Integração de atividades de programação no ensino de teste de software em um ambiente com elementos de jogos

Vinicius Bosa Petris[1], **Marco Aurélio Graciotto Silva**[1]

[1] Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Computação (DACOM)
Campo Mourão – PR – Brasil
viniciuspetris@alunos.utfpr.edu.br, magsilva@utfpr.edu.br

Abstract. *The objective of this study was to evaluate the effect of integrating programming activities while teaching of software testing using game elements. A study was carried out on the implementation of programs and unit tests in Java with the Code Defenders, evaluating the effect of the game elements and the quality of the programs. As a result, it was determined that the quality of Code Defender for teaching software testing is good, especially regarding students' interest. For the program's quality, most students were able to successfully implement the specified programs.*

Resumo. *O objetivo deste trabalho foi avaliar a integração de atividades de programação no ensino de teste de software usando elementos de jogos. Foi executado um estudo sobre implementação de programas e testes de unidade em Java com a ferramenta Code Defenders, avaliando-se o efeito do jogo e a qualidade dos programas. Como resultado, aferiu-se que a qualidade do Code Defender para ensino de teste é boa, destacando-se pelo interesse dos estudantes. Quanto à qualidade dos programas, a maioria dos estudantes conseguiu implementar com êxito os programas especificados.*

1. Introdução

Habilidades de programação e de teste de software são necessárias na formação de engenheiros de software que criam produtos de qualidade. Ambas são importantes e requerem prática e experiência para complementar a aprendizagem [Carrington 1997]. Entretanto, embora programação seja amplamente ensinada em cursos de Computação, os estudantes formados possuem deficiência em habilidades de teste de software [Scatalon et al. 2019].

Uma maneira para lidar com esse problema é abordar teste de software desde o começo do curso, integrando-o ao currículo e aplicando métodos de ensino no qual se ensina programação vinculada a testes. No entanto, os estudantes podem apresentar uma atitude negativa em relação ao aprendizado de teste tão precocemente, mesmo sabendo de sua importância [Scatalon et al. 2019]. Uma solução potencial é utilizar jogos ou elementos de jogos (gamificação) no aprendizado.

A gamificação é uma técnica para utilizar design de jogos em contextos não-jogos, com o objetivo de reter pessoas e engajá-las a realizarem determinadas tarefas [Deterding et al. 2011]. Ela permite transformar atividades que são de pouco interesse, por serem desinteressantes ou difíceis, em tarefas divertidas e competitivas [Fraser 2017].

Especificamente quanto ao ensino gamificado de teste de software, algumas abordagens podem ser citadas. O Testing Game [Valle et al. 2017b] alinha conteúdos teóricos

e práticos sobre teste de software, abordando três técnicas de teste: funcional, estrutural e baseada em defeitos. Cada uma das técnicas possui um determinado número de fases, que o jogador deve completar para avançar no jogo. Outra ferramenta é o Pex4Fun [Tillmann et al. 2011], que apresenta como principal elemento de gamificação os duelos de codificação, no qual os estudantes recebem uma implementação do professor, referentes a conceitos e tópicos estudados, e as implementam. A partir dessa ferramenta, originou-se o Code Hunt [Tillmann et al. 2014b], que possui a mesma característica de gamificação, porém o design de sua interface foi aprimorado, contando com animações, sons e frases com analogias a jogo de caça. Finalmente, o Code Defenders [Fraser 2017] é um jogo para ensinar teste de software com base nos princípios de teste de mutação, buscando motivar os estudantes a produzir testes mais rigorosos, empregando elementos de jogos como pontos, duelos e níveis [de Jesus et al. 2018].

Nesse contexto, este trabalho tem como objetivo avaliar o efeito da integração de atividades de programação no ensino de teste de software em um ambiente com elementos de jogos, explorando as seguintes questões de pesquisa: **QP1:** A integração de atividades de programação no ensino de teste de software em um ambiente com elementos de jogos permite um melhor aprendizado de teste de software? **QP2:** A qualidade do ambiente para ensino de teste de software integrado à programação é satisfatória?

