Na segunda possibilidade, Figura 13, o jogo é finalizado antes de o jogador ter passado por todas as etapas, isso ocorre quando o ele não consegue solucionar o problema durante os três minutos ou utilizou as três tentativas.

Figura 13 – Tela final: Game Over

Fonte: O autor.

3.2.1 Fases

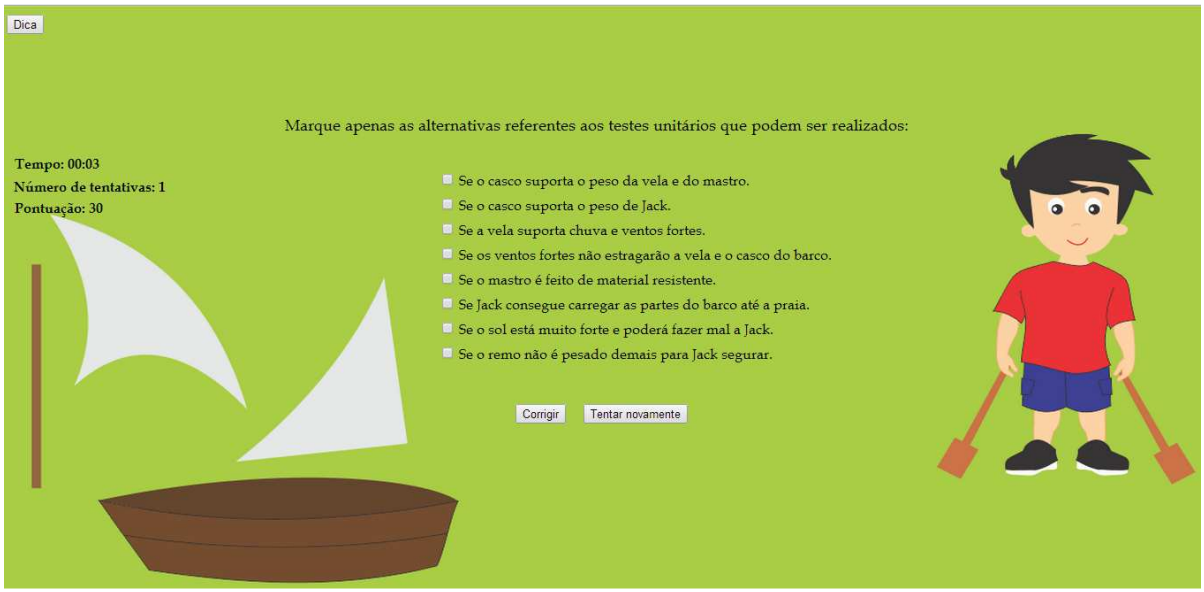
As fases do jogo foram elaboradas tendo como base os conceitos sobre as fases de Testes de Software. Nelas são apresentadas questões sobre o conteúdo abordado e o jogador deverá escolher entre as opções listadas a(s) alternativa(s) correta(s) relacionada(s).

As Figuras 14 e 15 representam a primeira fase do jogo, *Destroços do Avião*. Na figura 14 é apresentado ao enredo desta fase: *Jack* vai até os destroços do avião e encontra material para construir o barco. Depois de prontas, *Jack* decide testar as partes construídas antes de montar o barco. A Figura 15 mostra a fase em si, o jogador deverá selecionar entre as opções apresentadas quais são os testes unitários que deverão ser realizados nas partes construídas.

Figura 14 - Tela inicial da primeira fase

Fonte: O autor.

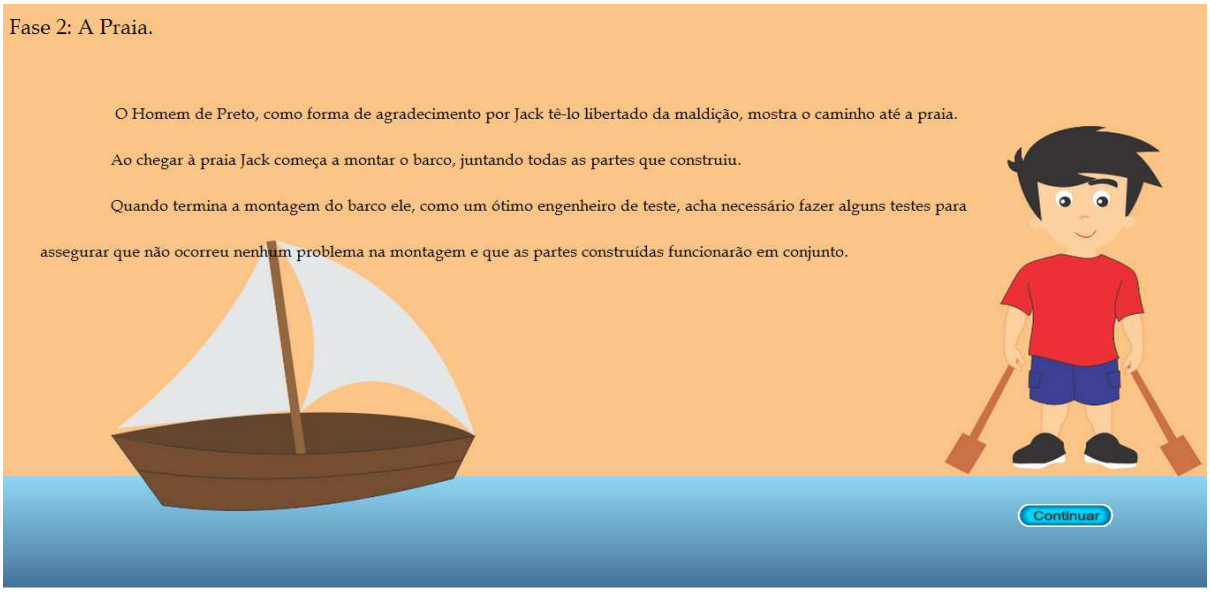
Figura 15 – Tela da primeira fase



Fonte: O autor.

