# Testing Game: An Educational Game to Support Software Testing Education

**Conference Paper** · September 2017

DOI: 10.1145/3131151.3131182



**3 authors:**

Pedro Henrique Dias Valle
Federal University of Juiz de Fora
**44** PUBLICATIONS **290** CITATIONS
SEE PROFILE

Rafaela Vilela da Rocha
Universidade Federal do ABC (UFABC)
**62** PUBLICATIONS **234** CITATIONS
SEE PROFILE

José Carlos Maldonado
University of São Paulo
**374** PUBLICATIONS **4,736** CITATIONS
SEE PROFILE

# Testing Game: An Educational Game to Support Software Testing Education

Pedro Henrique Dias Valle
Instituto de Ciências Matemáticas e de Computação (ICMC/USP)
São Carlos, São Paulo 13560-970
pedrohenriquevalle@usp.br

Rafaela Vilela Rocha
Instituto de Ciências Matemáticas e de Computação (ICMC/USP)
São Carlos, São Paulo 13560-970
rafaela.vilela@gmail.com

José Carlos Maldonado
Instituto de Ciências Matemáticas e de Computação (ICMC/USP)
São Carlos, São Paulo
jcmaldon@icmc.usp.br

## ABSTRACT

Software testing is an essential activity for software product quality assurance. For historical reasons, there is lack of qualified professionals in this area as well as of studentsmotivation in learning software testing related contents. To mitigate these problems, some approaches have been proposed in educational games perspective. Among them, there is the Testing Game, which is an educational game to support software testing education. The objective of this paper is to describe and analyze the development of the Testing Game. The method AIMED which is an agile method to support the development of open educational resources is used. This method supports the identification of aspects essential related to the quality of educational resources. Evidence are provided that the Testing Game considered positive aspects in the development of the game, such as: definition of project scope, licensing, availability of source code in repositories, among others. On the other hand, we identified some limitations in the development of the game, such as: prioritization and revision of artifacts, user tutorials, database, among others. It is important to emphasize that this description contributed to highlight the phases and functionalities of the game, as well as to perform an evaluation to verify if the Testing Game considered the necessary features for the development of educational resources, identifying limitations in the game, giving directions for future work in the evolution of the game and of similar initiatives.

## CCS CONCEPTS

**•Software and its engineering →Software verification and validation;** *Software testing and debugging;*

## KEYWORDS

Software Testing Education, Educational Resources, Games

**ACM Reference format:**
Pedro Henrique Dias Valle, Rafaela Vilela Rocha, and José Carlos Maldonado. 2017. Testing Game: An Educational Game to Support Software Testing Education. In *Proceedings of SBES'17, Fortaleza, CE, Brazil, September 20–22, 2017,* 10 pages.
DOI: 10.1145/3131151.3131182


*SBES'17, Fortaleza, CE, Brazil*

DOI: 10.1145/3131151.3131182

## 1 INTRODUÇÃO

Nos últimos anos, a indústria de desenvolvimento de software tem se deparado com a exigência de produtos de software de alta qualidade, e consequentemente as atividades de VV&T - Verificação, Validação e Teste têm sido aprimoradas, em especial as atividades de testes, com a utilização de técnicas, critérios e ferramentas para a execução de teste [10]. O teste de software tem por objetivo executar programas ou modelos com entradas específicas, analisando se eles se comportam conforme o esperado, realizando uma análise detalhada com o intuito de levá-los a falhar e, posteriormente eliminar os defeitos que originaram as falhas [10]. O teste envolve quatro fases: planejamento de teste, projeto de caso de teste, execução de testes e avaliação dos resultados dos testes [10].

Apesar do teste de ser reconhecido como uma atividade essencial no desenvolvimento de produtos de software com alta qualidade [10, 24], há uma carência de profissionais qualificados nessa área [9], uma vez que esses profissionais possuem dificuldades em aplicar técnicas, critérios e ferramentas de teste. Isso pode estar relacionado com a deficiência na formação dos profissionais de teste ou por desmotivação no ambiente de trabalho e também pelas estratégias de alocação e responsabilização desses profissionais nas equipes de desenvolvimento e teste [28].

Para verificar como os profissionais de teste são capacitados Valle, Barbosa e Maldonado (2015) [28] analisaram os currículos de referência propostos pela Sociedade Brasileira de Computação (SBC) e pela *Association for Computing Machinery* (ACM) para cursos de graduação da área de Computação. Esses currículos fornecem apoio e diretrizes para a definição, implantação e avaliação de cursos. Em geral, os currículos de referências em computação indicam que os conteúdos de teste devem ser abordados na disciplina de Engenharia de Software [1, 15, 17, 22, 27]. Além disso, foram realizadas pesquisas nos currículos das melhores universidades do Brasil e exterior. Em geral, observou-se que os cursos de graduação da área de computação não proporcionam aos estudantes uma visão integrada dos conteúdos de teste de software com outras disciplinas [28].

Outro fator que contribui para a deficiência na formação de profissionais de teste é a carência de ambientes que motivem os estudantes a realizarem atividades relacionadas com o teste de software, pois há uma dificuldade em ensinar esse conteúdo por meio de abordagens tradicionais que utilizem apenas aulas teóricas [8, 25]. Para mitigar esses problemas, algumas iniciativas têm sido propostas para motivar e capacitar os profissionais de teste de software como: módulos educacionais, ensino de teste com programação, jogos educacionais, entre outras [29]. Dentre essas iniciativas, encontram-se os jogos educacionais que são considerados importantes ferramentas que facilitam a aprendizagem de conceitos e ideias sobre os conteúdos

educacionais [6]. Eles motivam os jogadores enquanto eles jogam, permitindo adquirir conhecimentos educacionais combinados com a diversão [19].

No domínio de teste de software foram identificados 7 diferentes jogos, sendo eles: *iTest Learning* [3], Jogo das 7 Falhas [11], TestEG [7], JETS [5], U-TEST [26], iLearnTest [20] e JoVeTest [2]. No entanto, esses jogos apresentam algumas limitações como a ausência de conteúdos sobre o teste estrutural e mutação. Desta forma, desenvolveu-se um jogo educacional, denominado **Testing Game** [30], para auxiliar o ensino de teste de software em conjunto com programação. Esse jogo está disponível por meio da plataforma Web e aborda conteúdos de teste relacionados com as técnicas de teste funcional, estrutural e baseado em defeitos.

Para avaliar o Testing Game, utilizou-se uma metodologia experimental definida por Shull, Carver e Travassos (2011) [23] que é composta por quatro etapas: i) estudo de viabilidade; ii) estudo de observação; iii) estudo de caso (ciclo de vida); e iv) estudo de caso (indústria). Até o momento, o Testing Game foi avaliado utilizando apenas a primeira etapa que é o estudo de viabilidade. Essa avaliação pode ser consultada em [30, 31].

