iTest Learning: Um Jogo para o Ensino do Planejamento de Testes de Software

Virgínia Farias[1], Carla Moreira[1,3,4], Emanuel Coutinho[2,3,4], Ismayle S. Santos[3,4,a]

[1]Universidade Federal do Ceará (UFC)
Quixadá, CE – Brazil

[2]Instituto UFC Virtual – Universidade Federal do Ceará (UFC)
Fortaleza, CE – Brazil

[3]MDCC – Mestrado e Doutorado em Ciência da Computação
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brazil

[4]Grupo de Redes, Engenharia de Software e Sistemas - GREaT

virginia.farias7@gmail.com, emanuel@virtual.ufc.br, carlailane@ufc.br, ismaylesantos@great.ufc.br

Abstract. *The software test activity is one of the activities related to quality assurance software. Given the importance of this activity, software testing has been addressed in all curriculum of reference proposed by both the SBC and the ACM. Aiming to assist the teaching of software testing, games have been proposed in the literature. However, there is still a lack of games for certain phases of testing activity, such as the planning phase. In this context, this paper aims to present an educational game to support the teaching of software testing, providing a simulation environment to experiment with practices related to planning of software testing.*

Resumo. *A atividade de teste de software é uma das atividades relacionadas à garantia de qualidade de software. Em consequência à importância dessa atividade, o teste de software tem sido abordado em todos os currículos de referência propostos tanto pela SBC, quanto pela ACM. Visando auxiliar o ensino do teste de software, jogos têm sido propostos na literatura. No entanto, ainda existe uma carência de jogos para determinadas etapas da atividade de teste, como a etapa de planejamento. Neste contexto, este trabalho tem como objetivo apresentar um jogo educacional para apoiar o ensino de testes de software, proporcionando um ambiente de simulação para a experimentação de práticas relacionadas ao planejamento de testes de software.*

1. Introdução

Dentre as técnicas utilizadas para garantir a qualidade de um software tem-se a atividade de testes de software. Testar um software é executar um programa a procura de erros

[a] Bolsista da Capes do Programa de Mestrado MDCC/UFC

[Myers 2004]. Testes são necessários porque verificam o atendimento aos requisitos do usuário. Apesar da importância da atividade de testes de software, o ensino da disciplina Testes de Software tem recebido pouco destaque na maioria dos cursos de graduação em Ciência da Computação.

Embora o teste de software seja abordado em todos os currículos de referência propostos tanto pela SBC, quanto pela ACM [Silva et al. 2011b] essa é uma atividade que ainda não tem sido adequadamente abordada nos currículos. Segundo a pesquisa realizada por Wangenheim e Silva (2009) com 48 profissionais da área de software, o teste de software é um dos tópicos mais importantes dos cursos de Ciência da Computação, mas também é um dos tópicos cuja aprendizagem na graduação ainda deixa muito a desejar.

Visando auxiliar o ensino de teste de software, jogos podem ser desenvolvidos. Jogos têm sido utilizados para ajudar no ensino de diversas áreas do conhecimento. Estes, quando utilizados como ferramentas educacionais, podem permitir a experimentação de situações que seriam vivenciadas fora do contexto educacional, como, por exemplo, situações diretamente ligadas ao ambiente profissional [Silva 2010].

Neste contexto, este trabalho visa o desenvolvimento de um jogo educacional de simulação para apoiar o ensino de teste de software, de forma a simular a realização do planejamento de testes de software. Na seção 2 é descrito o iTest Learning. Na seção 3 trabalhos relacionados e na seção 4 a conclusão e trabalhos futuros.

2. O Jogo *iTest Learning*

O *iTest Learning* foi desenvolvido com o objetivo de apoiar o ensino de testes de *software* na fase de planejamento. Ele pertence à categoria de jogos educacionais de simulação, que provê um ambiente para a realização do planejamento de teste de *software* através de uma breve descrição de um projeto hipotético. O público alvo deste jogo são alunos de graduação de cursos da área de computação/informática, os quais possuem disciplinas que envolvem conteúdo relacionado ao teste de software.

