Jogo para o Apoio ao Ensino do Teste de Caixa-Preta

Lucio L. Diniz, Rudimar L. S. Dazzi

Mestrado em Computação Aplicada – Universidade do Vale do Itajaí (UNIVALI)
Itajaí – SC – Brasil

luciold2001@yahoo.com.br, rudimar@univali.br

Abstract. *Companies are increasingly investing in product quality, and one of the ways of guaranteeing this is through the activity of software testing. Despite its importance, software testing receives little attention in undergraduate curricula, and just a few hours being devoted to the teaching of it. There is also another factor, which is the difficulty of teaching software testing, either because of the lack of time, or because of the absence of practice, or because of the difficulty of motivating students. As a means of contributing to solving this problem, this paper proposes the use of a game as a teaching and learning strategy. The results from the quantitative and qualitative evaluations applied suggest that the game that was developed may be an efficient teaching technique to be used in teaching black-box testing.*

Resumo. *As empresas estão investindo, cada vez mais, na qualidade do produto, e uma das maneiras de garanti-la é através da atividade de teste de software. Apesar dessa importância, o teste de software recebe pouca atenção nos currículos de graduação, sendo poucas as horas dedicadas ao seu ensino. Além disso, existe outro fator, que é a dificuldade de ensinar teste de software, seja pela falta de tempo disponível, seja pela ausência de atividades práticas, ou pela dificuldade de se motivarem os alunos. Como forma de contribuir para a resolução desse problema, a iniciativa proposta por este artigo é a utilização de um jogo como estratégia de ensino e aprendizagem. O resultado da avaliação quantitativa e qualitativa realizada sugere que o jogo desenvolvido pode ser uma eficiente técnica de ensino a ser utilizada no ensino de teste de caixa-preta.*

1. Introdução

Atualmente, exige-se cada vez mais que as empresas construam *software* com menos defeitos e mais qualidade. Dessa forma, a qualidade passou de um diferencial entre as empresas para ser uma exigência do mercado, já que é um dos fatores críticos para o sucesso do negócio, a satisfação do cliente e a aceitação do produto. A falta de qualidade pode resultar em prejuízos financeiros, em perdas de imagem e de clientes, em insatisfação dos usuários e danos ao meio ambiente, podendo, inclusive, resultar em mortes (Laport *et al.*, 2007). Em 2002, o National Institute for Science & Technology (NIST) realizou um estudo em que foi estimado que falhas em *software* custam à economia dos Estados Unidos 59,5 bilhões de dólares por ano, o que leva à conclusão de que é necessário um maior investimento na qualidade de *software* (NIST, 2002).

Várias técnicas são utilizadas com o objetivo de garantir a qualidade de *software*, sendo que este artigo abordará a técnica de teste de software. O teste de software é uma importante técnica utilizada para garantir e melhorar a qualidade do software (Mary, 2000; Chen; Poon, 2004), tendo se tornado uma parte importante e valiosa dentro do

ciclo de vida do desenvolvimento de software. Segundo Myers (2004, p. 6), teste "é o processo de executar um programa com a intenção de encontrar erros".

Existem duas abordagens para o teste de *software*: uma utiliza a técnica de teste de caixa-preta; a outra, a técnica de teste de caixa-branca. Na técnica de teste de caixa-preta não é necessário nenhum conhecimento da lógica interna do sistema para se construirem os casos de testes (Perry, 2006). Já na técnica de teste de caixa-branca é necessário o conhecimento da lógica interna do sistema, para se desenvolverem os casos de testes (Perry, 2006).

Este artigo focará apenas a técnica de teste de caixa-preta, por se a mais utilizada na indústria (Chen; Poon, 1998; Kaner; Padmanabhan, 2007), além de ser mais eficaz na detecção de defeitos do que a técnica de teste de caixa-branca, conforme sugerem alguns estudos realizados (Davis, 1993; Kamsties; Lott, 1995).

Apesar da importância dos testes na garantia da qualidade do produto, a atividade de teste de *software* tem sido pouco explorada nos cursos de graduação (Carrington, 1997; Jones; Chatmon, 2001; Shepard *et al*., 2001; Chen; Poon, 2004, Elbaum *et al.*, 2007). Segundo Beizer (1990), 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.