É importante ressaltar que é necessário verificar se os jogos abordam grande parte dos aspectos essenciais a qualidade de recursos educacionais antes de serem inseridos em ambientes educacionais [32]. Desta forma, o objetivo desse trabalho foi utilizar o método AIMED [21](em inglês, *Agile, Integrative and open Method for open Educational resources Development*) para auxiliar a descrição do Testing Game. Essa descrição contribuiu para a realização de uma segunda avaliação do jogo, em que foram analisadas as particularidades necessárias para o desenvolvimento de recursos educacionais, identificando os aspectos que não foram considerados no desenvolvimento do jogo e que poderão ser utilizados como trabalhos futuros para sua evolução e iniciativas similares.

Este trabalho está organizado da seguinte forma: Na Seção 2 são apresentados os principais conceitos sobre teste de software. Além disso, é apresentada uma descrição do método AIMED que foi utilizado para auxiliar a descrição do Testing Game. Na Seção 3 é apresentada a descrição do jogo utilizando os processos do método AIMED. Na Seção 4 é discutido os aspectos que foram identificados por meio da descrição e que podem ser utilizados na evolução do jogo. Por fim, na Seção 5 são apresentadas as conclusões finais e possíveis trabalhos futuros.

## 2 REFERÊNCIAL TEÓRICO

Nesta seção são apresentados os principais conceitos sobre teste de software e as principais técnicas de teste de software, as quais foram consideradas no Testing Game. Além disso, é apresentada uma descrição do método AIMED utilizado para auxiliar o desenvolvimento de recursos educacionais. Neste artigo, o método AIMED foi utilizado para auxiliar a descrição do Testing Game.

### 2.1 Teste de Software

Nos últimos anos, a indústria de software tem aprimorado o desenvolvimento de produtos de software de alta qualidade. Para atender a essa exigência as atividades de VV& T - Verificação, Validação e Teste têm sido aprimoradas, em especial as atividades de teste de software, com a utilização de técnicas, critérios e ferramentas para auxiliar a execução de testes [10, 18]. O teste de software tem por objetivo executar programas ou modelos com entradas específicas, verificando se eles comportam-se de acordo com o esperado, com o intuito de levá-los a falhar e, posteriormente, eliminar os defeitos que originaram as falhas [10]. O teste de software é amplamente reconhecido como uma atividade essencial em qualquer processo de desenvolvimento de software bem sucedido [12].

A atividade de teste de software possui diferentes critérios que definem quais são partes constituintes do produto que precisam ser testadas para eventualmente revelar a presença de defeitos no software. Cada critério divide o domínio de entrada em diferentes subdomínios e, consequentemente, diferentes, em geral infinitos, conjuntos de casos de teste podem ser derivados para satisfazer um determinado critério. Em geral, é impossível garantir a inexistência de defeitos no software, então o teste é utilizado na prática para dar uma segurança da qualidade do produto desenvolvido [10]. Os critérios de teste de software, em nível de programa, podem ser classificados em três diferentes técnicas: **Funcional**, **Estrutural** e **Baseada em Defeitos** [10].

O Teste Funcional, ou teste de caixa-preta, é baseado apenas na especificação do produto em teste. Os critérios desta técnica podem ser aplicados em qualquer fase do teste de software (unidade, integração, sistema e aceitação), independentemente do paradigma utilizado, pois não se analisam detalhes de implementação [4, 10, 14]. Nessa técnica, os detalhes de implementação não são considerados e consequentemente não se conhece a estrutura interna do programa em teste. Essa técnica identifica apenas erros relacionados com o mal funcionamento de software e, em geral, não garantem que as partes críticas ou essenciais do produto em teste sejam exercitadas [10].

O Teste Estrutural, ou teste de caixa branca, é baseado na implementação do software a ser testado, isso requer a execução de partes ou componentes do software. Os critérios pertencentes a esta técnica são classificados em três categorias que são: **critérios baseados em complexidade**, **critérios baseados em fluxo de controle** e **critérios baseados em fluxos de dados**. Geralmente, no Teste Estrutural é utilizado uma representação do programa a ser testado, denominado "Grafo de Fluxo de Controle" (GFC) ou "Gráfo de Programa" [18]. Cada nó do GFC representa uma execução indivisível do código que termina em uma instrução simples ou condicional [10].

O Teste de Mutação ou Análise de Mutantes é o critério de teste mais conhecido da técnica de Teste Baseado em Defeitos. Este critério adiciona possíveis defeitos ao programa uniformemente. Para isso, utilizam-se os defeitos que são frequentemente cometidos pelos desenvolvedores. No teste de mutação, o programa testado é alterado várias vezes, criando um conjunto de programas alternativos (mutantes) [10, 16]. O principal objetivo deste critério é mostrar que o programa em teste não possui determinados tipos de defeitos [13].

A partir das três técnicas de teste de software apresentadas, é possível obter de forma sistemática um conjunto de casos de teste eficaz com relação à atividade de teste de software, e possivelmente revelar falhas nos programas testados. É importante ressaltar que essas técnicas não devem ser utilizadas separadamente, já que as mesmas são complementares na execução da atividade de teste.

Sendo assim, essas técnicas devem ser utilizadas em uma estratégia incremental de teste [10].

## 2.2 Método AIMED

O AIMED (*Agile, Integrative and open Method for open Educational resources Development*) é um método ágil que integra práticas de design pedagógico, design de jogo, modelagem de simulação, engenharia de software e gerenciamento de projetos; para auxiliar o desenvolvimento de recursos educacionais abertos eficientes e eficazes. Dentre os recursos educacionais considerados estão: jogos educacionais, jogos sérios, simulações interativas e artefatos gamificados [21]. Uma visão geral do método AIMED pode ser observada na Figura 1.

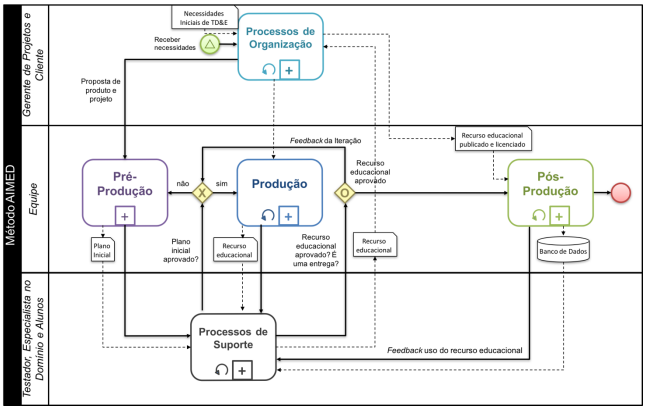


**Figura 1: Visão Geral do Método AIMED**

O método AIMED contém cinco macroprocessos com catorze processos [21], sendo eles:

(1) Processos Organizacionais: (1) Gerência – Executado ao longo da pré-produção e produção; (2) Licenciamento e (3) Publicação – Executados ao final da produção do recurso educacional;
(2) Processo de Pré-produção: (4) Planejamento Inicial – Executado logo após a concepção e aprovação do projeto (primeira atividade do processo de gerência);
(3) Processos de Produção: (5) Análise e Planejamento da iteração, (6) Projeto Iterativo, (7) Implementação Incremental, (8) Integração, teste e revisão da iteração. Estes processos são iterativos.
(4) Processos de Pós-produção: (9) Ambiente e Manutenção, (10) Execução e (11) Avaliação da Aprendizagem. Estes processos são realizados após a entrega do recurso educacional.
(5) Processos de Apoio: (12) Verificação dos Artefatos, (13) Validação, (14) Projeto Experimental. Estes processos são realizados ao longo de todo o ciclo de vida do recurso educacional.