O estudo contemplou a alteração e uso do Code Defenders para a integração de programação e teste de software. Foram realizadas atividades no Code Defenders para estudantes do curso de Ciência da Computação. Para a primeira questão de pesquisa foram avaliados o código do programa e os testes de unidade criados pelos estudantes. Analisando os testes de unidade através da cobertura de código, de desvios e de mutantes, o resultado apresentado foi satisfatório. Para responder a segunda questão de pesquisa foi utilizado o método de avaliação de jogos educativos MEEGA+, que estabelece um questionário como instrumento para esta avaliação [Petri et al. 2019]. Como resultado, o Code Defenders foi avaliado no nível de qualidade boa referente à qualidade do jogo, apresentando atividades desafiadoras, proporcionando sentimentos de confiança e satisfação aos jogadores e sendo relevante para os interesses dos estudantes.

O restante deste trabalho está organizado da seguinte forma. Na Seção 2 são apresentados trabalhos relacionados de ferramentas gamificadas para o ensino de teste de software. Na Seção 3 são descritas as alterações feitas no Code Defenders e o desenho experimental do estudo. Os resultados são apresentados na Seção 4 e discutidos na Seção 5. Por fim, na Seção 6 apresentamos as conclusões.

2. Trabalhos relacionados

Os jogos sérios são jogos que têm outros objetivos além do puro entretenimento e incluem todos os aspectos da educação: ensino, treinamento e informação. Jogos sérios podem ser uma forma motivadora de ensinar tópicos de ciência da computação. Um elemento fundamental nos jogos em geral é um alto nível de interação entre os jogadores. Um jogador deve estar ativamente envolvido no jogo para ter sucesso nele [Hakulinen 2011].

Além dos jogos sérios, a gamificação aparece como uma forma para motivar e aumentar a atividade e retenção do usuário. A gamificação consiste em usar elementos de design de jogos em contextos não relacionados a jogos e está gerando um intenso número de aplicações [Deterding et al. 2011]. A seguir, aborda-se o uso de jogos e de gamificação

no ensino de teste de software.

Pex4Fun e o Code Hunt trazem uma experiência de aprender a programar através da diversão, que é utilizada como um ingrediente vital para acelerar o aprendizado e a obtenção de uma habilidade necessária [Tillmann et al. 2014a]. O Pex4Fun é uma plataforma *web* [Tillmann et al. 2011] para o ensino baseado em jogos para as linguagens de programação C#, Visual Basic e F#. O professor é responsável por criar a especificação do problema na forma de um programa e os estudantes implementam a solução, a fim de tornar o comportamento do respectivo programa igual ao do professor. O comportamento não é diretamente mostrado para os estudantes, mas apresentado por casos de teste gerados a partir do código do professor, considerando uma técnica baseada em execução simbólica dinâmica [Xie et al. 2013].

O principal elemento de gamificação do Pex4Fun é o duelo de codificação. Ele é flexível o suficiente para permitir que o professor crie vários jogos para direcionar o aprendizado a uma série de habilidades, como programação, depuração, resolução de problemas, teste e especificação de escrita, com diferentes níveis de dificuldade [Xie et al. 2013]. Além do mecanismo de duelos, o Pex4fun utiliza recursos relacionados à dinâmica social, como visualizar e comparar as habilidades de duelo de cada jogador e apresentar a classificação dos jogadores através do sistema de pontuação [Tillmann et al. 2013].

A partir do Pex4Fun originou-se a Code Hunt, uma plataforma *web* que disponibiliza um jogo de codificação educacional para ensinar programação nas linguagens Java e C# [Tillmann et al. 2014c]. No Code Hunt, os duelos são divididos em setores. Cada setor tem um tema e diferentes níveis, que aumentam a dificuldade do jogo. Os níveis são desafios semelhantes aos duelos do Pex4Fun, com um código base para o qual o estudante deve implementar iterativamente, analisando as informações do Pex [Tillmann et al. 2014b]. Quando um jogador resolve um duelo, ele recebe uma quantidade de pontos. A plataforma possui uma tabela global de liderança.