Além de ter um nível de cobertura muito baixo na maioria dos currículos da Ciência da Computação e da Engenharia de *Software* (Shepard *et al.*, 2001), a atividade de teste tem sido ensinada somente no final do processo de aprendizagem desses cursos. Alguns autores defendem que princípios e técnicas de testes sejam integrados mais cedo ao currículo de graduação (Patterson *et al*., 2003; Shepard *et al.*, 2001; Elbaum *et al.*, 2007).

Outro problema levantado no ensino de teste de *software* nos cursos de graduação é a falta de atividades práticas. Como na programação, o teste exige experiência prática para complementar os princípios do ensino (Carrington, 1997), bem como para desenvolver atitudes e habilidades necessárias para a sua aplicação na vida profissional (Jones; Chatmon, 2001). Além disso, existe uma dificuldade de ensinar teste de *software* dentro do período de duração do curso (Kaner; Padmanabhan, 2007).

Várias soluções têm sido propostas para enfrentar esses problemas. Por exemplo, alguns autores propõem a seguinte metodologia de ensino: palestra, seguida de tutorial com exemplos ou exercícios (Kaner, 2004; Kaner; Padmanabhan, 2007). Outros autores, além das palestras e tutoriais, acrescentam, ainda, uma aula prática utilizando um projeto, para que os alunos derivem casos de testes a partir de especificações fornecidas (Chen; Poon, 1998; Chen; Poon, 2004; Murnane *et al*., 2007). Carrigton (1997), além de um projeto prático para elaboração de casos de testes, também inclui um projeto para a execução dos casos de testes derivados utilizando funcionalidades implementadas. Elbaum *et al.* (2007) propõem a criação de um tutorial web, com várias lições, exercícios práticos e *feedback*.

Para contribuir para a solução dos problemas expostos anteriormente, foi proposto neste artigo a utilização de um jogo educacional digital, chamado de Jogo das 7 Falhas, como estratégia de ensino. Jogos educacionais são jogos 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). Os jogos educacionais podem reduzir o tempo de formação, oferecer mais oportunidades para a prática e, consequentemente, aumentar a aquisição de conhecimentos (Brownfield; Vick, 1983; Ricci, 1994). Os jogos também se têm mostrado eficazes quando desenvolvidos para ensinar uma determinada habilidade (Griffiths, 2002) e para estimular a motivação e o interesse do aluno (Malone, 1983; Dempsey et al., 1994), já que permitem o **aprender fazendo** de situações reais com fornecimento de *feedback*.

Na próxima seção, é apresentado o Jogo das 7 Falhas. A seção 3 apresenta o resultado da avaliação qualitativa e quantitativa realizada. Na seção 4 são apresentadas as conclusões do artigo.

2. O Jogo das 7 Falhas

O jogo consiste em descobrir as falhas existentes nas funcionalidades de um software a ser testado em menos tempo possível. O jogo possui dois níveis de complexidade, baixa e média. Cada nível é composto por uma funcionalidade onde existem sete falhas a serem descobertas. O jogador só passará para o nível 2 caso descubra as sete falhas existentes no nível 1 dentro do tempo estimado (25 minutos para o nível 1; 40 minutos para o nível 2). Caso o tempo se esgote antes de o jogador identificar as sete falhas em cada nível, o jogo se encerra, e o jogador é eliminado. Caso o jogador descubra as sete falhas do nível 1 e as sete falhas do nível 2 dentro do tempo, o jogador é o vencedor, tendo alcançado o objetivo proposto pelo jogo.

O Jogo das 7 Falhas é um jogo single-player, no qual o jogador assume o papel de testador em uma equipe de teste de software de uma empresa fictícia chamada Diniz Quality Assurance, com a finalidade de descobrir as sete falhas existentes em cada funcionalidade testada, correlacionando as falhas com uma classe de equivalência ou um valor-limite.

Na tela inicial de boas vindas (Figura 1) é solicitado ao jogador que leia as regras do jogo (botão Regras) e os requisitos das funcionalidades a serem testadas (botão Requisitos). Também é solicitado, após a leitura dessas informações, que o jogador faça o seu login para iniciar o jogo. Ao pressionar o botão Regras, o jogador é apresentado à descrição da tarefa que deverá ser executada. Ela consiste do objetivo do jogo, de quanto tempo o jogador dispõe para alcançar o objetivo, das regras do jogo e de como a pontuação final é calculada. O objetivo do jogo é bem claro: encontrar as sete falhas dentro de cada nível o mais rápido possível. Ao pressionar o botão Requisitos, o jogador é apresentado aos requisitos das funcionalidades que deverão ser testados e validados. Por exemplo: características dos campos, regras de negócios, regras de botões e mensagens de erros e de sucesso que deverão ser apresentadas em determinadas situações.