O método AIMED é fundamentado em métodos ágeis quanto à produção rápida e contínua de artefatos [21]. Então, todos os artefatos que são entregues devem ser planejados, produzidos e avaliados em cada iteração. Os processos e atividades do método AIMED podem ser visualizados na Figura 2.

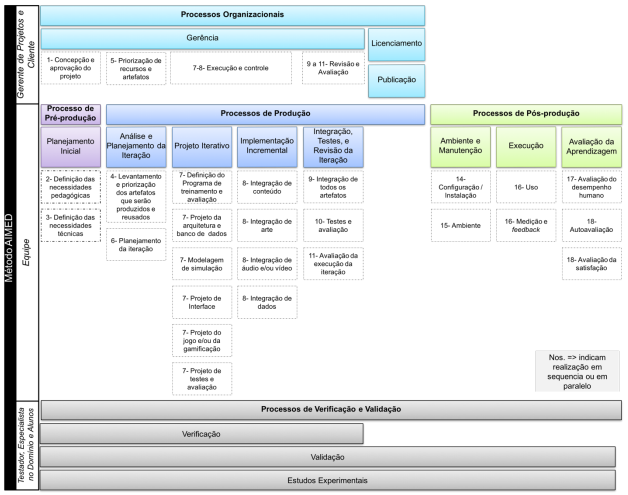


**Figura 2: Visão Geral das Atividades do Método AIMED**

Conforme mencionado na Seção 1, o método AIMED foi utilizado para auxiliar a descrição do Testing Game. Essa descrição pode ser observada na próxima seção.

# 3 DESCRIÇÃO DO TESTING GAME

Nesta seção é apresentada a descrição do Testing Game por meio do método AIMED. A descrição do jogo foi dividida em 3 iterações que correspondem as três técnicas de teste consideradas no jogo, sendo elas: teste funcional, teste estrutural e teste baseado em defeitos.

## 3.1 Processos Organizacionais

Os processos organizacionais estão relacionados com a gestão de projetos, licenciamento e publicação, os quais estão detalhados a seguir:

*3.1.1 Gerência.* O processo de Gerência contém as atividades e tarefas de planejamento e execução do projeto, controle de seus riscos e monitoramento de seu progresso. Essas atividades são executadas ao longo do desenvolvimento do recurso educacional. Esse processo é transversal aos processos de Pré-Produção e de Produção do método AIMED. Nesta etapa foram definidos os requisitos do jogo, porém eles não foram descritos formalmente. Além disso, identificou-se a visão do produto que está relacionada com as interfaces, módulos e tipos de licenças utilizadas.

*3.1.2 Licenciamento.* O processo de Licenciamento contém atividades e tarefas para definir políticas de licenciamento. Essas atividades promovem a garantia dos direitos de propriedade intelectual e o compartilhamento do recurso desenvolvido. Para auxiliar o desenvolvimento do jogo foi utilizado o motor de jogos Construct 2 com a licença Personal que é limitada quanto ao uso comercial. No entanto, é importante ressaltar que todos os direitos dos produtos desenvolvidos são dos desenvolvedores. Desta forma, o Testing Game foi disponibilizado por meio de uma licença *open source*.

*3.1.3 Publicação.* O processo de Publicação contém atividades e tarefas para criar os tutoriais e disponibilizar o recurso educacional desenvolvido em repositórios. Até o momento não foram

desenvolvidos tutoriais para auxiliar a utilização do jogo. É importante ressaltar que para utilizar o jogo não é necessário tutoriais de instalação, pois o jogo está disponível na plataforma Web, então não é necessário a instalação de bibliotecas e software. O Testing Game contém algumas dicas e enunciados durante as fases para facilitar o entendimento dos estudantes. Quanto a disponibilização do recurso educacional, o jogo está disponível no Bitbucket[1] que é um serviço de hospedagem de projetos semelhannte ao GitHub e GitLab. O código do Testing Game em HTML5 está disponível em: https://bitbucket.org/pedrohdvalle/testinggamehtml5.

## 3.2 Processo de Pré-produção:

O processo de Planejamento Inicial contém atividades e tarefas de planejamento do recurso educacional que será desenvolvido, contendo as necessidades de se desenvolver esse recurso educacional, tanto pedagógicas quanto técnicas.

*3.2.1 Definição das necessidades pedagógicas.* Nesta etapa, deve-se analisar as necessidades pedagógicas (aprendizagem, treinamento e avaliação) e fazer um planejamento inicial pedagógico. Para isso, foram definidos os seguintes itens:

- **Contexto e Problema:** O teste de software é reconhecido como uma atividade importante na garantia de qualidade de produtos de software. No entanto, há uma carência de profissionais qualificados nessa área e uma desmotivação por parte dos estudantes em aprender conteúdos relacionados com o teste de software. Para amenizar esses problemas, diferentes iniciativas têm sido propostas para auxiliar o ensino de teste de software, dentre elas, encontram-se os jogos educacionais. Desta forma, realizou-se o desenvolvimento de um jogo educacional denominado Testing Game.
- **Público-Alvo:** O público-alvo do jogo são alunos de graduação que cursam disciplinas com conteúdos relacionados com o teste de software. O jogo pode ser utilizado como um material de apoio às aulas de teste de software, pois os estudantes podem treinar suas habilidades adquiridas em sala de aula.
- **Conteúdo:** Os conteúdos considerados no jogo abordam as três principais técnicas de teste de software, a saber: teste funcional, teste estrutural e teste baseado em defeitos. O livro "Introdução ao Teste de Software" [10] foi utilizado para a elaboração dos conteúdos abordados no jogo e também como literatura complementar. Na Tabela 1 são apresentados os conteúdos abordados no jogo para cada técnica de teste considerada.
- **Avaliação:** Essa atividade é responsável por avaliar o desempenho dos estudantes com a utilização do recurso educacional. Até o momento, não foi avaliado o desempenho dos estudantes com a utilização do Testing Game. No entanto, pretende-se realizar essa avaliação como trabalhos futuros.

[1]Disponível: https://bitbucket.org/

*3.2.2 Definição das necessidades técnicas.* Nesta etapa, deve-se analisar as necessidades técnicas do jogo e realizar um planejamento inicial para o desenvolvimento. Para isso, foram definidas as seguintes atividades:

- **Objetivos de *Design*:** O Testing Game é um jogo com dimensão gráfica 2D para a plataforma Web. Nesse jogo, o estudante é representado por um *avatar* que pode realizar diversas ações, tais como: andar, correr, pular e atirar. O jogo contém 3 níveis que correspondem às três técnicas de teste de software, e cada nível contém fases que correspondem aos critérios de teste relacionados às técnicas consideradas. Cada nível do jogo é representado por uma porta que pode ser aberta com chaves encontradas durante o jogo. Para auxiliar a aprendizagem dos estudantes, o jogo contém módulos teóricos para os conteúdos abordados. Para ter acesso a esse conteúdo é necessário clicar no ícone representado por um livro no menu superior do lado direito no jogo. A pontuação do jogo é obtida a partir dos acertos dos estudantes.
- **Objetivos de Arte:** Para a definição do *avatar*, inimigos (representados por diferentes avatares em que os jogadores devem eliminar para continuar os desafios propostos no jogo), objetos e os cenários do jogo foram selecionadas imagens disponibilizadas na internet com licenças gratuitas. Essas imagens foram reutilizadas nas diferentes fases do jogo.
- **Objetivos Técnicos:** O Testing Game considerou a dinâmica *single player*, ou seja, possibilita a participação de apenas um estudante no jogo. Para auxiliar na programação do jogo, utilizou-se o motor de jogos Construct 2. Esse motor de jogos foi selecionado a partir de uma mapeamento sistemático realizado. Os sons utilizados foram obtidos por meio de repositórios disponibilizados na internet com licenças gratuitas.

