UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

FELIPE DE FREITAS JORGE

UM AMBIENTE PARA O ENSINO DE ATIVIDADES DE EXECUÇÃO DE TESTES DE SOFTWARE

QUIXADÁ
2013

FELIPE DE FREITAS JORGE

UM AMBIENTE PARA O ENSINO DE ATIVIDADES DE EXECUÇÃO DE TESTES DE SOFTWARE

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado 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]. Carla Ilane Moreira Bezzera.

QUIXADÁ
2013

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

J71a Jorge, Felipe de Freitas
Um ambiente para o ensino de atividades de execução de testes de software / Felipe de Freitas Jorge – 2013.
58 f. : il. color., enc. ; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2013.
Orientação: Profa. MSc. Carla Ilane Moreira Bezerra
Área de concentração: Computação

1. **Software - testes** 2. Jogos educativos 3. Engenharia de software I. Título.

CDD 005.14

FELIPE DE FREITAS JORGE

UM AMBIENTE PARA O ENSINO DE ATIVIDADES DE EXECUÇÃO DE TESTES DE SOFTWARE

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado 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 / dezembro / 2013.

BANCA EXAMINADORA

_____

Prof[a]. MSc. Carla Ilane Moreira Bezzera
Universidade Federal do Ceará-UFC

_____

Prof[a]. Dra. Andreia Libório Sampaio
Universidade Federal do Ceará-UFC

_____

Prof[a]. Dra. Paulyne Matthews Jucá
Universidade Federal do Ceará-UFC

Aos meus pais...

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Milton Jorge e Maria Josefina, que sempre me deram o apoio necessário para que eu pudesse seguir em frente na busca dos meus objetivos, sem eles nada disso seria possível.

Agradeço a todos os amigos que fiz ao longo do curso, através do convívio que tivemos pude aprender muito e crescer como ser humano.

Agradeço aos professores do curso que sempre se mostraram receptivos a ajudar quando precisávamos especialmente a professora Carla Ilane, minha orientadora, que deu todo o suporte necessário para a construção deste trabalho.

"As únicas pessoas que nunca fracassam são as que nunca tentam."
(Ilka Chase)

RESUMO

O teste de software é um processo fundamental no ciclo de desenvolvimento de software, através desse processo são identificados falhas e problemas que podem comprometer o devido funcionamento do projeto. A qualidade do software está diretamente ligada com a boa realização das práticas de teste o que torna esta atividade de fundamental importância para os profissionais de desenvolvimento de software e de áreas relacionadas. O ensino de teste de software deve apresentar ao aluno mais do que teorias e levantamentos de conceitos, ele deve também abordar um aspecto prático que é necessário para o entendimento dessa atividade de forma que os estudantes possam absorver de forma mais eficaz as atividades e o funcionamento dos processos de teste. A ferramenta *iTestLearning* foi desenvolvida com o objetivo de apoiar o ensino de teste de software através da disponibilização de um ambiente que pretende simular as etapas do processo de teste de software permitindo assim uma melhor experiência para o aprendizado dos estudantes das áreas de tecnologia da informação. Atualmente o jogo desenvolvido contempla duas fases do processo de teste de software as fases planejamento e projeto. É nesse contexto que este trabalho irá apresentar o desenvolvimento do modulo de fase de execução do jogo educacional *iTestLearning*, que tem como objetivo cobrir a terceira etapa do processo de testes (execução) tornando a ferramenta um meio mais eficiente e eficaz para apoiar o ensino de testes de software e um ambiente mais completo para a realização e aprendizagem das principais práticas de testes.

Palavras chave: Testes de software. Jogos educativos. Engenharia de software.

ABSTRACT

Software testing is a fundamental process in the software development lifecycle, through that process faults and problems that can compromise the proper functioning of the project are identified . Software quality is directly linked with the successful completion of the practical test which makes this activity extremely important for professional software development and related areas. The teaching of software testing should provide the student more than theories and concepts surveys, it must also address a practical aspect that is necessary for the understanding of this activity so that students can absorb more effectively the activities and functioning of test procedures. The ITestLearning tool was developed with the purpose of supporting the teaching of software testing by providing an environment that aims to simulate stages of software testing thus enabling a better experience for student learning in the areas of information technology process. Currently the game developed contemplates two phases of software testing phases planning and design process. It is in this context that this paper will present the development of the module implementation phase of the educational game iTestLearning, which aims to cover the third stage of the test (running) process making the tool more efficient and effective way to support the teaching of software testing and a more complete environment for conducting and learning from leading testing practices.

Keywords: Software testing . Educational games. Software engineering.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Em um cenário cada vez mais competitivo entre as empresas que desenvolvem software a qualidade de software é um atributo essencial. Qualidade, segundo a norma brasileira sobre o assunto (NBR ISO 8402), é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas. Modelos de gestão de processo propõem atividades de garantia da qualidade para a obtenção de melhores produtos nas organizações de desenvolvimento de software. Uma destas atividades é a de teste de software (OLIVEIRA e LAURINDO, 2008).

Teste de software é 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).

O processo de teste de software tem duas metas distintas: demostrar ao desenvolvedor e ao cliente que o software atende os requisitos e descobrir falhas e defeitos que o façam ter um comportamento incorreto, não desejável ou não em conformidade com a sua especificação (SOMMERVILE, 2011).

O teste de software possui três atividades básicas que são planejamento, projeto e execução. A fase de planejamento é onde será definida a estratégia de testes que deverão ser usados para que se tenha uma minimização dos riscos de negócios. A fase de projeto onde será feita uma identificação dos casos de testes mais adequados para que se possa apontar o número de falhas possíveis. A fase de execução, por sua vez, tem como objetivo executar os testes registrados na fase anterior e verificar o comportamento dos mesmos em um ambiente funcional do software (BASTOS et al., 2007).

Apesar de sua importância, a atividade de teste de software ainda não tem sido explorada de uma forma mais contundente em disciplinas de engenharia de software. Embora testes consumam mais da metade da vida profissional de um programador, menos de 5% da educação de um programador são dedicados a essa atividade (BEIZER, 1990).

Em uma pesquisa realizada por Wangenheim e Silva (2009), foi apontado que entre 48 profissionais da área de software o ensino de testes de software foi considerado pouco satisfatório o que mostra uma deficiência nessa disciplina nos principais cursos de Ciências da Computação.

Uma das maiores dificuldades em realizar o ensino de teste de software se dá pela necessidade de aplicar o processo na prática levando assim apenas um conhecimento a nível teórico aos graduandos das áreas de Computação. Nesse contexto, a literatura recomenda o desenvolvimento de jogos, pois os mesmos podem permitir a experimentação de situações que seriam vivenciadas fora do contexto educacional, como, por exemplo, situações diretamente ligadas ao ambiente profissional (SILVA, 2010).

Alguns trabalhos foram propostos para auxiliar o ensino das atividades de testes de software, dentre eles podemos citar o Jogo das 7 Falhas (DINIZ e DAZZI, 2011), *U-Test* (SILVA, 2010) e *BUG HUNT* (ELBAUM et al., 2007), que mostram o uso de jogos para simulações de testes específicos. Como por exemplo, o teste da caixa-preta realizados pelos jogos 7 falhas e *U-test*. Nesses dois jogos, o aluno deverá realizar os testes baseados nas técnicas de caixa-preta, teste que verifica a saída de dados usando entradas de vários tipos, através de informações que são disponibilizadas no jogo sobre um determinado projeto. No jogo *BUG HUNT* o aluno deverá realizar etapas relacionadas aos testes de caixa-preta, caixa-branca, teste que usa a perspectiva interna para modelar casos de teste, e JUnit (framework para automação de testes) e também a conceitos básicos de teste.

Dentre estes trabalhos não foram encontrados jogos que simulem todas as atividades de testes de software, composta por planejamento, projeto e execução dos testes. A execução desses testes precisa ser embasada em um planejamento prévio e uma modelagem de como possam ser executados esses testes, de forma que auxilie especialmente o teste caixa-preta.

Neste contexto, foi desenvolvido pela Universidade Federal do Ceará no Campus Quixadá um ambiente de apoio ao ensino de testes de software, na forma de jogos de simulação, denominado *iTestLearning*. Este foi desenvolvimento em 2011 com a criação da fase de planejamento e continuação em 2012 com a adição da fase de projeto. Através de avaliações feitas pelo modelo Savi et al. (2011), foi possível comprovar a eficiência do jogo dentro das atividades de teste de software. No entanto, para cobrir todas as atividades de testes é necessário simular os testes planejados e projetados de forma atender todas as fases dos testes de software.

Para suprir essa necessidade, este trabalho tem como objetivo desenvolver a fase de execução do sistema *iTestLearning* que terá como atividade simular os testes funcionais em sistemas hipotéticos, de forma a utilizar o planejamento e o projeto prévio dos testes para determinado projeto selecionado. Através desse módulo o jogo pretende se tornar mais

completo e fornecer ao aluno uma ferramenta com uma maior capacidade para beneficiar os graduandos em cursos da área da computação.

1.1 Objetivos

Objetivos gerais:

- Desenvolver o módulo da fase de execução do jogo *iTestLearning* para tornar a ferramenta mais completa e cobrir o ensino de execução de testes.

Objetivos específicos:

- Buscar na literatura trabalhos relacionados a jogos que apoiem o ensino de teste de software;
- Levantar os principais conceitos de teste de software;
- Efetuar levantamento de requisitos para desenvolvimento do produto;
- Utilizar o *game design* para definir processo o de criação do conteúdo e das regras de um jogo;
- Desenvolver o jogo utilizando a linguagem JAVA e os *frameworks* JSF 2.0 e Hibernate;
- Avaliar a fase de execução seguindo a metodologia desenvolvida por Savi (2011).

1.2 Estrutura do Trabalho

Este trabalho está estruturado em cinco seções, além desta introdução. A Seção 2, Fundamentação Teórica, apresenta os principais tópicos levantados para a fundamentação deste trabalho, nesta seção serão apresentados conceitos de teste de software, processos de teste de software, técnicas de teste e tipos de teste. Também será abordado o cenário do ensino de teste de software e a importância da utilização de jogos como uma ferramenta de apoio ao ensino. Por fim, será feita uma descrição sobre a ferramenta *iTestLearning* mostrando seu atual estado e as fases contempladas.

A Seção 3, Procedimentos Metodológicos, apresenta a metodologia utilizada para a execução do trabalho que foi dividida em quatro etapas: levantar conceitos de teste, identificar requisitos, desenvolver a fase de execução e avaliar o jogo.

A Seção 4, Desenvolvimento do módulo de execução, irá detalhar todo o procedimento que envolveu o processo de desenvolvimento bem como uma explicação sobre o funcionamento do módulo.

A Seção 5, Avaliação do módulo de execução, apresenta um detalhamento sobre a metodologia utilizada para a avaliação da fase de execução juntamente com a apresentação dos resultados coletados e uma análise sobre os mesmos.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão explorados os principais conceitos de teste de software, uma abordagem sobre o ensino de engenharia de software e também como o desenvolvimento de jogos interativos podem ser úteis para a melhoria do aprendizado de conceitos e práticas dessa área.

2.1 Teste de software

As exigências por softwares 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 (MALDONADO et al., 2004).

Um software é testado para descobrir erros que foram feitos inadvertidamente no momento em que foi projetado e construído (PRESSMAN, 2011). O processo de testar um software tem como função adicionar valor ao produto aumentando a qualidade e a confiabilidade do mesmo.

Como citado anteriormente, o teste de software tem duas metas distintas. A primeira meta conduz ao teste de validação, no qual se espera que o sistema seja executado reflitam o uso esperado do sistema. A segunda meta conduz o teste aos defeitos, no qual é procurado expor os defeitos muitas vezes simulando comportamentos anormais de um sistema (SOMMERVILLE, 2011).

É importante citar que segundo Sommerville (2011) os testes nem sempre podem demostrar que um software é livre de defeitos ou que ele se comportará conforme especificado em todas as circunstâncias. O que nos leva a concluir que o teste de software é algo que deve ser acompanhado durante todo o ciclo de vida de um produto.

O teste pode ser classificado de duas maneiras: teste baseado em especificação (*specification-based testing*) e teste baseado em programa (*program-based testing*). De acordo com tal classificação, têm-se que os critérios da técnica funcional são baseados em especificação e tanto os critérios estruturais quanto baseados em erros são considerados critérios baseados em implementação (HOWDEN, 1987).

As principais técnicas de teste de software são divididas em três categorias: funcionais estruturais e baseada em erros (MALDONADO et al., 2004). Entre os

procedimentos de teste conhecidos este trabalho enfoca na utilização de testes funcionais principalmente na aplicação do teste de caixa-preta que será utilizada como técnica essencial para a realização da fase de execução do jogo *iTestLearning*.

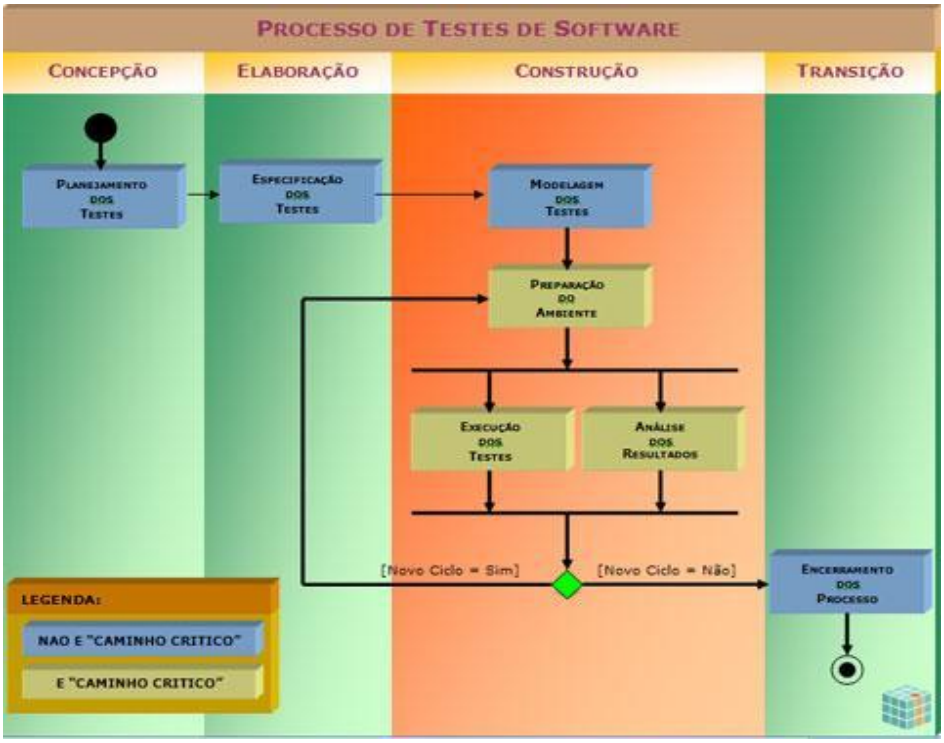
2.2 Processo de Teste de Software

O processo de teste de software é voltado para o alcance de um nível de qualidade de produto, que durante o processo de desenvolvimento de software muda conforme avanço das atividades - requisitos, protótipos, modelo de dados lógico, modelo de dados físico, código-fonte, módulos funcionais e finalmente um sistema (TOZELLI, 2008).