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

A aplicação prática no jogo permitirá ao aluno entender melhor a dinâmica da atividade de testes no desenvolvimento de software. O jogo fará acesso a um banco de dados, onde informações do aluno, projetos, questões e exemplos estarão armazenados. Também serão armazenadas as informações das atividades dos alunos para subsidiar a análise do professor sobre o desempenho deles.

O jogo não cobrirá todos os itens que um planejamento de testes pode conter. Os itens cobertos são definidos com base em Silvia (2011a), que apresenta uma definição

de uma metodologia de teste de software para micro e pequenas empresas com base nos modelos de documentos disponibilizados pelo IEEE Standard 829 (1998). Tais itens são descritos a seguir:

- *Escopo*: descreve uma visão geral sobre o projeto que está sendo testado. O escopo é fornecido pelo próprio jogo, de maneira que seja dada uma breve descrição do projeto para o qual será feito o planejamento de teste de *software*;
- *Itens de Teste*: descreve os itens que serão objetos de teste. No jogo, o jogador deverá escolher quais os itens a serem testados de acordo com o projeto escolhido;
- *Tipos de Teste*: identifica os tipos de testes a serem realizados. Também está representado no jogo em uma etapa no qual o jogador define os tipos de teste que farão parte do projeto;
- *Níveis de Teste*: é uma das dimensões de teste que define "quando", ou melhor, a que fase do projeto se aplica um determinado teste. Este item não é contemplado no modelo de Silvia (2011a), mas foi introduzido no escopo do jogo para reforçar o conhecimento no processo de testes dentro do processo de desenvolvimento de software. O jogador terá que reconhecer quais os níveis de teste de software existem e o que eles representam;
- *Ferramentas*: este item corresponde à escolha das ferramentas que serão utilizadas para a realização dos testes. As ferramentas estão contidas na seção de recursos físicos descrita em [Silva 2011a];
- *Artefatos*: descreve os documentos que serão deverão ser desenvolvidos ao longo das atividades de teste. É uma fase do jogo que tem como objetivo fazer com que o jogador saiba quais artefatos podem ser gerados durante o processo de testes.

2.1. Tecnologia do Jogo

O jogo foi desenvolvido na linguagem Java utilizando o *framework* Java Server Faces, que se destina a simplificar o desenvolvimento de interfaces de usuário baseadas em web. Foram utilizados ainda os *frameworks* Hibernate, para persistência dos dados, e o Primefaces, suíte de componentes JSF customizados. Como servidor web foi utilizado o Tomcat, como banco de dados o PostgreSql e XHTML na camada de visão.

2.2. Modelagem do Jogo

Na Figura 1 é apresentado o fluxo de atividades do jogo, que consiste em realizar um planejamento de testes a partir de cenários propostos que representam especificações de sistemas hipotéticos. Essas especificações contêm características que determinarão os itens de teste, tipo de teste, níveis de teste e critérios de aceitação a serem adotados no desenvolvimento do sistema.