O Testing Game é um jogo educacional sobre teste de software, com conteúdos teóricos e práticos [Valle et al. 2017b]. Ele as técnicas de teste funcional, estrutural e baseada em defeitos. Adota-se uma estratégia incremental de teste organizada em iterações, em que cada iteração representa uma técnica de teste e possui um determinado número de fases. A cada fase o jogador tem acesso a um módulo de conteúdo teórico sobre os respectivos assuntos [Valle et al. 2017a].

O Code Defenders é uma aplicação *web* que implementa uma abordagem de gamificação para teste de mutação, utilizando o *framekwork* JUnit para a criação de testes automatizados na linguagem de programação Java. O jogo estimula a criação de conjuntos de casos de teste de qualidade (que distinguem mutantes do programa correto) e de mutantes difíceis de serem distinguidos do programa correto (requisito de teste) ou funcionalmente equivalentes (mutante equivalente). Para isso, os jogadores podem assumir um dos seguintes papéis: atacante, que pretende criar mutantes, que são modificações sutis do programa em teste; ou defensor, que escreve testes que revelam essas modificações e matam mutantes. Ambas as funções incentivam e treinam o entendimento de defeitos de software e como eles são detectados pelos testes [Rojas e Fraser 2016c].

O Code Defenders baseia-se na competitividade para envolver os jogadores. Através de um sistema de pontuação, incentiva-se a produção de mutantes e de conjunto de

casos de teste [Clegg et al. 2017]. De modo geral, os atacantes vencem se produzem mutantes que sobrevivem a teste dos defensores por mais tempo. Os defensores vencem se produzem casos de teste que matam a maioria dos mutantes [Rojas e Fraser 2016a].

Para criar um jogo, o primeiro recurso disponível para a escolha do jogador é especificar uma classe de teste, escolhendo-a em uma lista predefinida de exemplo ou fazendo o envio de uma classe diferente. O segundo recurso é o modo de jogo: duelo (um atacante vs um defensor) ou batalha (time de atacantes vs time de defensores). Por fim, são configurados o número total de rodadas e o nível de dificuldade, fácil ou difícil. No nível fácil, os mutantes são totalmente mostrados aos defensores, enquanto que, no nível difícil, existe o ocultamento de informações: atacantes não tem acesso aos casos de teste (mas tem acesso aos dados de cobertura do programa sob teste) e defensores não tem acesso à alteração do código feita no mutante, mas da linha em que está localizada a alteração no programa sob teste [Rojas e Fraser 2016b].

3. Método

Em relação a este trabalho, o Code Defenders foi escolhido por estar disponível como software livre e permitir a criação, compilação e execução de casos de teste automatizados, sendo necessária, como principal alteração para este estudo, a flexibilização do início do jogo, permitindo a criação de programas pelos estudantes diretamente pela ferramenta.

Originalmente, o Code Defenders permite escolher programas do próprio banco de dados ou subir um programa completo na plataforma. Neste trabalho, o jogador pode criar seu próprio código dentro da plataforma com o objetivo de promover o aprendizado integrado de programação e teste. Assim, são proporcionados dois momentos para exercitar diretamente habilidades de programação: na criação do programa e na criação dos casos de teste automatizados. Como na criação dos mutantes são observados erros de programação que levam a defeitos no software, indiretamente também é exercitada a habilidade de programação, com a criação de mutantes que são mais difíceis de satisfazer ou que permitem identificar erros no programa (mutante revelador de erro).

Além dessa alteração, foram realizados ajustes quanto ao mecanismo de gamificação. Na ferramenta, não existia um limite quanto à criação de mutantes ou de casos de teste. Isso pode promover a tentativa e erro, o que não proporciona uma experiência de aprendizado tão adequada quanto a reflexão em ação [Edwards 2004]. Para inibir esse comportamento, foi estabelecido um custo para cada atividade e um crédito para ser gasto, incentivando os jogadores a refletir para criar mutantes que não irão morrer facilmente e para criar casos de teste de maior qualidade. A ferramenta Code Defenders com as alterações está disponível em `https://github.com/vnc10/CodeDefenders`.