## 3.3 Processos de Produção

O objetivo principal do processo de Produção é desenvolver o recurso educacional, de forma iterativa e incremental. Para a descrição do Testing Game foram definidas 3 iterações, as quais correspondem aos três níveis do jogo. Essas iterações estão descritas a seguir:

*3.3.1 Iteração 1 - Teste Funcional.* A Iteração 1 representa o primeiro nível do Testing Game, no qual são abordados os conteúdos relacionados com a técnica de teste funcional. Nesse nível há 8 fases que correspondem aos critérios de teste funcional, tais como: particionamento em classes de equivalência e análise de valor limite.

- **Análise e Planejamento:** Neste processo, analisou-se e priorizou os artefatos produzidos e reusados na fase sobre teste funcional. Algumas imagens foram reusadas de um repositório disponível na internet, tais como portas, que representam os níveis do jogo, ícone de conteúdo teórico, inimigos, tiros, personagem do jogo. Outras imagens foram criadas utilizando o software Microsoft PowerPoint 2013. Além disso, alguns artefatos foram criados a partir dos recursos disponibilizados pela Construct 2, tais como: casos

**Tabela 1: Conteúdos abordados no Testing Game**

| Técnicas de Teste de Software | Conteúdo |
|---|---|
| **Teste Funcional** | Critérios de Teste de Software, Especificação do Programa BubbleSort, Critério de Particionamento em Classes de Equivalência, Tabela de Classe de Equivalência, Critério de Análise de Valor Limite e Conjunto de Caso de Teste Mínimo. |
| **Teste Estrutural** | Grafo de Fluxo de Controle, Critério Todos-Nós, Critério Todas-Arestas, Usos de Variável (definição, uso computacional, uso predicativo), Grafo Def-Uso, Critério Todas-Definições, Critério Todos-Usos, Conjunto Mínimo de Casos de Teste e Caminho Não-Executável. |
| **Teste Baseado em Defeitos** | Operadores de Mutação em Java, Programas Mutantes, Mutantes Equivalentes, Escore de Mutação e Conjunto de Caso de Teste Mínimo. |

de teste, enunciados, *labels* de tempo e pontuação, entre outros.

- **Projeto Iterativo:** Neste processo, definiram-se as fases do jogo para o nível que corresponde ao teste funcional. Para avançar para a próxima fase, o estudante deve concluir a fase atual com sucesso. Neste nível foram definidas 8 fases que abordam os critérios de particionamento em classe de equivalência, análise de valor limite, conjunto de caso de teste mínimo, entre outros. A seguir, apresenta-se uma breve descrição das fases do teste funcional.
  - **Fase 1:** Nesta fase, o jogador deve arrastar os critérios para a técnica de teste que os correspondem. As técnicas consideradas estão representadas por dois retângulos. Quando o jogador arrastar um critério para a técnica correta, um sinal sonoro é realizado indicando que o jogador acertou, caso contrário, um sinal sonoro é emitido indicando que a ação realizada é incorreta, e o critério volta para seu local. Nesta fase está disponível um módulo de conteúdo teórico sobre o teste funcional. Na Figura 3, pode-se visualizar a interface da Fase 1.

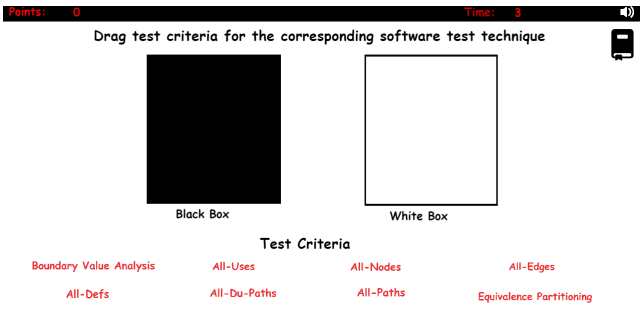


**Figura 3: Primeira Fase do Teste Funcional**

  - **Fase 2:** Nesta fase é apresentado ao jogador a especificação do programa utilizado, a saber: Bubble Sort, para aplicar os conceitos de teste durante a execução do jogo. A especificação diz que o programa Bubble Sort só aceitará vetores com o tamanho igual a 4. Em seguida, é apresentado um enunciado para o jogador que diz que ele deve encontrar vetores com o tamanho igual a 4. Para isso, o jogador deve eliminar os inimigos e os vetores inválidos. Após o jogador completar o desafio, é apresentada uma lista com os vetores válidos e inválidos da fase. Na Figura 4 é apresentada a interface da Fase 2.


**Figura 4: Segunda Fase do Teste Funcional**

  - **Fase 3:** A terceira fase é semelhante a fase 2. A única diferença é que o jogador deve encontrar vetores com tamanho igual a 4 e que eles contenham apenas números inteiros conforme a especificação do programa Bubble Sort.
  - **Fase 4:** Nesta fase, o jogador deve preencher uma tabela de equivalência a partir da especificação do programa Bubble Sort. Nela, há um módulo com conteúdo teórico sobre o critério "Particionamento em Classes de Equivalência".
  - **Fase 5:** Nesta fase, o jogador deve encontrar casos de teste válidos para satisfazer o critério de "Particionamento em Classes de Equivalência". Para isso, ele deve eliminar os inimigos e os casos de teste que são inválidos. Na Figura 5 é apresentada a interface da Fase 5.
  - **Fase 6:** Nesta fase, o jogador deve selecionar os casos de teste disponíveis na tela para as classes de equivalência do programa Bubble Sort.
  - **Fase 7:** Nesta fase, o jogador deve analisar a classe de equivalência Tamanho do Vetor para encontrar todos os casos de teste disponíveis na fase para satisfazer o critério "Analise de Valor Limite". Para isso, o jogador

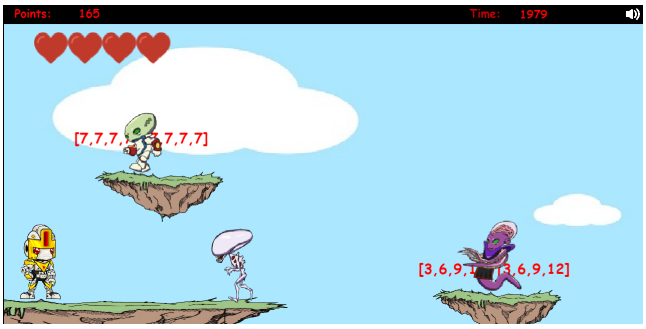


**Figura 5: Quinta Fase do Teste Funcional**

deve eliminar todos os inimigos e os casos de teste que são inválidos.

