Uma Avaliação Inicial do Jogo para o Ensino de Testes de Software iTestLearning sob a Ótica de um Software Educativo

Emanuel Ferreira Coutinho[1], Carla Ilane Moreira Bezerra[2]

[1]Instituto UFC Virtual – Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

[2]Campus Quixadá – Universidade Federal do Ceará (UFC) – Quixadá, CE – Brasil

emanuel@virtual.ufc.br, carlailane@ufc.br

Abstract. *Adapting theory to practice is usually an arduous task to conduct due to technical and human factors. Usually in Software Engineering disciplines, requirements are produced and implemented, requiring validations and consequently Software Testing. One difficulty in teaching Software Testing is to apply it in practice. In this context, Educational Softwares can help in the teaching-learning process from the contextualisation of certain contents to a certain reality. For this, an environment for supporting software tests teaching was developed, denominated iTestLearning. The objective of this work is to present an evaluation of iTestLearning from the viewpoint of a method that evaluates educational software and from the viewpoint of the students that used it in the discipline of Software Engineering.*

Resumo. *Adequar teoria à prática é normalmente uma tarefa árdua de se conduzir devido a fatores técnicos e humanos. Normalmente em disciplinas de Engenharia de Software, requisitos são produzidos e implementados, necessitando de validações, e consequentemente Testes de Software. Uma dificuldade em ensinar Testes de Software é aplica-lo na prática. Nesse contexto, Softwares Educativos podem auxiliar no processo de ensino-aprendizagem a partir da contextualização de determinados conteúdos para uma determinada realidade. Para isso, um ambiente de apoio ao ensino de testes de software foi desenvolvido, denominado iTestLearning. O objetivo desse trabalho é apresentar uma avaliação do iTestLearning do ponto de vista de um método que avalia software educativo e do ponto de vista dos alunos que o utilizaram na disciplina de Engenharia de Software.*

1. Introdução e Motivação

Disciplinas que alinham a teoria à prática são normalmente difíceis de se conduzir devido à necessidade de se tratar fatores técnicos (linguagens de programação, ferramentas, componentes, etc) e humanos (comunicação, gestão, disponibilidade, etc) ao mesmo tempo [Coutinho et al. 2016]. De maneira geral, disciplinas de cursos de graduação mais técnicos tendem a ter uma abordagem curricular muito voltada para produção e tecnologias [Coutinho e Moreira 2017]. Normalmente, cursos mais tradicionais relacionados à Computação acabam produzindo diversos tipos de softwares. Devido a interdisciplinaridade entre áreas, muitas vezes há um processo de desenvolvimento entre áreas totalmente diferentes, onde geralmente se produz requisitos

em uma área e em outra se implementa um aplicativo. Estes aplicativos ou sistemas gerados necessitam de uma validação quanto aos requisitos propostos.

O processo de testes de software possui três atividades básicas: planejamento, projeto e execução. Na fase de planejamento é definida a estratégia de testes que deverá ser utilizada para minimizar os riscos de negócios. Na fase de projeto é realizada uma identificação dos casos de testes mais adequados para que se possa apontar o número de falhas possíveis. A fase de execução 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].

Wangenheim e Silva [Wangenheim e Silva 2009] indicam em seu estudo que para os profissionais da área de software, o ensino de testes de software é pouco satisfatório, o que demonstra uma deficiência nessa disciplina nos principais cursos de Ciências da Computação.

Uma das maiores dificuldades para o ensino de testes de software ocorrer se dá pela necessidade da aplicação do processo na prática, com muitas vezes tendo um aprendizado apenas a nível teórico. Nesse contexto, a literatura recomenda o desenvolvimento de jogos, pois os mesmos possibilitam a experimentação de situações que seriam vivenciadas fora do contexto educacional [Silva 2010]. Por exemplo, situações diretamente ligadas ao ambiente profissional.

Desta maneira, os Softwares Educativos podem auxiliar no processo de ensino-aprendizagem a partir da contextualização de determinados conteúdos para a realidade de cada aluno [Morais 2003]. Os Softwares Educativos são concebidos com o objetivo principal de facilitar o processo de ensino-aprendizagem, sendo compostos por um conjunto de recursos computacionais projetados com a intenção de serem empregados em um contexto educacional [Sancho 1998].