A tabela com o ranking dos jogadores foi feito com a intensão de motivar os alunos a buscar o máximo de pontos, o que pode ser conseguido com aprofundamento nos conhecimentos e no desempenho na descoberta dos problemas a serem testados.

Figura 1. Tela de boas vindas

Após efetuar o login, o jogador recebe o título de Analista de Teste Júnior, sendo exibida a tela de início do jogo (nível 1). Neste momento, também são sorteadas as sete falhas que farão parte da jogada. A cada novo login do usuário, as sete falhas são sorteadas aleatoriamente dentre as 33 possíveis. O sorteio teve como objetivo proporcionar aos jogadores novos desafios e motivá-los a jogarem o jogo mais vezes, já que, a cada jogada, eles terão um novo desafio a completar.

Ao iniciar o jogo, o jogador terá 25 minutos para identificar as sete falhas existentes na funcionalidade Cadastro de Usuário, que estarão em desacordo com os requisitos da funcionalidade. Para identificar as falhas, o usuário irá executar os casos de testes elaborados anteriormente através da utilização das técnicas de classe de equivalência e análise de valor-limite. A cada caso de teste executado que não originar uma falha, será exibida, na parte inferior, a mensagem correspondente ao resultado esperado e um feedback informando a qual classe de equivalência ou valor-limite o caso de teste executado corresponde.

Para cada caso de teste executado que originar uma falha, será exibido, na parte inferior, uma mensagem informando que uma falha foi descoberta e que é necessário selecionar a classe de equivalência ou o valor-limite que a originou. Também será exibida, abaixo da mensagem, uma lista com sete classes de equivalência e/ou valores-limite, para que o jogador selecione, entre elas, a que deu origem à falha simulada. A Figura 2 corresponde a um exemplo onde foi executado um caso de teste para simular essa falha.

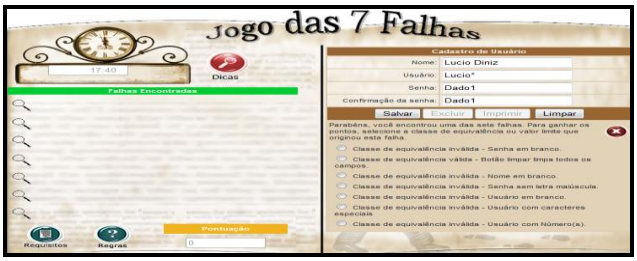

Figura 2. Tela exibida após a simulação de uma falha

O jogador, após simular uma falha, deverá, então, selecionar a classe de equivalência ou o valor-limite que originou a falha. Caso o jogador selecione a classe de equivalência ou o valor limite correto, ele receberá dez pontos. A classe de equivalência ou o valor-limite selecionado irá para a parte esquerda da tela, no item falhas encontradas, e será exibido o feedback “Parabéns você selecionou a classe de equivalência ou valor limite que originou a falha e ganhou 10 pontos.” Caso o jogador selecione a classe de equivalência ou o valor-limite errado, ele perderá dez pontos, e será exibido o feedback

"Infelizmente você não selecionou a classe de equivalência ou valor limite que originou a falha e perdeu 10 pontos."

Caso o jogador não encontre as sete falhas do nível 1 dentro dos 25 minutos, será exibido um feedback, informando que o tempo expirou, sendo o jogador eliminado. Caso o jogador encontre as sete falhas dentro dos 25 minutos, ele será promovido à analista de teste pleno, e passará para o segundo nível do jogo. Além disso, o jogador ganhará um ponto a cada intervalo de 10 segundos que sobrar do tempo disponível. Por exemplo: se o jogador descobrir as sete falhas em 24 minutos, ou seja, 60 segundos menos que o tempo disponível, ele ganhará mais seis pontos. Essa recompensa foi colocada para diferenciar os jogadores que descobriram as sete falhas em menos tempo (lembrando que o objetivo do jogo é descobrir as sete falhas em cada nível o mais rápido possível).