A abordagem proposta foi avaliada em um estudo de caso, considerando tanto a perspectiva da qualidade dos casos de teste e do código, e quanto de elementos de jogos associados. A população do estudo consistiu em estudantes do curso de Ciência da Computação. A amostra foi obtida por conveniência, através de convite a estudantes de disciplinas de Engenharia de Software no primeiro semestre do ano letivo de 2021. Os participantes implementaram dois programas cada um, e criaram uma partida para cada implementação. A partir destes jogos criados, eles participaram dos próprios jogos e dos jogos de seus colegas, alternando no papel de atacante e defensor.

Para a execução do estudo foram desenvolvidas atividades para dois problemas

típicos de disciplinas de algoritmos e estruturas de dados: ordenação (*bubble sort*) e estruturas de dados (fila). Para esses problemas, foram criadas implementações de referência dos algoritmos em Java. Quando possível, o código do programa e casos de teste foram extraídos de bibliotecas existentes, como Apache Commons Collections [ASF 2001].

Para avaliar os programas criados pelos estudantes, utilizaram-se os casos de teste de referência, com cobertura de 100% de instruções e desvios e de ao menos 85% para mutação. Para avaliação dos casos de teste, foram consideradas as coberturas de instruções e desvios calculadas pela ferramenta JaCoCo [Hoffmann et al. 2009] e a pontuação de mutação calculada pela ferramente PIT [Coles et al. 2016]. Quanto à avaliação dos elementos de jogos, foi escolhido o método MEEGA+ [Petri et al. 2019].

4. Resultados

O estudo abrangeu estudantes do curso de Ciência da Computação, com amostra obtida por conveniência, através de convite em disciplina de Engenharia de Software no primeiro semestre acadêmico de 2021. Foram convidados 12 estudantes, dos quais 7 participaram.

4.1. Análise da integração de atividades de programação e teste de software (QP1)

A primeira parte do estudo foi realizada em aula síncrona com duração de 1h50m. Após consentida a participação no estudo, todos os estudantes responderam ao questionário sobre a experiência de programação e teste de software.

Todos os estudantes possuíam experiência na linguagem C e apenas um em Java. Em relação a teste de software, a maioria (4) afirmou ter alguma experiência com teste de unidade. Através destes números pode-se concluir que a maioria dos estudantes não possuía experiência prévia com a linguagem de programação Java. Não saber Java, mas ser competente em programação, é conveniente para o estudo, dado que os erros cometidos provavelmente serão alinhados com as hipóteses de programador competente e, espera-se, referentes a detalhes da linguagem.

Seguindo com o estudo, cada participante acessou a plataforma do Code Defenders. Houve uma breve explicação de como jogar na plataforma, iniciar uma partida, criar mutantes e testes de unidade. Foram disponibilizados documentação e vídeo com explicações para uso durante o estudo. Para a primeira implementação foi proposto o desenvolvimento do *bubble sort* e assim como seus mutantes e testes de unidade. A primeira implementação teve a participação de todos os estudantes que participaram da primeira etapa do estudo, ou seja, 7 implementações. Para a segunda parte do estudo, foi disponibilizada uma atividade referente à estrutura de fila para realização de forma assíncrona (fora do período de aula). Novamente, todos os participantes realizaram essa atividade.

A partir dos códigos implementados pelos participantes, eles mesmo criaram suas partidas, sendo uma para o código de *bubble sort* e outra para a fila no modo de jogo fácil, onde tanto o defensor consegue ver os códigos alterados para as criações de mutantes quanto o atacante consegue ver o código do teste de unidade e suas linhas de cobertura. Esse modo de jogo tem como objetivo não estragar a experiência deles durante o aprendizado de teste de unidade, pois como a maioria deles não tem experiência com a linguagem Java, o modo difícil iria dificultar o aprendizado, desmotivando-os a jogar. Após criar a partida, cada estudante participou como atacante ou defensor das partidas dos colegas.

Referente à qualidade das implementações dos algoritmos e estrutura de dados propostos, das 7 implementações do algoritmo de ordenação, 4 passaram por todos os testes e 3 falharam em, ao menos, um caso de teste. Quanto à estrutura de fila, três participantes desenvolveram códigos com falhas reveladas pelos casos de teste.