A Figura 1 demostra o modelo de processo de teste de software, o detalhamento dos passos estão descritos a seguir.

A primeira etapa corresponde ao planejamento de testes. Este nível caracteriza-se pela definição de uma proposta de testes baseada nas expectativas do Cliente em relação à prazos, custos e qualidade esperada, possibilitando dimensionar a equipe e estabelecer um esforço de acordo com as necessidades apontadas pelo Cliente (BARTIE, 2007).

Figura 1 - Processo de Teste de Software.



Fonte: BARTIE, 2007.

Após a finalização da fase de planejamento dá-se início a etapa de especificação de testes que terá como objetivos básicos elaborar/revisar casos e roteiros de teste. No

cumprimento dos objetivos é preciso que se observe que: 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 (BASTOS et al., 2007).

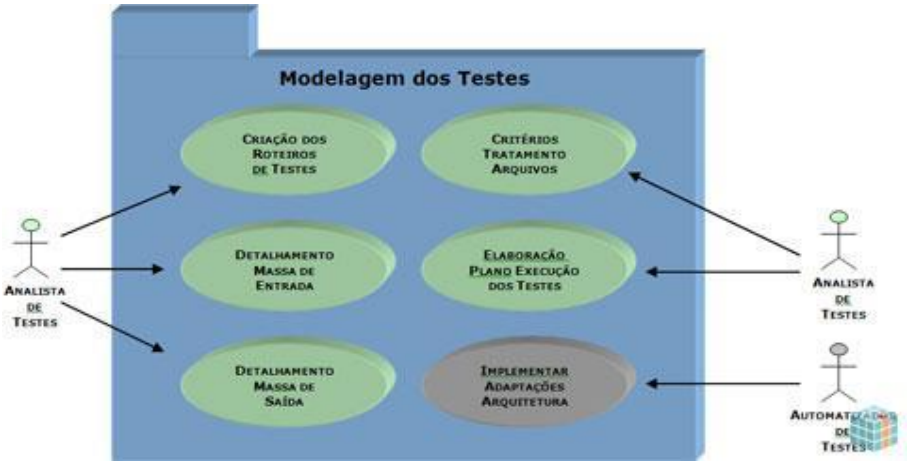
A fase seguinte corresponde à etapa de modelagem de testes onde será feita a identificação de todos os elementos necessários para a implementação de cada caso de testes especificado. Fazem parte desta etapa a modelagem das massas de testes e a definição dos critérios de tratamento de arquivos (envelhecimento, descaracterização e comparação de resultados). Durante essa etapa se segue os processos de preparação do ambiente que visa à disponibilização física de um ambiente de testes, execução dos testes que tem como objetivo a execução e conferência dos testes planejados e por fim a análise de resultados onde será feito a análise e confirmação dos resultados relatados durante a fase de execução dos testes (BARTIE, 2007).

Figura 2 - Etapa de Especificação de Testes

Fonte: BARTIE, 2007.

A última etapa do processo de teste de software corresponde ao encerramento do processo onde será feito a avaliação de todo o Processo de Teste de Software, comparando os resultados alcançados em relação ao que foi inicialmente planejado. Neste momento, diversos indicadores são extraídos, visando avaliar qualitativamente e quantitativamente o desempenho do trabalho, através de comparações históricas de projetos anteriores (BARTIE, 2007).

Figura 3 - Etapa de Modelagem de Teste



Fonte: BARTIE, 2007.

2.3 Técnicas de Testes de Software

Como citado anteriormente, as principais técnicas de teste de software são divididas em três categorias: funcionais, estruturais e baseada em erros. Testes funcionais são baseados em técnicas onde os critérios e requisitos de teste são estabelecidos a partir da função da especificação do software. Testes estruturais são aqueles em que os critérios e os requisitos são derivados essencialmente a partir das características de uma particular implementação em teste. Por fim na técnica baseada em erros os critérios e requisitos de teste são oriundos do conhecimento sobre erros típicos cometidos no processo de desenvolvimento de software (MALDONADO et al., 2004).

Atualmente na literatura existem diversas técnicas que procuram explorar o uso de testes de diversas maneiras e diferentes metodologias, dentre as técnicas de teste de software mais conhecidas podemos citar:

- Teste de Caixa-Preta: Uma abordagem para o teste onde os testadores não tem acesso ao código-fonte de um sistema ou seus componentes. Os testes são derivados a partir da especificação do sistema (SOMMERVILLE, 2011). 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);

- Teste de Caixa-Branca: É baseado em um exame rigoroso do detalhe procedimental. Caminhos lógicos internos ao software e colaborações entre componentes são testados, definindo-se casos de teste que exercitam conjuntos específicos de condições e/ou ciclos de vida (PRESSMAN, 2011);
- Teste de Regressão: Que se caracteriza pelo teste de aplicativos já estáveis que passam por uma correção de erros, ou de aplicativos já existentes que são evoluídos para uma nova versão e suas funcionalidades são alteradas (KANER, 1997).

2.4 Tipos de Testes de Software

Os tipos de teste referem-se às características do software que podem ser testadas e compreende: Teste de Funcionalidade; Teste de Interface; Teste de Desempenho; Teste de Carga (*Stress*); Teste de Usabilidade; Teste de Volume; Teste de Segurança (CRESPO et al., 2004). O detalhamento de cada um deles está escrito logo em seguida:

- Teste de Funcionalidade: 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 (BASTOS et al., 2007);
- Teste de Interface: Verifica a interação do usuário em relação ao aplicativo, garantindo o 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 (BASTOS et al., 2007);
- Teste de Desempenho: Também chamado teste de performance, tem como principal objetivo verificar se o desempenho está consistente com os requisitos definidos, empregando um volume de transações nas situações previstas de pico máximo de acesso e concorrência, e tempo de resposta obtidos nos testes, e assim, compará-los com os valores limites especificados (BARTIE, 2002);
- Teste de Carga (Stress): 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 (BASTOS et al., 2007);

- Teste de Usabilidade: O teste de usabilidade é um processo no quais participantes representativos avaliam o grau que um produto se encontra em relação a critérios específicos de usabilidade (RUBIN, 1994);
- 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 (BASTOS et al., 2007);
- Teste de Segurança: tenta verificar se todos os mecanismos de proteção embutidos em um sistema o protegerão, de fato, de acessos indevidos (DIAS, 2011).

2.5 Ensino de Engenharia de Software

Engenharia de software é uma disciplina da engenharia relacionada a todos os aspectos de software (SOMERVILLE, 2011). Em uma conferência (em 1969) Fritz Bauer a definiu como “a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais”.

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, o que 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 (MALDONADO et al., 2004).

Cada vez mais, profissionais da área de TI devem estar capacitados, portanto é necessário que o ensino da disciplina de engenharia de software seja priorizado e transmitido de uma maneira eficiente nos cursos da área da computação.

A qualidade da educação em ES pode contribuir significativamente à melhoria do estado da arte do desenvolvimento de software e auxiliar a solução de alguns problemas tradicionais e crises relacionadas com as práticas da indústria de software (GIBBS, 1994).

A Disciplina de Engenharia de Software é preocupada com a aplicação de teoria, conhecimento e prática para o desenvolvimento efetivo e eficiente de sistemas de software que satisfaçam os requisitos dos usuários (ACM/IEEE, 2008).

O principal problema encontrado no ensino de Engenharia de Software é que modelo tradicional de ensino adotado na Educação em Engenharia está apoiado na transmissão de conhecimentos, que normalmente focaliza os aspectos conceituais das diversas teorias, sem a sua necessária contextualização (BEHOLT et al., 2001) por isso é importante que o ensino dessa disciplina não se resuma apenas ao formato tradicional de aulas, entre as possíveis abordagens pode se considerar substituições de aulas expositivas por discussão de casos práticos (GNATZ et al., 2003), dinâmicas de grupo, o uso de jogos (WANGENHEIN e SHULL, 2009) e *Capstone projects* (um esforço em grupo em que alunos executam um projeto do início ao fim) (GOOLD e HORAN, 2002).

2.6 Jogos em Engenharia de Software

O ato de jogar é definido por PRENSKY (2001) como uma atividade não obrigatória, governada por regras, que produz resultados incertos e que apresenta elementos que imitam a realidade. Sintetizado nesta frase talvez estejam presentes os elementos que tornam o jogo um atrativo, principalmente, para os "nativos digitais". A incerteza de resultados e a não obrigatoriedade são características chave (FERNANDES e WERNER, 2009).

Jogos podem ser utilizados para facilitar e auxiliar o ensino em diferentes áreas do conhecimento. De maneira geral, jogos 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).

No ambiente educacional, jogos 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).

Na engenharia de software jogos podem ser utilizados para simular ambientes que representem uma experimentação de situações que seriam vivenciadas fora do contexto educacional, como, por exemplo, situações diretamente ligadas ao ambiente profissional (SILVA, 2010). Os projetos desenvolvidos em sala de aula pelos estudantes de ES, em função do tempo e da natureza didática, não permitem evidenciar diversos aspectos do processo de

desenvolvimento (BENITTI, 2008). Mesmo que o professor possa explicar a maioria destas ações em aulas expositivas, os estudantes não terão a oportunidade de participar de um processo de desenvolvimento de software completo (BENITTI, 2008). Neste contexto as utilizações de jogos para auxiliarem o ensino de software podem suprir a necessidade de se realizar uma ligação entre teorias e práticas.

Foram encontrados jogos com o objetivo de apoiar o ensino de teste de software, dentre os trabalhos pesquisados serão descritos durante esta seção os seguintes jogos: Jogo das 7 Falhas (DINIZ e DAZZI, 2011), *U-Test* (SILVA, 2010) e *BUG HUNT* (ELBAUM et al., 2007).

O Jogo das 7 Falhas (DINIZ e DAZZI, 2011) é um jogo *single-player*, no qual o jogador assume o papel de uma equipe de teste de software onde terá como objetivo descobrir as sete falhas existentes em cada funcionalidade testada, correlacionando as falhas com uma classe de equivalência ou um valor-limite. Para que o jogador consiga avançar para os demais níveis será necessário identificar os sete erros em uma determinada funcionalidade em um determinado período de tempo que irá variar dependendo do nível (25 minutos para o nível 1 e 40 minutos para o nível 2). Ao fim se o jogador conseguir encontrar os 7 erros no período de tempo determinado ele vencerá o jogo.

U-test (SILVA, 2010) é um jogo que tem como objetivo geral realizar uma simulação onde o jogador deve assumir o papel de um testador que tem como função escrever testes de unidades para funções pertencentes á um sistema hipotético. Ao inicio do jogo o usuário passa por uma “entrevista” sobre determinados conceitos de software, obter sucesso nesta etapa o usuário estará habilitado para as seguintes. Logo em seguida o jogador recebe um artefato e a partir dela serão instanciado novos desafios. O jogo possui quatro etapas com diferentes objetivos e ao fim o usuário obterá sua pontuação.

BUG HUNT (ELBAUM et al., 2007) é mais um exemplo de um jogo que apoia o ensino de teste de software que tem seu objetivo determinado por quatro etapas. Durante a primeira a ferramenta busca familiarizar o usuário com o básico de teste de execução onde ele deverá descobrir um determinado problema usando pistas providas pelos casos de teste criados. Durante a segunda etapa serão introduzidos conceitos de teste caixa-preta, para se concluir o objetivo o aluno deverá desenvolver casos de teste que demostrem os problemas do software fictício. No terceiro nível será explorada a técnica de teste de caixa-branca e por fim será apresentado aos conceitos de automação de teste. Ao completar as quatros etapas será apresentado aos alunos o resultado final.

2.7 iTestLearning

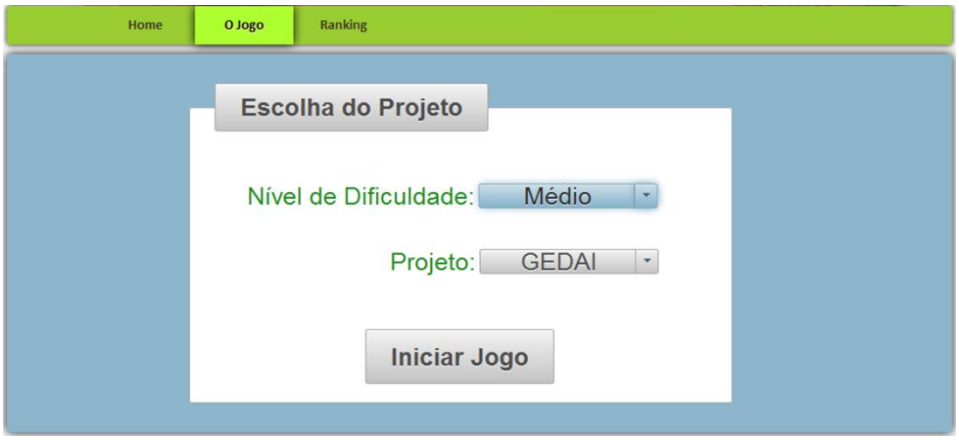
O *iTestLearning* é um jogo educacional que visa apoiar o ensino de teste de software simulando um ambiente onde o jogador deverá realizar as fases de planejamento e de projeto de teste de um determinado sistema a partir de descrições feitas sobre eles (FARIAS et al., 2012).

Sua concepção e desenvolvimento deram-se início em 2011 durante o trabalho de conclusão de curso e teve sua continuidade com o projeto de monitoria de graduação da Universidade Federal do Ceará – Campus Quixadá.

As tecnologias adotadas para o desenvolvimento foram baseadas nos frameworks JSF 2.0 e jQuery foram utilizadas para o desenvolvimento visual do jogo (telas e animações), *Hibernate* para a persistência de dados e JAVA como linguagem de programação padrão para desenvolvimento do projeto.

O jogo exige conhecimentos nas áreas de engenharia de software, para que se tenham compreensão sobre os processos de desenvolvimento de software, e Teste de Software, onde conceitos sobre teste de software serão necessários para a realização das etapas do jogo, por esse motivo ele é voltado para alunos de cursos de graduação da área de computação ou informática (FREITAS et al., 2012). A figura 4 demostra a tela inicial do jogo.