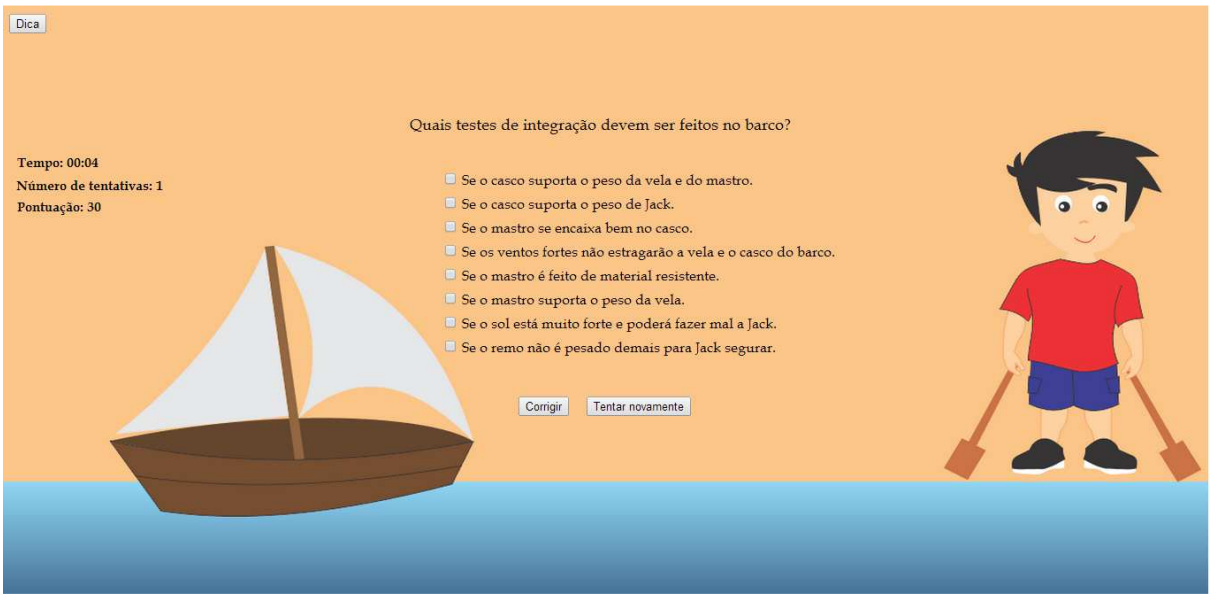
As Figuras 16 e 17 ilustram *A Praia*, a segunda fase do jogo. A Figura 16 mostra o contexto da fase: ao chegar à praia *Jack* monta o barco e começa a realizar teste no barco para verificar seu funcionamento quando as partes construídas são colocadas em conjunto. Já a Figura 17 apresenta a fase, onde o jogador deverá selecionar as alternativas corretas relacionada aos testes de integração que serão realizados no barco.

Figura 16 - Tela inicial da segunda Fase



Fonte: O autor.

Figura 17 – Tela da segunda fase



Fonte: O autor.

A terceira fase, *A Lagoa*, é ilustrada nas Figuras 18 e 19. Na Figura 18 está uma parte do contexto da fase: após *Jack* se machucar é *James* quem constrói o barco e *Jack* assumirá o papel de usuário final e testará o funcionamento do barco. A Figura 19 demonstra a tela que o jogado tem algumas alternativas de testes e deverá escolher qual teste *Jack* irá realizar.

Figura 18 - Tela inicial da terceira fase

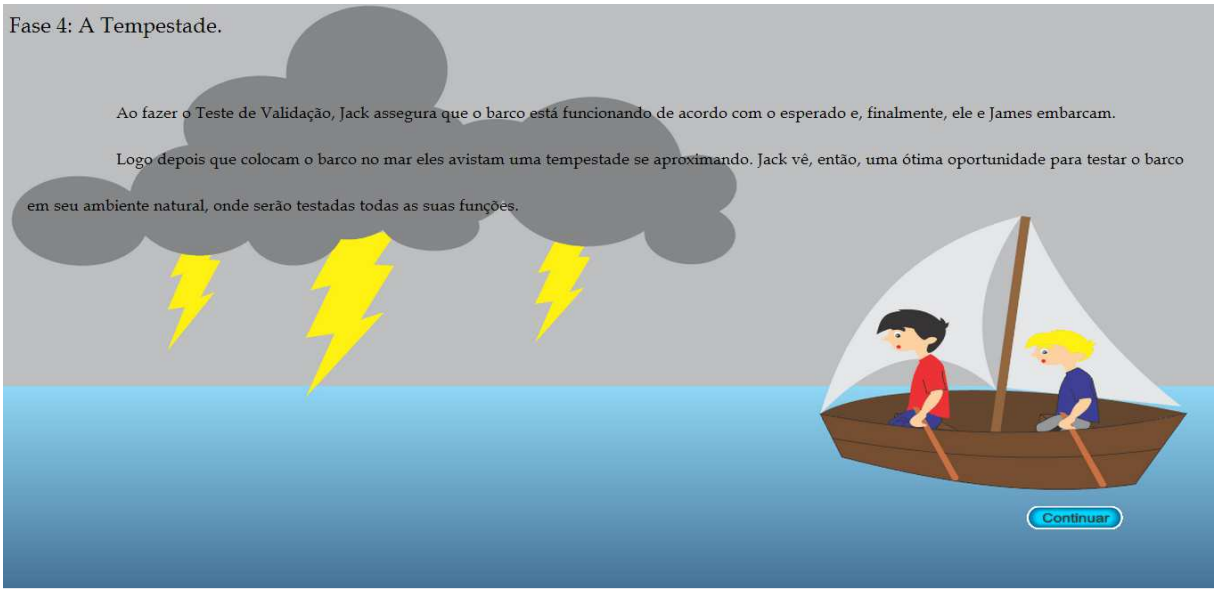
Fonte: O autor.

Figura 19 - Tela da terceira fase

Fonte: O autor.