O uso dos Softwares Educativos tem, em potencial, uma boa aceitação, pois estes acrescentam elementos lúdicos ao processo de ensino-aprendizagem [Silva el al. 2016]. No entanto, dificuldades podem ser enfrentadas pelos educadores na implantação de Softwares Educativos no ambiente escolar. A escolha por um Software Educativo que corresponda às expectativas do professor e às necessidades pedagógicas do conteúdo abordado exige atenção, sendo necessário garantir que o Software Educativo atenda a um conjunto de requisitos para que seu uso seja efetivo. O Software Educativo precisa funcionar corretamente, então seus elementos pedagógicos precisam estar de acordo com os propósitos dos conteúdos abordados e a interface deve ser condizente com as características, necessidades e limitações dos usuários. Problemas em qualquer um destes níveis pode prejudicar o processo de construção do conhecimento. Portanto, é necessário, antes de decidir adotar um Software Educativo, certificar-se de que este será um elemento agregador no aprendizado, avaliando-o antes de seu uso.

Neste contexto, um ambiente de apoio ao ensino de testes de software foi desenvolvido, na forma de um jogo de simulação, denominado iTestLearning [Sousa et al. 2012]. Em sua primeira versão havia apenas a fase de planejamento, sendo esta avaliada em [Bezerra e Coutinho 2013]. O jogo evoluiu com a adição da fase de projeto [Bezerra et al. 2014] e fase de execução [Jorge et al. 2015]. As avaliações foram realizadas por meio do modelo descrito em Savi et al. [Savi et al. 2011], onde 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.

O objetivo desse trabalho é apresentar uma avaliação do jogo para o ensino de testes de software iTestLearning do ponto de vista de um método que avalia software educativo e do ponto de vista dos alunos que o utilizaram na disciplina de Engenharia de Software. O restante do artigo está dividido nas seguintes seções: uma breve contextualização da ferramenta e referencial teórico; a metodologia planejada; os resultados e análises, e por fim conclusões e trabalhos futuros.

2. Contextualização e Referencial Teórico

Esta seção descreve brevemente sobre Testes de Software, um pouco sobre a ferramenta iTestLearning e sobre o método de Reeves para a avaliação de softwares educacionais.

2.1. Testes de Software

O estudo de testes de software é obrigatório em todo curso de Engenharia de Software. Livros de autores como [Pressman 2006] e [Sommerville 2007], que são referências comuns para disciplinas de Engenharia de Software trazem muitos conceitos e tipos de testes de software, destacando sua importância para o desenvolvimento de aplicações.

De acordo com o IEEE [IEEE 2004] o teste de software é a "verificação dinâmica do funcionamento de um programa utilizando um conjunto finito de casos de teste, adequadamente escolhido dentro de um domínio de execução, em geral, infinito, contra seu comportamento esperado". Essa definição é interessante por destacar que o teste envolve a execução do programa, que o teste exaustivo (com um conjunto infinito de casos de testes) é inviável na prática e que ele compara o comportamento real do sistema contra um comportamento esperado.

Os testes podem ser classificados quanto ao objetivo. Testes funcionais verificam se o comportamento do sistema está de acordo com as especificações. Testes de usabilidade analisam dentre outras coisas o quão fácil é para o usuário usar o software. Testes de desempenho verificam os requisitos de desempenho do software. Outros tipos de testes e mais detalhes podem ser encontrados em [IEEE 2004].

Com o objetivo de guiar a execução dos testes, um processo de teste pode ser utilizado. Segundo Müller et al. [Müller et al. 2007], em geral os processos de testes apresentam as seguintes atividades:

- Planejamento e controle: O planejamento consiste em definir os objetivos dos testes especificando as atividades para alcançar esses objetivos. A atividade de controle consiste em comparar constantemente o progresso atual contra o planejado. O planejamento pode ser documentado em um documento denominado "Plano de Testes", que deve conter dentre outras coisas: recursos, cronograma, itens que serão testados, equipe e riscos;
- Análise e Modelagem (Projeto): Os objetivos gerais do teste são transformados em condições e modelos de teste tangíveis. Um dos principais resultados dessa etapa é o documento de especificação dos casos de testes, onde são descritos os casos e procedimentos de testes;

- Implementação e execução: Procedimentos ou roteiros de teste são implementados pela combinação dos casos de teste. Além disso, o ambiente de testes é preparado e os testes são executados;
- Avaliação do critério de saída e relatórios: A execução do teste é avaliada mediante os objetivos definidos para produzir um relatório de testes e verificar se são necessários mais testes;
- Atividades de encerramento de testes: Coleta de dados de todas as atividades para consolidar a experiência da execução dos testes.

