# Jogo Digital 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

***Abstract.*** *Despite its importance in quality assurance, software testing receives little attention in undergraduate curricula, only a few hours being devoted to teaching it. Besides this, there is another factor, which is the difficulty of teaching s*oftware testing*, either because of the lack of time, or because of the absence of practice, or because of the difficulty of motivating students. To contribute to the resolution of this problem is proposed in this paper the use of game as a strategy for teaching and learning. The results from the quantitative and qualitative evaluations applied to Computer Science students suggest that the game that was developed may be an efficient teaching technique to be used in teaching black-box testing.*

***Resumo.*** *Apesar da importância na garantia da qualidade, 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. Para contribuir com a resolução desse problema, é proposta neste artigo a utilização de um jogo como estratégia de ensino e aprendizagem. O resultado da avaliação quantitativa e qualitativa aplicadas a alunos do curso de Ciência da Computação sugerem 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 funcional; a outra, a técnica de teste estrutural. A técnica de teste funcional é conhecida como técnica de teste de caixa-preta porque não é necessário nenhum conhecimento da lógica interna do sistema para se construirem os casos de testes (Perry, 2006). Nessa técnica, os casos de teste são construídos a partir de uma especificação do programa (Chen; Poon, 2004). Entre os principais métodos de seleção de casos de testes da técnica funcional, estão: classes de equivalência e análise de valor-limite (Jorgensen, 1995). Já a técnica estrutural recebe o nome de teste de caixa-branca, porque nela é necessário o conhecimento da lógica interna do sistema, para se desenvolverem os casos de testes (Perry, 2006). Nesta última, os casos de teste são construídos a partir do código fonte do programa (Chen; Poon, 2004).

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 (Basili; Selby, 1987; 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, existindo a necessidade da sua inclusão nos currículos dos cursos de graduação (Carrington, 1997; Jones; Chatmon, 2001; Shepard *et al*., 2001; Lamb; Kelly, 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 (Jones, 2001; 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*., 2005; 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, 1980; 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

Nesta seção é apresentada uma breve descrição do funcionamento do Jogo das 7 Falhas e da modelagem utilizada. Com isso pretende-se esclarecer a funcionalidade e mecânica do jogo e também dar uma visão de como ele foi modelado.

### 2.1. Funcionamento do 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.

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.

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.

A tela inicial do jogo, conforme pode ser vista na Figura 1, é dividida em duas partes:

1. Lado esquerdo – Nele se encontram: os controles do jogo; o relógio, que já estará em andamento (será iniciado com 25 minutos); um botão de dicas e a lista de falhas, que é preenchida à medida que as falhas são descobertas. Também nesse lado ficam a pontuação do jogo e os botões **Requisitos**, onde estarão listados os requisitos da funcionalidade do nível 1, e **Regras**, contendo o objetivo e as regras do jogo.
2. Lado direito – Na sua parte superior se encontra a funcionalidade a ser testada, que, no caso do nível 1, é o **Cadastro de Usuário**, com os campos **Nome**, **Usuário**, **Senha** e **Confirmação**, e os botões **Salvar** e **Limpar** (habilitados), além de **Excluir** e **Imprimir** (desabilitados). Na parte inferior, após uma falha ser simulada, são exibidas sete classes de equivalência e/ou valores-limite, para que o jogador possa selecionar uma delas. Ali, também, são apresentados o *feedback* e as mensagens, ao longo do jogo.


**Figura 1. Tela inicial do jogo (nível 1)**

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 exibida, 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.

Considere-se que um dos erros sorteados seja que o sistema esteja permitindo cadastrar um usuário com caracteres especiais, quando o requisito informa que isso não seria permitido. A Figura 2 corresponde a um exemplo onde foi executado o seguinte caso de

teste para simular essa falha: **Passo 1 – Preencher o campo Código com um valor válido; Passo 2 – Preencher o campo "Usuário" com um nome que possua pelo menos um caractere especial (!, @, #, %, *, ...); Passo 3 – Preencher o campo "Senha" com uma senha válida; Passo 4 – Preencher o campo "Confirmação" com o mesmo valor preenchido no campo "Senha"; Passo 5 – Pressionar o botão "Salvar"**. Este caso de teste tem como resultado esperado: **Exibir a mensagem: "O campo Usuário não pode conter caracteres especiais"**. Como este foi um dos sete erros sorteados, a mensagem não será exibida, sendo, então, simulada a falha. Após a execução desse caso de teste pelo jogador, será exibida, na parte inferior do lado direito, a seguinte mensagem: "Parabéns, você encontrou uma das sete falhas. Para ganhar os pontos, selecione a classe de equivalência ou valor limite que originou esta falha". Logo abaixo dessa mensagem, é exibida a lista com sete classes de equivalência/valores limite que foram sorteados aleatoriamente, contendo sempre a classe de equivalência ou valor-limite correto entre os sete, para que o jogador possa selecioná-lo.