A mecânica do nível 2 é a mesma do nível 1. A diferença realmente está no grau de complexidade das falhas; por isso, não é apresentado em mais detalhes. Da mesma forma que no nível 1, caso o jogador não encontre as sete falhas do nível 2 em 40 minutos, será eliminado. Caso descubra as sete falhas do nível dois dentro do intervalo de 40 minutos, será promovido a analista de teste sênior e sairá como vencedor, ganhando também um ponto extra a cada intervalo de 10 segundos que sobrar dos 40 minutos.

3. Resultados Obtidos

O experimento foi aplicado com 21 alunos do curso de Ciência da Computação da UNIVALI, em Itajaí, nos dias 04/11/2010 e 11/11/2010. O experimento foi dividido em duas partes. A primeira parte do experimento, ocorrida no dia 04/11/2010, teve a duração de 3 horas e 30 minutos. Após o esclarecimento sobre os objetivos do experimento e sobre a livre participação de todos, foi iniciada a aula sobre teste de software. O conteúdo da aula de teste de software foi apresentado em duas horas e 15 minutos.

Após a conclusão da explanação do conteúdo sobre teste de software, os participantes foram apresentados à especificação de requisitos das funcionalidades Cadastro de Usuário e Cadastro de Material. Também foram informados de que a próxima atividade a ser realizada seria derivar casos de testes a partir desses requisitos, utilizando as técnicas de classe de equivalência e análise de valor-limite, vistas anteriormente. Para a execução dessa atividade, foi dado o tempo de 1 hora. Ao final deste período, os participantes foram dispensados.

A segunda parte do experimento ocorreu no dia 11/11/2010 e teve a duração de 3 horas. Após os participantes ocuparem seus lugares, os mesmos foram instruídos a responder uma avaliação sobre o conteúdo ministrado na primeira parte do experimento. Após as devidas instruções sobre a avaliação, foi passado aos participantes o pré-teste. O pré-teste teve a duração de 30 minutos. Após a realização do pré-teste foi feita a divisão do grupo através do sorteio, sendo o grupo A experimental e o grupo B de controle.

O grupo A (experimental) permaneceu na sala do laboratório, onde foram instruídos sobre a utilização, as regras e os objetivos do Jogo das 7 Falhas. O grupo B (controle) foi para outro laboratório, onde foram aplicados os exercícios sobre classe de

equivalência e análise de valor limite, acompanhados de outro profissional, que teve a função de orientá-los.

A atividade do jogo realizada pelo grupo experimental teve a duração aproximada de 1 hora. Após a finalização da execução do jogo, os participantes preencheram um formulário de avaliação do jogo. Após o preenchimento da avaliação do jogo, os participantes foram informados da realização do pós-teste. O pós-teste teve a duração de trinta minutos, e, após a sua finalização, os participantes foram liberados.

A atividade realizada pelo grupo de controle teve a duração aproximada de 1 hora. Após a sua finalização, os participantes foram informados da realização do pós-teste, que teve a duração de 30 minutos.

3.1. Avaliação Quantitativa

Após a realização do experimento, as provas do pré-teste e do pós-teste foram corrigidas, e os dados coletados foram tabulados, de forma a permitir a realização das análises e testes das hipóteses de pesquisa. A primeira hipótese de pesquisa testada foi o efeito de aprendizagem nos níveis de conhecimento, compreensão e aplicação do grupo experimental não é superior ao do grupo de controle (H0).

As notas obtidas pelos participantes do grupo experimental e de controle estão disponíveis na Tabela 1, onde se encontram o número do participante de cada grupo, as notas obtidas no pré-teste e no pós-teste, além da diferença entre essas notas.