Figura 4 - Tela para escolha de um projeto a ser jogado

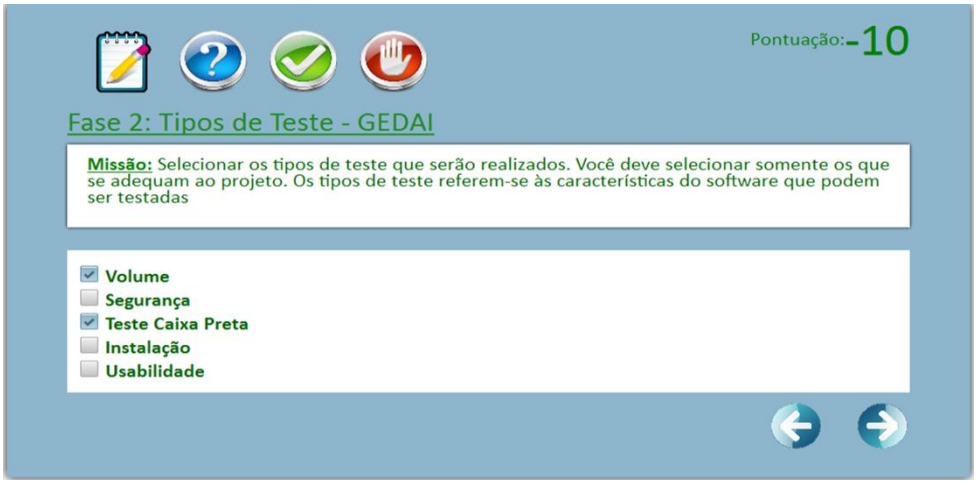


Fonte: BEZERRA e COUTINHO (2013).

Atualmente o jogo conta com duas etapas: a fase de planejamento e a fase de projeto. Durante a fase de planejamento o aluno deverá realizar o planejamento dos testes englobando durante o procedimento seis (6) etapas definidas: itens de teste, tipos de teste, níveis de teste, critérios de aceitação, ferramentas e artefatos. Em cada uma destas etapas o aluno será desafiado a selecionar entres as opções referentes ao item de planejamento qual melhor se

adequa ao projeto que está sendo jogado. Ao longo do progresso na etapa vão sendo apresentados ao jogador diversos conceitos referentes a teste de software como, por exemplo, ferramentas para testes (Selenium, Jmeter) e técnicas de teste. A Figura 5 demonstra a fase de planejamento.

Figura 5 - Fase de Planejamento



Fonte: BEZERRA e COUTINHO (2013).

A fase de projeto dar-se início após a finalização da primeira fase, mediante a obtenção de uma pontuação mínima necessária na primeira fase, e consiste em selecionar os casos de testes válidos de acordo com a descrição de uma especificação que dependendo do nível será representada através de um caso de uso, estórias de usuário ou requisitos. Os casos de testes listados apresentam um fluxo representado uma operação realizada no sistema através da entrada de dados e repostas do sistema.

Para que se possa cobrir de forma completa o processo de teste de software este trabalho tem como objetivo o desenvolvimento da fase de execução sendo integrada a fase de projeto da ferramenta.

3 PROCEDIMENTOS METODOLÓGICOS

O objetivo deste trabalho é o desenvolvimento do módulo da fase de execução em que, a partir de uma integração deste módulo com as fases de planejamento e projeto, o jogo *iTestLearning* se tornará uma ferramenta mais completa para auxiliar e dar suporte a professores e alunos no ensino de testes de software.

O jogo consiste na simulação de casos de teste funcionais em um ambiente que irá realizar a projeção de sistemas hipotéticos. Dentre as técnicas de teste funcional que podem ser aplicados ao processo da fase de execução terá como modelo o Teste de Caixa-Preta onde

serão usados os casos de teste selecionados durante a fase de projeto e a partir deles serão feitas simulações para verificar a saída dos dados em um ambiente funcional de forma que o aluno possa observar se as saídas estão coerentes com as entradas.

A metodologia utilizada para a construção do módulo de fase de execução segue as seguintes etapas: (i) Levantantamento de conceitos de testes e jogos para o ensino de testes de software, (ii) Identificar requisitos, (iii) Desenvolver módulo de execução do jogo e (iv) Avaliar o jogo.

A etapa (i) de levantamento de conceitos tem como objetivo fazer um levantamento bibliográfico sobre os principais conceitos de testes de software com foco em testes funcionais e avaliar trabalhos que mostrem uso de jogos para auxiliarem o ensino de teste de software. Para elaboração da fase de execução do jogo pretendido foram pesquisados trabalhos relacionados e modelos que pudessem servir como uma referência para a construção de um ambiente que simulasse testes funcionais Como citados anteriormente, este trabalho irá se basear nos projetos 7 Falhas (DINIZ e DAZZI, 2011), *U-Test* (SILVA, 2010) e *BUG* HUNT (ELBAUM et al., 2007).

A etapa (ii) consistiu na identificação de requisitos será feita partir da análise dos trabalhos relacionados citados anteriormente serão identificados os principais requisitos que podem ser reutilizados para a construção do módulo e quais devem ser devem ser adicionados para que possam cobrir o que as ferramentas estudadas não fornecem.

O desenvolvimento do módulo de execução, contemplado na etapa (iii), será composto por várias atividades. Ao inicio será elaborado a modelagem do jogo onde diagramas de classe e de caso de uso serão construídos para auxiliarem no desenvolvimento do jogo e para complementar a documentação da ferramenta. Também será necessário o desenvolvimento de diagramas de atividade para que possa ser demostrado a integração da fase de execução com os demais módulos da ferramenta. Protótipos de tela irão ser desenvolvidos com a finalidade de validar os casos de uso e verificar o funcionamento do jogo. Para que o módulo a ser desenvolvido possua o devido aspecto de jogo será utilizado o *Game Design Document* que tem como função definir o processo de criação dos conteúdos e das regras de um jogo (BRATHWAITE e SCHREIBER, 2009). Através da utilização deste documento poderemos determinar os principais pontos a serem considerados em relação a regras, objetivos e mecânicas do jogo, o Apêndice B mostrará mais detalhes. A codificação do jogo segue o padrão atual utilizado no jogo *iTestLearning* baseado na linguagem de

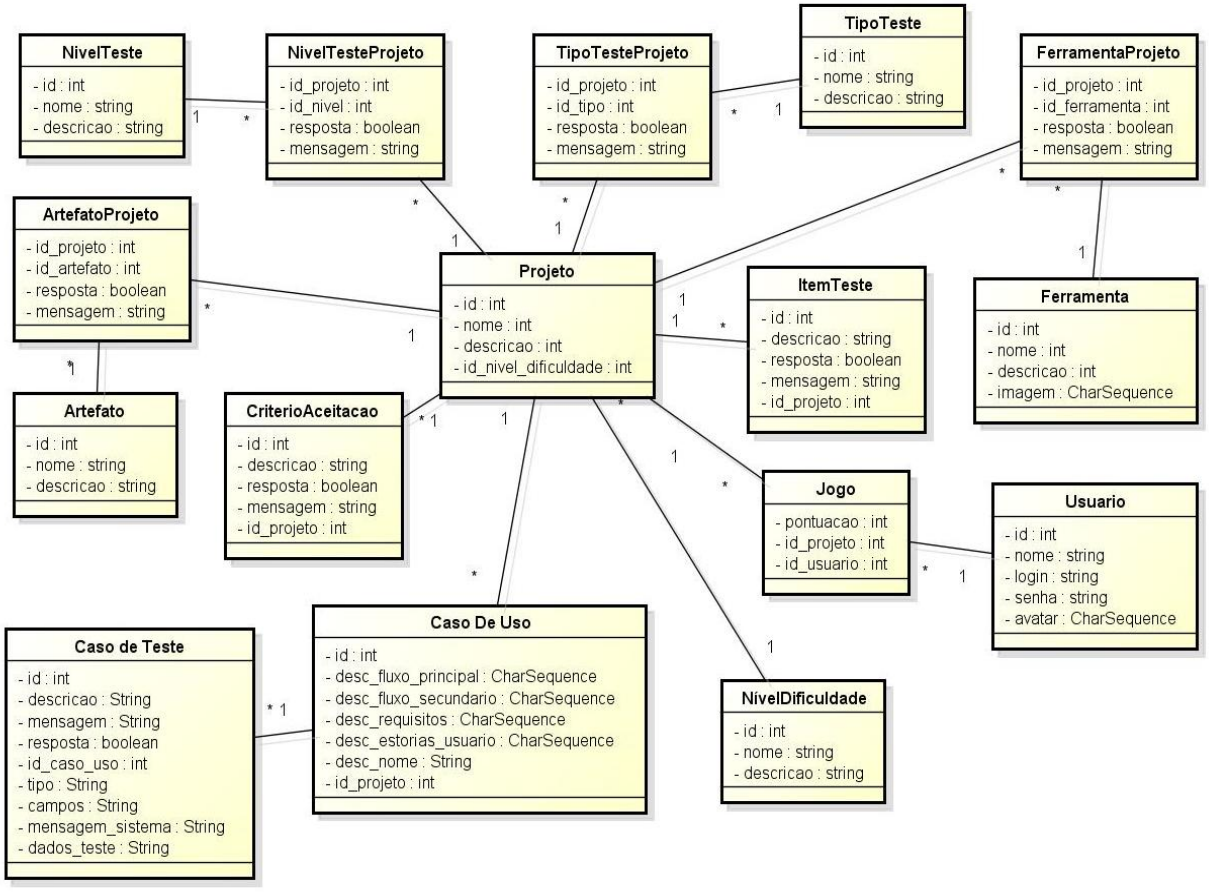
programação JAVA e frameworks como JSF 2.0 e *Hibernate* para a realização da persistência em banco de dados.

A avaliação do módulo, descrito na etapa (iv) seguirá o modelo Savi et al. (2011) desenvolvido principalmente para medir a eficiência em jogos que podem ser utilizados para apoio ao ensino e aprendizagem. Este modelo busca avaliar um jogo através de contextos como a motivação que jogador terá para usar a ferramenta, se trará uma experiência positiva aos usuários e o aprendizado adquirido como o jogo. Com base nestes elementos os alunos deverão avaliar determinados itens em uma escala com variação de -2 a +2, que correspondem a concordâncias negativas e positivas, respectivamente.

4 DESENVOLVIMENTO DO MÓDULO DE EXECUÇÃO DO ITESTLEARNING

Do processo da metodologia descrito acima já foram feitos os devidos levantamentos bibliográficos juntamente a uma análise dos requisitos necessários para o desenvolvimento do módulo. Em relação à fase de desenvolvimento foram construídos o diagrama de fluxo e de classe, que podem ser visualizados nas figuras 6 e 7 respectivamente.

Figura 6 – Diagrama de classe



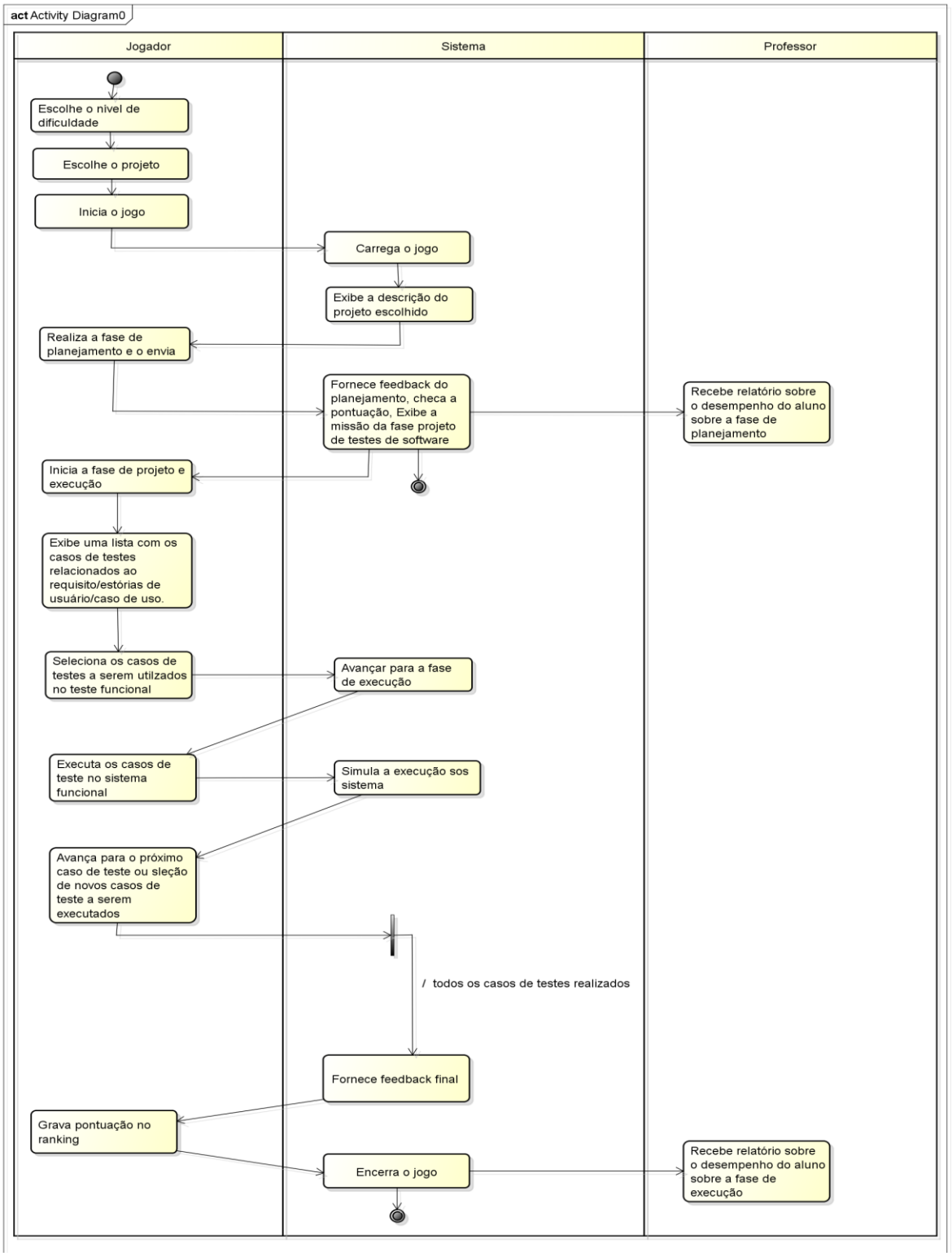
Fonte: o autor.

No presente trabalho, foram realizados os seguintes passos dos procedimentos metodológicos: levantamento de conceitos de testes e jogos para o ensino de testes de software, identificar requisitos e o game design da fase de desenvolvimento do módulo de execução do jogo.

Foram identificados os requisitos necessários para a implementação da fase de execução através da leitura de artigos sobre trabalhos relacionados, na qual foi possível realizar uma coleta das principais propriedades a serem consideradas para o desenvolvimento do módulo. A elicitação dos requisitos foi feitas através da utilização de casos de usos devido à necessidade de visualizar e validar a interação do usuário com o software. Os quatros casos de uso definidos foram: o jogador inicia a fase de execução, o jogador seleciona os casos de teste, o jogador realiza a execução dos casos de teste e o jogador termina a fase de execução. A descrição de cada um deles se encontra no apêndice A.