2.2. 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 [Sousa et al. 2012] [Bezerra e Coutinho 2013] [Bezerra et al. 2014] [Jorge et al. 2015].

Atualmente o jogo conta com três etapas: a fase de planejamento, fase de projeto e fase de execução. Durante a fase de planejamento [Sousa et al. 2012] [Bezerra e Coutinho 2013], o aluno deverá realizar o planejamento dos testes englobando durante o procedimento seis etapas definidas: itens de teste, tipos de teste, níveis de teste, critérios de aceitação, ferramentas e artefatos. Em cada uma dessas etapas o aluno será desafiado a selecionar entre as opções referentes ao item de planejamento qual melhor se adequa ao projeto que está sendo jogado. Ao longo do progresso nas etapas, diversos conceitos referentes a teste de software vão sendo apresentados ao jogador, tais como ferramentas para testes (Selenium, Jmeter) e técnicas de teste. A Figura 1 demonstra a fase de planejamento.

Figura 1. Fase de planejamento do jogo iTestLearning [Sousa et al. 2012] [Bezerra e Coutinho 2013]

A fase de projeto se inicia 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 respostas do sistema [Bezerra et al. 2014]. A Figura 2 apresenta telas da fase de projeto.

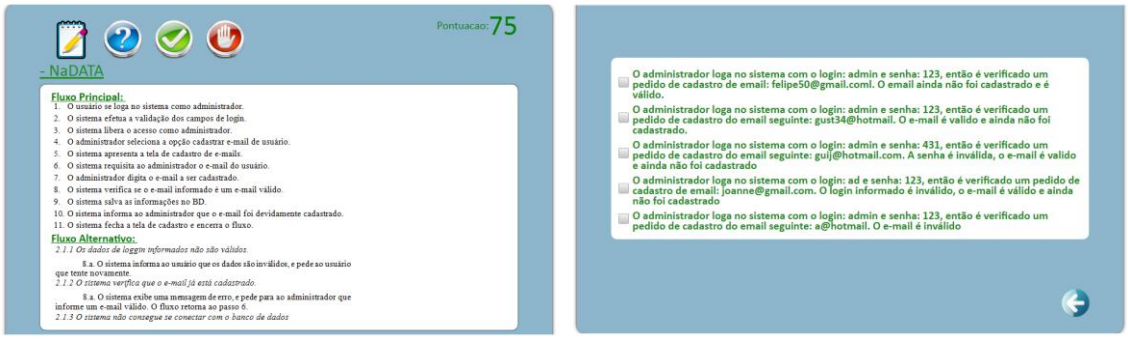


Figura 2. Fase de projeto do jogo iTestLearning [Bezerra et al. 2014]

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 aplicadas ao processo da fase de execução, foi utilizada como modelo o Teste de Caixa-Preta, onde serão usados os casos de teste selecionados durante a fase de projeto. A partir destes casos de teste simulações são realizadas para a verificação da saída dos dados em um ambiente funcional, de forma que o aluno possa observar se as saídas estão coerentes com as entradas [Jorge et al. 2015]. A Figura 3 exibe telas da fase de execução.

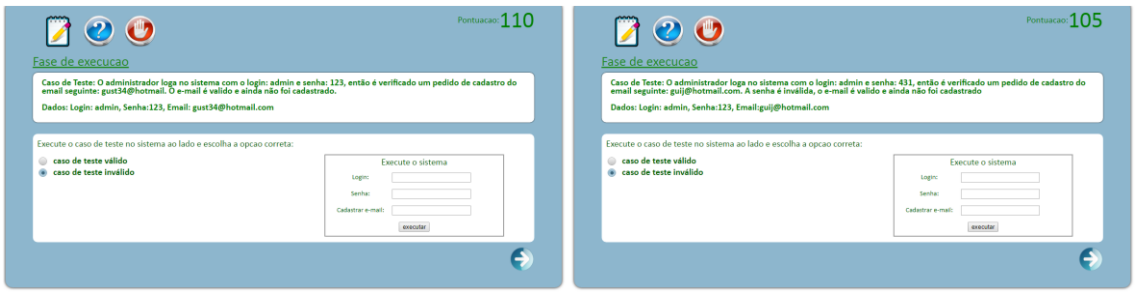


Figura 3. Fase de execução do jogo iTestLearning [Jorge et al. 2015]

2.3. Método de Reeves

Tecnologias de maneira geral disponibilizam vários tipos de softwares para um melhor aproveitamento educacional e como uma nova forma de aprendizagem [Rodrigues et al 2017]. Tais softwares podem ser classificados conforme suas características, podendo ser tutoriais, simulação, jogo, entre outros [Valente, 1999].