- **Fase 8:** Por fim, nesta fase, o jogador deve apenas pegar a chave que abre a porta do próximo nível do jogo. O próximo nível corresponde aos conteúdos relacionados com o teste estrutural.

- **Implementação Incremental:** O processo de Implementação Incremental contém atividades e tarefas de criação, revisão, reúso e remixagem de artefatos. Esses artefatos incluem arte, áudio e vídeo. Os recursos de imagem e áudio foram obtidos por meio de websites sob licença gratuita. A programação das fases foi realizada com o auxílio da Construct 2. Os Recursos de vídeos e banco de dados não foram considerados na versão atual do jogo.
- **Integração, Testes e Revisão:** Após a definição, criação e reutilização dos artefatos, realizou-se a integração dos recursos de arte, multimídia e conteúdo. Em seguida, houve a integração das fases para que juntas formassem o primeiro nível do jogo. Para verificar a eficiência das fases desenvolvidas foram testadas suas principais funcionalidades. Após os testes realizados, a Iteração 1 foi aprovada como um recurso educacional pronto.

*3.3.2 Iteração 2 - Teste Estrutural.* A Iteração 2 representa o segundo nível do Testing Game com 10 fases. Nesse nível são abordados os conteúdos relacionados com o teste estrutural.

- **Análise e Planejamento:** Neste processo foram identificados e definidos os artefatos utilizados para o desenvolvimento da Iteração 2. Alguns artefatos foram reutilizados da Iteração 1, tais como: portas, ícone de conteúdo teórico, moedas, personagens, inimigos, entre outros. No entanto, alguns artefatos foram criados exclusivamente pra o desenvolvimento da Iteração 2, sendo eles: lupa para representar código fonte, grafos de fluxo de controle, imagens de *background*, bandeiras, entre outros. É importante ressaltar que todos esses artefatos foram obtidos por meio de licenças gratuitas.
- **Projeto Iterativo:** Neste processo foram definidas as fases da Iteração 2. Assim como na Iteração 1, os estudantes só têm acesso a próxima fase se concluirem a fase atual com sucesso. Neste nível contêm 10 fases com conteúdos relacionados com os critérios Todos-Nós e Todos-Usos, Uso de Variável (definição, uso computacional e uso predicativo), Grafo de Fluxo de Controle, Grafo Def-Uso, conjunto mínimo de teste, caminho não-executável e outros.
  - **Fase 1:** Na primeira fase, é apresentado ao jogador o código do programa Bubble Sort. Em seguida, o jogador deve encontrar o GFC que corresponde ao código do programa. Para isso, ele deve eliminar os GFCs que não correspondem ao programa e eliminar os inimigos durante os desafios que são propostos. Nesta fase está disponível um módulo de conteúdo teórico sobre o teste estrutural. Na Figura 6, pode-se visualizar a interface da fase 1.

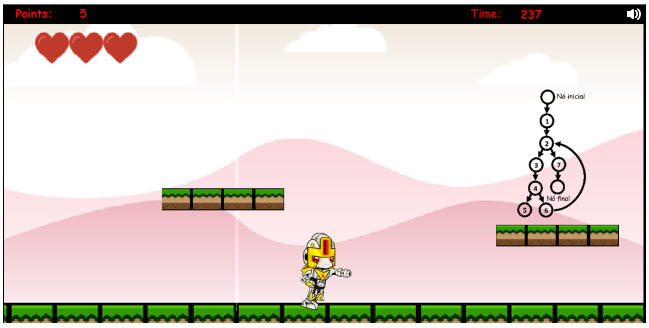


**Figura 6: Primeira fase do Teste Estrutural**

  - **Fase 2:** Na segunda fase, o jogador deve encontrar casos de teste para satisfazer o critério "Todos-Nós" para o programa Bubble Sort. Quando o jogador encontra um caso de teste válido, o GFC é atualizado na tela para demonstrar quais os nós foram cobertos. É importante ressaltar que o jogador deve eliminar os casos de teste inválidos e os inimigos.
  - **Fase 3:** A terceira fase é semelhante a fase 2 desta Iteração. No entanto, ao invés de encontrar casos de teste para satisfazer o critério "Todos-Nós", o jogador deve encontrar casos de teste para satisfazer o critério "Todas-Arestas".
  - **Fase 4:** Na quarta fase, o jogador deve encontrar o conjunto mínimo de casos de teste para satisfazer os critérios "Todos-Nós" e "Todas-Arestas". Para isso, o jogador deve eliminar os casos de teste que são inválidos ou que fornecem a mesma cobertura dos casos de teste que foram selecionados.
  - **Fase 5:** Na quinta fase, o jogador deve apenas observar os usos das variáveis do programa Max. Esse programa foi utilizado para que os jogadores pudessem entender quais são os tipos de usos de variáveis para que nas próximas fases eles utilizassem esses conceitos no programa Bubble Sort. Na Figura 7 é ilustrada a interface da fase 5.
  - **Fase 6:** Na sexta fase, o jogador deve encontrar todas as variáveis do programa Bubble Sort que tiveram uso predicativo. Para isso, os jogadores devem eliminar os inimigos e as variáveis que não tiveram uso predicativo. No final dessa fase é apresentado ao jogador

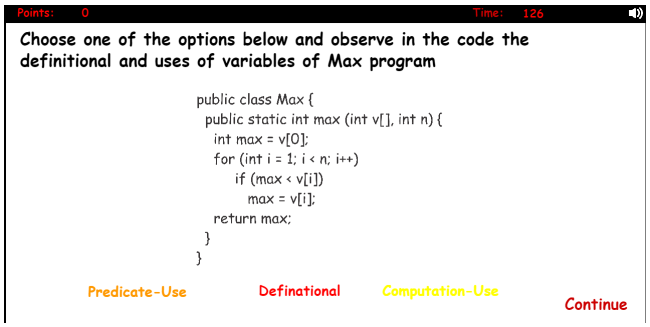


**Figura 7: Quinta fase do Teste Estrutural**

o GFC com as variáveis que tiveram uso predicativo. Nesta fase está disponível um módulo de conteúdo teórico sobre os critérios da família de fluxo de dados.

- **Fase 7:** A sétima fase é semelhante a fase 6. No entanto, ao invés de encontrar as variáveis que tiveram uso predicativo, o jogador deve encontrar todas as variáveis do programa Bubble Sort que tiveram uso computacional.
- **Fase 8:** Na oitava fase, o jogador deve observar o grafo Def-Uso do programa Bubble Sort e preencher uma tabela que demostra em quais nós do grafo as variáveis foram definidas. Na Figura 8, pode-se visualizar a interface da fase 8.