**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.

A tela do nível 2, conforme pode ser vista na Figura 3, também é dividida em duas partes:

1. Lado esquerdo – Ali se encontram os controles do jogo e o relógio, que já estará em andamento. Neste caso, será iniciado com 40 minutos (os 15 minutos a mais no nível 2 em relação ao nível 1 são em função da quantidade de campos a serem testados e da complexidade das falhas a serem descobertas). Pode-se observar que, no nível 2, não existe o botão Dica, justamente para se ter um grau de dificuldade maior do que o apresentado no nível 1. Nesse lado, também fica a lista de falhas, que será preenchida à medida que as falhas forem sendo descobertas, a pontuação do jogo e os botões **Requisitos**, onde estarão listados os requisitos da funcionalidade do nível 2, e **Regras**, contendo o objetivo e as regras do jogo.
2. Lado direito – Na parte superior, encontra-se a funcionalidade a ser testada. No caso, é **Cadastro de Material**, com os campos **Código**, **Material**, **Tipo**, **Quantidade**, **Preço**, **Total** e **Data**, e os botões **Salvar** e **Limpar** (habilitados), além de **Excluir** e **Imprimir** (desabilitados). Na parte inferior, após uma falha ser simulada, são exibidas sete classes de equivalência e/ou valores-limite, para que o jogador possa selecionar um deles. Ainda na parte inferior, são apresentados o *feedback* e as mensagens, ao longo do jogo.

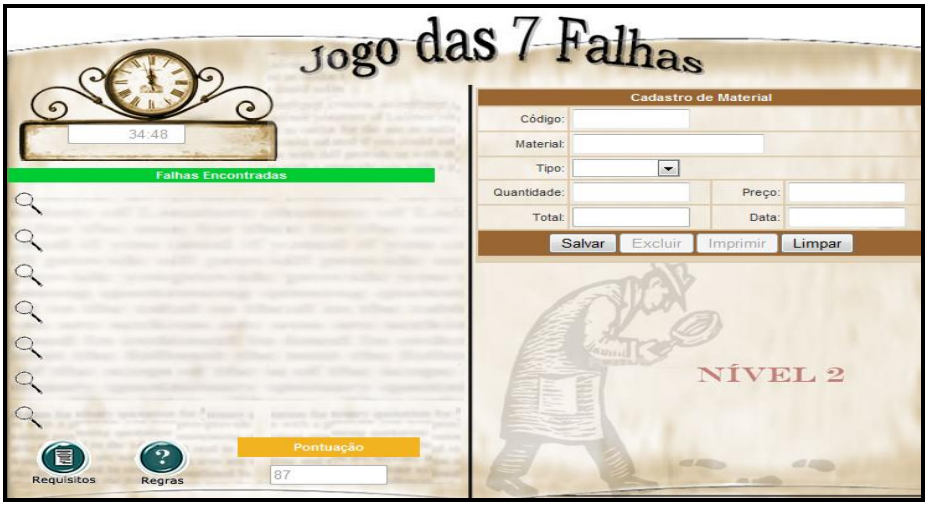


**Figura 3. Tela do nível 2**

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.

## 2.2. Modelagem Básica do Jogo

Nesta seção serão apresentados o diagrama de classe, a estrutura do banco de dados e a tecnologia utilizada na implementação do jogo.

O diagrama de classe descreve os tipos de objetos e suas relações estáticas. Nele são representadas as classes, com seus atributos, os métodos e suas

associações/relacionamentos (Pfleeger, 2001). Na Figura 4, é apresentado o diagrama de classes do Jogo das 7 Falhas.