Analisando o progresso dos participantes, os participantes 1, 2 e 5 implementaram ambos os códigos perfeitamente, já com os participantes 3 e 4 ocorreu o inverso, implementaram ambas de forma errônea. O participante 6 implementou o código do *bubble sort* corretamente, porém implementou errado o código da fila. Vale ressaltar que a segunda implementação é mais complexa que a primeira, o que pode ter sido um fator para ele não ter conseguido. O participante 7 não conseguiu implementar o algoritmo de ordenação corretamente, enquanto implementou corretamente, embora de forma parcial, a estrutura de dados de fila, mostrando um possível avanço no aprendizado.

Durante os jogos, o Code Defenders salva toda tentativa de criação de teste de unidade, mesmo quando o teste não compila corretamente. Ao todo, a plataforma salvou 81 tentativas, divididas em 42 para a primeira atividade e 39 para a segunda. Dessas 81 tentativas, 33 testes foram compilados corretamente e enviados para a partida, de forma que 10 foram para o *bubble sort* e 23 para a fila.

Os 10 testes de unidade referentes ao código do *bubble sort* foram desenvolvidos em 3 partidas: na primeira foram desenvolvidos 5 testes, na segunda 4 e no última 1. Dos participantes, apenas três criaram com sucesso casos de teste. Em todos as partidas foram alcançados 100% de coberturas de instruções e de desvios. Analisando os testes de unidade quanto ao critério de análise de mutantes, todos os participantes que tiveram êxito na criação de seus testes alcançaram um total de 85% de cobertura de mutação.

A implementação dos testes na fila foi dividida em 4 partidas, sendo 4 testes para a primeira, 9 para a segunda, 9 para a terceira e 1 para a quarta, totalizando 23 casos de teste. Na primeira partida, todos os 4 testes de unidade tiveram sua implementação correta, porém 2 falharam. A cobertura de instruções foi apenas de 57% e de desvios foi de 40%, enquanto a cobertura de mutantes alcançou 63%. Na segunda partida, dos 9 testes, 5 falharam, 1 não continha assertivas e 3 passaram. O resultado dos testes que passaram foi satisfatório, já que alcançou um total de 92% de cobertura de instrução, 70% de cobertura de desvio e 75% de cobertura de mutação. Na terceira partida tivemos 2 jogadores defendendo. O primeiro jogador fez apenas um teste, o qual passou e teve um total de cobertura de instrução de 26%, cobertura de desvio de 20% e cobertura de mutação de 33%. O segundo defensor implementou 8 testes de unidade e alcançou resultados satisfatórios de cobertura: 79% para instrução, 80% para desvio e 77%. Na última partida foi realizado apenas um teste de unidade que teve um número de 57% de cobertura de instruções, 40% de cobertura de desvios e 63% de cobertura de mutação.

Em relação aos casos de teste, observou-se que uma parcela considerável continha erros de compilação (praticamente 60%), o que indica que os estudantes enfrentaram dificuldades em relação à linguagem de programação. Proporcionalmente, a quantidade de erros de implementação dos testes de unidade foi menor na segunda atividade (23 casos de teste corretos de um total de 39) do que na primeira (10 de 42). Isso sugere que as habilidades de programação melhoraram com o emprego da abordagem.

4.2. Análise do Code Defenders quanto à gamificação (QP2)

Ao final da segunda atividade, cada participante preencheu o formulário referente ao instrumento MEEGA+, para avaliação dos atributos de jogo do Code Defenders quanto ao aprendizado. Assim, foi disponibilizado questionário com 35 perguntas, sendo 3 perguntas abertas sobre a experiência dentro do jogo e 32 perguntas de múltipla escolha com respostas em escala Likert de cinco níveis, com alternativas de resposta que variam de discordo totalmente a concordo totalmente.