O desenvolvimento do *Game Design Document* foi baseado em seu *template* que se caracteriza por sete pontos principais: objetivos, história e narrativa, *gameplay*, arte, documentação técnica, gerenciamento e análise do projeto e post mortem. Dentre eles os tópicos história e narrativa e arte são ainda pontos a serem desenvolvidos no jogo, gerenciamento e análise de projeto não é aplicável ao âmbito que está sendo desenvolvida a ferramenta e post mortem só deve ser descrito após a finalização do projeto, por esses motivos estes tópicos não foram aplicados na construção do documento. Os demais pontos estão descritos no apêndice B.

Figura 7 - Diagrama de fluxo

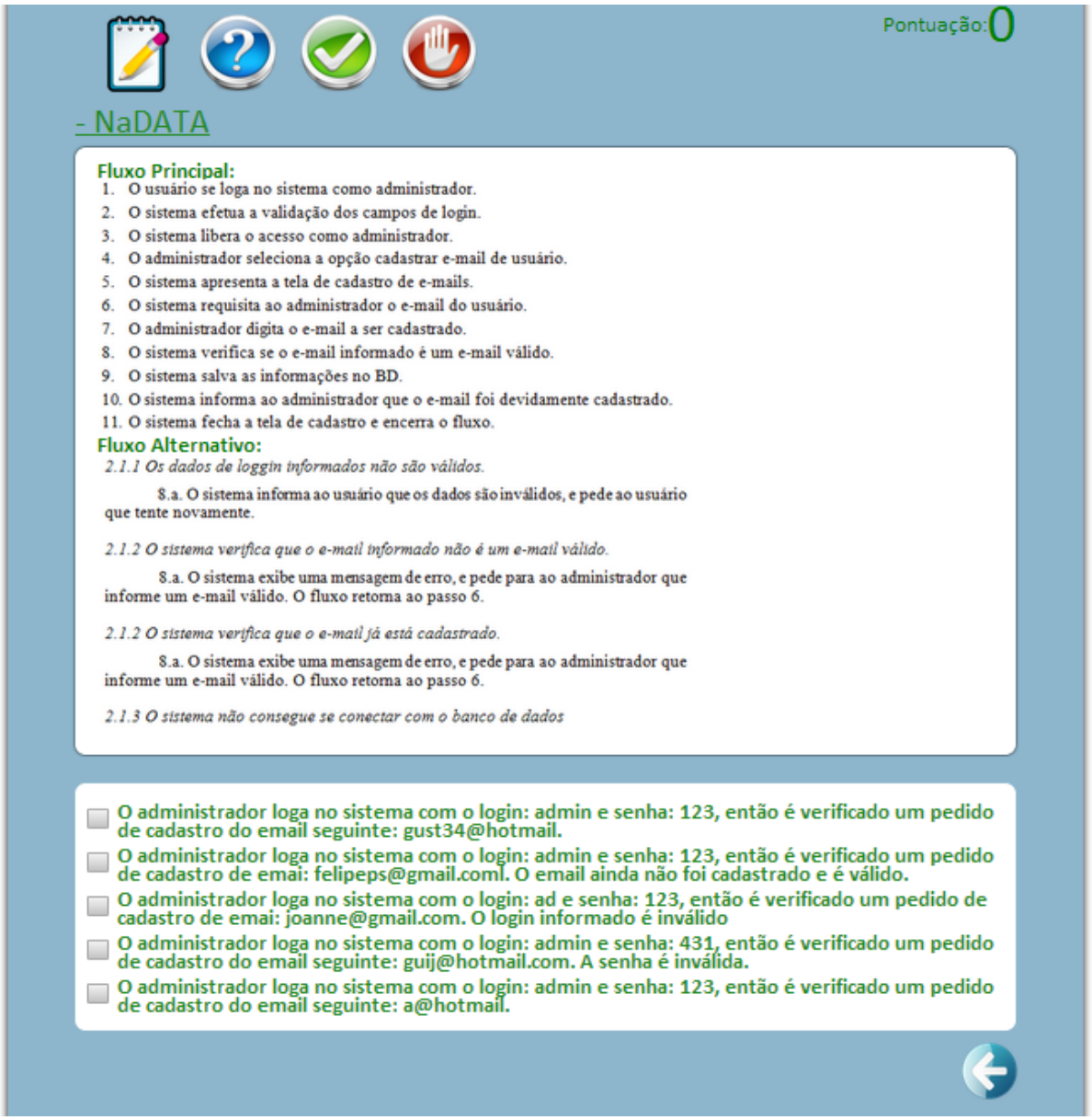


Fonte: o autor.

O *gameplay* do jogo funciona do jogo possui a seguinte dinâmica, seu inicio se dá após a finalização da fase de planejamento ao conseguir êxito nessa etapa o jogador estará habilitado para passar a fase seguinte. Na próxima fase, o sistema apresenta a descrição

detalhada de um caso de uso e uma lista de casos de teste no qual o usuário fará uma análise dos casos a serem executados na etapa de execução. A Figura 7 ilustra essa etapa da fase.

Figura 8 - Seleção de casos de teste na fase de projeto.



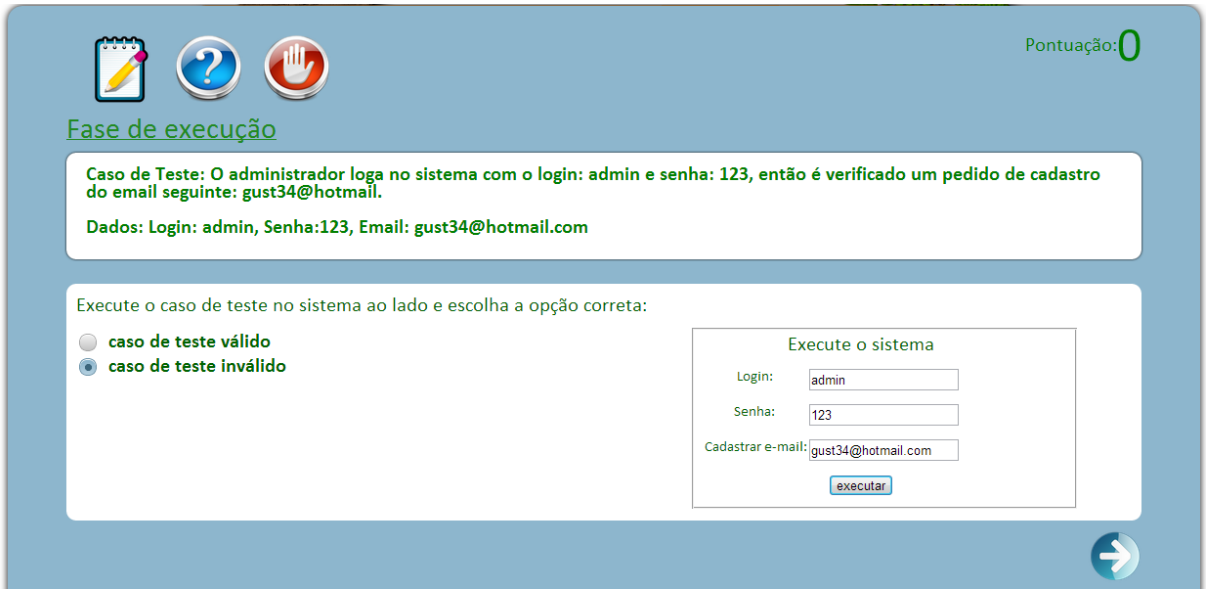
Fonte: BEZERRA e COUTINHO. (2013)

Os números de casos de uso variam de acordo com cada projeto já os casos de teste podem variar entre quatro e cinco casos de teste por caso de uso. O aluno deverá selecionar um número mínimo de casos de testes para prosseguir para a etapa de execução que varia dependendo do nível de dificuldade do projeto (no mínimo dois para o fácil, três para o médio e quatro para o difícil).

A tela da fase de execução, apresentada na Figura 8, possui dois componentes, as opções a serem selecionadas que podem ser casos de testes válidos e casos de testes inválidos

e o sistema onde deverá ser executado o sistema a partir de dados pré-fornecidos. Ao entrar com os dados no sistema e clicar “executar” o jogador deverá observar o comportamento do sistema e definir a opção correta.

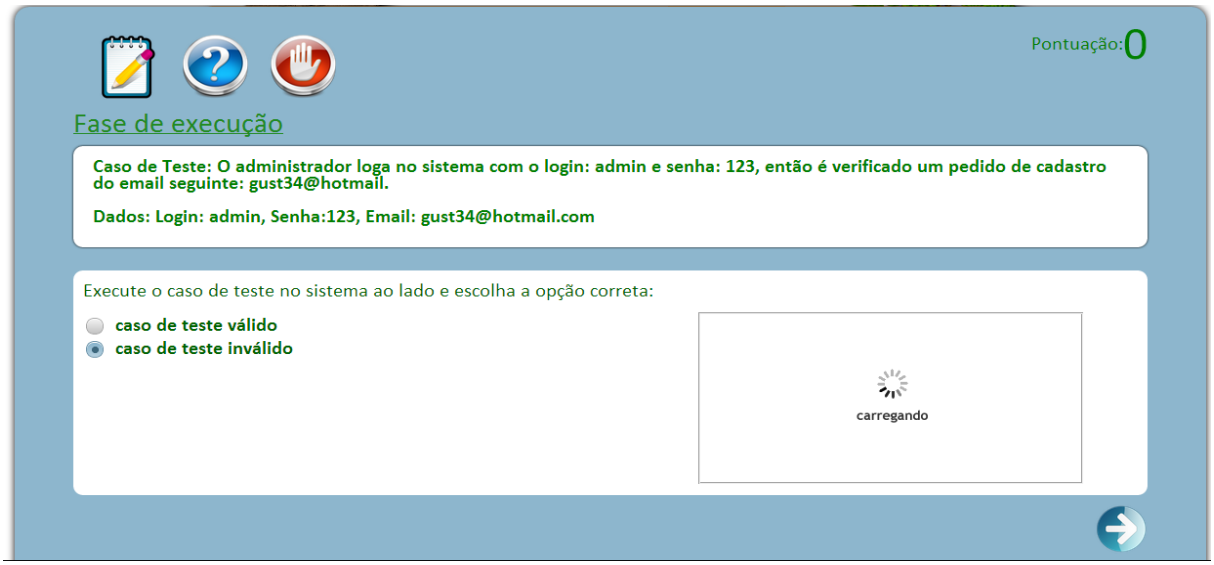
Figura 9 – Simulação do caso de teste na fase de execução.



Fonte: O autor

A execução irá simular o comportamento do sistema relacionado aos dados enviados e apresentará ao jogador as respostas sobre o evento que serão feitas através de alertas ou telas de finalização de processos. O aluno deverá observar essa resposta e marcar caso de teste válido quando o sistema simulado apresentar um comportamento correto ou caso de teste inválido quando apresentar um comportamento inesperado. A Figura 9 demonstra a funcionalidade sendo executada.

Figura 10 - Funcionalidade do caso de teste sendo executada.



Fonte: O autor.

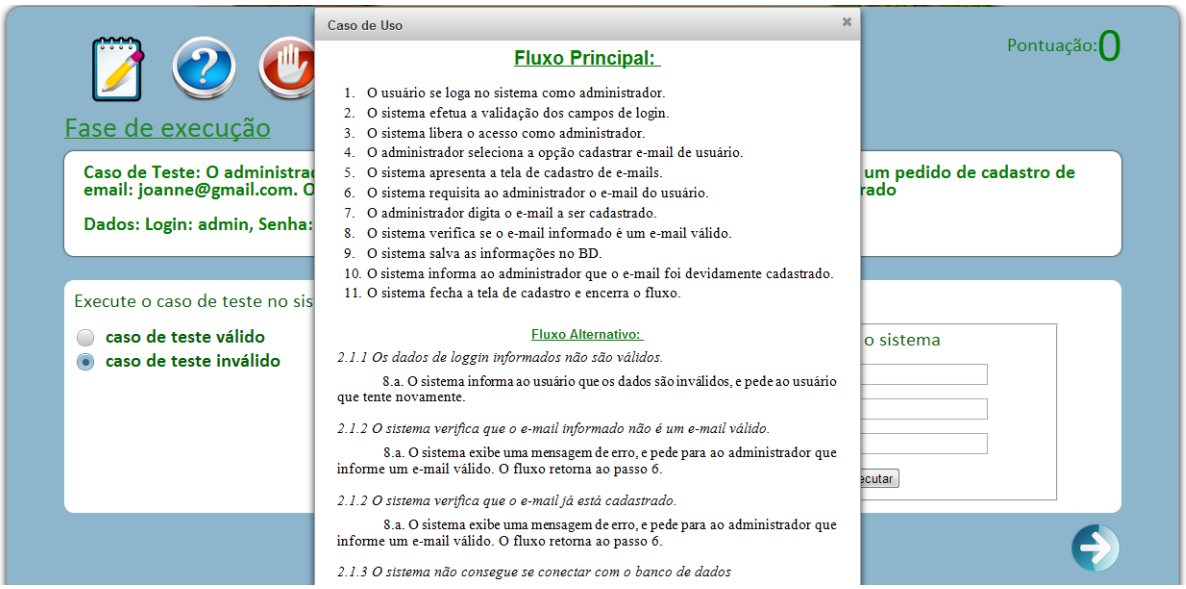
Após a execução o sistema retornará a resposta através de alertas, conforme mostrado na Figura 10, que mostrarão mensagens correspondentes às saídas do sistema em relação aos dados submetidos.

Figura 11 - Saída da execução do caso de teste.

Fonte: O autor

Durante a execução dessa etapa 3 ícones, demostrados na Figura 11, estarão disponíveis: caso de uso, que apresentará a descrição do caso de uso sempre que o jogador precisar reler para o melhor entendimento do caso de teste, dúvidas, onde terá descrições sobre o funcionamento da fase de execução, e encerrar o jogo.