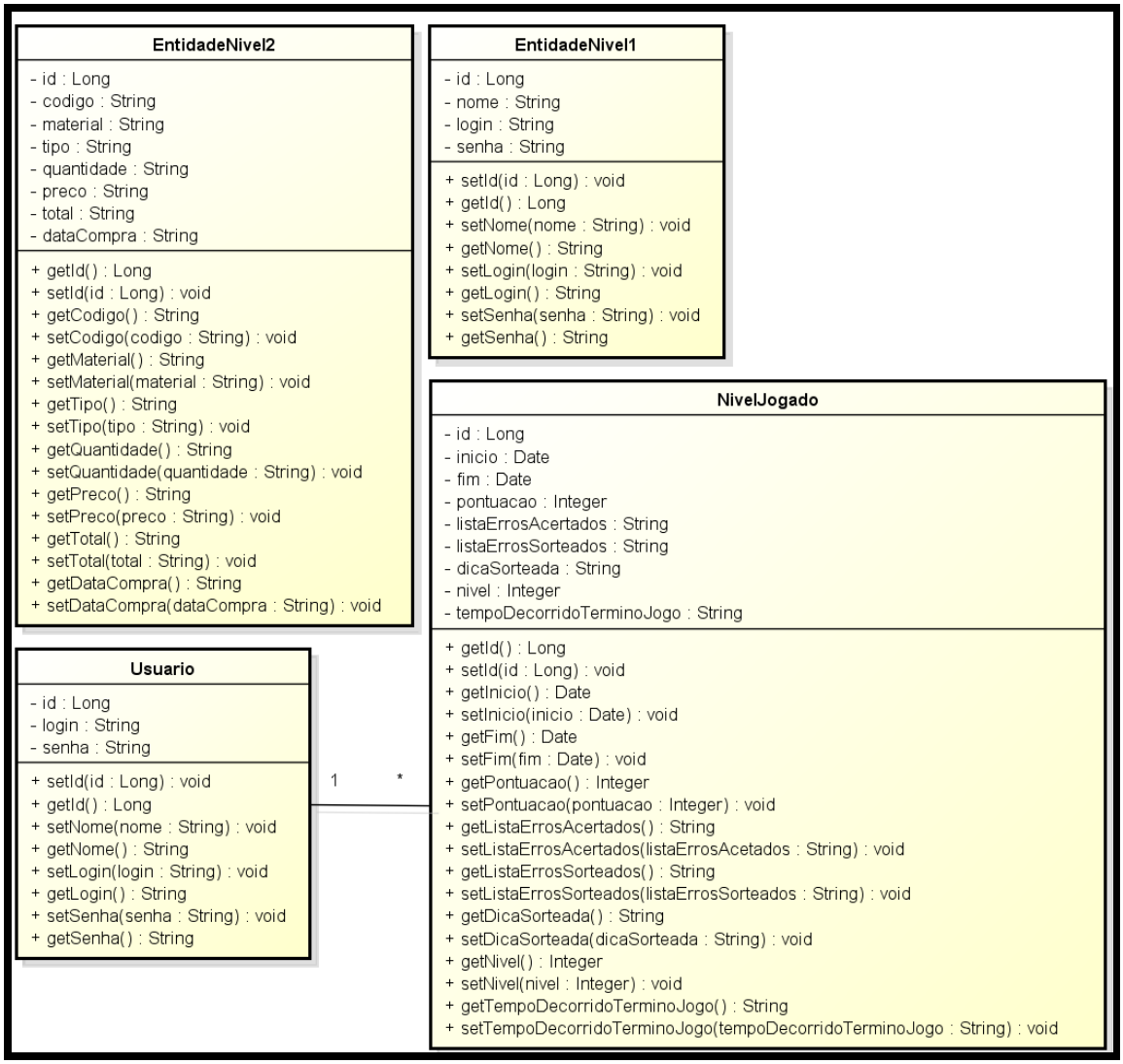


**Figura 4. Diagrama de Classe**

A estrutura de banco de dados utilizada é composta por três tabelas. A primeira, chamada **Niveljogado** e representada na Figura 5, foi implementada para armazenar informações dos jogadores, tais como: pontuação, dica utilizada, tempo, falhas encontradas e falhas sorteadas. A segunda, **Entidadenivel1**, foi utilizada para implementar a funcionalidade **Cadastro de Usuário** e está representada na Figura 6. A terceira, **Entidadenivel2**, foi utilizada para implementar a funcionalidade **Cadastro de Material** e está representada na Figura 7.

| Niveljogado | | |
|---|---|---|
| PK | Id | int |
| | DicaSorteada | varchar(255) |
| | Fim | datetime |
| | Inicio | datetime |
| | ListaErrosAcertados | varchar(255) |
| | ListaErrosSorteados | varchar(255) |
| | Nivel | int(11) |

| | | |
|---|---|---|
| | Pontuacao | int(11) |
| | TempoDecorridoTerminoJogo | varchar(255) |
| | Usuario_id | int(20) |