O processo de avaliação de um software educativo consiste em observar suas características e, também, suas implicações para o uso educacional [Pereira et al., 2016]. Na literatura, existem vários métodos para avaliação de softwares educativos abordando tanto aspectos pedagógicos quanto de interface do usuário, tais como o método de Reeves, a técnica de Mucchielli, a avaliação de LORI, a técnica de TICESE, dentre outros. Neste trabalho utilizaremos o Método de Reeves.

O método de Reeves apresenta duas abordagens complementares para a avaliação de um software educacional [Reeves e Harmon 1994]. Basicamente o método é composto por 10 critérios de avaliação da interface do usuário e 14 critérios para a avaliação de aspectos pedagógicos. Nessa abordagem, cada critério é associado a uma escala não numérica, que utiliza de conceitos antagônicos. Os critérios são avaliados em um procedimento gráfico, que consiste em realizar uma marcação sobre uma escala com dois sentidos, sob a forma de uma seta bidirecional. À esquerda da seta situam-se os conceitos mais negativos, e à direita os mais positivos. Quando o avaliador decide o que for marcar para cada critério, ele deve unir os pontos de cada item. A conclusão da avaliação é pela análise da disposição dos pontos marcados nas escalas.

3. Metodologia

A metodologia adotada neste trabalho constou de uma série de atividades a serem executadas durante a disciplina Engenharia de Software pelos alunos e uma avaliação do jogo com o método de Reeves para avaliação de software educacional.

Durante a disciplina Engenharia de Software, uma atividade individual foi criada onde os alunos deveriam acessar o jogo, utilizá-lo e responder a um questionário projetado no Google Forms com 4 questões de múltipla escolha e 3 questões abertas. No início do formulário os alunos concordaram com um termo de consentimento da pesquisa. As questões estão dispostas na Tabela 1. A questão 1 utiliza a escala “Ruim”, “Regular”, “Indiferente”, “Bom”, “Ótimo”. As questões 2, 3 e 4 são na escala “Não entendi nada”, “Entendi pouco”, “Entendi mais ou menos”, “Entendi quase tudo”, “Entendi tudo”. As questões 5, 6 e 7 são livres, podendo o aluno responder o que achar mais adequado. A utilização do jogo se deu antes da aula de testes de software, onde os alunos teriam que lidar apenas com seus conhecimentos prévios ou de experiências com testes de software. Após o período de preenchimento do formulário, o professor consolidou os dados e realizou as análises.

Tabela 1. Formulário aplicado aos alunos (Tipo da questão: M = múltipla escolha, A = questão aberta e livre)

ID	Tipo	Questão
1	M	De maneira geral (ignorando os erros, interface gráfica e as funcionalidades não implementadas) o que você achou do jogo?
2	M	Em que nível de entendimento das atividades da fase de planejamento de testes você se encontra após o jogo?
3	M	Em que nível de entendimento das atividades da fase de projeto de testes você se encontra após o jogo?
4	M	Em que nível de entendimento das atividades da fase de execução de testes você se encontra após o jogo?
5	A	Você achou o jogo adequado para iniciar os estudos em processos de testes de software?
6	A	Que sugestões você daria para que a dinâmica do jogo seja mais motivadora?
7	A	Quais os principais pontos fracos do jogo em relação a ensinar as atividades de teste de software?

A outra etapa da pesquisa foi o emprego do método de Reeves [Reeves e Harmon 1994] para uma análise do jogo iTestLearning como software educativo. Para isso, pessoas que conheciam o jogo iriam responder os critérios de interface do usuário

e pedagógicos. Após a contabilização de cada resultado, gráficos para a análise seriam elaborados.

4. Resultados e Análises

Esta seção descreve os resultados encontrados na pesquisa e suas análises. Ela está subdividida em questões de múltipla escolha, questões abertas e no método de Reeves.

4.1. Questões de Múltipla Escolha

Para essa etapa da pesquisa, 15 alunos da turma de Engenharia de Software do Curso de Graduação em Sistemas e Mídias Digitais da Universidade Federal do Ceará responderam o questionário. Todos esses alunos possuem conhecimentos em programação e em interfaces de usuário (usabilidade e *design*) e conceitos de gamificação e narrativas de jogos. A Figura 4 exibe gráficos de pizza para as quatro questões de múltipla escolha.