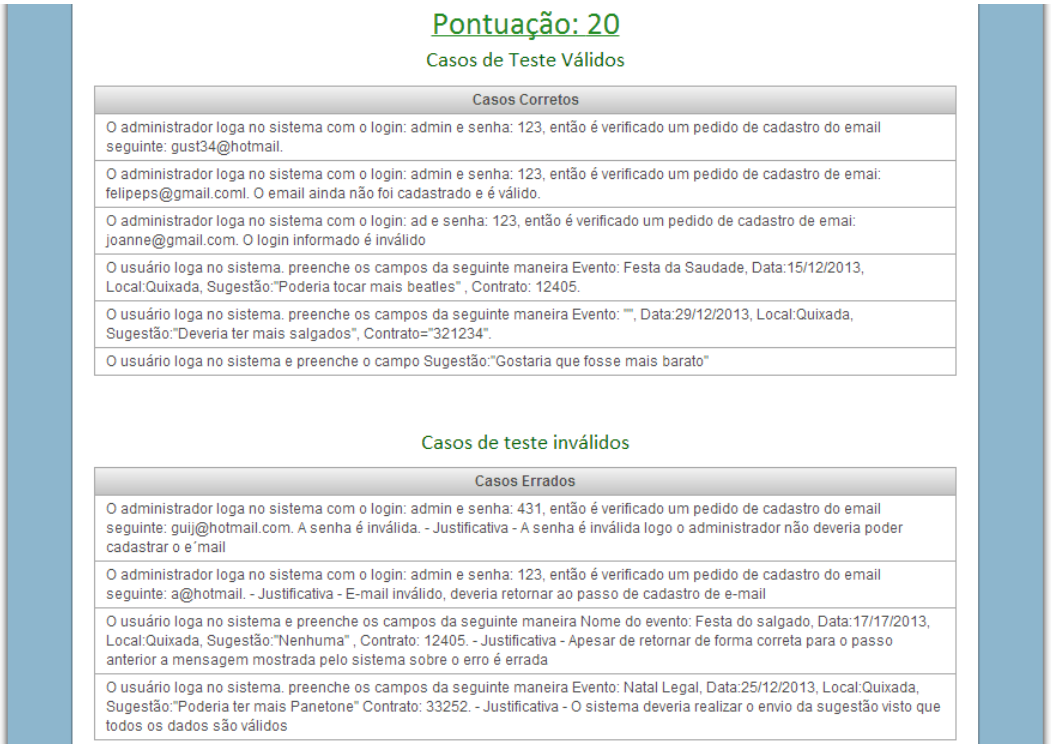
Figura 12- Tela de informação fornecida pelo ícone caso de teste.



Fonte: O autor

Ao final, uma tela de resultado dará um *feedback* ao jogador contendo todos casos de testes referentes ao projeto jogado, os mesmos estarão divididos em casos corretos (casos de teste válidos) e casos errados (casos de teste inválidos) que também mostrará uma justificativa para informar o motivo que o torna inválido. A tela final é demonstrada na Figura 12.

Figura 13 - Tela de *feedback* final da fase de projeto e execução.



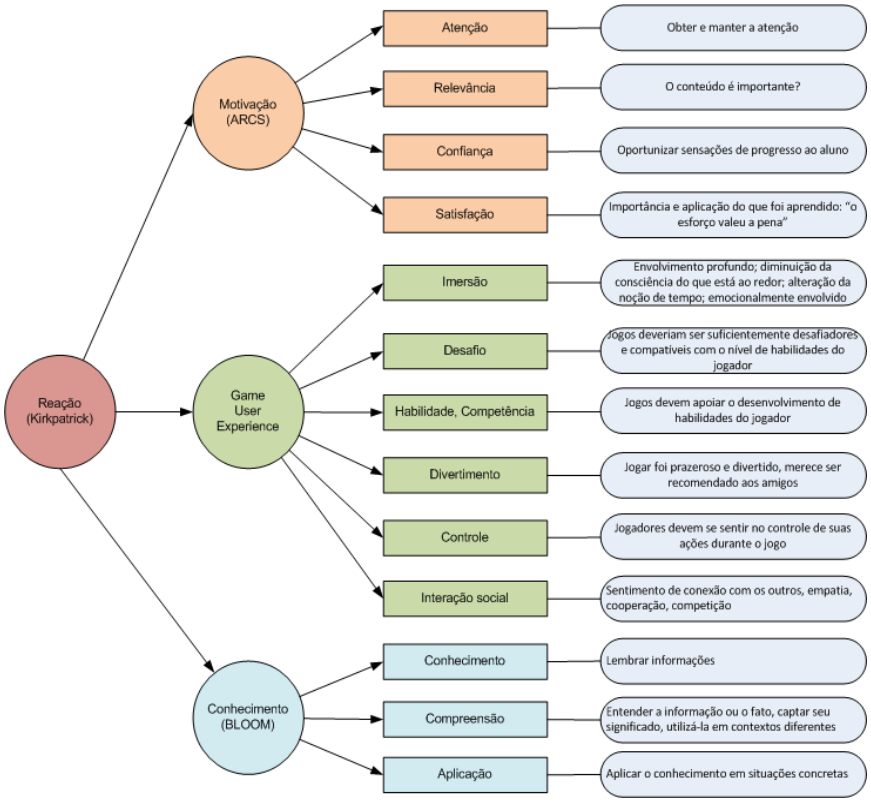
Fonte: O autor

5 AVALIAÇÃO DO MÓDULO DE EXECUÇÃO DO ITESTLEARNING

O modelo adotado para avaliação do módulo de fase de execução será a metodologia SAVI especialmente utilizada para avaliação de jogos educacionais. Esse modelo foi construído em três principais bases, são elas motivação (modelo ARCS), *game user experience* e conhecimento (*bloom*) onde cada uma delas abordam diferentes pontos que serão levados em consideração para a realização da avaliação (SAVI et al., 2011). O modelo pode ser melhor visualizado na figura 13.

Para que se possa alcançar a motivação durante o processo de aprendizagem serão avaliados os quatro pontos abordados pelo modelo ARCS. O primeiro, **Atenção**, refere-se às respostas cognitivas dos alunos aos estímulos instrucionais (HUANG; HUANG; TSCHOPP, 2010). A atenção é um elemento motivacional e também pré-requisito para aprendizagem. O desafio é obter e manter um nível satisfatório da atenção dos alunos ao longo de um período de aprendizagem (KELLER, 2009). O segundo ponto, **Relevância**, embora necessários, a atenção e curiosidade do aluno não são condições suficientes para a motivação aos estudos. O aluno também precisa perceber que a proposta educacional seja consistente com seus objetivos, que ele consiga conectar o conteúdo da aprendizagem com seu futuro profissional ou acadêmico. Quantas vezes vemos estudantes perguntarem “porque tenho que estudar isso?”. Caso não se apresente uma boa resposta, estamos diante de um problema de relevância (KELLER, 2009). Também representa o nível de associação que os alunos conseguem perceber entre seus conhecimentos prévios e as novas informações (HUANG; HUANG; TSCHOPP, 2010). O terceiro ponto, **Confiança**, está relacionado em criar expectativas positivas aos estudantes. Isso pode ser alcançado ao se proporcionar experiências de sucesso decorrentes de suas próprias habilidades e esforço. Este fator tem influência na persistência dos estudantes (KELLER, 2009; HUANG; HUANG; TSCHOPP, 2010). Por fim o ponto **Satisfação** aborda que os alunos precisam ter sentimentos positivos sobre a experiência de aprendizagem, e isso pode vir com recompensas e reconhecimento (SAVI et al, 2011).

Figura 14 - Modelo SAVI



Fonte: SAVI et al., 2011

Game user experience descreve pontos sobre a experiência do usuário em relação a interação como o jogo. São discutidos nesse conceito seis características. A primeira característica é a **Imersão** no qual bons jogos levam o jogador a ter uma experiência de profundo envolvimento no jogo, que geralmente provoca um desvio de foco do mundo real (SAVI et al., 2011). A **Interação social** é um elemento de diversão nos jogos com envolvimento com outras pessoas e está relacionado com o sentimento de compartilhar um ambiente com outras pessoas e de se ter um papel ativo nele (SAVI et al., 2011). **Desafio** que é um dos aspectos mais importantes nos bons jogos (SWEETSER; WYETH, 2005). Um jogo precisa ser suficientemente desafiador, estar adequado ao nível de habilidade do jogador e apresentar variações no nível de dificuldade (SAVI et al. 2011). O ponto **diversão** tem como requisito proporcionar sentimentos de diversão, prazer, relaxamento, distração e satisfação (POELS; KORT; IJSSELSTEIJN, 2007). O quinto ponto relaciona-se com o C**ontrole** e está relacionado com a sensação de independência, domínio (TAKATALO; HÄKKINEN; KAISTINEN, 2010) autonomia, poder e liberdade (POELS; KORT; IJSSELSTEIJN, 2007). Por fim a competência é uma medida combinada de habilidades do jogador e sentimentos positivos de eficiência. “Eu senti que poderia atender as demandas das situações do jogo” (TAKATALO; HÄKKINEN; KAISTINEN, 2010).

A terceira base do modelo consiste na taxonomia de *Bloom* que foi criada dentro de um contexto acadêmico na década de 1950 com o objetivo de apoiar os processos de projeto e avaliação educacional (CHAPMAN, 2009b). Essa taxonomia descreve seis conhecimentos que devem ser desenvolvidos pelos alunos: **Conhecimento** que consiste em lembrar informações sobre: fatos, datas, palavras, teorias, métodos, classificações, lugares, regras, critérios, procedimentos etc. **Compreensão** que visa entender a informação ou o fato, captar seu significado, utilizá-la em contextos diferentes. **Aplicação** do conhecimento em situações concretas. **Análise** das partes e suas inter-relações. **Síntese** das partes não organizadas para formar combinando-as para formar um todo e por fim **Avaliar** o valor do conhecimento.

A avaliação funcionará em cima dos três aspectos descritos (motivação, *game user experience* e conhecimento, esquematizados na figura 13) através da percepção do aluno, para isso o modelo desenvolveu um questionário que possui itens sobre variados aspectos dos parâmetros do modelo de avaliação que são descritos na forma de afirmações onde o aluno atribuirá uma escala que varia de discordo fortemente até concordo fortemente (SAVI et al, 2011).

Tabela 1 - Questionário do modelo de avaliação.

MOTIVAÇÃO **Atenção** - Houve algo interessante no início do jogo que capturou minha atenção. - O design da interface do jogo é atraente. **Relevância** - Ficou claro para mim como o conteúdo do jogo está relacionado com coisas que eu já sabia. - Eu gostei tanto do jogo que gostaria de aprender mais sobre o assunto abordado por ele. - O conteúdo do jogo é relevante para meus interesses. - Eu poderia relacionar o conteúdo do jogo com coisas que já vi, fiz ou pensei. - O conteúdo do jogo será útil para mim. **Confiança** - O jogo foi mais difícil de entender do que eu gostaria. - O jogo tinha tanta informação que foi difícil identificar e lembrar dos pontos importantes - O conteúdo do jogo é tão abstrato que foi difícil manter a atenção nele. - As atividades do jogo foram muito difíceis. - Eu não consegui entender uma boa parcela do material do jogo. **Satisfação** - Completar os exercícios do jogo me deu um sentimento de realização. - Eu aprendi algumas coisas com o jogo que foram surpreendentes ou inesperadas. - Os textos de feedback depois dos exercícios, ou outros comentário do jogo, me ajudaram a sentir recompensado pelo meu esforço. - Eu me senti bem ao completar o jogo. EXPERIÊNCIA DO USUÁRIO **Imersão** - Eu não percebi o tempo passar enquanto jogava. - Eu perdi a consciência do que estava ao meu redor enquanto jogava.	- Me senti mais no ambiente do jogo do que no mundo real. - Me esforcei para ter bons resultados no jogo. - Houve momentos em que eu queria desistir do jogo. - Me senti estimulado a aprender com o jogo. **Desafio** - Eu gostei do jogo e não me senti ansioso ou entediado. - O jogo me manteve motivado a continuar utilizando-o. -Minhas habilidades melhoraram gradualmente com a superação dos desafios - O jogo oferece novos desafios num ritmo apropriado. - Este jogo é adequadamente desafiador para mim, as tarefas não são muito fáceis nem muito difíceis. **Habilidade / Competência** - Me senti bem sucedido. - Eu alcancei rapidamente os objetivos do jogo. - Me senti competente. - Senti que estava tendo progresso durante o desenrolar do jogo. **Interação Social** - Senti que estava colaborando com outros colegas. - A colaboração no jogo ajuda a aprendizagem. - O jogo suporta a interação social entre os jogadores. **Divertimento** - Eu gosto de utilizar este jogo por bastante tempo. - Quando interrompido, fiquei desapontado que o jogo tinha acabado. - Eu jogaria este jogo novamente. - Algumas coisas do jogo me irritaram. - Fiquei torcendo para o jogo acabar logo. - Achei o jogo meio parado. CONHECIMENTO - Depois do jogo consigo lembrar de mais informações relacionadas ao tema apresentado no jogo. - Depois do jogo consigo compreender melhor os temas apresentados no jogo. - Depois do jogo sinto que consigo aplicar melhor os temas relacionados com o jogo.

Fonte: (SAVI et al., 2011).

O questionário utilizado para a avaliação da fase de execução da ferramenta foi adaptado do modelo SAVI, no qual foram retiradas as categorias de divertimento e interação social, pois não se adequam as características do módulo. O questionário utilizado encontra-se no anexo **A** onde pode ser visto a categoria de cada um dos itens.

5.1 Execução da Avaliação

A avaliação foi realizada com 11 alunos da disciplina de qualidade de software, apenas uma turma da disciplina estava disponível por isso a experiência foi centrada somente nos alunos dessa turma. Todos os envolvidos no experimento já tinham cursado a disciplina de engenharia de software e possuíam conhecimentos teóricos da área de teste de software. A

justificativa para a escolha desse perfil refere-se à necessidade do aluno já possuir conhecimentos sobre testes para a devida utilização da ferramenta.

Os alunos participaram de uma aula que teve como foco as fases de planejamento e execução do processo de teste de software como também os principais conceitos de casos de teste. Também foi realizada uma apresentação do jogo *iTestLearning* demostrando suas funcionalidades básicas, durante esse evento a professora orientadora não estava presente.