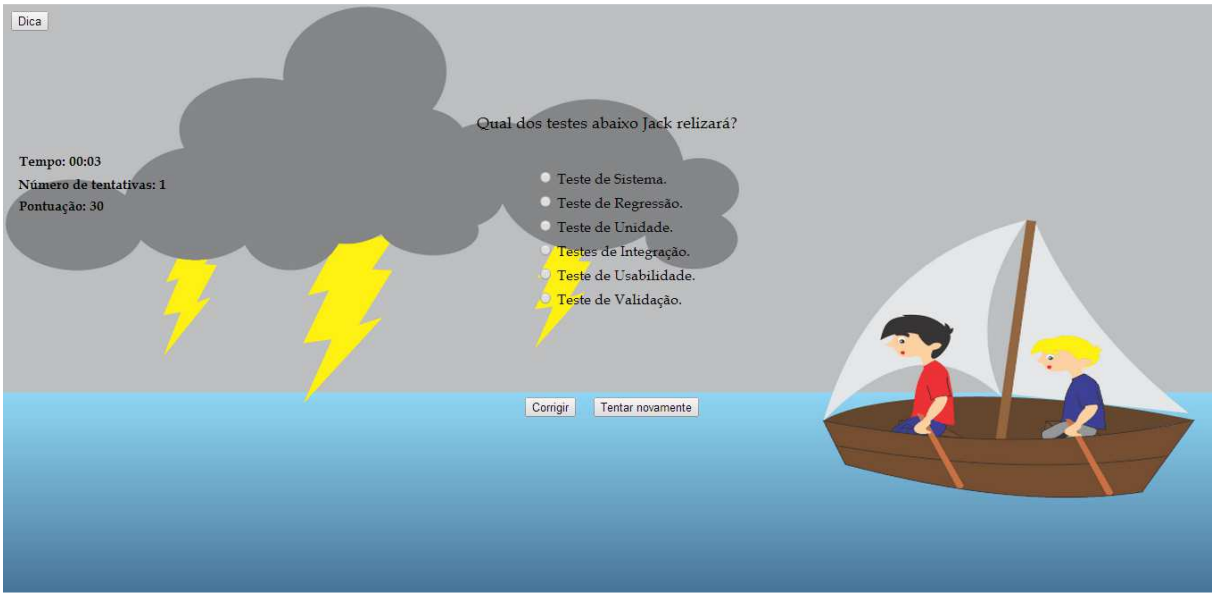
A última fase, *A Tempestade*, é representada pelas Figuras 20 e 21. Na Figura 20 explica que após *Jack* e *James* colocarem o barco no mar eles avistam uma tempestade. *Jack* aproveita para verificar o funcionamento do barco em seu ambiente natural. A Figura 21 mostra alternativas de testes e o jogador deverá selecionar o teste que Jack irá fazer.

Figura 20 - Tela inicial da quarta fase



Fonte: O autor.

Figura 21 - Tela da quarta fase



Fonte: O autor.

3.2.2 Desafios

Nos desafios os conteúdos de Teste de Software são abordados com um maior grau de dificuldade do que nas fases do jogo.

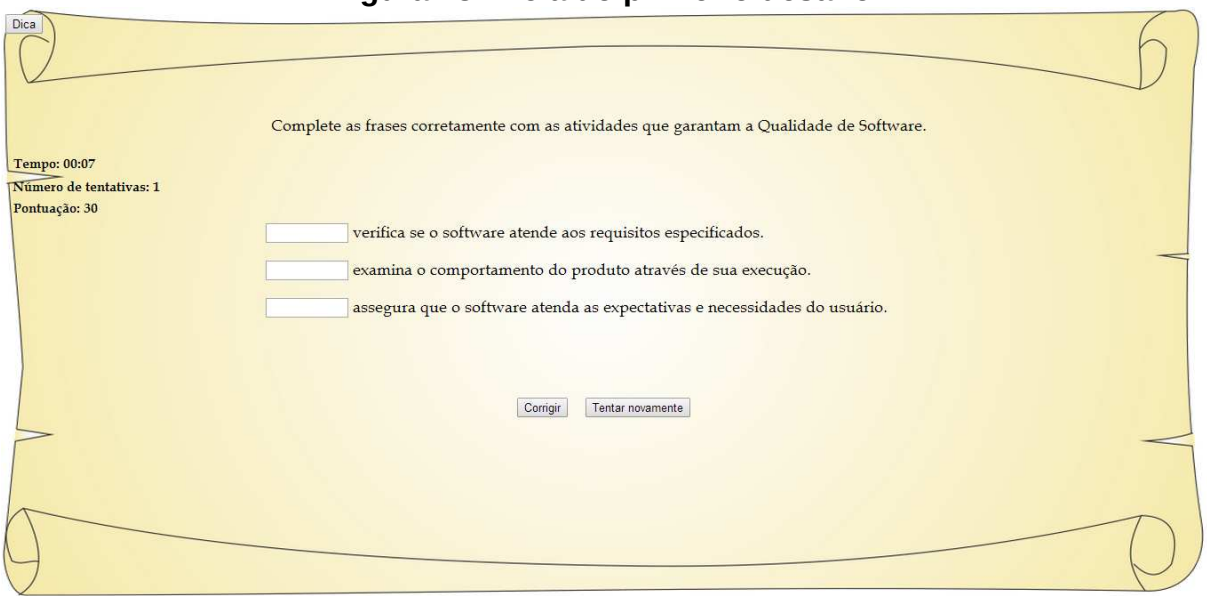
Nesse tipo de etapa o jogador se depara com problemas que para que ele possa encontrar a solução exigirá dele um maior conhecimento sobre os assuntos abordados.

O primeiro desafio, *O Homem de Preto*, está representado nas Figuras 22 e 23. A Figura 22 apresenta o contexto histórico do desafio: *Jack* encontra uma figura misteriosa e assustadora, que se apresenta como o Homem de Preto. Ele pede ajuda a *Jack* para se libertar da maldição a que está preso. Para ajudá-lo *Jack* deverá desvendar as três palavras-chaves que completam as frases que estão escritas em um papiro que o Homem de Preto carrega. A Figura 23 representa a fase, nela o jogador deverá completar as lacunas com as palavras que conceituam as três frases apresentadas.

Figura 22 - Tela inicial do primeiro desafio

Fonte: O autor.

Figura 23 - Tela do primeiro desafio



Fonte: O autor.

As Figuras 24 e 25 representam *As Caixas*, o segundo desafio do jogo. Na Figura 24 é apresentado o contexto histórico: *Jack* é capturado pelo *Os Outros,* habitantes de uma tribo canibal. O chefe da tribo descobre que *Jack* é um engenheiro de teste e propõe a ele um desafio. O desafio é apresentado na Figura 25. Nesse desafio o jogador deverá ter conhecimentos sobre as metodologias de Teste de Software para que possa classificar as características entre Teste de Caixa Preta e Teste de Caixa Branca.

Figura 24 - Tela inicial do segundo desafio

Fonte: O autor.

Figura 25- Tela do segundo desafio

Fonte: O autor.

Nas Figuras 26 e 27 mostra o terceiro desafio, *O Baú*. Nesse desafio *James* encontra um baú e descobre que pra abrí-lo o jogador terá que colocar em sequência as atividades realizadas durante o Processo de Teste de Software (Figura 26). Como mostra a Figura 27 o jogador deverá sequenciar essas atividades.