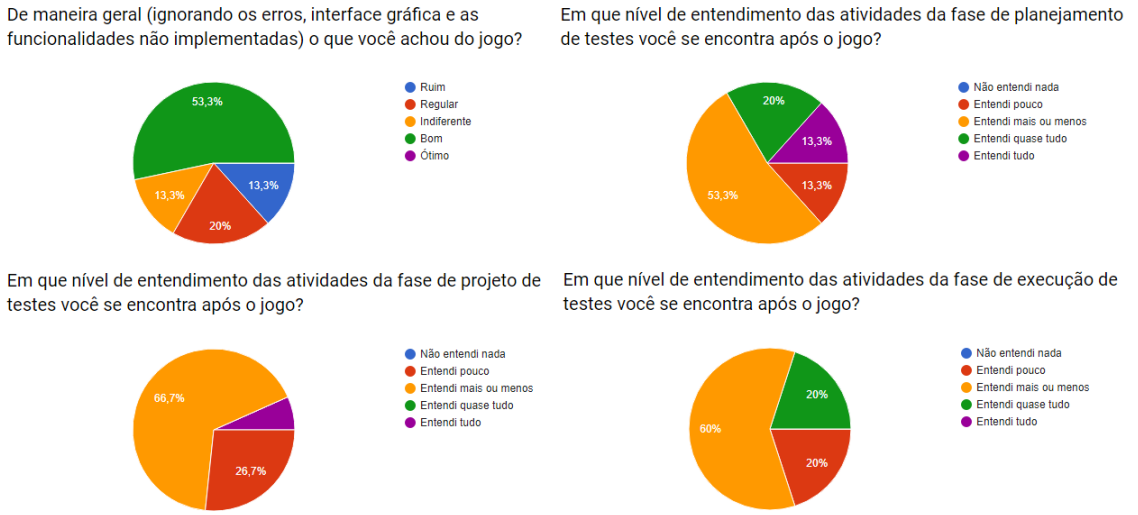


Figura 4. Gráficos com os resultados das quatro questões de múltipla escolha

De maneira geral, os alunos acharam o jogo bom, ignorando erros conhecidos, mesmo sem os conhecimentos prévios da teoria sobre testes de software. Ao se considerar cada umas das três fases cobertas pelo jogo, percebeu-se que o aplicativo deveria ter ajustes para uma melhor compreensão por parte dos alunos. Isso também ficou visível pelos comentários das questões abertas e pela avaliação com o método de Reeves. Apesar que nas três fases não ocorreu nenhuma resposta “Não entendi nada”, de maneira geral houveram ocorrências que o jogo era ruim.

As três fases obtiveram como a maioria das respostas “Entendi mais ou menos”, o que implica que é necessário identificar os motivos e reformular a aplicação. Especificamente para as fases de projeto e execução, praticamente dois terços dos alunos não compreenderam bem suas atividades, o que requer um ajuste maior.

4.2. Questões Abertas

Você achou o jogo adequado para iniciar os estudos em processos de testes de software?

Atualmente o jogo possui alguns erros de navegação, sendo conhecida a necessidade de ajustes nos erros para melhorar seu desempenho. Isso impactou um pouco na avaliação dos alunos, pois muitos indicaram que esse fato prejudicou os estudos. Um aluno relatou que para iniciar os estudos o jogo requeria um conhecimento prévio sobre técnicas, ferramentas e teorias, que ele só iria saber ao longo da disciplina. Houve a observação que o jogo é uma ferramenta para testar conhecimentos, e não para iniciar os estudos, e que ela é semelhante a modelos de estudo tradicionais. Por fim, de maneira explícita, quatro alunos apontaram que sim, o jogo era adequado para estudos de teste de software, e seis apontaram que não.

Que sugestões você daria para que a dinâmica do jogo seja mais motivadora?