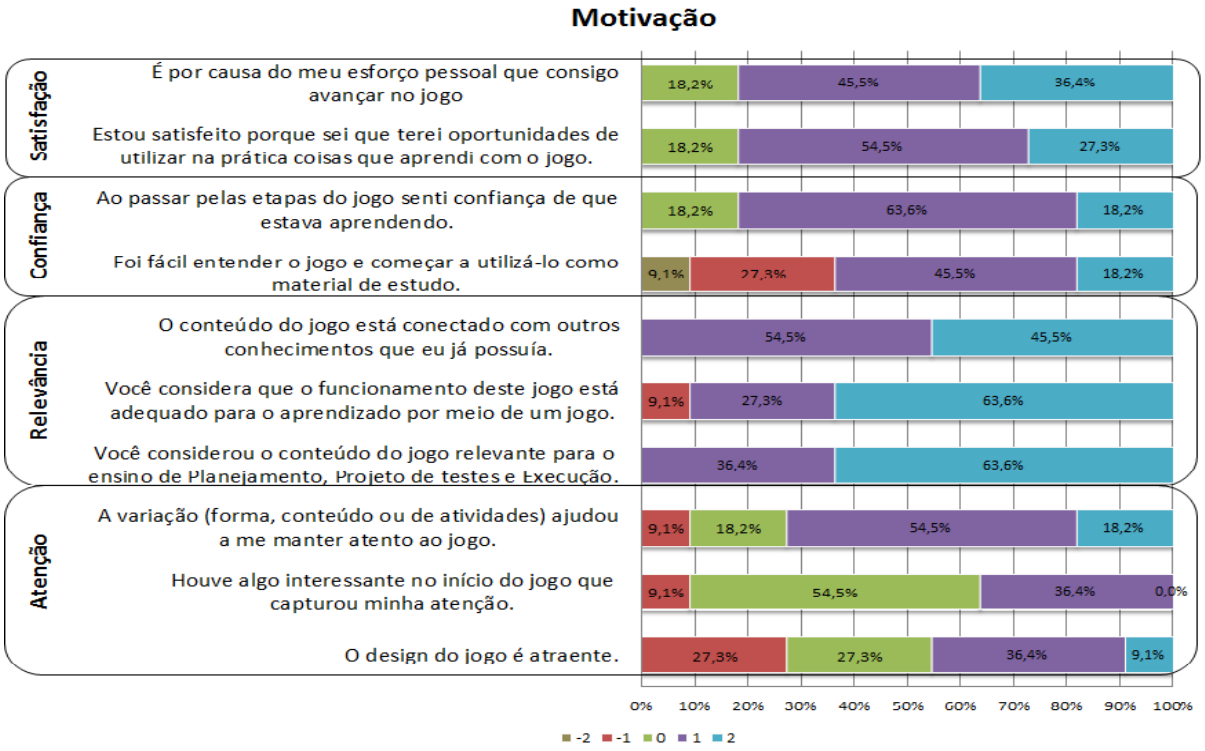
Após a finalização da aula os alunos foram submetidos à realização de práticas através do uso da ferramenta, então foram escolhidos dois projetos a serem jogados. O primeiro foi realizado em conjunto com a turma para que se tivesse uma explicação de como funcionava o jogo e os alunos pudessem tirar suas dúvidas em relação à dinâmica e funcionalidades básicas da ferramenta.

O segundo projeto escolhido foi jogado individualmente, a fase de planejamento foi descartada, visto que não era o foco do trabalho, então cada aluno executou todas as etapas da fase de projeto e execução do jogo *iTestLearning* e tiveram que explorar os recursos fornecidos pela ferramenta e completarem todos os níveis existentes do determinado projeto.

5.2 Análise dos Resultados

Os resultados coletados foram compilados e demonstrados através dos gráficos seguintes que demonstram o percentual entre as avaliações feitas pelos alunos entre os três pontos do modelo SAVI motivação, experiência do usuário e aprendizagem.

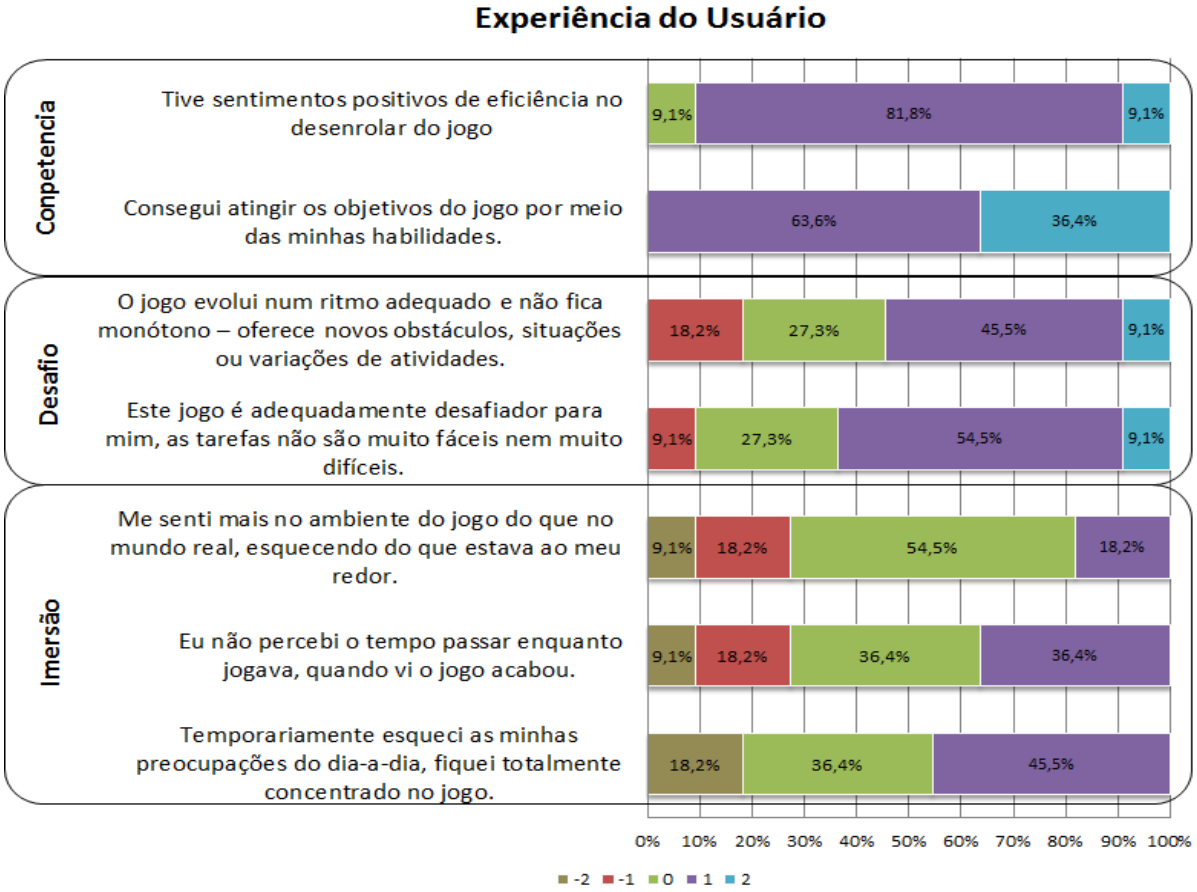
Figura 15 - Percentual das respostas do aspecto motivação.



Fonte: O autor.

O primeiro gráfico na Figura 14 representa a média das respostas em relação ao requisito motivação composto por 10 questões dentre eles a maioria dos itens tiveram suas maiores marcas porcentagens em respostas +1 e +2 o que revela que os alunos concordaram com a maioria dos pontos relacionados a essa categoria e indicando que o ponto motivação foi atingido com êxito. O aspecto relevância foi o que obteve maior êxito onde podemos ver que a grande maioria dos itens obteve respostas máximas entre +1 e +2. Entre os itens que devem ser melhorados podemos observar o ponto “Foi fácil entender o jogo e começar a utilizá-lo como material de estudo” do quesito confiança que teve 27,3% das respostas marcadas em -1 e 9,1% marcadas como -2 o que sugere que o jogo poderia ser mais intuitivo e também o ponto “design do jogo” que recebeu 27,3% das respostas em -1 o que mostra que o jogo necessita de uma melhora no conteúdo gráfico.

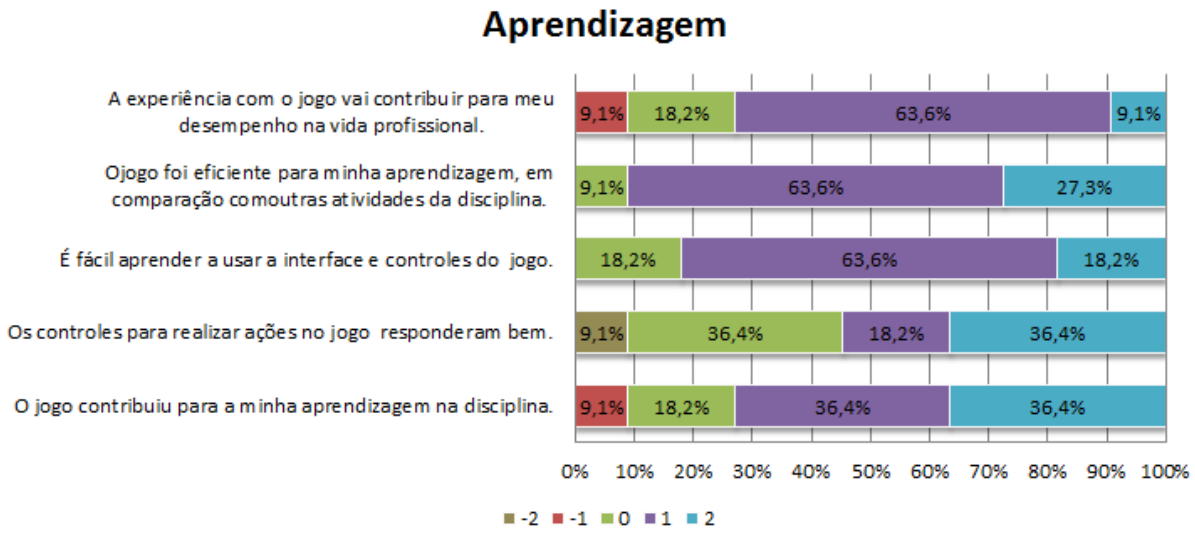
Figura 16 – Percentual das respostas do aspecto Experiência do Usuário.



Fonte: O autor.

O segundo gráfico na Figura 15 demonstra a experiência do usuário onde dos 7 itens avaliados obtivemos a maioria das respostas centralizadas nos pontos 0 e +1 o que mostra que conseguimos uma média satisfatória neste quesito. Os itens referentes à categoria competência foram o que demonstraram maior grau de aprovação principalmente do item "Consegui atingir os objetivos do jogo por meio das minhas habilidades" que obteve 63,6% das respostas marcadas em 1 e 36,4% em 2 o que indica que o jogo conseguiu passar ao usuário o sentimento de que ele realmente estava usando suas habilidades para passar de nível. É possível também afirmarmos que o jogo se mostrou eficiente nos quesitos desafio com uma ressalva apenas no item "O jogo evolui num ritmo adequado e não fica monótono" que necessita de uma melhoria para tornar o jogo mais dinâmico. O quesito que demonstrou menor eficiência foi o de imersão, no qual podemos ver que as respostas entre -1 e -2 compuseram entre 20% e 30% dos itens o que revela que esse quesito precisa ser melhor trabalhado para que possa se criar um ambiente que prenda melhor a atenção dos jogadores.

Figura 17 - Percentual das respostas do aspecto Aprendizagem.



Fonte: O autor.

O terceiro gráfico na Figura 16 demonstra a categoria aprendizado onde dos 5 itens avaliados obtivemos a maioria das reposta centralizadas no ponto +1 demonstrando a eficiência da ferramenta no ensino e aprendizagem de teste de software. Os itens “O jogo foi eficiente para minha aprendizagem, em comparação com outras atividades da disciplina” e “É fácil aprender a usar a interface e controles do jogo” obtiveram mais de 70% das suas repostas centralizadas em +1 e +2 indicando que o jogo está alinhado com seu objetivo de apoiar o ensino de teste de software e de facilidade de uso da ferramenta. Apenas o item “Os controles para realizar ações no jogo responderam bem” obteve respostas no ponto -2, o que sugere a necessidade de pequenos ajustes para que os controles funcionem de maneira eficiente.

5.3 Análise dos Pontos Fortes e Pontos de Melhoria

Este tópico tem como objetivo mostrar os pontos fortes e pontos de melhoria sugeridos pelos alunos que participaram da atividade, estes estão dispostos na Tabela 2.

Sobre os pontos fortes do jogo podemos destacar que, segundo os alunos, o módulo de Execução do *iTestLearning* contribui de forma positiva para o entendimento do processo de teste, os passos do jogo ficaram bem entendidos pelos que realizaram e principalmente pela facilidade da utilização do entendimento e manuseio das funcionalidades da ferramenta. Outro ponto elogiado pelos alunos foram o conteúdo que o jogo possui e a interação com o sistema.

Como pontos negativos entre os comentários destacam-se a necessidade de uma melhor elaboração do *design* do jogo para torná-lo mais atraente, fato também descrito no

tópico 5.1, e também se faz necessário à reescrita de alguns casos de teste apresentados para que possa ficar mais claro o que se deve esperar da saída do sistema.

Tabela 2 - Comentários dos alunos sobre pontos fortes e melhorias do jogo.

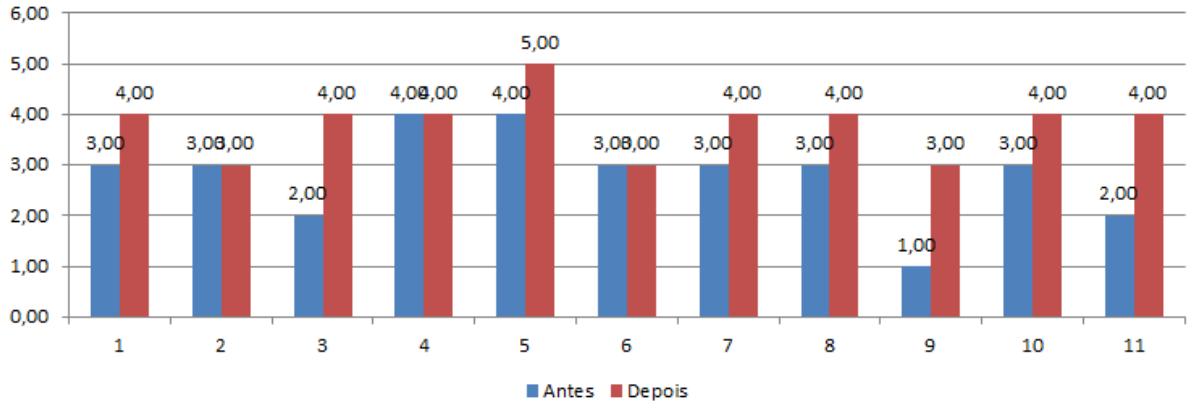
Alunos	Pontos fortes do jogo	Sugestão de melhorias
1	Aborda um tema muito importante na minha graduação, Ajuda a entender melhor o conteúdo.	Os textos dos casos de teste ficaram um pouco confusos. Algumas vezes ficou confuso saber qual seria o resultado final do sistema.
2	Passos bem explicados.	
3	A metodologia usada para fazer as perguntas são bem intuitivas e prazerosas de responder.	
4	O jogo torna-se interessante pelo fato de colocarmos valores e analisar se o sistema está retornando o esperado ou se está se comportando de maneira inesperada. Essa interação é um ponto forte da execução do jogo. Consegui identificar bem os casos, e responder corretamente as questões.	A especificação nos casos de teste não ficou muito clara. Faltou especificar os campos que realmente são válidos e os nãos válidos de maneira melhor.
5	Boa explicação nos casos de uso. Fácil entendimento da interface	
6	Linguagem simples, fácil manuseio.	
7	Linguagem simples. Interface gráfica que contribui para o manuseio do jogo. Abordagem bastante relevante das questões.	Melhor formatação das opções que estão em uma mesma linha. Separar as informações por campos.
8	De fácil manuseio e objetivo.	Melhorias quanto aos textos das questões. Os textos podem gerar confusão, não deixando claras as informações necessárias.
9	O conteúdo bem elaborado.	Interface gráfica e facilitar o início do jogo em relação como o jogo funcionará.
10	Como é um jogo atrai mais atenção de quem esta jogando e estudando do que a quem esta somente estudando.	Poderia melhorar a interface e deixar mais claro qual o objetivo do jogo.
11	Aumento no conhecimento em relação a casos de testes para a melhoria da qualidade do software.	