Figura 26 – Tela inicial do terceiro desafio

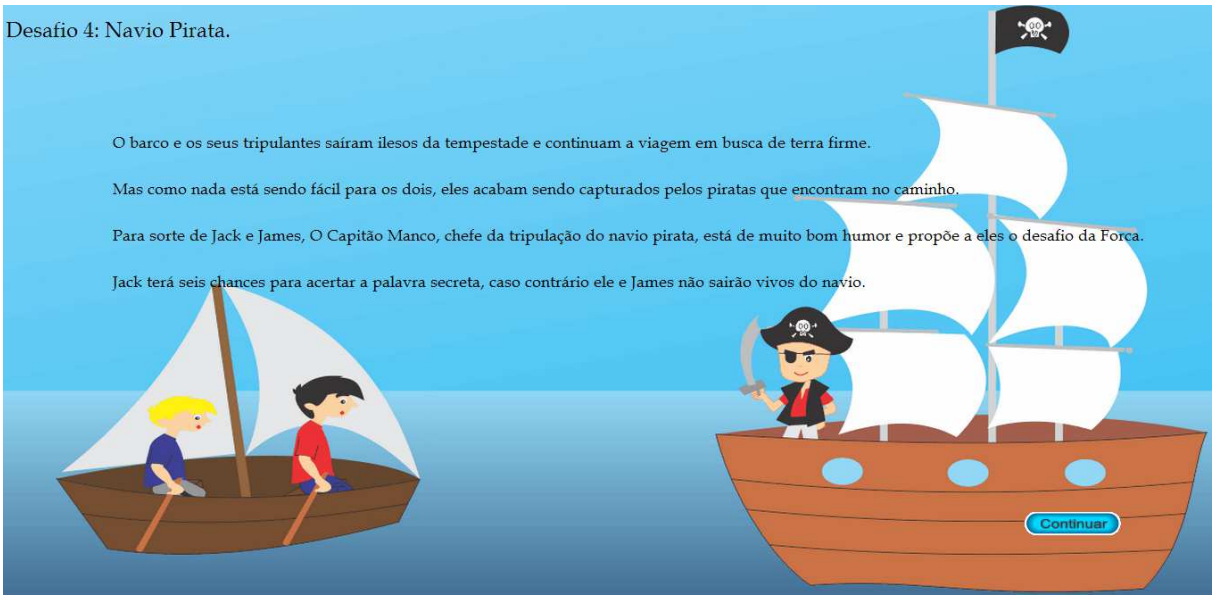
Fonte: O autor.

Figura 27 – Tela do terceiro desafio

Fonte: O autor.

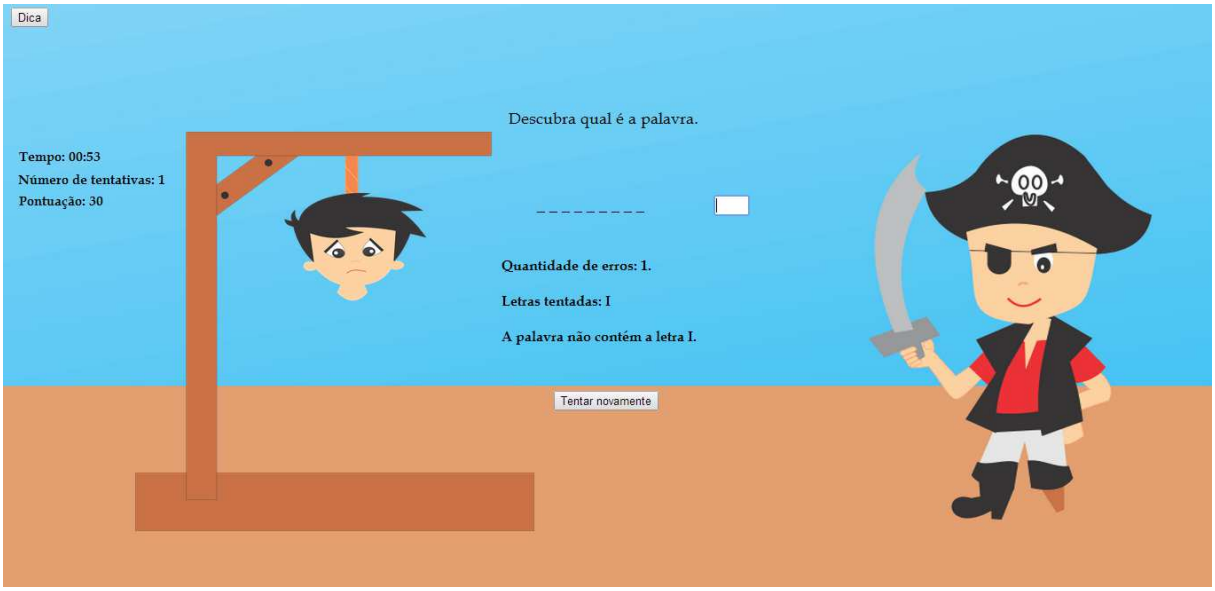
Já as Figuras 28 e 29 representam *A Forca*, o quarto desafio. *Jack* e *James* são capturados por piratas que encontram pelo caminho e *Jack, então é* colocado na forca, porém o Capitão Manco, chefe da tripulação do navio dos piratas, faz um desafio: *Jack* terá que descobrir a palavra secreta para ser libertado (Figura 28). A Figura 29 representa o desafio: o jogador deverá descobrir qual é a palavra proposta para salvar *Jack*.

Figura 28 – Tela inicial do quarto desafio



Fonte: O autor.

Figura 29 – Tela do quarto desafio



Fonte: O autor.

3.3 Tecnologia Utilizada

O jogo foi desenvolvido utilizando *HTML 5*, *CSS* e *Javascript*. A linguagem *HTML 5* foi escolhida porque, entre outras coisas, as aplicações desenvolvidas por essa linguagem rodam em qualquer dispositivo que tenha um navegador *web*, além de ser uma tecnologia gratuita.

Foi utilizado o *Notepad*++, editor de texto de código aberto, para a construção dos arquivos *.html* e *.css*.

Foram utilizadas também as ferramentas *CorelDRAW* e o *Adobe Flash*, para a construção das imagens e das animações utilizadas no jogo, respectivamente.