Destaques para a necessidade de redução dos erros na aplicação, como era esperado. Em relação às fases de testes, sentiu-se uma necessidade de um breve resumo de cada etapa, explicando a importância da fase. Assim seria uma tutoria mais adequada. A utilização de uma função de "ajuda/informação" para alguns termos, como nomes de software. O jogo também tem que ser autocontido, ou seja, possuir todas as informações disponíveis nele próprio. Do ponto de vista da aplicação, melhorar interface, usabilidade e a comunicação com o jogador, principalmente os textos, pois em algumas situações o entendimento foi confuso. Para a melhoria da motivação, talvez utilizar outra abordagem diferente de um quiz em alguns pontos, e incluir elementos de gamificação e uma narrativa melhor. Frases de estímulo que indiquem se o aluno está indo bem, de uma maneira geral, a cada resposta enviada também colaborariam para a motivação. Em relação ao *feedback*, não há uma identificação clara dos pontos fracos. A utilização de ícones junto dos nomes das ferramentas de teste e a utilização de mais *feedbacks* visuais indicando o progresso (melhor aproveitamento da gamificação). Por fim, caso o aluno quisesse continuar o jogo sem ter que reiniciá-lo, uma fase intermediária para corrigir erros seria interessante, pois o aluno poderia ir para a próxima fase sem ler tudo que errou ou até identificar o que errou, pois estaria junto com os acertos.

Quais os principais pontos fracos do jogo em relação a ensinar as atividades de teste de software?

Alguns alunos apontaram a necessidade de base em testes de software (conceitos e ferramentas). Isso implicava para alguns que a aplicação não é um jogo em si, servindo mais para testar conhecimentos, assemelhando-se a uma avaliação. Alguns alunos indicaram que haviam poucas definições, em contraste com outros alunos que relataram textos demais. Um aspecto interessante foi a ausência de foco ou reflexão nos erros, possibilitando aprender mais com os erros. Nas fases, a que recebeu mais relatos foi a execução, com alguns testes confusos e repetitivos. A falta de motivação em usar o jogo foi relatada como consequência de muito texto para ler, e por não ser um jogo em si. Por fim, erros no jogo prejudicaram seu desempenho.

4.3. Método de Reeves

Para uma avaliação de um software educativo, o método de Reeves foi utilizado. Duas pessoas responderam ao método. Essas duas pessoas possuíam conhecimento na ferramenta e já a utilizaram em disciplinas de Engenharia de Software para o ensino de Testes de Software. Os resultados do método para os critérios de interface com o

usuário estão apresentados na Figura 5 e os resultados para os critérios pedagógicos estão dispostos na Figura 6.

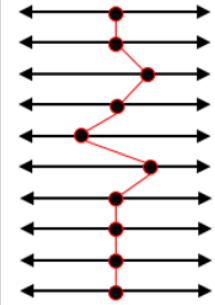
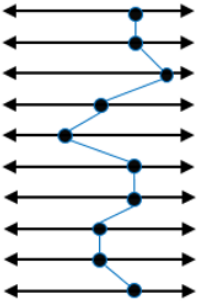
Critérios de interface com o usuário	Avaliador 1	Avaliador 2
Facilidade (difícil a fácil)		
Navegação (difícil a fácil)		
Carga cognitiva (não gerenciável a gerenciável/intuitiva)		
Mapeamento (nenhum a poderoso)		
Design da tela (princípios violados a princípios respeitados)		
Compatibilidade espacial do conhecimento (incompatível a compatível)		
Apresentação da informação (confusa a clara)		
Integração das mídias (não coordenada a coordenada)		
Estética (desagradável a agradável)		
Funcionalidade geral (não funcional a altamente funcional)		

Figura 5. Critérios de interface com o usuário [Reeves e Harmon 1994] para o iTestLearning

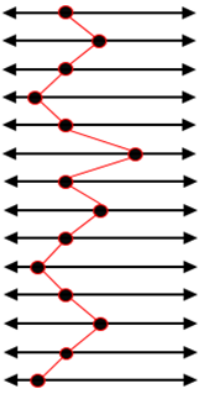
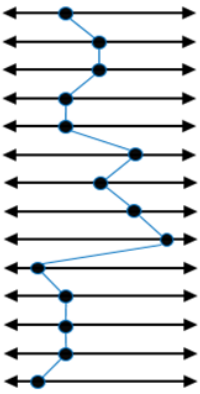
Critérios pedagógicos	Avaliador 1	Avaliador 2
Epistemologia (objetivista a construtivista)		
Filosofia pedagógica (instrutivista a construtivista)		
Psicologia subjacente (comportamental a cognitiva)		
Objetividade (precisamente focalizado a não focalizado)		
Sequenciamento instrucional (reducionista a construtivista)		
Validade experimental (abstrato a concreto)		
Papel do instrutor (provedor de materiais a agente facilitador)		
Valorização do erro (aprendizagem sem erro a com experiência)		
Motivação (extrínseca a intrínseca)		
Estruturação (alta a baixa)		
Acomodação de diferenças individuais (não existente a multifacetada)		
Controle do aluno (não existente a irrestrito)		
Atividade do usuário (matemagênica a generativa)		
Aprendizado cooperativo (não suportado a integral)		