**Figura 5. Níveljogado**

| Entidadenivel1 | | |
|---|---|---|
| PK | Id | Int(20) |
| | Login | varchar(8) |
| | Nome | varchar(255) |
| | Senha | varchar(255) |

**Figura 6. Entidadenivel1**

| Entidadenivel2 | | |
|---|---|---|
| PK | Id | Int(20) |
| | Codigo | varchar(255) |
| | DataCompra | varchar(255) |
| | Material | varchar(255) |
| | Preco | varchar(255) |
| | Quantidade | varchar(255) |
| | Tipo | varchar(255) |
| | Total | varchar(255) |

**Figura 7. Entidadenivel2**

O Jogo das 7 Falhas foi implementado basicamente com a linguagem de programação Java, utilizando o *framework* Java Server Faces. Também foi utilizado o *framework* RichFaces para dar suporte às funcionalidades Ajax dentro do sistema. Quanto aos servidores de aplicação, foram utilizados os servidores Glasfish Server Application e o JBOSS Server Application. A arquitetura utilizada foi o Model-View-Controller (MVC), que é um padrão de arquitetutra de *software* que tem como objetivo separar a lógica de negócio da lógica de apresentação. Para o *design* da camada de visão, foram utilizadas páginas XHTML. Em relação ao banco de dados, foi utilizado o MySQL.

## 3. Resultado da Avaliação

Esta seção apresenta a descrição do experimento realizado, bem como os resultados obtidos através dele.

### 3.1. Descrição do Experimento

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. No primeiro dia foi ministrada a aula sobre teste de *software*. 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.

No dia 11/11/2010 os participantes foram instruídos a responder uma avaliação (pré-teste) sobre o conteúdo ministrado na primeira parte do experimento. 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. A atividade do jogo teve

a duração aproximada de 1 hora. Após a finalização da execução do jogo, os participantes do grupo A (experimental) preencheram um formulário de avaliação do jogo e em seguida realizaram o pós-teste.

O grupo B (controle) foi para outro laboratório, onde foram aplicados os exercícios práticos em papel abordando o mesmo conteúdo disponibilizado no jogo, acompanhados de outro profissional, que teve a função de orientá-los. A atividade dos exercícios teve a duração aproximada de 1 hora, e após a sua finalização, os participantes realizaram o pós-teste.

### 3.2. Resultado da 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 hipótese de pesquisa testada foi a $H_0$ (o efeito de aprendizagem nos níveis de conhecimento, compreensão e aplicação do grupo experimental não é superior ao do grupo de controle).

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**

| 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**.

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

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**.

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 $H_0$ 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 |

No passo seguinte, foram calculadas: a soma das classificações do grupo experimental (**T1),** e a soma das classificações do grupo de controle (**T2)**. O detalhamento do cálculo é apresentado na Tabela 5.

**Tabela 5: Cálculo de T1 e T2**

| |
|---|
| T1 = 20 + 20 + 3 + 11,5 + 15 + 20 + 17,5 + 9,5 + 15 + 7,5 = 139 |
| T2 = 1 + 11,5 + 15 + 2 + 13 + 6 + 9,5 + 17,5 + 4,5 + 4,5 + 7,5 = 92 |

O resultado da maior classificação é **T1** com o valor de 139. O valor da maior classificação será utilizado na variável **Tx** da fórmula do cálculo do valor de **U**. Em seguida, foi realizado o cálculo de **U**.

$$U = 10 * 11 + 10 * (10 + 1) / 2 – 139$$

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. 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. 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.

### 3.3. Resultado da 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” “sim”, respectivamente.

A Tabela 6 consolida as respostas dadas pelos participantes ao questionário de avaliação do Jogo das 7 Falhas.

**Tabela 6: Perguntas e respostas da avaliação do Jogo das 7 Falhas**

| **Respostas** | **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 | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 3 | 3 | 4 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| tarefas? | | | | | | | | | | |
| Você gostou de jogar o jogo de teste de *software*? | 4 | 4 | 4 | 3 | 4 | 4 | 3 | 4 | 3 | 4 |

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

Para se ter uma melhor avaliação do jogo, também foram incluídas algumas questões discursivas no questionário de avaliação do jogo:

a) "O que mais você gostou no jogo?"

b) "O que menos você gostou no jogo?"

c) "Você teria alguma sugestão de melhoria para o jogo?"