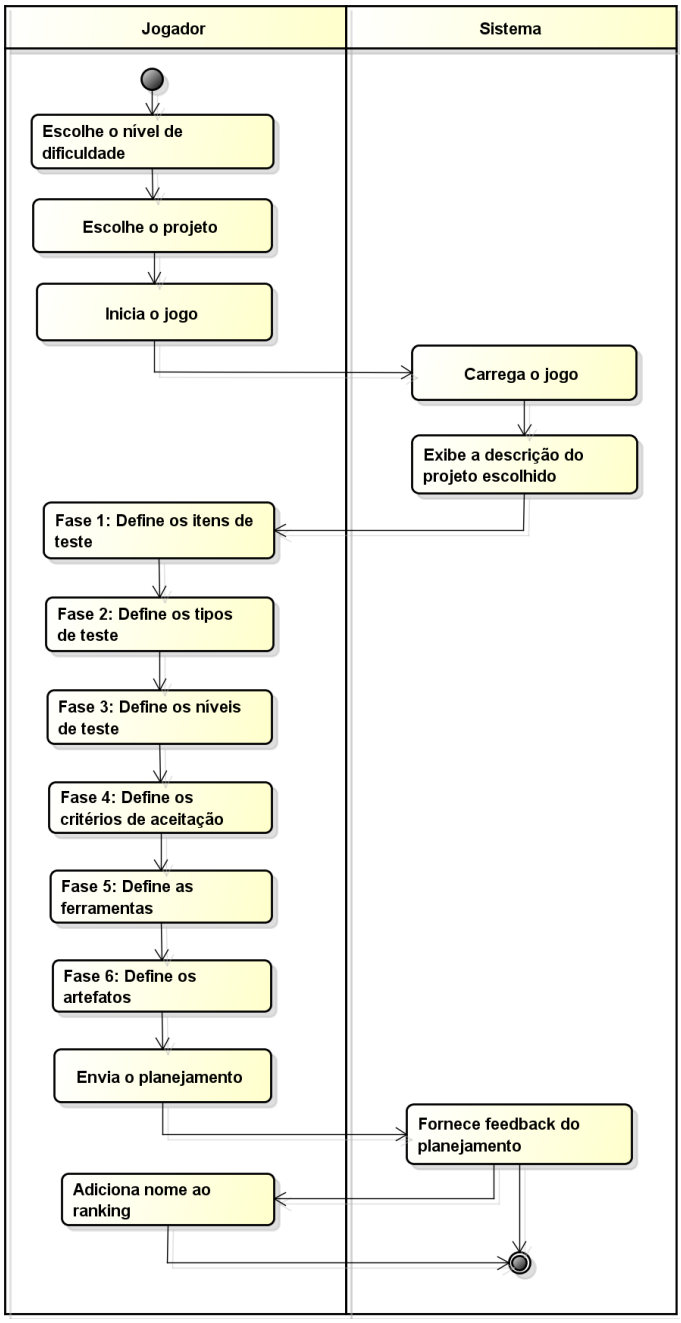

Figura 1: Diagrama de Atividades do iTest Learning

O jogador poderá escolher o nível de dificuldade do jogo (fácil, médio, difícil), a partir do qual poderá selecionar o projeto para iniciar o jogo. O jogo possui 6 fases: Na Fase 1 os jogadores devem escolher os itens do projeto a serem testados (de acordo com a descrição do projeto), na Fase 2 o jogador define quais tipos de teste serão realizados durante o processo de teste (Interface, Usabilidade, Segurança, Estresse,etc.), na Fase 3 o jogador definirá por quais níveis de teste o projeto passará (Unidade, Integração, Sistema, Aceitação, etc.), na Fase 4 serão definidos os critérios de aceitação que farão com que um teste executado seja aprovado ou não (de acordo com a descrição do projeto), na Fase 5 o jogador deve escolher quais ferramentas serão utilizadas no processo de testes (Selenium, JMeter, Marathon, Mantis, etc.) e na Fase 6 o jogador

indica quais artefatos podem ser gerados no processo de teste de software (Plano de Teste, Especificação de Casos de Testes, Relatório de Testes, etc.).O jogador só poderá passar para a fase seguinte quando concluir a atual. Ele ganhará pontos por cada acerto, e será penalizado em seus pontos a cada erro. A qualquer momento, o jogador poderá tirar dúvidas em relação aos conceitos abordados através da funcionalidade de ajuda, que estará disponível no decorrer do jogo. Quando o planejamento for concluído, a aplicação dará um *feedback* para o jogador, mostrando seus erros e/ou acertos. Nesse *feedback* também será exibida uma proposta de como poderia ser o planejamento dos testes, já que não há como definir uma resposta absoluta.

2.3. Dinâmica do Jogo

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

O jogo possui três níveis de dificuldade: fácil, médio, difícil. Podem existir diversos projetos que são relacionados a algum nível de dificuldade. Para iniciar o jogo, é preciso escolher um desses níveis de dificuldade e depois escolher um projeto referente ao nível de dificuldade escolhido. Ao escolher o projeto, é apresentada uma descrição do projeto escolhido ao jogador. Em seguida, o jogador deverá realizar o planejamento de testes de software, composto por 6 fases conforme a Figura 01, para o projeto selecionado. A Figura 2 ilustra a tela de interface de usuário na Fase 01 do jogo, onde o jogador deve definir quais os itens do teste para o projeto sob teste.