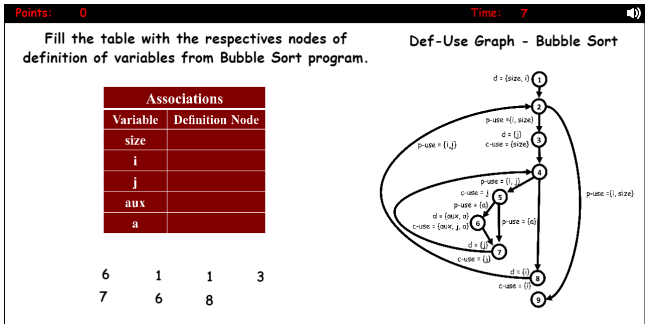


**Figura 8: Oitava fase do Teste Estrutural**

- **Fase 9:** Na nona fase, o jogador deve encontrar as associações requeridas para satisfazer o critério "Todos-Usos" para o programa Bubble Sort. Para isso, o jogador deve eliminar as associações incorretas que forem encontradas durante a fase.
- **Fase 10:** Por fim, na décima fase, o jogador deve encontrar o caminho do GFC do programa Bubble Sort que é um caminho não-executável. Para isso, o jogador deve eliminar os caminhos executáveis do GFC. Nesta subfase está disponível um módulo de conteúdo teórico que explica o que é um caminho não-executável.

- **Implementação Incremental:** O processo de Implementação Incremental é responsável pela criação, revisão, reúso e remixagem de artefatos. Os recursos de imagem e áudio utilizados neste nível foram obtidos sob licença gratuita. Para auxiliar a programação das fases foi utilizado a Construct 2. É importante ressaltar que os recursos de vídeos e banco de dados não foram considerados na versão atual do jogo.
- **Integração, Testes e Revisão:** Após a definição, criação e reutilização dos artefatos, realizou-se a integração dos recursos de arte, multimídia e conteúdo. Em seguida, houve a integração das fases para formarem o segundo nível do jogo. É importante ressaltar que foram testadas as principais funcionalidades desse nível. Para isso, foram geradas combinações entre as funcionalidades e uma sequência de passos que representam a execução do jogo. Em seguida, a Iteração 2 foi aprovada como um recurso educacional pronto.

*3.3.3 Iteração 3 - Teste de Mutação.* A Iteração 3 representa o terceiro e último nível do Testing Game. Neste nível são abordados conteúdos relacionados com o teste baseado em defeitos, especificamente o critério de teste de mutação.

- **Análise e Planejamento:** Nesse processo foram identificados e criados artefatos que foram utilizados para o desenvolvimento da Iteração 3. Para esta Iteração foram criados poucos artefatos, pois grande parte dos artefatos utilizados foram reusados das Iterações 1 e 2.
- **Projeto Iterativo:** Neste processo foi definido o programa educacional das fases da Iteração 3. Neste nível contém 5 fases com conteúdos sobre operadores de mutação em Java, programas mutantes, mutantes equivalentes, escore de mutação e conjunto de caso de teste mínimo. Assim como nas Iterações 1 e 2, os estudantes só têm acesso a próxima fase se conseguirem concluir a fase atual com sucesso.
  - **Fase 1:** Na primeira fase, o jogador deve encontrar o programa mutante gerado a partir dos operadores de mutação: arimético, atribuição, relacional e lógico. Para isso, o jogador deve eliminar todos os programas mutantes que não foram gerados por essas classes de operadores de mutação. Na Figura 9 é apresentada a interface da fase 1 do teste de mutação.

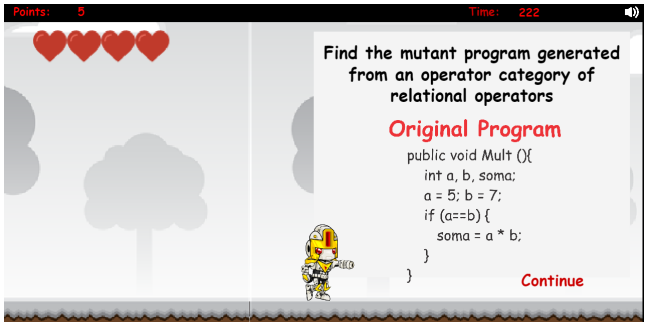


**Figura 9: Primeira fase do Teste de Mutação**

- **Fase 2:** Na segunda fase, o jogador deve encontrar casos de teste que eliminem os programas mutantes do

programa denominado Op. Para isso, o jogador deve eliminar os inimigos e os casos de teste inválidos.

- **Fase 3:** A fase 3 é semelhante a fase 2. No entanto, o jogador deve encontrar casos de teste para eliminar os mutantes do programa Bubble Sort. Ele pode visualizar no código quais as mutações foram eliminadas com a seleção do caso de teste. Além disso, o jogador pode visualizar seu *score* de mutação que está disponível no topo da tela no lado direito de forma fixa.
- **Fase 4:** Nessa fase, o jogador deve eliminar os programas mutantes que são considerados mutantes equivalentes. Para isso, o jogador deve eliminar os inimigos, barreiras e mutantes que não são equivalentes. O local que houve mutação é marcado com cor vermelha no código do programa Bubble Sort. Na Figura 10 é apresentada a interface da fase 4 do teste de mutação.

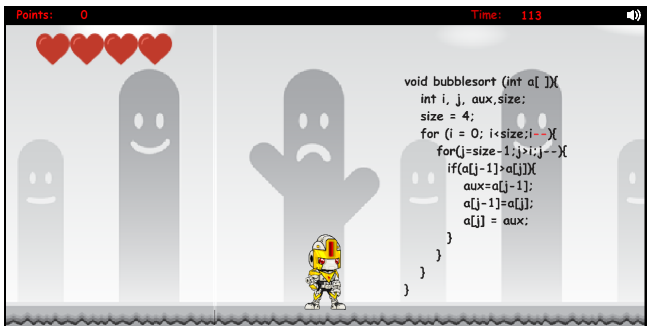


**Figura 10: Quarta fase do Teste de Mutação**

- **Fase 5:** Por fim, na quinta e última fase, o jogador deve selecionar casos de teste para eliminar os programas mutantes. Nessa fase são apresentados aos jogadores 3 códigos de programas diferentes, o primeiro contém uma estrutura apenas sequencial, o segundo contém uma estrutura condicional e o último uma estrutura de repetição. Desta forma, os jogadores podem observar que é possível aplicar o teste de mutação em qualquer estrutura de programa.

- **Implementação Incremental:** O processo de Implementação Incremental considera a criação, revisão, reúso e remixagem de artefatos. Esses artefatos incluem arte, áudio e vídeo, os quais foram obtidos por meio de licença gratuita. É importante ressaltar que até o momento não foram considerados os recursos de vídeos e banco de dados. Porém, pretende-se eliminar essas limitações na próxima versão do jogo.
- **Integração, Testes e Revisão:** Após a definição de cada conteúdo abordado na Iteração 3, considerou-se os recursos de arte e áudio assim como nas Iterações 1 e 2. Após a integração desses recursos, realizou-se os testes com as principais funcionalidades do Testing Game. Para isso, criou-se um conjunto de passos que representavam a execução do jogo. Por fim, a Iteração 3 foi aprovada como recurso educacional pronto para auxiliar o ensino de teste de software.