As respostas obtidas nessas questões indicam que aquilo de que os participantes mais gostaram no jogo foi implementar o aprendizado com algo interativo, encontrar erros testando os seus requisitos, o fato de o jogo simular um pequeno sistema com erros comuns e a interface gráfica do jogo. Em relação àquilo de que menos gostaram, mencionaram: a dificuldade de encontrar alguns erros e a tecla *Backspace* que, quando apertada indevidamente, retorna o jogo ao início. Entre as sugestões de melhoria, as mais citadas foram a criação de listas de testes já realizados e a criação de erros mais fáceis.

### 3.4. Discussão

Esta seção tem por objetivo tratar questões relativas às eventuais ameaças à validade do experimento realizado.

A primeira ameaça interna identificada foi a possibilidade de os participantes receberem o mesmo teste mais de uma vez, o que poderia fazer com que eles se familiarizassem com as respostas, obtendo, dessa maneira, uma nota maior no pós-teste do que no pré-teste. Para tratar/reduzir essa ameaça, embora se tenha mantido uma correspondência entre as questões do pré-teste e do pós-teste, foi efetuada uma ligeira alteração nos cenários e nos valores das questões.

A segunda ameaça interna à validade dos resultados foi a possibilidade de existir um intervalo entre os eventos de medição e tratamento, possibilitando, dessa forma, que os participantes estudassem entre o pré-teste e o pós-teste. Para tratar/reduzir essa ameaça, o pré-teste foi aplicado no mesmo dia em que o pós-teste, não havendo intervalo disponível para que os alunos estudassem entre um evento e outro.

Outra ameaça identificada foi a possibilidade de existir diferença de tempo dedicado ao estudo, entre os participantes dos grupos experimental e de controle. Com o objetivo de tratar/reduzir esta ameaça, para o grupo de controle, optou-se pela realização de uma atividade focada no ensino de teste (exercício prático em papel abordando o mesmo conteúdo disponibilizado no jogo), em lugar de um jogo-placebo, para que todos os participantes estivessem ocupando seu tempo com atividades relacionadas ao ensino de teste.

Em relação à ameaça externa, foi identificada a possibilidade de participantes de um determinado grupo terem conhecimentos prévios sobre o assunto abordado (classe de equivalência e análise de valor-limite). Para tratar/reduzir essa ameaça, a seleção dos participantes dos grupos experimental e de controle foi feita de forma aleatória, através

de sorteio. A ameaça foi tratada apenas no primeiro e no terceiro experimentos, já que o segundo experimento não teve a divisão dos participantes em grupo experimental e de controle.

## 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 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. O estudo permitiu também obter um feedback sobre o Jogo das 7 Falhas, além de sugestões de melhorias, que serão direcionadas a sua evolução em trabalhos futuros.

Sugere-se, ainda, como trabalhos futuros, a realização de novas avaliações qualitativas e quantitativas sobre o Jogo das 7 Falhas; se possível, com um número maior de participantes. Os novos experimentos poderiam, inclusive, utilizar cursos extracurriculares sobre teste de *software* ou turmas de pós-graduação que tivessem a disciplina de Teste de *Software*. Também poderiam ser utilizadas outras atividades de ensino (distintas de exercício) como tratamento para o grupo de controle.

## Referências

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

Basili, V.; Selby, R. Comparing the Effectiveness of Software Testing Strategies. **IEEE Transactions on Software Engineering**, p. 1278-1296, 1987

Brathwaite, B.; Schreiber, I. “Challenges for game designers”. Boston: Charles River Media, 2009.

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. An Experiential Approach to Incorporating Software Testing into the Computer Science Curriculum. **Proceedings of the Frontiers in Education Conference**, p. F3D 7-F3D, 2001.

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.

Jorgensen, P. C. **Software testing: a craftsman's approach**. CRC Press, 1995.

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.

Malone, T. C. What makes things fun to learn? heuristics for designing instructional computer games. **Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems**, p. 162-169, 1980.

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.

Murnane, T.; Reed, K.; Hall, R. Towards Describing Black-Box Testing Methods as Atomic Rules. **Proceedings of the 29th Annual International Computer Software and Applications Conference**, v. 01, p. 437-442, 2005.

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.

Pfleeger, S. L. **Software engineering:** theory and practice. Second Edition, Prentice Hall, 2001.

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.