Figura 6. Critérios pedagógicos [Reeves e Harmon 1994] para o iTestLearning

Do ponto de vista de facilidade e navegação, os avaliadores estão considerando com tendência a um software fácil. Entretanto, apesar disso, o *design* da tela deixa a desejar. No caso do *design* melhorar, é possível que os demais itens também melhorem sua avaliação, principalmente apresentação da informação e estética. Pela análise gráfica do método de Reeves, para os critérios de interface com o usuário pode-se dizer de maneira geral para os dois avaliadores que o software iTestLearning está mediano nesse critério.

Em relação aos critérios pedagógicos, muito há de se analisar para a melhoria do software. Um aspecto que se destacou foi quanto à motivação, onde os avaliadores deram respostas quase opostas. Muitos desses critérios não quer dizer certo ou errado, e sim ao estilo do software educativo. Um trabalho a ser feito é definir quais as estratégias que o software deve ter para melhor atender às necessidades dos usuários. A acomodação das diferenças individuais é outro aspecto a ser investido. Apesar que o jogo possui diferentes níveis de dificuldade, a sequência e atividades são as mesmas, sem adaptações da aplicação. Por fim, o aprendizado cooperativo não é incentivado na aplicação. Pela análise gráfica do método de Reeves, para os critérios pedagógicos, pode-se dizer de maneira geral para os dois avaliadores que o jogo iTestLearning está carente de aspectos de acomodação de diferenças, aprendizado cooperativo, estruturação e atividades construtivistas.

Alguns critérios para os dois avaliadores se destacaram. A objetividade obteve valores próximos a precisamente focalizado, ressaltando que o jogo possui uma forma utilizada nos tutoriais e treinamentos, o que é verdade. Estruturação obteve valores altos, que também é verdade, indica que o jogo é altamente estruturado quando sua sequência e caminhos já foram determinados previamente. Por fim, aprendizado cooperativo não foi suportado, pois o jogo não permite trabalho em pares ou grupos.

O critério que obteve maior valor na escala em comum para os dois avaliadores foi o de validade experimental, que varia de abstrato a concreto, sendo mais próximo de concreto, que é quando se contextualiza o conteúdo apresentado em situações reais. A motivação obteve valores inversos. Sua variação vai de extrínseca (vem de fora do ambiente de aprendizado) a intrínseca (vem de dentro do ambiente de aprendizado). Nesse critério, consultamos os avaliadores. O que pontuou como extrínseca informou que sua motivação era de alguém que lhe mandaria usar a aplicação, ou porque seria uma avaliação. O que pontuou intrínseca informou que sua motivação era porque achou a aplicação interessante e ficou curiosa em seguir adiante nas fases.

Alguns desses critérios coincidiram com avaliações dos alunos durante o formulário. Apesar dos avaliadores terem considerando os critérios de interface com o usuário do jogo medianos, muitos dos alunos apontaram problemas de usabilidade, erros, desmotivação e excesso de texto. Em relação aos critérios pedagógicos, motivação foi um problema apontado por um dos avaliadores, mas indicado por vários alunos. Talvez o perfil dos alunos com experiência em gamificação e narrativas mais ativas esteja indicando um novo direcionamento para o jogo. A valorização do erro também foi um ponto comum a ser relatado, necessitando um maior reflexo no jogo e um *feedback* melhor e preferencialmente mais visual.

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma avaliação do jogo para o ensino de testes de software iTestLearning, e suas três fases: planejamento, projeto e execução. O trabalho se deu por meio de uma avaliação do jogo em um questionário por parte dos alunos da disciplina Engenharia de Software, sob aspectos do uso da aplicação, e pela aplicação do método de Reeves para avaliação de softwares educativos.

Como trabalhos futuros, pretende-se ajustar a aplicação para a correção de erros identificados, e melhoria das funcionalidades. Baseado no resultado do método de Reeves, diversas ações podem ser tomadas para a melhoria do jogo, desde a melhoria da interface com o usuário, quanto a aspectos de cooperação e acompanhamento das atividades e progresso do aluno. Também é importante investir no aspecto diversão do jogo, para motivar ainda mais os alunos. Para isso, pretende-se pesquisar diversas formas de deixar o jogo mais atrativo, como por exemplo desafios, surpresas e narrativas. Por fim, realizar uma avaliação do jogo com um rigor científico maior.