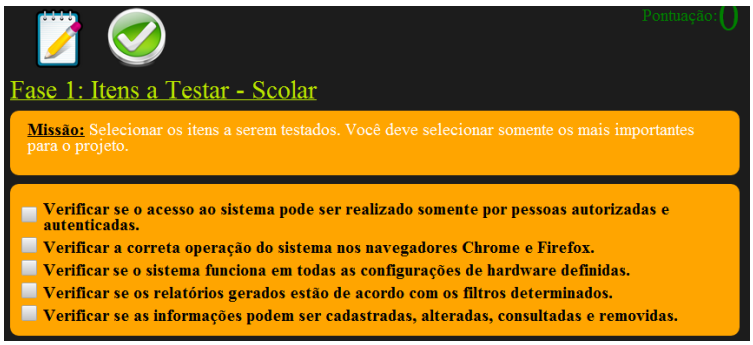

Figura 2: Tela da Fase 1 do jogo

Durante todo o jogo é possível tirar dúvidas em relação aos conceitos que são abordados em cada fase. Para isso, basta clicar no ícone de ajuda no *menu* do jogo. Há ainda, durante todo o projeto, em todas as fases, a opção de visualização da descrição do projeto. Isso permite que o jogador possa ter acesso a essa descrição sem necessitar sair da fase na qual está. Para tal, basta clicar no ícone correspondente, representado por uma caderneta de anotações. Por último, há "*check" que* exibe as respostas propostas para a fase corrente. Isso permite que o jogador possa ter um *feedback* imediato de suas ações, mesmo que ao final do jogo haja um *feedback* geral em relação ao planejamento realizado pelo jogador. Porém, uma vez utilizada essa opção, não será mais possível jogar na fase. Também não é permitido voltar para a fase anterior, uma vez que já passou por ela.

Ao final do jogo é exibido uma tela com um *feedback* contendo todo o planejamento feito pelo jogador, conforme ilustrado na Figura 3. Logo abaixo na mesma tela há também um planejamento que seria recomendado para o projeto escolhido. O *feedback* contém uma breve descrição do porquê de cada escolha, de forma a fazer o jogador refletir e também discutir as respostas com outros jogadores. Assim, o jogador poderá fazer uma comparação do seu planejamento com o planejamento recomendado pelo sistema iTest Learning. É preciso lembrar que o planejamento recomendado não quer dizer que é o correto, pois seu objetivo nesse jogo é orientar a execução das atividades, o sistema apenas indica uma sugestão de planejamento. Depois do término do jogo, o jogador tem a opção de ir para o *ranking* do jogo e verificar sua colocação.