Tabela 1. Notas do pré-teste e do pós-teste do grupo experimental e de controle

Grupo Experimental				Grupo de Controle			
Participante	Pré-teste	Pós-teste	Diferença	Participante	Pré-teste	Pós-teste	Diferença
1	6,8	10,0	3,2	1	6,0	3,8	-2,2
2	10,0	10,0	0,0	2	10,0	8,6	-1,4
3	4,6	5,4	0,8	3	10,0	9,0	-1,0
4	9,6	8,6	-1,0	4	9,0	5,2	-3,8
5	6,2	9,0	2,8	5	9,2	8,8	-0,4
6	9,6	10,0	0,4	6	8,2	6,4	-1,8
7	9,6	9,6	0,0	7	4,4	7,2	2,8
8	5,2	7,2	2,0	8	10,0	9,6	-0,4
9	8,6	9,0	0,4	9	7,0	5,8	-1,2
10	5,8	6,8	1,0	10	4,2	5,8	1,6
				11	7,6	6,8	-0,8

De posse dos dados da Tabela 1, foi aplicado o teste estatístico Mann-Whitney referente ao efeito de aprendizagem relativo. Para o teste do efeito de aprendizagem relativo, foram utilizadas as diferenças dos resultados do pré-teste e do pós-teste de ambos os grupos. O primeiro passo foi classificar as diferenças, ignorando o grupo ao qual elas pertenciam. Para realizar a classificação, todos os resultados das diferenças foram colocados em ordem ascendente, iniciando-se em 1, a partir do menor resultado. No caso de haver duas ou mais diferenças iguais, a classificação foi obtida através do cálculo da média das suas posições.

Uma vez classificadas as diferenças, os dados foram transferidos para outra tabela, que separa a classificação para cada grupo. Tais dados estão demonstrados na Tabela 2.

Tabela 2: Classificação das diferenças por grupo

Grupo Experimental			Grupo de Controle		
Participante	Diferença	Classificação	Participante	Diferença	Classificação

1	3,2	21	1	-2,2	2
2	0,0	11,5	2	-1,4	4
3	0,8	15	3	-1,0	6,5
4	-1,0	6,5	4	-3,8	1
5	2,8	19,5	5	-0,4	9,5
6	0,4	13,5	6	-1,8	3
7	0,0	11,5	7	2,8	19,5
8	2,0	18	8	-0,4	9,5
9	0,4	13,5	9	-1,2	5
10	1,0	16	10	1,6	17
			11	-0,8	8

No passo seguinte, os valores das classificações do grupo experimental foram calculados sob a identificação **T1**, e os valores do grupo de controle foram calculados sob a identificação **T2**. A soma das classificações do grupo experimental gerou o valor **T1**; a soma das classificações do grupo de controle gerou o valor **T2**. O detalhamento do cálculo é apresentado na Tabela 3.

Tabela 3: Cálculo de T1 e T2

T1 = 21 + 11,5 + 15 + 6,5 + 19,5 + 13,5 + 11,5 + 18 + 13,5 + 16 = 146
T2 = 2 + 4 + 6,5 + 1 + 9,5 + 3 + 19,5 + 9,5 + 5 + 17 + 8 = 85

Após calculados os valores de T1 e T2, foi selecionado o maior resultado. Nesse caso, o resultado da maior classificação é T1 com valor 146. O valor da maior classificação será utilizado na variável Tx da fórmula do cálculo do valor de U. Os valores de n1, n2 e nx foram obtidos da seguinte forma:

a)**n1:** número de participantes do grupo experimental;
b)**n2:** número de participantes do grupo de controle;
c)**nx:** número de participantes do grupo com maior somatório de classificação.

No caso do experimento realizado, o valor de n1 é igual a 10, n2 é igual a 11 e nx é igual a 10, pois havia 10 participantes no grupo experimental (n1), 11 participantes no grupo de controle (n2) e o grupo com maior somatório de classificação (nx) foi o grupo experimental com 10 participantes. De posse dos valores dessas variáveis, foi realizado o cálculo de U.

$$U = n1 * n2 + nx * (nx + 1) / 2 – Tx$$

O valor calculado para U foi 19. O passo seguinte foi utilizar a tabela de valores críticos de U para o teste de Mann-Whitney. Na tabela de valores críticos de U (MATH.USASK, 2010), o valor da interseção entre n1 igual a 10 e n2 igual a 11 é 26.

Para que o resultado do teste seja significativo, o valor de U obtido deve ser igual ou menor que o valor crítico de U. Como o valor calculado de U (19) é ligeiramente inferior ao valor crítico (26), a conclusão é a de que o efeito de aprendizagem relativo entre os grupos experimental e grupo de controle são significativamente diferentes. Dessa forma, pode se afirmar com 95% de certeza que, para esse experimento, a hipótese H0 não é verdadeira, o que evidencia, sob as condições descritas nesse experimento, uma melhora significativa do efeito de aprendizagem relativo do grupo experimental em relação ao grupo de controle.