3.4 Validação do Jogo

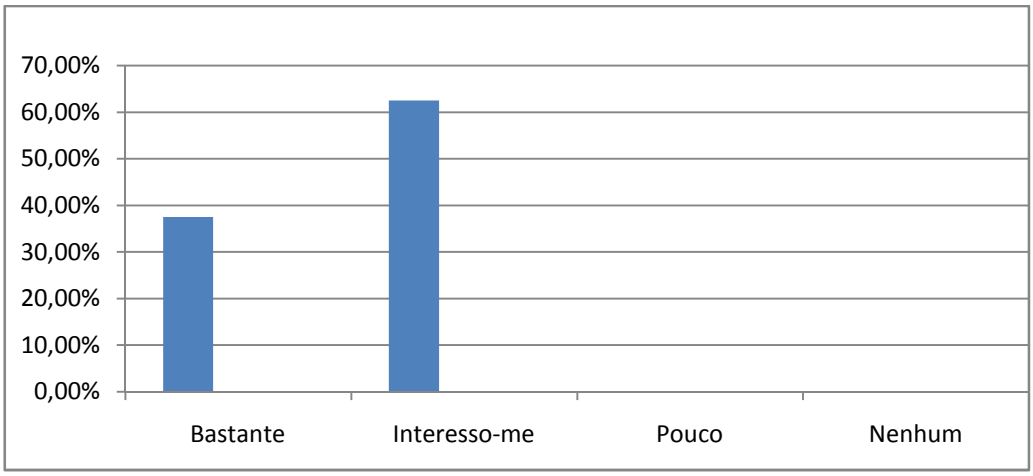
Com o objetivo de verificar a validação do jogo, "As aventuras de *Jack Test*" foi disponibilizado aos alunos da Faculdade de Tecnologia e Ciência (FTC), campus de Vitória da Conquista do curso de Sistema de Informação que cursam a disciplina

de Engenharia de Software e posteriormente aplicado um questionário que aborda questões referentes à utilização e importância do jogo.

Os oitos alunos que cursam a disciplina de Engenharia de Software responderam o questionário.

Quando foi questionado o grau de interesse à Teste de Software 62,5% dos alunos afirmam que se interessam bastante pelo assunto e os outros 37,5% indicaram que se interessam pelo assunto. A Figura 30 apresenta esses dados.

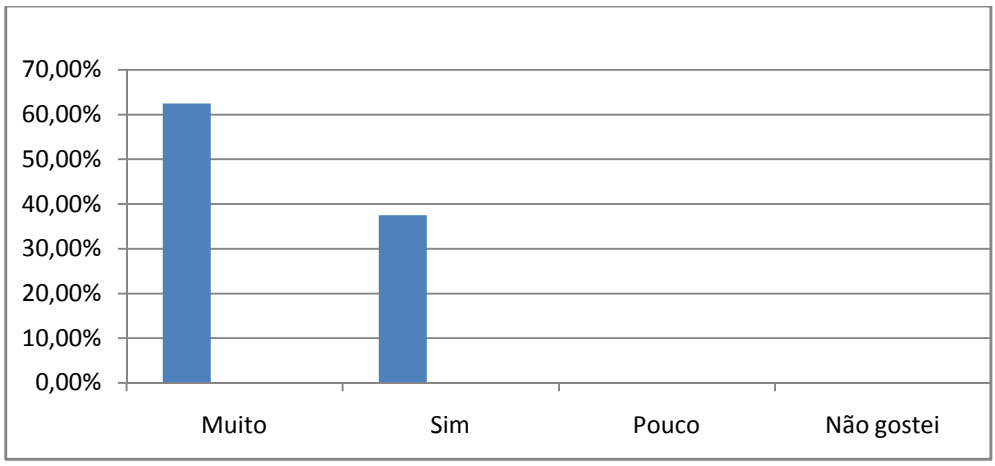
Figura 30 - Grau de interesse sobre Teste de Software



Fonte: O autor.

Ao serem questionados sobre a percepção do jogo 37,5% dos alunos responderam "muito", ou seja, que gostaram muito do jogo e se sentiram muito motivados para executar as tarefas e os outros 62,5% disseram que "sim". A Figura 31 representa esses dados.

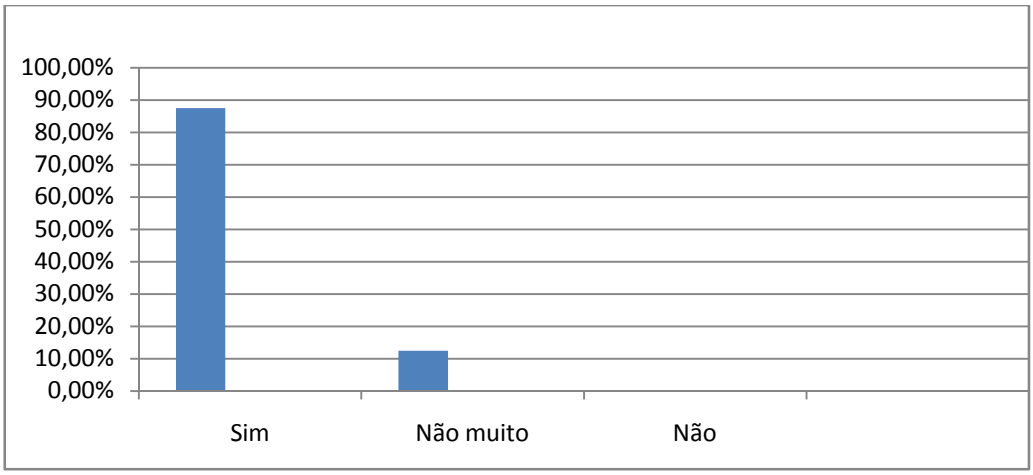
Figura 31 - Nível de motivação em relação ao jogo



Fonte: O autor.

Quando foi solicitado que o aluno informasse o grau de clareza que o conteúdo do jogo foi exposto apenas 12,5% afirmaram que o conteúdo do jogo está um pouco confuso e os demais alunos, 87,5%, disseram que o conteúdo é exposto de forma clara. A Figura 32 demonstra isso.