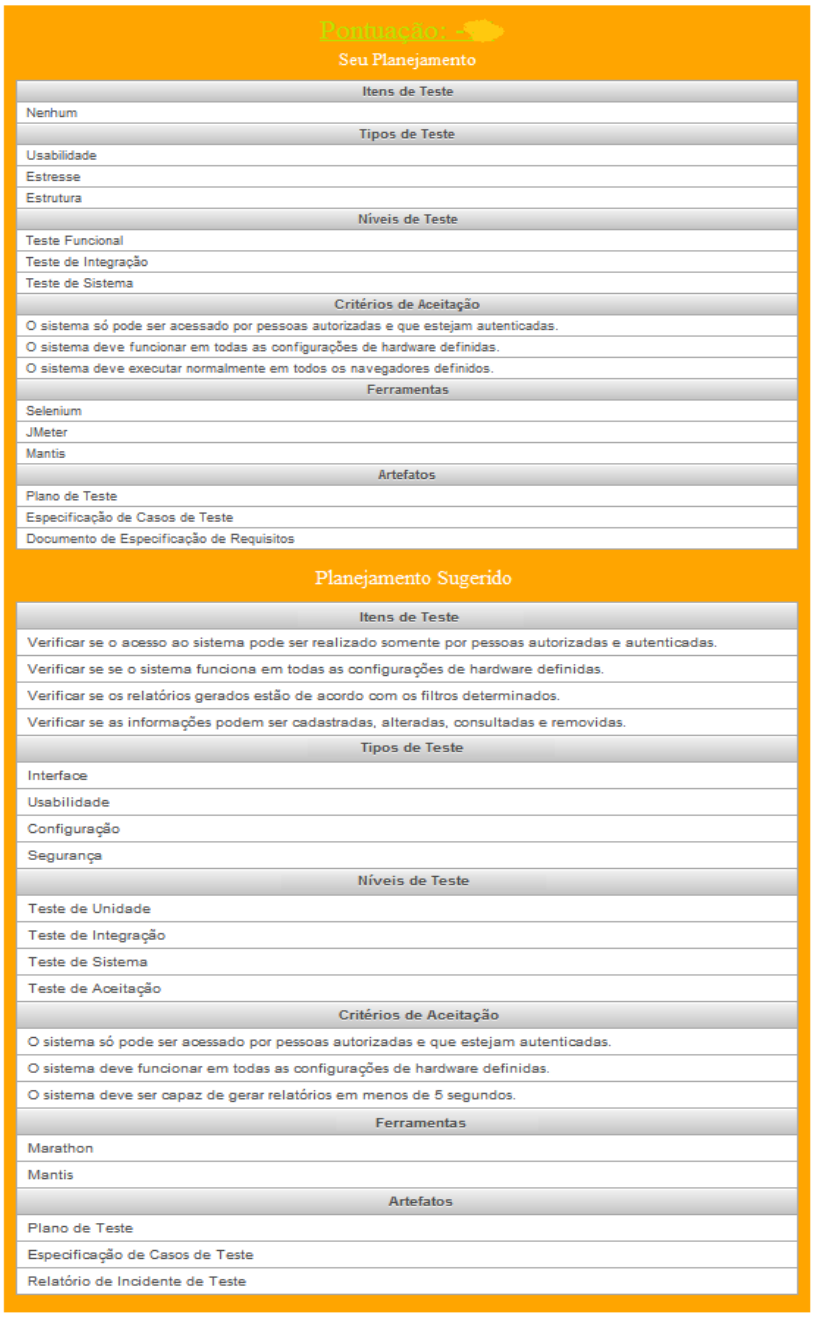

Figura 3: Tela do *feedback* (Planejamento do participante e sugerido)

3. Trabalhos Relacionados

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

Diante desse contexto, diversos jogos têm sido propostos para apoiar o ensino de Engenharia de Software [Gonçalves et al. 2011, Souza et al. 2010, Thiry et al. 2010, Monsalve et al. 2010]. Contudo, com relação ao ensino da disciplina Teste de Software só foi encontrado na literatura um jogo educacional, o Jogo das 7 Falhas [Diniz e Dazzi 2011], um jogo tutorial *web* Bug Hunt com lições, exercícios e *feedback* [Elbaum et. al. 2007] e o jogo educacional U-Test [Silva, 2010].

O Jogo das 7 Falhas tem por objetivo ensinar técnicas de caixa-preta e para isso o jogador deve identificar 7 falhas em uma determinada funcionalidade e relacionar as falhas identificadas à técnica de classe de equivalência ou valor-limite empregadas. Já com relação ao tutorial *web* Bug Hunt, ele apresenta conceitos relacionados a testes de caixa-branca e caixa-preta e apresenta alguns exercícios relacionados aos conceitos apresentados. O U-Test permite ao jogador assumir o papel de um testador responsável por escrever teste de unidade para funções já escritas de um sistema hipotético de forma a aplicar técnicas para seleção de dados de entrada para o teste de unidade.

Assim, existem na literatura jogos voltados para o ensino de Teste de *Software* em tópicos específicos, porém nesta pesquisa não foi identificada nenhuma referência relacionada ao aprendizado para o planejamento dos testes.