Para classificar o jogo na escala MEEGA+, foi utilizado um script fornecido na documentação do MEEGA+ e utilizado o software estatístico R para o cálculo. Através do software é calculado a média das pontuações fornecidas de todos os participantes que avaliaram o jogo, aplicando a Teoria de Resposta ao Item (TRI). Após o cálculo da média, foi realizada uma transformação desses escores em uma escala (50,15), aplicando a fórmula no escore médio: $\Theta(50, 15) = 50 + 15 * \Theta(0.1) * (\text{média})$, em que a média é 0,294098133, obtendo assim o resultado de 49,559. Com base neste valor final, a qualidade do Code Defenders foi em avaliada no nível de qualidade boa, em que o jogo às vezes apresenta atividades desafiadoras, oferecendo novos desafios para os estudantes. Ele fornece atenção moderadamente focada aos jogadores, embora os estudantes não se esqueçam de seus arredores. Às vezes, o jogo também proporciona sentimentos de confiança e satisfação aos jogadores. Frequentemente o jogo apresenta momentos de interação social e diversão entre os jogadores. Frequentemente, o jogo é considerado relevante para os interesses dos estudantes e, geralmente, os estudantes reconhecem que o conteúdo do jogo está relacionado ao curso. Frequentemente, o jogo contribui de forma eficiente para o aprendizado do estudante. Em termos de usabilidade, o jogo costuma ter regras claras e é fácil de jogar, embora, normalmente, não apresente um design totalmente atraente [Petri et al. 2019].

Em relação às perguntas abertas, três respostas foram obtidas. A primeira pergunta tratava sobre a experiência do jogador na plataforma e qual o fator que mais gostou no jogo. Um dos participantes destacou que preferiu jogar de atacante do que de defensor: "A parte dos ataques me prendeu mais no jogo." O motivo pode se entender que, quando participando de um jogo na plataforma pela primeira vez, jogar de atacante é mais tentador por ser mais fácil, porém não cumpre com o propósito do jogo, que é aprimorar a habilidade de desenvolver teste de unidade. Em contraposição, um outro participante mostrou-se interessado em desenvolver teste de unidade dentro das partidas nos quais participou: "Achei interessante para quem cria os casos de teste, é desafiador criar o máximo de casos de teste para que nenhum erro de um atacante passe pela sua defesa." Finalmente, a última resposta é um comentário sobre a atratividade da plataforma, pois a abordagem de teste de mutação pode ser vista como algo inédito para os participantes:

A segunda pergunta questionava sobre o que poderia ser melhorado no jogo. A primeira resposta pode ser interpretada pela falta de material na plataforma de testes de unidades: "Especificação e uma explicação melhor sobre os testes." De fato, a plataforma não tem conteúdo integrado sobre testes, tanto que foi passado aos participantes documentação para consulta durante a atividade, porém fora da plataforma. Na segunda resposta, o participante afirmou que ficou confuso após jogar como atacante devido a uma questão de usabilidade referente ao rótulo dos botões. Na última resposta pode se fazer uma relação com a primeira, visto que é a falta de material didático sobre as regras dos jogos, apesar de serem apresentados para eles durante a aula, ainda houve uma deficiência

na parte da plataforma: “Regras mais claras, talvez um exemplo prático.”

A última pergunta era se o participante gostaria de fazer algum comentário sobre qualquer ponto que desejasse. Ela teve duas respostas. Na primeira resposta, o participante concordou que o Code Defenders ajudou no desenvolvimento de teste de unidade, apesar de ter encontrado alguma dificuldade. Já na segunda, o participante achou que teria mais dificuldades por se tratar de uma linguagem pelo qual ele não usa, porém ele superou a dificuldade e ainda destacou como pontos positivos o desenvolvimento dos testes e ter um pouco mais de prática com o Java: “Neste jogo, eu pensava que teria muita dificuldade, principalmente por não ter muita afinidade com Java, mas acabou sendo interessante, consegui fazer os códigos e os casos de teste, sem tanta dificuldade. Foi possível entender como fazer casos de teste e a ter mais contato com a linguagem Java.”

5. Discussão

De modo geral, considerando os resultados da correção dos casos de teste e da cobertura desses em relação aos critérios de cobertura de instruções, desvios e análise de mutantes, foi possível observar indícios que, para algoritmos simples, o aprendizado integrado de programação e teste de software possui melhores resultados. Para problemas mais complexos, envolvendo algoritmos e estruturas de dados, a qualidade da implementação da fila foi variável, com um grupo de estudantes cuja implementação passou em todos os casos de teste de referência e outro grupo em que a maioria dos casos de teste falharam. Resultado similar pode ser observado quanto à qualidade dos casos de teste criados pelos estudantes, com maior qualidade para aqueles criados para a primeira atividade e com apenas dois estudantes cujos conjuntos de teste obtendo coberturas superiores à 70% para os critérios de teste analisados na segunda atividade.