Além da avaliação realizada com enfoque no efeito de aprendizagem relativo, foi realizada a avaliação do efeito de aprendizagem absoluto. Da mesma forma, foi aplicado o teste estatístico de Mann-Whitney. O primeiro passo foi classificar as notas do pós-teste, ignorando o grupo ao qual elas pertenciam. Em seguida os dados foram

transferidos para outra tabela, que separa a classificação para cada grupo. Tais dados estão demonstrados na Tabela 4.

Tabela 4: Classificação das notas por grupo

Grupo Experimental			Grupo de Controle		
Participante	**Nota**	**Classificação**	**Participante**	**Nota**	**Classificação**
1	10,0	20	1	3,8	1
2	10,0	20	2	8,6	11,5
3	5,4	3	3	9,0	15
4	8,6	11,5	4	5,2	2
5	9,0	15	5	8,8	13
6	10,0	20	6	6,4	6
7	9,6	17,5	7	7,2	9,5
8	7,2	9,5	8	9,6	17,5
9	9,0	15	9	5,8	4,5
10	6,8	7,5	10	5,8	4,5
			11	6,8	7,5

Da mesma forma que no teste anterior foram feitos os cálculos e o valor calculado para U foi 26. O passo seguinte foi utilizar a tabela de valores críticos de U para o teste de Mann-Whitney. Para que o resultado do teste seja significativo, o valor de U obtido deve ser igual ou menor que o valor crítico. Como o valor de U é igual ao valor crítico, a conclusão é a de que o efeito de aprendizagem absoluto entre os grupos experimental e de controle é significativamente diferente. Dessa forma, pode-se afirmar com 95% de certeza que, sob as condições descritas nesse experimento, ocorreu uma melhora significativa no efeito de aprendizagem absoluto do grupo experimental em relação ao de controle..

3.2. Avaliação Qualitativa

Com o objetivo de efetuar a avaliação qualitativa do Jogo das 7 Falhas, foi aplicado aos participantes um questionário de avaliação do jogo com as seguintes perguntas:

a) "Você achou o jogo agradável?"

b) "O jogo é atrativo e conseguiu motivá-lo a executar as tarefas?"

c) "Você gostou de jogar o jogo de teste de *software*?"

Para cada uma dessas questões do questionário de avaliação do jogo, o participante deveria apontar um valor de 1 a 4, correspondendo às respostas "não", "mais para não", "mais para sim" e "sim", respectivamente.

A Tabela 5 consolida as respostas dadas pelos participantes ao questionário de avaliação do Jogo das 7 Falhas.

Tabela 5: Resultado do Questionário de Avaliação do Jogo das 7 Falhas

Perguntas	**Respostas dos Participantes**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Você achou o jogo agradável?	4	4	4	4	3	4	4	4	3	3
O jogo é atrativo e conseguiu motivá-lo a executar as tarefas?	4	4	4	3	4	4	4	3	3	4
Você gostou de jogar o jogo de teste de *software*?	4	4	4	3	4	4	3	4	3	4

Os dados demonstram que 30% dos participantes responderam "mais para sim" e 70% responderam "sim" para as perguntas, o que indica que os participantes consideraram o jogo agradável, atrativo, e que gostaram de jogá-lo.

4. Conclusões

Embora o teste de software seja uma das técnicas mais utilizadas para garantir a qualidade de software, pouca atenção tem sido dada a essa atividade nos currículos de graduação. Além disso, os cursos oferecidos enfrentam algumas dificuldades, tais como: falta de atividades práticas, dificuldade para motivar os alunos, falta de tempo para a transmissão do conhecimento e dificuldades para ensinar as habilidades de elaboração e execução de casos de teste. A partir dessa motivação, buscou-se a concepção, a implementação e a avaliação de um jogo educacional a ser utilizado como apoio ao ensino de teste de software.