Referências

Bastos, A., Rios, E., Cristalli, R., Moreira, T. (2007) Base de Conhecimento em Teste de Software. 2.ed. São Paulo: Martins Fontes.

Bezerra, C. I. M. e Coutinho, E. (2013). Avaliação do jogo itestlearning: Um jogo para o ensino de planejamento de testes de software. In XXI Workshop sobre Educação em Computação (WEI2013).

Bezerra, C. I. M., Coutinho, E. F., Santos, I. S., Monteiro, J. M., e Andrade, R. M. C. (2014). Evolução do jogo itestlearning para o ensino de testes de software: Do planejamento ao projeto. In XIX Conferência Internacional sobre Informática na Educação (TISE2014), Fortaleza.

Coutinho, E. F. e de Oliveira Moreira, L. (2017). Um panorama sobre a interdisciplinaridade da engenharia de software no curso de graduação em sistemas e mídias digitais. Revista Sistemas e Mídias Digitais (RSMD), 2(1).

Coutinho, E. F., Gomes, G. A. M., e Junior, A. J. M. L. (2016). Applying design thinking in disciplines of systems development. In 8th Euro American Conference on Telematics and Information Systems (EATIS2016).

IEEE (2004) SWEBOK - Guide to the software engineering body of knowledge. IEEE Computer Society.

Jorge, F. F., Bezerra, C. I. M., Coutinho, E. F., Monteiro, J. M., e Andrade, R. M. C. (2015). A evolução do jogo itestlearning para o ensino das atividades de execução de testes de software. In XX Conferência Internacional sobre Informática na Educação (TISE 2015) - Nuevas Ideas en Informática Educativa TISE 2015.

Morais, R. X. T. (2003). Software educacional: a importância de sua avaliação e do seu uso nas salas de aula. Monografia (Bacharel em ciências da computação) – Faculdade Lourenço Filho. 51p. Fortaleza.

Müller, T., Graham, D., Friedenberg, D., Veendendal, E. (2007) Base de Conhecimento para Certificação em Teste - Foundation Level Syllabus. ISTQB - Comissão Internacional para Qualificação de Teste de Software.

Pereira, W. S., Cardoso Filho, R. J., Silva, W. R. A., Silva, R. S. T., Dantas, V. F., Aguiar, Y. P. C. (2016) Validação de uma abordagem combinada para avaliação de software educativo: avanços e desafios. Revista Tecnologias na Educação. Vol.16, No. 8, Setembro. p. 06.

Pressman, R. S. (2006). Engenharia de software. McGraw-Hill, 6 edition.

Reeves, T. e Harmon, S. W. (1994). Systematic Evaluation Procedures for Interactive Multimedia for Education and Training, capítulo 15. Idea Group Publishing.

Rodrigues, L. R., Luiz, S. S. S., Sebastião, J. F., Padilha, T. P. P. (2017) Avaliação de Softwares Educativos Voltados para Conscientização e Prevenção de Acidentes de Trânsito: um estudo de caso. II Congresso sobre Tecnologias na Educação (Ctrl+E 2017), Mamanguape - PB.

Sancho, J. (1998). Para uma tecnologia educacional. Porto Alegre: ArtMed.

Savi, R., Wangenheim, C., Borgatto, A. (2011) Um Modelo de Avaliação de Jogos Educacionais na Engenharia de Software. Anais do Simpósio Brasileiro de Engenharia de Software (SBES 2011), São Paulo.

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

Silva, R.S., Silva, W.R., Filho, R. C., Pereira, W., Aguiar, Y., Dantas, V. (2016) Avaliação de Software Educativo: a complexidade de escolher uma abordagem adequada. I Congresso Regional sobre Tecnologias na Educação (Ctrl+E 2016), Natal - RN.

Sommerville, I. (2007). Engenharia de software. Pearson Prentice Hall, 8 edition.

Sousa, V., Bezerra, C. I. M., Coutinho, E., e Santos, I. S. (2012). itest learning: Um jogo para o ensino do planejamento de testes de software. In V Fórum de Educação em Engenharia de Software (FEES 2012), Natal - RN.

Valente, J. A. (org). (1999). O computador na sociedade do conhecimento. Campinas: Unicamp/NIED.

Wangenheim, C. G., Silva, D. A. (2009) Qual conhecimento de engenharia de software é importante para um profissional de software? In Anais do Fórum de Educação em Engenharia de Software, Fortaleza.