No questionário respondido ao final do estudo, os estudantes relatam a motivação em criar casos de teste, indicando o potencial benéfico do uso de gamificação para ensino de teste. Assim, quanto à primeira questão de pesquisa, o emprego de gamificação na integração de atividades de programação e de teste de software apresenta indícios de que é possível conciliar esses objetivos educacionais e alcançar resultados de aprendizagem satisfatórios, embora seja necessário investigar como melhorar essa integração.

Na configuração atual da ferramenta Code Defenders, ainda não é possível alterar o código do programa durante a criação da partida ou enquanto outros jogadores testam ou criam mutantes. Isso é um limitador quanto ao emprego de teste durante a programação. Na ferramenta PeX4Fun/Code Hunt [Tillmann et al. 2014b], a criação do programa guiada por casos de teste apresenta-se como um atrativo importante. Em trabalho futuro, permitir o desenvolvimento e correção incrementais do programa, a partir dos resultados referentes aos casos de teste e criação de mutantes, é outro ponto a ser explorado.

Quanto à segunda questão de pesquisa, o estudo indica que a qualidade do ambiente gamificado para o ensino de teste de software integrado à programação é boa. No entanto, ainda existem oportunidades para melhoria. Um dos aspectos a serem aperfeiçoados é a integração de conteúdos referentes aos tópicos e melhorias de aspectos de gamificação e usabilidade. De forma similar ao Testing Game [Valle et al. 2017b], é necessário alinhar os conteúdos teóricos e práticos, apresentando-os aos estudantes, sem quebrar a estrutura lúdica. Uma forma de realizar isso é pela oferta de tutoriais, ensinando a mecânica do jogo e, paralelamente, associando os conceitos e habilidades de programação e teste, de

forma escalonada. Outro ponto é corrigir deficiências de usabilidade, que atrapalham a experiência dos estudantes durante o uso da ferramenta.

6. Conclusões

Para o desenvolvimento de software com mais qualidade é necessário promover o uso de técnicas de teste de software integradas com atividades de programação. Uma solução potencial é abordar gamificação junto ao aprendizado de teste de software. Neste artigo foi abordada a alteração e utilização da ferramenta Code Defenders para integrar atividades de programação no ensino de teste de software em um ambiente com elementos de jogos. Assim, cada estudante pode criar um programa Java dentro na plataforma ao invés de apenas submeter o arquivo com o código escrito. Além disso, foi adicionado um mecanismo para incentivar os jogadores a refletir para criar mutantes e criar casos de teste, evitando a tentativa e erro possível na versão original da ferramenta.

Quanto à primeira questão de pesquisa sobre a integração de atividades de programação no ensino de teste de software em um ambiente com elementos de jogos, pode-se notar, através dos códigos de programa e dos testes de unidades desenvolvidos pelos estudantes, que eles foram capazes de alcançar o objetivo proposto pela atividade. Para a segunda questão de pesquisa, a qual se refere à qualidade do Code Defenders para o ensino, constatou-se que a plataforma possui uma boa qualidade de gamificação, apresentando atividades desafiadoras para os estudantes.

Dado o objetivo do presente estudo em avaliar o efeito da integração de atividades de programação no ensino de teste de software em um ambiente com elementos de jogos, a ferramenta apresentou-se como como uma alternativa a estudantes da computação que buscam aprender ou melhorar a habilidade de desenvolvimento de testes de unidade e praticar o desenvolvimento da linguagem de programação Java. O ambiente oferece elementos de jogos para o estudante se sentir fora de uma atividade comum à sala de aula, trazendo uma experiência mais divertida a eles.

Referências

ASF (2001). Commons collections. Programa de computador.

Carrington, D. (1997). Teaching software testing. In *2nd Australasian Conference on Computer Science Education*, p. 59–64. ACM.

Clegg, B., Rojas, J. M., Fraser, G. (2017). Teaching software testing concepts using a mutation testing game. In *39th ICSE – SEET*, p. 33–36.