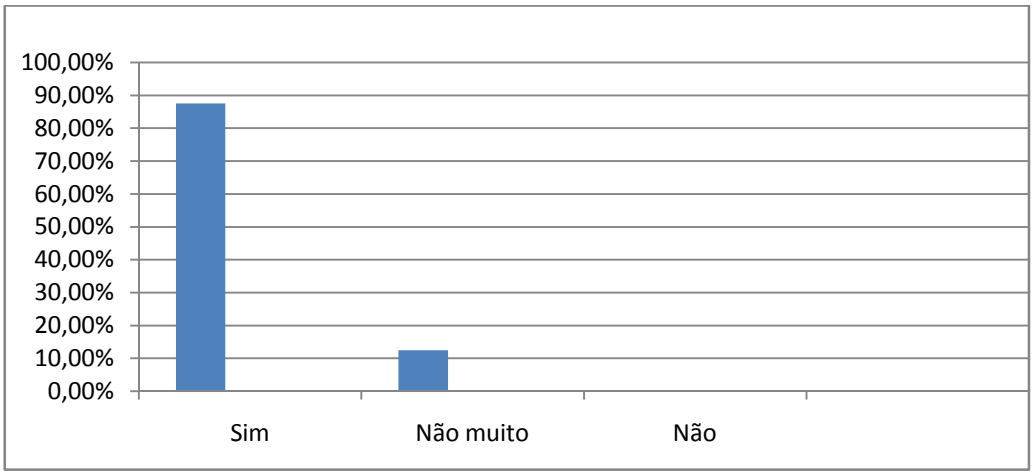
Figura 32 - Nível de clareza do conteúdo do jogo



Fonte: O autor.

Ao serem questionados se o jogo reforça os conceitos dados em sala de aula 87,5% dos alunos responderam que "sim". A Figura 33 mostra os dados levantados.

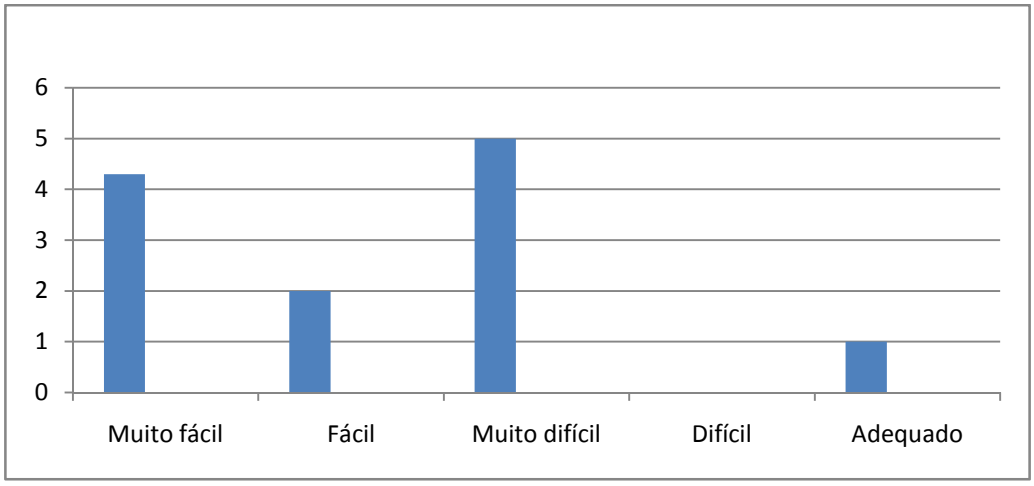
Figura 33 - Conteúdo relevante para o aprendizado



Fonte: O autor.

Quando foram questionados sobre o grau de dificuldade 25% dos alunos consideraram o jogo fácil, 62,5% consideraram muito difícil e 12,5% o consideram adequado. A Figura 34 ilustra esses dados.

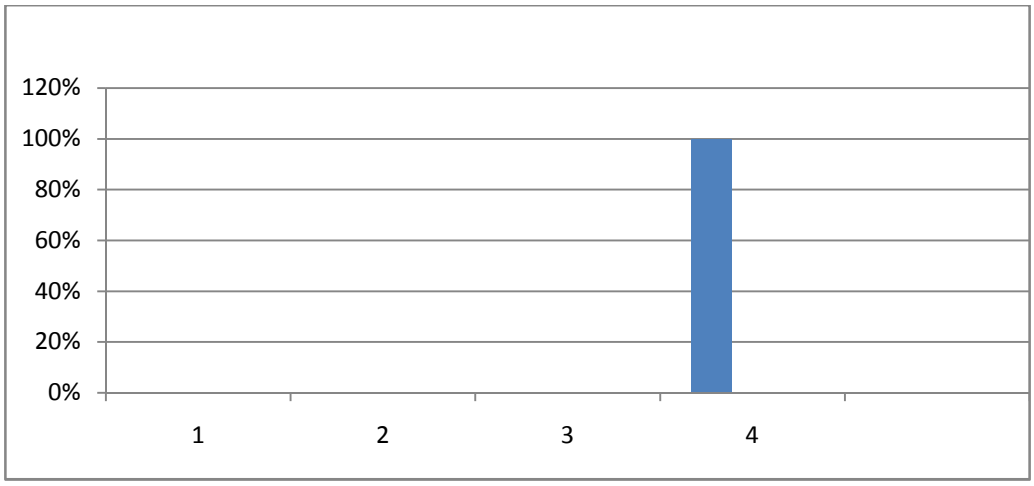
Figura 34 - Grau de dificuldade do jogo



Fonte: O autor.

Depois de ter finalizado o jogo todos os alunos responderam que saberia conceituar as quatro fases de Teste de software. A Figura 35 apresenta esse dado.

Figura 35 - Conhecimento após utilizarem o jogo

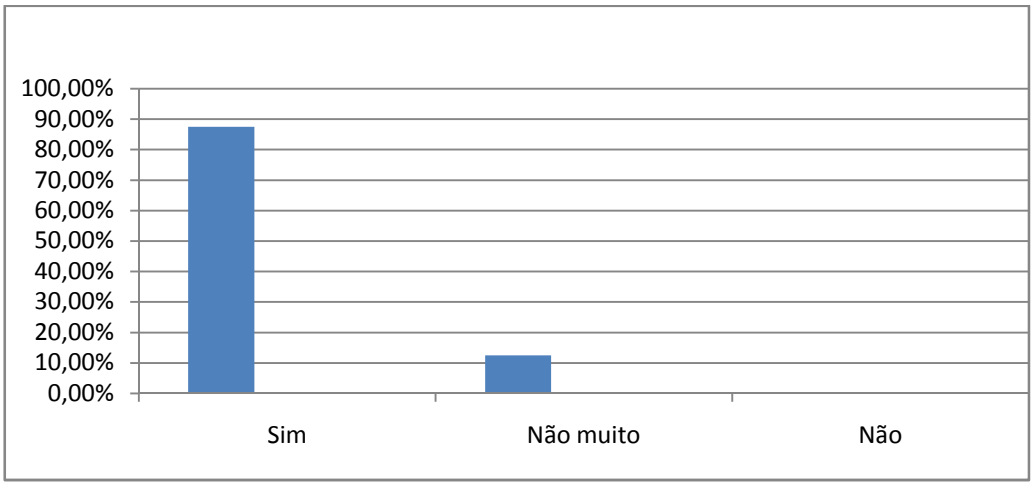


Fonte: O autor.