## 3.4 Processos de Pós-produção

Neste processo, o recurso educacional já foi desenvolvido e pode ser utilizado como instrumento de apoio à aprendizagem de conteúdos educacionais. Para isso, é necessário considerar a execução, ambiente e avaliação da aprendizagem conforme apresentado a seguir:

- **Ambiente:** O Testing Game está disponível por meio plataforma Web no servidor do Scirra Arcade[2]. É importante ressaltar que para ter acesso ao jogo não é necessário instalar recursos, tais como: banco de dados, aplicativos, *plugins*, entre outros.
- **Execução:** Nesta etapa, os estudantes já podem utilizar o Testing Game como um recurso educacional de apoio ao ensino de teste de software. Até o momento, o Testing Game foi utilizado apenas no estudo de viabilidade realizado para avaliar a qualidade e usabilidade do jogo [30].
- **Avaliação da Aprendizagem:** Na versão atual do jogo, não foram realizados experimentos com estudantes para verificar a aprendizagem adquirida com a utilização do Testing Game. Porém, pretende-se realizar uma avaliação como trabalhos futuros para avaliar a aprendizagem dos estudantes. No entanto, foi realizado um estudo de viabilidade [23] para avaliar a usabilidade e a qualidade do jogo sob o ponto de vista dos estudantes com relação à motivação, experiência do usuário e aprendizagem.

## 3.5 Processos de Apoio

Esses processos estão relacionados com a verificação e validação dos artefatos. Além disso, são considerados estudos experimentais. É importante ressaltar que estes processos são realizados ao longo de todo o ciclo de vida do recurso educacional. A seguir, apresenta-se a descrição desses processos:

- **Verificação e Validação:** Os processos verificação e validação são processos transversais na metodologia de desenvolvimento de recursos educacionais, eles avaliam as saídas dos processos (Pré-Produção, Produção e Pós-Produção). Desta forma, os artefatos gerados foram avaliados para serem reusados em outras iterações, tanto em relação ao desenvolvimento correto entre os processo (verificação) quanto se o jogo está sendo implementado corretamente e completamente, conforme necessidades/requisitos dos usuários (validação).
- **Estudos Experimentais:** Para avaliar a usabilidade e qualidade do Testing Game com relação à motivação, experiência do usuário e aprendizagem, realizou-se um estudo de viabilidade de acordo com uma metodologia experimental [23]. Nesse estudo, aproximadamente 85,64% dos estudantes avaliaram a qualidade do jogo de forma positiva sob o ponto de vista dos estudantes. Quanto à usabilidade, foram identificados poucos problemas [30, 31].

[2]Disponível em: https://goo.gl/50pmWP

## 4 DISCUSSÃO

O Testing Game foi desenvolvido utilizando um processo *ad-hoc*. No entanto, é importante ressaltar que os jogos educacionais devem ser avaliados antes de serem utilizados como recursos educacionais para verificar se eles possuem boa qualidade [32]. Para isso, utilizou-se o método AIMED para auxiliar a descrição e avaliação do desenvolvimento do jogo, verificando se o Testing Game contém os principais aspectos essenciais à qualidade de recursos educacionais.

Por meio dessa descrição, identificaram-se aspectos positivos no desenvolvimento do jogo, tais como: i) definição do escopo, pois foram definidas as técnicas e critérios de teste abordadas no jogo; ii) visão de produto (licença, módulos, interfaces) bem definida; iii) disponibilização do código fonte em repositórios; iv) disponibilização de módulos extras com conteúdos teóricos abordados no jogo; v) estabelecimento de licença software livre; vi) definição de conteúdos abordados em cada nível; e vii) acesso ao jogo por meio de um *link*, dispensando a utilização de tutoriais de acesso.

No entanto, alguns aspectos do jogo podem ser revisitados na evolução do jogo. Dentre eles, destacam-se: i) priorização e revisão de artefatos; ii) ausência de tutoriais para auxiliar os estudantes na utilização do jogo; iii) ausência do estabelecimento de uma lista de requisitos; iv) ausência de tutorial para os estudantes como um guia de uso (telas/menus, funcionalidade/fase, comandos); v) falta de divisão de papéis no desenvolvimento do jogo; vi) ausência de um planejamento inicial com os principais riscos; vi) ausência de uma avaliação para verificar a aprendizagem dos estudantes na utilização do jogo; e vii) ausência de banco de dados para armazenar os dados dos estudantes.

Com relação ao Testing Game, pretende-se desenvolver uma nova versão para atender as limitações identificadas por meio da descrição do jogo e implementar a característica de *multiplayer*, possibilitando que os estudantes troquem experiências entre si. Além disso, pretende-se integrar ferramentas de teste para auxiliar os estudantes na execução de testes. Com relação ao método AIMED, seria interessante considerar a evolução do método para auxiliar o desenvolvimento de recursos educacionais com equipes de desenvolvimento geograficamente distribuídas. Na atividade responsável pelo licenciamento do recurso educacional seria interessante uma lista com as principais licenças para os recursos educacionais. Além disso, na atividade responsável pela programação poderia haver uma lista com sugestões de ferramentas de apoio ao desenvolvimento dos recursos educacionais.

## 5 CONCLUSÃO E TRABALHOS FUTUROS

O teste de software é reconhecido como uma atividade importante para o desenvolvimento de produtos de software de boa qualidade. No entanto, há uma carência de profissionais qualificados nessa área e uma dificuldade em ensinar teste de software por meio de aulas teóricas tradicionais. Para amenizar esses problemas, desenvolveu-se um jogo denominado Testing Game [30]. Porém, é necessário realizar uma avaliação dos jogos antes de serem inseridos em ambientes educacionais para verificar se eles possuem boa qualidade.

Para isso, realizou-se uma descrição e avaliação do desenvolvimento do Testing Game por meio do método AIMED. Essa descrição auxiliou a identificação de aspectos positivos do jogo e aspectos que não foram considerados no jogo, tais como: tutoriais de uso, lista de requisitos, planejamento de riscos, entre outros. Esses aspectos serão considerados na evolução do jogo para que o Testing Game possa contemplar grande parte dos itens essenciais do método AIMED. É importante ressaltar que o Testing Game aborda conteúdos relacionados com os testes estrutural e baseado em defeitos, os quais não foram contemplados nos jogos de teste que foram identificados. Além disso, o jogo foi avaliado de forma positiva quanto à sua qualidade e usabilidade, e consequentemente pode ser utilizado como instrumento motivador da aprendizagem de conteúdos relacionados com o teste.

## 6 AGRADECIMENTOS

Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio concedido a este trabalho.

## REFERÊNCIAS

[1] ACM and IEEE. 2013. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. (2013).

[2] Ana. Karoline. T. Barbosa, Larissa L. E. Neves, and Arilo C. Dias Neto. 2016. JoVeTest - Jogo da Velha para Auxiliar no Ensino e Estudo de Teste de Software. In *IX Fórum de Educação em Engenharia de Software*. SBC, Maringá, Brasil, 65–76.

[3] Carla IM Bezerra, Emanuel F Coutinho, Ismayle S Santos, José Maria Monteiro, and Rossana MC Andrade. 2014. Evolução do Jogo Itest Learning para o Ensino de Testes de Software: Do Planejamento ao Projeto. In *XIX Conferência Internacional sobre Informática na Educação (TISE)*. Nuevas Ideas En Informática Educativa, Fortaleza, Brasil.