5.4 Análise da percepção de Aprendizagem

Para analisar o ganho com relação à percepção de aprendizagem dos alunos foram elaboradas duas perguntas com o objetivo de medir quanto os alunos julgaram ter compreendido melhor o assunto em relação ao conhecimento que eles possuíam antes do uso da ferramenta. Para isso, foi utilizada uma escala que varia de 1 a 5 no qual o aluno deveria selecionar o nível de compreensão que ele possuía sobre um determinado assunto antes e depois de usar a ferramenta.

O gráfico demonstrado na Figura 17 tem como objetivo demonstrar o nível de aprendizagem dos participantes. Para os resultados desse gráfico foram perguntados aos alunos o nível de aprendizagem sobre o conhecimento dos conceitos e funcionalidades dos casos de testes antes e depois do uso da ferramenta.

Figura 18 – Percepção de aprendizagem sobre o conhecimento de casos de teste

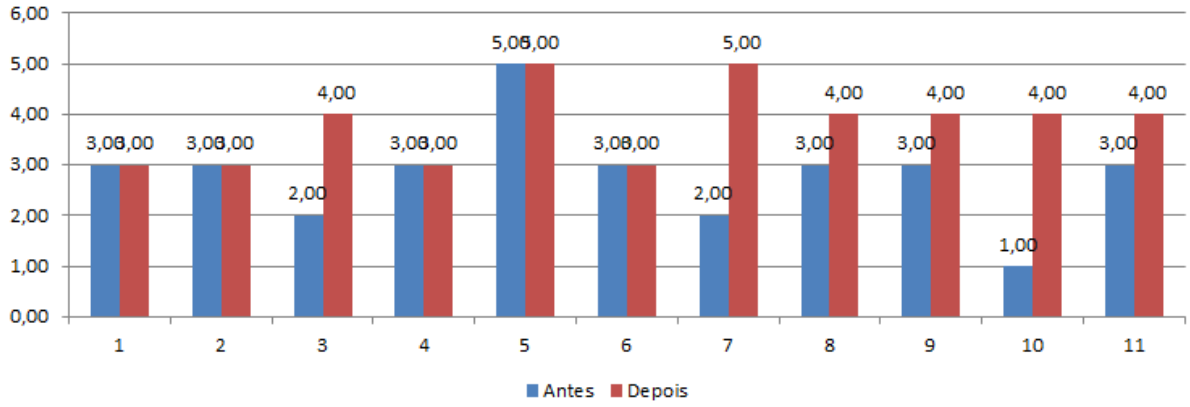


Fonte: O autor.

Entre os 11 alunos, 8 demonstraram que obtiveram um aumento do conhecimento individual sobre casos de teste enquanto 3 demonstraram que não obtiveram ganhos significativos. Por isso, pode-se inferir que 72,7% dos alunos conseguiram ter uma maior compreensão sobre o conceito de casos de teste e suas utilidades.

O gráfico ilustrado na Figura 18 tem como objetivo demonstrar o nível de aprendizagem dos alunos em relação ao conhecimento individual sobre o processo de execução de testes antes e depois do uso da ferramenta.

Figura 19 – Percepção de aprendizagem sobre execução de testes.



Fonte: O autor.

Com relação ao nível de aprendizado sobre execução de casos de teste dos 11 alunos 6 afirmaram que o jogo trouxe um aumento da compreensão em relação as atividades

de teste de software o que representa 54,5% dos alunos tiveram um aumento em média de 2 pontos, entre eles o aluno 10 foi o que registou a melhor marca com um aumento de 3 pontos.

Pelos dois gráficos podemos verificar que o jogo contribuiu de forma significativa em relação à percepção do aprendizado individual de cada aluno em relação a casos de teste e do processo de execução o que indica que o jogo pode ser utilizado como uma ferramenta de estudo de testes de software.

6 CONSIDERAÇÕES FINAIS

O processo de teste de software é uma das mais importantes atividades no processo de desenvolvimento de software, atividade no qual é apresentada de maneira quase sempre de maneira teórica nos cursos de TI. A necessidade da prática para que se tenha um melhor aproveitamento no aprendizado e uma maior absorção do conteúdo pelo estudante em relação ao processo de teste de software é a principal motivação desse trabalho que teve como foco o desenvolvimento da fase de execução da ferramenta *iTestLearning.*

Para se alcançar os objetivos foi necessário realizar uma revisão bibliográfica relacionada aos principais conceitos de teste de software focado principalmente nos teste funcionais e no processo de teste de software. Outro ponto fundamental foram as pesquisas sobre jogos educacionais e como eles podem ser utilizados para contribuir com o ensino de engenharia de software. Os trabalhos relacionados utilizados 7 Falhas (DINIZ e DAZZI, 2011), U-Test (SILVA, 2010) e BUG HUNT (ELBAUM et al., 2007), foram de fundamental importância para a criação desse projeto, pois através deles foi possível tirar as principais ideias para a construção da fase de execução.

A metodologia adotada foi seguida de maneira eficiente onde todos os passos foram realizados desde o levantamento de requisitos, construção do *game design*, desenvolvimento e avaliação. Dentre os pontos da metodologia, a avaliação do módulo foi o passo mais interessante realizado, pois a interação com os alunos foi bastante importante para a conclusão do objetivo e também como uma fonte para a inspiração para trabalhos futuros.

Entre as dificuldades do trabalho podemos destacar a utilização de apenas uma turma para a realização da avaliação o que demostra a utilização de um grupo pequeno de alunos para a execução da experiência, por esse fato não podemos afirmar que a aprendizagem do conteúdo através do jogo é completamente efetiva, mas pelos resultados da avaliação podemos concluir ferramenta tem potencial para a contribuição nesse processo. Outro ponto que podemos destacar é que durante a avaliação a professora orientadora não estava presente o que beneficiou a conclusão do experimento pelo motivo que sem a presença da professora os alunos sentiram-se mais a vontade para realizar critica e sugestões ao jogo. A realização do questionário também foi outro ponto a destacar, no qual os alunos tiveram liberdade para respondê-lo fora da sala de aula e sem supervisão o que representou uma maior liberdade e credibilidade nas respostas.

Através desse trabalho podemos concluir que o projeto conseguiu obter êxito nos objetivos a serem alcançados, como podemos analisar pelo tópico 5.1 o jogo conseguiu marcar bons índices de aceitação dos alunos nos critérios do modelo SAVI. O tópico 5.3 indica também que o jogo contribuiu de maneira significativa para o aprendizado dos alunos que participaram do experimento sobre o processo de execução de teste.

Entre os pontos negativos podemos citar a necessidade de uma maior atenção ao *design* do jogo para torná-lo mais atrativo e proporcionar um ambiente que crie uma maior imersão ao jogador, visto que esse foi o ponto que mais recebeu críticas. Esse fato deve-se ao curto espaço de tempo para o desenvolvimento da fase no qual as funcionalidades foram priorizadas em relação ao design.

Como trabalhos futuros, deve-se ter uma maior dedicação ao conteúdo geral do jogo com a criação de animações, personagens, enredos, entre outros. Através da inserção desses elementos teremos uma maior *gameficação* do jogo o que o tornará mais completo e conseguirá criar um ambiente mais interessante atraindo assim um maior numero de usuários que irão usar a ferramenta como um material de estudo.

REFERÊNCIAS

ABNT. NBR ISO 8402. **Gestão da qualidade e garantia da qualidade** – terminologia. 1994.

ACM/IEEE. **Computer Science Curriculum**, 2008.

ANSI/IEEE Standard 729-1983, **Institute of Electrical and Electronics Engineers**, New York, 1983.

BARTIE, A. **Processo de Teste de Software,** iMasters, 2007, Disponível em: <http://imasters.com.br/artigo/6102/software/processo-de-teste-de-software-parte-01>. Acesso em 28/05/2013;

BARTIE, A. **Garantia da Qualidade de Software: Adquirindo Maturidade Organizacional**. Rio de Janeiro: Elsevier, 2002 - 9ª Reimpressão.

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

BENITTI, F. B. V.; MOLLÉRI, J. S. **Utilização de um RPG no Ensino de Gerenciamento e Processo de Desenvolvimento de Software**. Anais do XXVIII Congresso da SBC – 2008.

BELHOT, R. V.; FIGUEIREDO, R. S.; & MALAVÉ, C. O. **O uso da simulação no ensino de engenharia. In Congresso Brasileiro de Ensino de Engenharia**, XXIX COBENGE (pp. 445-451), 2001.

BEIZER, B. **Software Testing Techniques**. 2 ed. New York: Van Nostrand Reinhold, 1990.

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

BEZERRA, C.; COUTINHO, E. **Avaliação do Jogo iTestLearning: Um Jogo para o Ensino de Planejamento de Testes de Software**, 2013.

CHAPMAN, Alan**. Bloom's taxonomy - learning domains. Businessballs**, 2009. Disponível em: <http://www.businessballs.com/bloomstaxonomyoflearningdomains.htm>. Acesso em: novembro. 2013.

CRESPO, A. N.; SILVA, O. J.; BORGES, C. A., SALVIANO, C. F.; TEIVE, M.; JUNIOR, A.; JINO, M. **Uma metodologia para teste de software no contexto da melhoria de processo. Simpósio Brasileiro de Qualidade de Software**, 2004.

DIAS, C. **Teste de Software**. UDESC, Joinville, 2011.

DINIZ, L.; DAZZI, L. S**. Jogo para o Apoio ao Ensino do Teste de Caixa-Preta. Anais do Simpósio Brasileiro de Informática na Educação**. Vol. 1. No. 1. 2011.

ELBAUM, S et al., **Bug hunt: Making early software testing lessons engaging and affordable. Software Engineering**, 2007. ICSE 2007. 29th International Conference on. IEEE, 2007.

FARIAS, V.; MOREIRA, C.; COUTINHO, E.; SANTOS, I. S. "**iTestLearning: Um Jogo para o Ensino do Planejamento de Testes de Software.**" Fórum Educacional de Engenharia de Software, Natal - RN. 2012.

FERNANDES, L.; WERNER C. M. L**. Sobre o uso de Jogos Digitais para o Ensino de Engenharia de Software. Simpósio Brasileiro de Engenharia de Software**, 2009.

FREITAS, F.; MOREIRA, C.; COUTINHO, E.; SANTOS, I. S.; FERREIRA M. S.; PORDEUS, P. I. A.; SILVA, P. J. S. "**Ensino de Projeto de Testes de Software: Uma Evolução do Jogo iTestLearning**" Encontros Universitários UFC, Quixadá-CE. 2012.

GIBBS, W. **Software's chronic crisis**. Scientific American 271 3 (1994), pp. 86–95.
GNATZ, M.; KOF. L.; PRILMEIER, F.; SEIFERT, T**. A Practical Approach of Teaching Software Engineering**, Proc. 16th Conf. Software Eng. Education and Training, pp. 120–128, 2003.

GOOLD, A.; HORAN, P. **Foundation software engineering practices for capstone projects and beyond. Proc. 15th Conference on Software Engineering Education and Training**, IEEE CS Press, pp 140-146, 2002.
HOWDEN, W. E. **Software Engineering and Technology: Functional Program Testing and Analysis**. McGrall-Hill Book Co, New York, 1987.

HUANG, W.; HUANG, W.; TSCHOPP, J. **Sustaining iterative game playing processes in DGBL: The relationship between motivational processing and outcome processing**. Comput. Educ., v. 55, n. 2, p. 789-797, 2010.

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

KELLER, J. M. **Motivational Design for Learning and Performance: The ARCS Model Approach**. Springer, 2009.

KANER, C. **Improving the maintainability of automated test suites, Proceedings of the International Quality Week**, 1997

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

MALDONADO, J. C.; BARBOSA, E. F.; VICENZI, A. M. R.; DELAMARO, M. E.; SOUZA, S. R. S.; JINO, M. **Introdução ao teste de software**. São Carlos, 2004.

MORAES.; RENATO.; LAURINDO,F. J. B**. Teste De Software e Qualidade De Software: Uma Visão Geral**, 2008.

PRENSKY, M. **Digital Game-Based Learning**, McGraw-Hill, 2001.

PRESSMAN, R. S. (2011), **Engenharia de Software: Uma abordagem profissional**, 7° ed. McGraw-Hill.

POELS, K.; KORT, Y. D.; IJSSELSTEIJN, W. **"It is always a lot of fun!": exploring dimensions of digital game experience using focus group methodology** In: PROCEEDINGS OF THE 2007 CONFERENCE ON FUTURE PLAY. Toronto, Canada: ACM, 2007.p.83-89.

RUBIN, J. **Handbook of Usability Testing: How to Plan, Design and Conduct Effective Tests**. New York: John Wiley & Sons, Inc., 1994. 330 p.

SAVI, R.; WANGENHEIN, C., BORGATTO, A. (2011). "**Um Modelo de Avaliação de Jogos Educacionais na Engenharia de Software**". Anais do XXV Simpósio Brasileiro de Engenharia de Software (SBES 2011), São Paulo.

SILVA, A. C. **Jogo educacional para apoiar o ensino de técnicas para elaboração de testes de unidade**. Diss. Dissertação (Mestrado), Univali, São José, 2010.

SOMMERVILLE, I. **Engenharia de Software**, *9º ed*. Pearson Addison-Wesley, São Paulo, 2011.

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.

SWEETSER, P.; WYETH, P. **GameFlow: a model for evaluating player enjoyment in games**. Comput. Entertain., , v. 3, n. 3, p. 3-3, 2005.

TAKATALO, J.; HÄKKINEN, J.; KAISTINEN, J.; Nyman, G. Presence, Involvement, and Flow in Digital Games. In: BERNHAUPT, R. **Evaluating User Experience in Games: Concepts and Methods**. Springer, 2010, p, 23-46.

TOZELLI, P.; **Técnicas de teste**. Processo de Teste de Software, 2008.

WANGENHEIN, C. G.; SILVA, D. A. **Qual conhecimento de engenharia de software é importante para um profissional de software?** In Anais do Fórum de Educação em Engenharia de Software, Fortaleza, 2009.