4. Conclusões e Trabalhos Futuros

Neste artigo, foi apresentado um jogo *web* para auxiliar no ensino de teste de software focado na fase de planejamento de forma a auxiliar no estímulo e motivação do aluno para o aprendizado do conteúdo ministrado, a partir da prática com simulação de problemas para possíveis sistemas.

O jogo desenvolvido, o *iTest Learning*, se destaca por focar em um tópico específico da área de teste, o planejamento. Com o jogo, espera-se que os alunos possam aprender, de forma prática, a construir um plano de teste para um dado projeto. Destaca-se que o jogo não exclui a participação do professor no processo de aprendizagem, mas ao contrário visa fornecer subsídios para que os professores possam acompanhar o desempenho dos alunos.

Como trabalhos futuros pretende-se a extensão do jogo para as fases de projeto e execução de testes, de modo que o aluno simule todas as fases envolvidas na atividade de testes de software. Também é previsto uma avaliação do jogo pelos alunos das disciplinas de Verificação e Validação do Curso de Engenharia de Software do campus Quixadá de forma a validar a eficácia do jogo como meio para favorecer o aprendizado dos conceitos ministrados de planejamento de testes de software.

Referências

Aldrich, C. (2005) “Learning by Doing: A Comprehensive Guide to Simulations, Computer Games, and Pedagogy in E-learning and Other Educational Experiences”. Hoboken: Wiley.

Diniz, L. L., Dazzi, R. L. S. (2011) “Jogo Digital para o Apoio ao Ensino do Teste de Caixa-Preta”. In: X Simpósio Brasileiro de Qualidade de Software, Curitiba.

Elbaum, S., Person, S. e Dokulil, J. (2007) “Bug hunt: making early software testing lessons engaging and affordable”. In: 29th International Conference on Software Engineering (ICSE'07), Minneapolis, p. 688 – 697.

Gonçalves, R. Q., Thiry, M., Zoucas, A. (2011) “Avaliação da Aprendizagem em Experimentos com Jogo Educativo de Engenharia de Requisitos”. In: X Simpósio Brasileiro de Qualidade de Software (SBQS), Curitiba.

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

Monsalve, E. S., Werneck, V. M. B., Leite, J. C. S. P. (2010) “SimulES-W: Um Jogo para o Ensino de Engenharia de Software”. In: III Fórum em Educação de Engenharia de Software (FEES), Simpósio Brasileiro de Engenharia de Software (SBES), Salvador, p. 17-26.

Myers, G. (2004) “The Art of Software Testing”. John Wiley & Sons, 2ª edição.

Silva, A. C. (2010). “Jogo Educacional para Apoiar o Ensino de Técnicas para Elaboração de Testes de Unidade”. Dissertação de Curso de Mestrado, Computação Aplicada, UNIVALI, São José.

Silva, A. R. (2011a) “Uma Metodologia de Testes em Software para Micro e Pequenas Empresas Estruturada em Multicritério”. Dissertação de Curso de Mestrado, Informática Aplicada, UNIFOR, Fortaleza.

Silva, T. G.; Müller, F. M.; Bernardi. G. (2011b) “Panorama do Ensino de Engenharia de Software em Cursos de Graduação Focado em Teste de Software: Uma Proposta de Aprendizagem Baseada em Jogos”. In RENOTE - Revista Novas Tecnologias na Educação, ISSN 1679-1916.

Souza, M. M., Resende, R. F., Prado, L. S., Fonseca, E. F., Carvalho, F. A., Rodrigues, A. D. (2010) “SPARSE: Um ambiente de Ensino e Aprendizado de Engenharia de Software Baseado em Jogos e Simulação”. In: XXI Simpósio Brasileiro em Informática na Educação, João Pessoa.

Thiry, M., Zoucas, A., Gonçalvez, R. Q. (2010) “Promovendo a Aprendizagem de Engenharia de Requisitos de Software através de um Jogo Educativo”. In: XXI Simpósio Brasileiro em Informática na Educação, João Pessoa.

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.