[4] E M Bizerra Junior, D Silva Silveira, M Lencastre Pinheiro Menezes Cruz, and FJ Araujo Wanderley. 2012. A Method for Generation of Tests Instances of Models from Business Rules Expressed in OCL. *Latin America Transactions* 10, 5 (2012), 2105–2111.

[5] Tarcila Gesteira da Silva and Felipe Martins Muller. 2012. *Jogos Sérios em Mundos Virtuais: uma abordagem para o ensino-aprendizagem de teste de Software*. Master's thesis. Universidade Federal de Santa Maria.

[6] Sara De Freitas and Paul Maharg. 2011. *Digital games and learning*. Bloomsbury Publishing, New York.

[7] Bruno CÉSAR de Oliveira. 2013. TestEG - UM SOFTWARE EDUCACIONAL PARA O ENSINO DE TESTE DE SOFTWARE. In *Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras*. UFLA, Lavras, Brasil, 1–92.

[8] Draylson Micael de Souza, Sofia Larissa da Costa, Nemesio Freitas Duarte Filho, and Ellen Francine Barbosa. 2013. Um Estudo Experimental do Ambiente ProgTest no Ensino de Programação. In *X Workshop Latinoamericano Ingeniería de Software Experimental (ESELAW)*. CIbSE.

[9] J. C.; de Souza, D. M.; Maldonado and E. F. Barbosa. 2012. Aspectos de Desenvolvimento e Evolução de um Ambiente de Apoio ao Ensino de Programação e Teste de Software. In *XXIII Simpósio Brasileiro de Informática na Educação*. SBC, Rio de Janeiro, Brasil, 1–10.

[10] M.E. Delamaro, J.C. Maldonado, and M. Jino. 2016. *Introdução ao teste de software* (2 ed.). Elsevier, Rio de Janeiro.

[11] Lucio Lopes Diniz and Rudimar Luis Scaranto Dazzi. 2011. JOGO DAS SETE FALHAS: UM JOGO EDUCACIONAL PARA APOIO AO ENSINO DO TESTE CAIXA PRETA. In *Computer on the Beach*. Universidade do Vale do Itajaí, Florianópolis, 1–10.

[12] Gordon Fraser and Andrea Arcuri. 2013. Whole test suite generation. *IEEE Transactions on Software Engineering* 39, 2 (2013), 276–291.

[13] Yue Jia and M. Harman. 2011. An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering* 37, 5 (Sept 2011), 649–678.

[14] M.A. Khan and M. Sadiq. 2011. Analysis of black box software testing techniques: A case study. In *International Conference and Workshop on Current Trends in Information Technology (CTIT)*. IEEE, Dubai, United Arab Emirates, 1–5.

[15] Richard J LeBlanc, Ann Sobel, Jorge L Diaz-Herrera, Thomas B Hilburn, and others. 2006. Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. (2006).

[16] P. Lochab, A. Singhal, and A. Bansal. 2014. Generation of mutation operators for aspect-oriented software systems. In *V International Conference Confluence The Next Generation Information Technology Summit*. IEEE, Noida, India, 748–752.

[17] Chengying Mao. 2008. Towards a question-driven teaching method for software testing course. In *International Conference on Computer Science and Software*

*Engineering*. IEEE, Wuhan, China.

[18] Glenford J Myers, Corey Sandler, and Tom Badgett. 2011. *The art of software testing*. John Wiley & Sons, New Jersey, USA.

[19] José Francisco Barbosa Neto and Fernando de Souza da Fonseca. 2013. Jogos educativos em dispositivos móveis como auxílio ao ensino da matemática. *RENOTE* 11, 1 (2013).

[20] T. P. B. Ribeiro and A. C. R. Paiva. 2015. iLearnTest: Educational game for learning software testing. In *X Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, Aveiro, Portugal, 1–6.

[21] R.V. Rocha, P.H.D. Valle, J.C. Maldonado, I.I. Bittencourt, and S. Isotani. 2017. AIMED: Agile, Integrative and open Method for open Educational resources Development. In *XVII IEEE International Conference on Advanced Learning Technologies*. IEEE, Timisoara, Romania, 1–6.

[22] SBC. 2005. Currículo de Referência para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação. (2005).

[23] Forrest Shull, Jeffrey Carver, and Guilherme H. Travassos. 2001. An Empirical Methodology for Introducing Software Processes. In *VIII European Software Engineering Conference Held Jointly and IX International Symposium on Foundations of Software Engineering (ACM SIGSOFT)*. ACM, New York, USA, 288–296.

[24] Rodolfo Adamshuk Silva, Evandro Westphalen Carlos Gomes, and Simone Nasser Matos. 2012. Plano de Teste para Validação do Subframework de Análise Semântica de Fórmulas. In *IX International Conference on Information Systems and Technology Management (CONTECSI)*. São Paulo, Brasil, USP, 4182–4208.

[25] Joanna Smith, Joe Tessler, Elliot Kramer, and Calvin Lin. 2012. Using peer review to teach software testing. In *IX annual international conference on International computing education research*. ACM, Melbourne, Australia, 93–98.

[26] Marcello Thiry, Alessandra Zoucas, and Antônio C da Silva. 2011. Empirical study upon software testing learning with support from educational game.. In *XXIII International Conference on Software Engineering and Knowledge Engineering (SEKE)*. IEEE, Miami Beach, USA.

[27] Heikki Topi, Joseph S Valacich, Ryan T Wright, Kate Kaiser, Jay F Nunamaker Jr, Janice C Sipior, and Gert-Jan de Vreede. 2010. IS 2010: Curriculum guidelines for undergraduate degree programs in information systems. *Communications of the Association for Information Systems* (2010).

[28] P. H. D. Valle, E. F. Barbosa, and J. C. Maldonado. 2015. CS curricula of the most relevant universities in Brazil and abroad: Perspective of software testing education. In *XVII International Symposium on Computers in Education (SIIE)*. IEEE, Setúbal, Portugal, 62–68.

[29] Pedro Henrique Dias Valle, Ellen Francine Barbosa, and José Carlos Maldonado. 2015. Um Mapeamento Sistemático sobre Ensino de Teste de Software. In *XXVI Simpósio Brasileiro de Informática na Educação*. SBC, Maceió, Brasil, 71 – 80.

[30] Pedro Henrique Dias Valle and José Carlos Maldonado. 2016. *Jogos educacionais: uma contribuição para o ensino de teste de software.* Master's thesis. Universidade de São Paulo.

[31] Pedro Henrique Dias Valle, Armando Maciel Toda, Ellen Francine Barbosa, and José Carlos Maldonado. 2017. Educational Games: A Contribution to Software Testing Education. In *XLVII Annual Frontiers in Education (FIE)*. IEEE.

[32] Pedro Henrique Dias Valle, Ricardo Ferreira Vilela, Paulo Afonso Parreira Júnior, and Ana Carolina Gondim Inocêncio. 2013. HEDEG-Heurísticas para Avaliação de Jogos Educacionais Digitais. In *XVIII Conferência Internacional sobre Informática na Educação (TISE)*. Nuevas Ideas En Informática Educativa, Porto Alegre, Brasil.