O aluno informou, também, se o jogo foi relevante para o seu aprendizado, 87,5% deles responderam que "sim" e apenas 12,5% informaram que "não muito",

que apenas reafirmou os conceitos dados em sala de aula. A Figura 36 mostra esses dados.

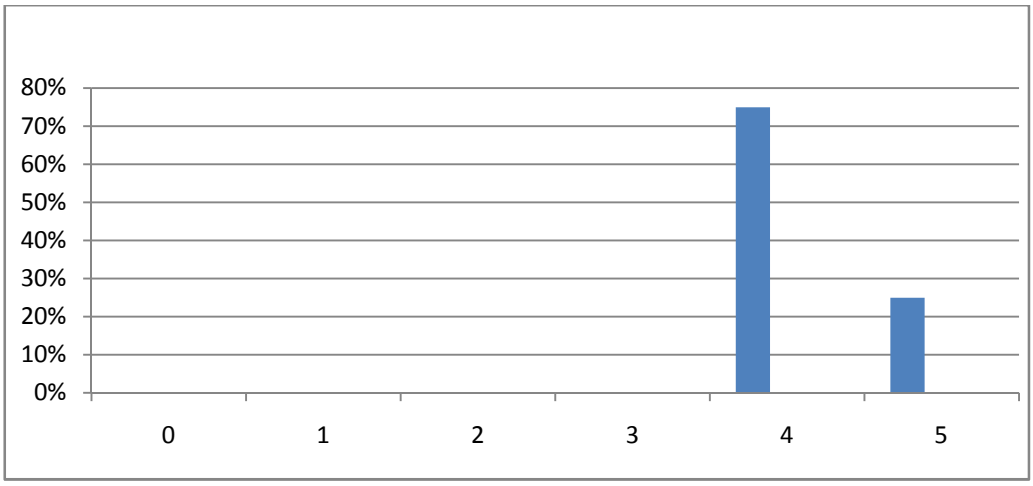
Figura 36 - Contribuição na aprendizagem



Fonte: O autor.

E por fim foi pedido ao aluno que conceituasse o jogo de 0 (ruim) a 5 (ótimo), 75% dos alunos deram a nota 4 e 25% deles deram a nota 5. A Figura 37 mostra os dados levantados.

Figura 37 - Conceito do jogo de 0 (ruim) a 5 (ótimo)



Fonte: O autor.

4 TRABALHOS RELACIONADOS

Diversos jogos foram desenvolvidos para auxiliar o ensino na área de Engenharia de Software, porém poucos foram direcionados aos Testes de Software.

Durante uma breve pesquisa foram encontrados os seguintes jogos:

O Jogo das 7 Falhas (Diniz, 2011) é um jogo educacional de simulação e foi desenvolvido em *Java,* com o intuito de simular a execução de casos de teste de caixa preta. O jogador assume o papel do testador de uma equipe de teste de software de uma empresa fictícia e deverá identificar as 7 falhas existentes nas funcionalidades do software que está sendo testado.

O jogo possui dois níveis de complexidade: baixa e média. O jogador só avançará pra o segundo nível se conseguir descobrir as sete falhas existentes no primeiro nível.

Além dos alunos de Ciência da Computação e Sistemas de Informação, esse jogo tem como público-alvo profissional de software já formado em uma dessas áreas e que trabalhem ou não com testes de software.

O Jogo das 7 Falhas não está mais disponível on-line. A Figura 38 representa a tela inicial do jogo.

Figura 38 – Tela inicial do Jogo das 7 Falhas

Fonte: Diniz (2011)

O U-Test (Silva, 2010) é um jogo educacional de simulação que tem como foco auxiliar o ensino de teste de unidade Foi desenvolvido utilizando a linguagem *Hypertext Preprocessor* (PHP).

Neste jogo o jogador, também, assume o papel do testador responsável por escrever teste de unidade para funções que já estão escritas de um sistema hipotético. Durante o jogo o jogador terá que aplicar técnicas de teste de software que foram aprendidas em sala de aula. Para o jogo é necessário que o jogador tenha noções básicas sobre programação, Engenharia de Software e Teste de Software.

Seu público-alvo são os alunos dos cursos de Ciência da Computação ou Sistema de Informação e que estejam cursando a disciplina de Engenharia de Software.

A Figura 39 representa a tela inicial do jogo.

Figura 39 – Tela inicial de U- Test

Fonte: Silva (2010)

iTest Learning (Farias, Moreira, Coutinho e Santos) também é um jogo educacional de simulação, que tem como objetivo apoiar o ensino de teste de software na etapa de planejamento. Simula um ambiente para a realização do planejamento de teste de software, descrevendo um projeto hipotético. Foi desenvolvido utilizando a linguagem *Java*.

O jogo possui três níveis de dificuldade: fácil, médio e difícil e o jogador deverão escolher em qual nível quer jogar.

Este jogo também tem como público-alvo os alunos dos cursos de graduação da área de computação e informática que cursam disciplinas que envolvam conteúdos de Teste de Software.

A Figura 40 representa a tela inicial do jogo.

Figura 40 – Tela inicial do iTest Learning

Fonte: Farias, Moreira, Coutinho e Santos

O Quadro 5 mostra uma comparação entre os jogos encontrados durante a pesquisa e o jogo "As aventuras de *Jack Test*".

Quadro 5 - Comparação entre os jogos

	Jogo das 7 Falhas	**U-Test**	**iTest Learning**	**As aventuras de Jack Test**
Público-alvo	Estudantes e Profissionais	Estudantes	Estudantes	Estudantes
Idioma	Português	Português	Português	Português
Gênero	Simulação	Simulação	Simulação	Aventura e estratégia
Plataforma	Web	Web	Web	Web
Ano de desenvolvimento	2011	2010		2014
Tempo cronometrado para terminar o jogo ou a etapa	Não	Não	Não	Sim
Nível de aprendizagem (taxonomia revisada de Bloom)	Lembrar, entender e aplicar	Lembrar, entender e aplicar	Lembrar, entender e aplicar	Lembrar e entender
Linguagem de desenvolvimento	Java	PHP, ActionSciprt	Java	HTML 5, JavaScript e CSS

Fonte: O autor.