Coles, H., Laurent, T., Henard, C., Papadakis, M., Ventresque, A. (2016). PIT: A practical mutation testing tool for Java (demo). In *25th ISSTA*, p. 449–452. ACM.

de Jesus, G. M., Ferrari, F. C., de Paula Porto, D., Fabbri, S. C. P. F. (2018). Gamification in software testing: A characterization study. In *III Braziliam Symposium on Software Testing (SAST 2018)*, p. 39–48. ACM.

Deterding, S., Dixon, D., Khaled, R., Nacke, L. (2011). From game design elements to gamefulness: Defining "gamification". In *15th MindTrek Conference*, p. 9–15. ACM.

Edwards, S. H. (2004). Using software testing to move students from trial-and-error to reflection-in-action. In *35th SIGCSE Technical Symposium on Computer Science Education*, p. 26–30. ACM.

Fraser, G. (2017). Gamification of software testing. In *12th International Workshop on Automation of Software Testing*, p. 2–7. IEEE.

Hakulinen, L. (2011). Using serious games in computer science education. In *11th Koli Calling International Conference on Computing Education Research*, p. 83–88. ACM.

Hoffmann, M. R., Mandrikov, E., Friedenhagen, M., Liba, R., Beck, C., Janiczak, B., others (2009). JaCoCo – java code coverage library. Programa de computador.

Petri, G., von Wangenheim, C. G., Borgatto, A. F. (2019). MEEGA+: Um modelo para a avaliação de jogos educacionais para o ensino de computação. *Revista Brasileira de Informática na Educação (RBIE)*, 27(3):52–81.

Rojas, J. M. Fraser, G. (2016a). Code Defenders: A mutation testing game. In *9th ICST Workshops – International Workshop on Mutation Analysis*, p. 162–167.

Rojas, J. M. Fraser, G. (2016b). Teaching mutation testing using gamification. In *European Conference of Software Engineering Education 2016*, p. 1–5. Shaker.

Rojas, J. M. Fraser, G. (2016c). Teaching software testing with a mutation testing game. In *27th Annual Workshop Psychology of Programming Interest Group*, p. 290–293.

Scatalon, L. P., Carver, J. C., Garcia, R. E., Barbosa, E. F. (2019). Software testing in introductory programming courses: A systematic mapping study. In *50th Technical Symposium on Computer Science Education*, p. 421–427.

Tillmann, N., Bishop, J., Horspool, N., Perelman, D., Xie, T. (2014a). Code Hunt: Searching for secret code for fun. In *7th International Workshop on Search-Based Software Testing*, p. 23–26. ACM.

Tillmann, N., De Halleux, J., Xie, T. (2011). Pex4Fun: Teaching and learning computer science via social gaming. In *24th Conference on Software Engineering Education and Training*, p. 546–548. IEEE Computer Society.

Tillmann, N., de Halleux, J., Xie, T. (2014b). Transferring an automated test generation tool to practice: From Pex to Fakes and Code Digger. In *29th International Conference on Automated Software Engineering*, p. 385–396. ACM.

Tillmann, N., de Halleux, J., Xie, T., Bishop, J. (2014c). Code Hunt: Gamifying teaching and learning of computer science at scale. In *1st ACM Conference on L@S*, p. 221–222.

Tillmann, N., De Halleux, J., Xie, T., Gulwani, S., Bishop, J. (2013). Teaching and learning programming and software engineering via interactive gaming. In *35th International Conference on Software Engineering*, p. 1117–1126.

Valle, P. H. D., Rocha, R. V., Maldonado, J. C. (2017a). Testing game: An educational game to support software testing education. In *31st Brazilian Symposium on Software Engineering*, p. 289–298. ACM.

Valle, P. H. D., Toda, A. M., Barbosa, E. F., Maldonado, J. C. (2017b). Educational games: A contribution to software testing education. In *2017 IEEE Frontiers in Education Conference*, p. 1–8.

Xie, T., Tillmann, N., de Halleux, J. (2013). Educational software engineering: Where software engineering, education, and gaming meet. In *3rd International Workshop on Games and Software Engineering*, p. 36–39. IEEE.