As avaliações do jogo apresentadas neste artigo indicam o mesmo como uma atividade efetiva de ensino, capaz de motivar os alunos, podendo servir, dessa forma, como apoio ao instrutor no ensino do teste de caixa-preta. É importante ressaltar ainda, a necessidade de se aplicar o jogo em conjunto com outras estratégias de ensino (aula expositiva e projeto de derivação de casos de testes a partir dos requisitos que serão utilizados no jogo.

Referências

Beizer, B. Software testing techniques. Second Edition Van Nostrand Reinhold, 1990.

Brownfield, S.; Vik, G. Teaching basic skills with computer games. **Training and Developmental Journal**, v. 37, n. 2, p. 52-56, 1983.

Carrington, D. Teaching software testing. **Proceeding of the 2nd Australian conference on Computer science education**. Australia, 1997. p. 59-64.

Chen, T. Y.; Poon, P. L. Experience with teaching black-box testing in a computer science/software engineering curriculum. **IEEE Transactions on Education**, v. 47, n. 01, p. 42-50, 2004.

Chen, T. Y.; Poon, P. L. **Teaching black box testing**. Proceedings of the 1998 International Conference on Software Engineering: Education & Practice, 1998. p. 324.

Davis, A. **Software requirements**: objects, functions and states, revision. New Jersey: Prentice Hall, Upper Saddle River, 1993.

Dempsey, J. V.; Rasmussen, K.; Lucassen, B. **Instructional gaming:** implications for instructional technology. Paper presented at the Annual Meeting of Association for Educational Communications and Technology, Nashville, 1994.

Elbaum, S. *et al.* Bug hunt: making early software testing lessons engaging and affordable. In: 29th INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE'07), 2007.

Griffiths, M. D. The educational benefits of videogames. **Education and Health**, v. 20, n. 3, p. 47-51, 2002.

Jones, E. L.; Chatmon, C. L. A perspective on teaching software testing. **Proceedings of the seventh annual consortium for computing in small colleges**, p. 92-100, 2001.

Kamsties, E.; Lott, C. An empirical evaluation of three defect-detection techniques. **Proceedings of the 5th European Software Engineering Conference**, p. 362-383, 1995.

Kaner, C. Teaching domain testing: a status report. Proceedings of the 17th Conference on Software Engineering Education and Training table of contents, **ACM**, p. 112-117, 2004.

Kaner, C.; Padmanabhan, S. Practice and transfer of learning in the teaching of software testing. **Proceedings of the 20th Conference on Software Engineering Education & Training**, p. 157-166, 2007.

Laporte, C. Y.; April, A.; Bencherif, K. **Teaching software quality assurance in an undergraduate software engineering program**. Software Quality Professional, 4-10, 2007.

Malone, T. C. Guidelines for designing educational computer programs. **Childhood Education**, v. 59, n. 4, p. 241-47, 1983.

Mary, J. H. Testing: a roadmap. international conference on software engineering. **Proceedings of the Conference on The Future of Software Engineering**, Limerick, Ireland June 04-11, 2000.

MATH.USASK. Disponível em: <http://math.usask.ca/~laverty/S245/Tables/wmw.pdf> Acesso em: 20 maio 2010.

Murnane, T.; Reed, K.; Hall, R. On the Learnability of Two Representations of Equivalence Partitioning and Boundary Value Analysis. **Proceedings of the 2007 Australian Software Engineering Conference**, p. 274-283, 2007.

Myers, G. J. **The art of software testing**. Wiley: Second Edition, 2004.

NIST - National Institute for Science & Technology - Planning Report 02-3: **The economic impact of inadequate infrastructure for software testing**. Prepared by RTI for the National Institute of Standards & Technology, USA, May 2002. Disponível em: <http://www.nist.gov/director/prog-ofc/report02-3.pdf>. Acesso em: 10 fev. 2011.

Patterson, A.; Kolling, M.; Rosenberg, J. Introducing unit testing with BlueJ. **Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'03)**, Thessaloniki, Greece, 2003, p 11-15.

Perry, W. E. **Effective methods for software testing**. Indianápolis, Indiana: Third Edition, Wiley, 2006.

Ricci, K. E. The use of computer-based videogames in knowledge acquisition and retention. **Journal of Interactive Instruction Development**, n. 7, v. 1, p. 17-22, 1994.

Shepard, T.; Lamb, M.; Kelly, D. More Testing Should be Taught. **Communications of the ACM**, v. 44, n. 06, p. 103-108, 2001.

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.