5 CONCLUSÃO E TRABALHOS FUTUROS

Os jogos educacionais estão cada vez mais presentes no cotidiano dos alunos e professores. O principal motivo da sua utilização é que os jogos possuem um fator motivacional.

Os dois principais objetivos desse trabalho foram demonstrar a importância dos jogos para o ensino na área da Computação e o desenvolvimento de um jogo educacional que aborda os conceitos da disciplina de Teste de Software.

Por meio da análise das respostas do questionário aplicados aos alunos da Faculdade de Tecnologia e Ciência (FTC) que cursam a disciplina de Engenharia de Software verificou-se que o jogo cumpre o objetivo para qual ele foi proposto, que é auxiliar o ensino de teste de software.

A principal dificuldade para a realização desse trabalho foi encontrar jogos educacionais que abordassem os conceitos de Teste de Software para realizar um estudo comparativo entre eles e o jogo desenvolvido.

Pode-se concluir que os jogos educacionais são de extrema importância para que o aluno absorva os conceitos dados em sala de aula e para que esse futuro profissional realize testes eficientes nos softwares construídos, dessa forma, desenvolva um produto de qualidade.

Como sugestão de trabalho futuro é proposto o desenvolvimento de um jogo focado na área Manutenção de Software, pois, como Testes de Software, é uma área da Engenharia de Software que é pouco abordada nos cursos de graduação da área da computação.

REFERÊNCIAS

BARTIÉ, Alexandre. **Garantia de qualidade de software: adquirindo maturidade organizacional**. Rio de Janeiro: Editora Campus, 2002.

DINIZ, Lúcio Lopes, DAZZI, Rudimar Luís Scaranto. **Jogo digital para o apoio ao ensino do teste de caixa-preta**. In: X Simpósio Brasileiro de Qualidade de Software, Curitiba, 2011

DINIZ, Lúcio Lopes, **Jogo das 7 falhas: um jogo educacional para apoio ao ensino do teste de caixa preta**. Dissertação do Curso de Mestrado, Computação Aplicada, UNIVALI, São José, (2011).

FALBO, Ricardo. A. **Engenharia de software***.* 2012, 2013. Notas de Aula. UEFS.

FARIAS, Virgínia, MOREIRA, Carla, COUTINHO, Emanuel, SANTOS, Ismayle S.*.* ***iTest Learning*****: um jogo para o ensino do planejamento de testes de software**.

FERRAZ, Ana Paula C.M. **Taxonomia de Bloom: revisão teórica e apresentação das adequações do instrumento para a definição de objetivos institucionais**.

GRANADO, Regina Célia. **O jogo na educação: aspectos didático-metodológicos do jogo na educação matemática**, 2011.

MENDES, Antonio. **Arquitetura de software: desenvolvimento orientado para arquitetura**. Rio de Janeiro: Campus, 2002.

PRESMAN, Roger S. **Engenharia de software**. São Paulo: Editora: Pearson, 2009.

RIOS, Emerson, MOREIRA, Trayahú. **Teste de software**. Rio de Janeiro: Editora Alta Books, 2006.

RODRIGO, Motta, TRIGUEIRO, José Júnior. ***Short game design document*** **(SGDD): documento de game design aplicado a jogos de pequeno porte e advergames. Um estudo de caso do *advergame Rockergirl Bikeway***, 2013.

SILVA, Antônio Carlos. **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.

SILVA, Marcelo Ferreira. **A Taxonomia de Bloom revisada: contribuições para a elaboração do processo de avaliação**. Universidade Tecnológica Federal do Paraná, II Ciclo de Seminários Pedagógicos, 2013.

SILVEIRA, Luís Silveira. **Jogo educacional para apoiar o ensino de melhoria de processo de software com foco no nível G de maturidade do MPS-BR**. Dissertação do Curso de Mestrado, Computação Aplicada, UNIVALI, São José, 2012.

SOMMERVILLE, Ian. **Engenharia de software**, 8a. ed. São Paulo: Editora: Pearson, 2010.

WANGENHEIM, Chistiane Gesse Von. **Como ensinar com jogos?**, XXXII Congresso da Sociedade Brasileira de Computação, Curitiba, 2012.

APÊNDICE I

Questionário

Questionário destinado aos alunos de Sistemas da Informação da FTC. As respostas serão utilizadas no Trabalho de Conclusão de Curso de Kely Santos Macêdo.

1 – Indique qual é o seu grau de interesse por Teste de Software.

- o Interesso-me bastante pelo assunto.
- o Interesso-me pelo assunto.
- o Possuo pouco interesse pelo assunto.
- o Não possuo nenhum interesse pelo assunto.

2 – Você achou o jogo atrativo e se sentiu motivado para executar as tarefas?

- o Muito.
- o Sim.
- o Pouco
- o Não gostei.

3 – O conteúdo do jogo é exposto de forma clara, o texto é simples e natural?

- o Sim, entendi claramente.
- o Não muito, o conteúdo do jogo está um pouco confuso, o texto pouco ajuda;
- o Não, o conteúdo é totalmente confuso, o texto não ajuda em nada.

4 – O jogo reforça os conceitos que foram dados em sala de aula?

- o Sim.
- o Não muito.
- o Não.

5 – Em relação ao grau de dificuldade do jogo você considera que ele foi:

- o Muito fácil, precisa aumentar o grau de dificuldade.
- o Fácil.

- o Muito difícil, precisa diminuir o grau de dificuldade.
- o Difícil.
- o Adequado, não precisa ser modificado.

6 – Depois de ter finalizado o jogo você saberia conceituar há quantas fases de Teste de Software?

- o 1
- o 2
- o 3
- o 4

7 – O conteúdo do jogo foi relevante para seu aprendizado?

- o Sim.
- o Não muito. Não me ensinou nada novo, apenas reafirmou os conceitos dados.
- o Não me ajudou em nada.

8 – Conceitue o jogo de 0 (ruim) a 5 (ótimo):

- o 0
- o 1
- o 2
- o 3
- o 4
- o 5