WANGENHEIN, C. G. V.; SHULL, F. **To Game or Not to Game**? IEEE Software, 26 (2), pp. 92-94, 2009.

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 – Definição dos requisitos do jogo

Casos de usos da fase de execução

1. O JOGADOR SELECIONA OS CASOS DE TESTE

Breve Descrição

O jogador realizou a fase de planejamento e iniciou a fase de projeto e agora deverá realizar uma seleção dos casos de teste a serem executados para um determinado caso de uso.

Fluxo de Eventos

Fluxo Básico

1. O jogador inicia a fase de projeto.

2. O sistema exibe uma lista com os casos de testes.

3. O jogador seleciona quais casos de teste devem ser executados.

4. O jogador avança para a etapa de execução.

Fluxos Alternativos

1. O jogador encerra o jogo.

Requisitos Especiais

Requisito de Desempenho

O tempo de resposta de cada tela do jogo deve ser de no máximo 3 (três) segundos.

Condições Prévias

4.1 O jogador deve ter escolhido um projeto para jogar.

4.2 O jogador deve ter definido o nível de dificuldade.

4.3 O jogador deverá obter êxito na fase de Planejamento.

Condições Posteriores

5.1 O jogador inicia a etapa de execução.

2. O JOGADOR REALIZA A EXECUÇÃO DOS CASOS DE TESTE

Breve Descrição

O jogador selecionou os casos de teste e inicializou a fase de execução e agora irá executa-los em um ambiente funcional.

Fluxo de Eventos

Fluxo Básico

1. O sistema exibe uma tela com o caso de teste a ser executado.

2. O jogador executa o caso de teste em um sistema funcional.

3. O Sistema irá simular a execução do processo.

4. O usuário deverá selecionar se o caso de teste é valido ou inválido.

5. O jogador avança para o próximo caso de teste.

Fluxos Alternativos

1. O jogador encerra o jogo.

Requisitos Especiais

Requisito de Desempenho

O tempo de resposta de cada tela do jogo deve ser de no máximo 3 (três) segundos.

Condições Prévias

4.1 O jogador deve ter escolhido um projeto para jogar.

4.2 O jogador deve ter definido o nível de dificuldade.

4.3 O jogador deverá obter êxito na fase de Planejamento.

4.4 O jogador selecionou os casos de teste.

4.5 O jogador iniciou a fase de execução.

Condições Posteriores

5.1 O Sistema exibe uma nova lista para selecionar casos de teste para o próximo caso de uso

5.2 O jogador termina a fase de execução

3. O JOGADOR TERMINA DE EXECUTAR OS TESTES SELECIONADOS.

Breve Descrição

O jogador selecionou e executou todos os casos de teste referentes aos casos de uso do projeto escolhido.

Fluxo de Eventos

Fluxo Básico

1. O jogador termina a fase de execução.

2. O sistema verifica que não há mais casos de testes a serem selecionados.

3. O Jogador avança para o *feedback* final.

4. O sistema exibe o *feedback* e a pontuação final do jogador.

5. O jogador confere seu desempenho e salva sua pontuação.

Fluxos Alternativos

1. O jogador encerra o jogo.

2. O jogador confere o ranking.

Requisitos Especiais

Requisito de Desempenho

O tempo de resposta de cada tela do jogo deve ser de no máximo 3 (três) segundos.

Condições Prévias

4.1 O jogador deve ter escolhido um projeto para jogar.

4.2 O jogador deve ter definido o nível de dificuldade.

4.3 O jogador deverá obter êxito na fase de Planejamento.

4.4 O jogador selecionou os casos de teste.

4.5 O jogador iniciou a fase de execução.

4.6 O jogador não possui mais casos de testes a serem selecionados.

Condições Posteriores

5.1 O sistema encerra o jogo.

APÊNDICE B – Definição dos requisitos do jogo

Game Design

O jogo *iTestLearning* é um jogo *single-player* do gênero educacional/simulação que tem como público alvo estudantes do curso de computação, engenharia de software e sistemas de informação. O objetivo do jogo é auxiliar o ensino de teste de software simulando a elaboração das fases de planejamento, projeto e execução de testes de software.

A fase de execução do jogo seguirá os preceitos citados a cima e a os demais pontos serão descritos a seguir.

1. Objetivo da Fase

O objetivo principal da fase será a execução dos casos de teste propostos na fase anterior (fase de projeto) em um ambiente que realizará a simulação de um projeto de forma funcional onde será possível realizar os testes de maneira interativa.

2. *Gameplay*

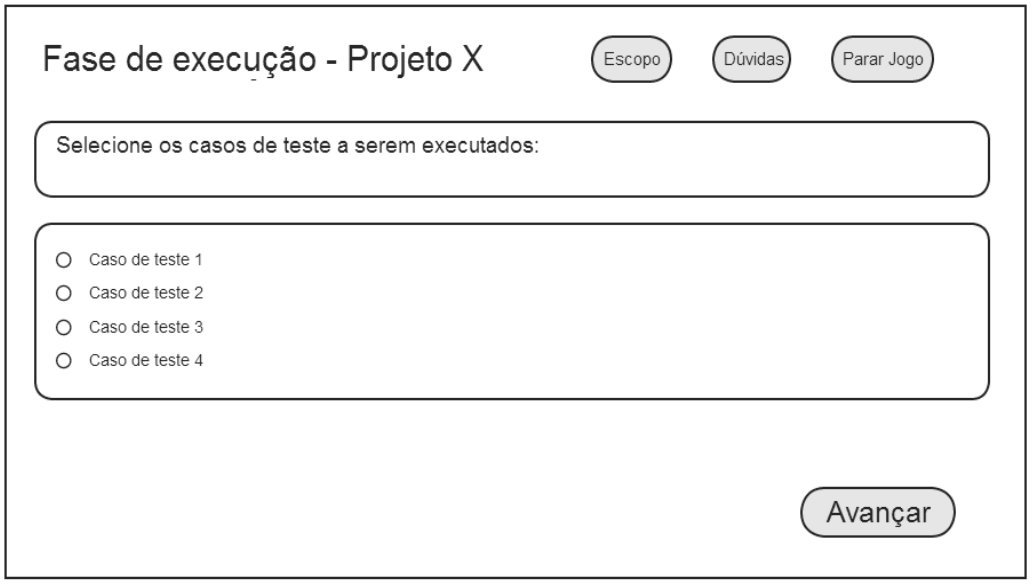
Após o êxito na fase anterior o jogador poderá ter acesso à fase de execução, durante essa etapa o sistema funcionará com duas telas. A primeira apresentará a fase de projeto no qual o usuário deverá selecionar os casos de testes que irão ser executados. Após a seleção, em uma nova tela será mostrado o sistema (escolhido pelo usuário ao inicio do jogo) e a simulação do processo que executará o caso de teste. O jogador nesse momento irá realizar a execução do caso de teste no qual o usuário deverá entrar com os dados fornecidos e executar o sistema que de acordo com a sua resposta o usuário deverá selecionar entre caso de teste válido e inválido. Esse processo será seguido até o fim da lista de casos de teste selecionados. Durante o procedimento da fase será necessário o preenchimento dos campos obrigatórios caso contrário o sistema não irá permitir um avanço. A pontuação seguirá o mesmo padrão já definidos nas outras etapas do jogo, em níveis fáceis acertos valem 10 pontos e erros descontam 5 pontos, em níveis médios e difíceis os acertos também geram 10 pontos ao jogador e erros descontam 7 e 10 pontos respectivamente.

3. Design da Fase e Mecânica.

Como descrito anteriormente o fase possuirá duas telas a primeira com o objetivo de selecionar o s casos de testes (fase de projeto) a serem testados e a segunda dará inicio a fase de execução onde será simulado o sistema funcional do projeto escolhido pelo jogador.

O protótipo de primeira tela pode ser visto na figura 20.

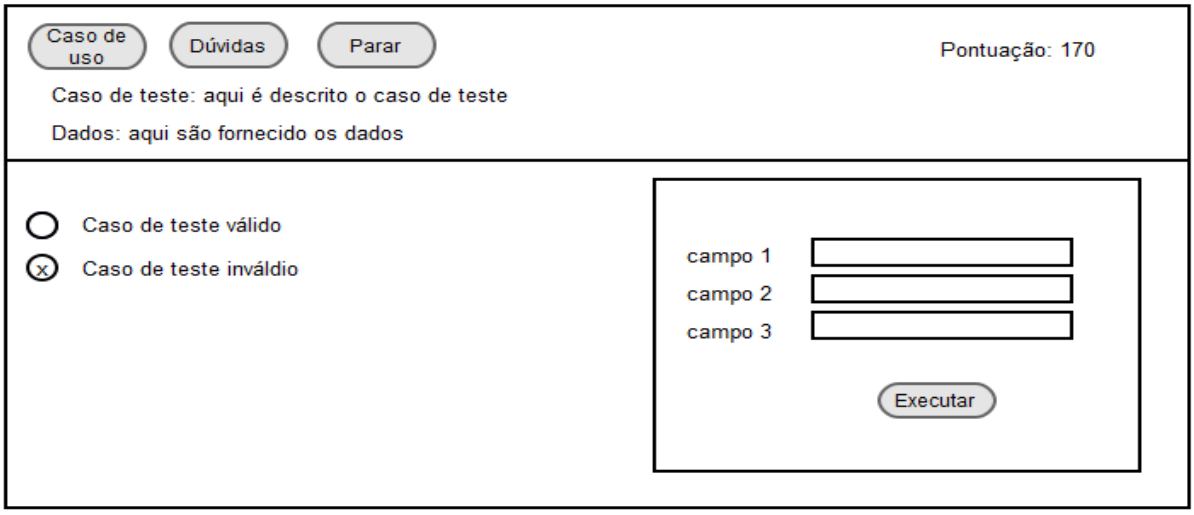
Figura 20 - Seleção de casos de teste a serem executados



Fonte: o autor.

Antes do início da fase de execução o sistema apresentará uma lista onde serão selecionados os casos de teste que irão ser executados referentes a um caso de uso (nível fácil), estórias de usuário (nível médio) ou requisitos (nível difícil) que são definidos na fase de projeto. O jogador terá as opções de escopo, dúvidas e parar jogo. Ao selecionar o item "escopo" será possível a visualização da descrição do projeto. O ícone de "dúvidas" irá mostrar as principais informações referentes à fase de execução. Por fim o ícone de parar jogo irá encerrar a sessão, não será possível salvar a pontuação do usuário ao escolher essa opção. Ao selecionar os casos de teste o jogador então poderá começar a executa-los ao clicar no botão "avançar".

Figura 21 - etapa de execução



Fonte: o autor.

Houve uma modificação em relação ao protótipo da fase de execução (figura 21) no qual os logs foram substituídos pelas opções casos de teste válido e válidos. O protótipo de tela da etapa de execução pode ser vista na figura 21. Após selecionar os casos de teste e avançar, dará inicio a fase de execução. No canto superior esquerdo será possível visualizar o caso de uso referente à execução, o caso de teste que será utilizado para orientar e os dados a serem utilizados na execução. A tela direita representa uma simulação do sistema, nessa tela será onde jogador irá realizar a execução do caso de teste. Os demais ícones presentes na tela terão as mesmas funções já descritas anteriormente.

Após executar todos os casos de teste relacionados ao caso de uso o sistema retornará a tela da figura 20 para que sejam feitas uma nova seleção de casos de teste para a execução do próximo caso de uso definido.

Ao finalizar a realização da fase será feito o encerramento geral do jogo mostrando ao usuário sua pontuação total e um *feedback* geral de seu desempenho. O jogo então irá realizar o salvamento da pontuação e o jogo poderá ser encerrado.

4. Documentação Técnica

A fase de execução do jogo *iTestLearning* é desenvolvida na linguagem utilizada como padrão do jogo JAVA juntamente com a utilização do *framework* JSF 2.0 para o desenvolvimento das telas.

Os requisitos mínimos para o jogo são:

- Conexão com internet mínima de 56kbp/s.
- Browsers compatíveis: Internet Explorer 9, ou superior / Mozzila Firefox / Chrome / Safari / Opera.
- Instalação do JRE.

ANEXOS

ANEXO A – Avaliação do módulo de execução

O questionário utilizado possui 22 itens divididos entre as categorias do modelo SAVI, cada um dos itens foi avaliado pelos alunos através de uma escala que varia entre +2 (forte concordância) e -2 (forte discordância). A tabela abaixo mostra os itens avaliados e suas categorias correspondentes:

Tabela 3 - Questionário utilizado

1	O design do jogo é atraente.	Atenção	
2	Houve algo interessante no início do jogo que capturou minha atenção.	Atenção	
3	A variação (forma, conteúdo ou de atividades) ajudou a me manter atento ao jogo.	Atenção	
4	Você considerou o conteúdo do jogo relevante para o ensino de Planejamento e Projeto de testes.	Relevância	
5	Você considera que o funcionamento deste jogo está adequado para o aprendizado por meio de um jogo.	Relevância	
6	O conteúdo do jogo está conectado com outros conhecimentos que eu já possuía.	Relevância	
7	Foi fácil entender o jogo e começar a utilizá-lo como material de estudo.	Confiança	
8	Ao passar pelas etapas do jogo senti confiança de que estava aprendendo.	Confiança	
9	Estou satisfeito porque sei que terei oportunidades de utilizar na prática coisas que aprendi com o jogo.	Satisfação	
10	É por causa do meu esforço pessoal que consigo avançar no jogo	Satisfação	
11	Temporariamente esqueci as minhas preocupações do dia-a-dia, fiquei totalmente concentrado no jogo.	Imersão	
12	Eu não percebi o tempo passar enquanto jogava, quando vi o jogo acabou.	Imersão	
13	Senti-me mais no ambiente do jogo do que no mundo real, esquecendo-se do que estava ao meu redor.	Imersão	
14	Este jogo é adequadamente desafiador para mim, as tarefas não são muito fáceis nem muito difíceis.	Desafio	
15	O jogo evolui num ritmo adequado e não fica monótono – oferece novos obstáculos, situações ou variações de atividades.	Desafio	
16	Consegui atingir os objetivos do jogo por meio das minhas habilidades.	Competência	
17	Tive sentimentos positivos de eficiência no desenrolar do jogo	Competência	
18	O jogo contribuiu para a minha aprendizagem na disciplina.	Aprendizagem de curto termo	
19	Os controles para realizar ações no jogo responderam bem.	Aprendizagem de curto termo	
20	É fácil aprender a usar a interface e controles do jogo.	Aprendizagem de curto termo	
21	O jogo foi eficiente para minha aprendizagem, em comparação com outras atividades da disciplina.	Aprendizagem de curto termo	
22	A experiência com o jogo vai contribuir para meu desempenho na vida profissional.	Aprendizagem de longo termo	
23	Cite pontos fortes que você identificou no jogo	Comentário	
24	Cite sugestões para melhoria do jogo.	Comentário	

Fonte: O autor.