UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO MENDES DE SOUZA

INVESTIGAÇÃO SOBRE GAMIFICAÇÃO EM DISCIPLINAS INTRODUTÓRIAS DE PROGRAMAÇÃO

MONOGRAFIA

CAMPO MOURÃO
2017

BRUNO MENDES DE SOUZA

INVESTIGAÇÃO SOBRE GAMIFICAÇÃO EM DISCIPLINAS INTRODUTÓRIAS DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso de graduação apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso de Bacharelado em Ciência da Computação do Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Marco Aurélio Graciotto Silva

CAMPO MOURÃO

2017

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO

Às **18:00** do dia **29 de novembro de 2017** foi realizada na sala **E102** da UTFPR-CM a sessão pública da defesa do Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação do(a) acadêmico(a) **Bruno Mendes De Souza** com o título **Investigação sobre gamificação em disciplinas introdutórias de programação**. Estavam presentes, além do(a) acadêmico(a), os membros da banca examinadora composta por: **Prof. Dr. Marco Aurélio Graciotto Silva** (orientador), **Prof. Dr. Marcos Silvano Orita Almeida** e **Prof. Dr. Radames Juliano Halmeman**. Inicialmente, o(a) acadêmico(a) fez a apresentação do seu trabalho, sendo, em seguida, arguido(a) pela banca examinadora. Após as arguições, sem a presença do(a) acadêmico(a), a banca examinadora o(a) considerou _____ na disciplina de Trabalho de Conclusão de Curso **2** e atribuiu, em consenso, a nota _____ (_____). Este resultado foi comunicado ao(à) acadêmico(a) e aos presentes na sessão pública. A banca examinadora também comunicou ao acadêmico(a) que este resultado fica condicionado à entrega da versão final dentro dos padrões e da documentação exigida pela UTFPR ao professor Responsável do TCC no prazo de **onze dias**. Em seguida foi encerrada a sessão e, para constar, foi lavrada a presente Ata que segue assinada pelos membros da banca examinadora, após lida e considerada conforme.

Observações: _____

_____

_____

_____

_____

_____

Campo Mourão, **29 de novembro de 2017**

Prof. Dr. Marcos Silvano Orita Almeida
Membro 1

Prof. Dr. Radames Juliano Halmeman
Membro 2

Prof. Dr. Marco Aurélio Graciotto Silva
Orientador

A ata de defesa assinada encontra-se na coordenação do curso.

Agradecimentos

A oportunidade de realizar um trabalho como este, a experiência adquirida, e os novos conhecimentos conquistados, são resultados muitos satisfatórios. Agradeço a Universidade Tecnológica Federal do Paraná e ao corpo docente do Departamento Acadêmico de Computação (DACOM) pelos anos em que estive aqui.

Esta pesquisa não foi realizada somente por uma mim, mas por todos aqueles que de alguma forma contribuíram para minha formação. Agradeço primeiramente ao meu orientador, Marco Aurélio Graciotto Silva, pela melhor orientação que eu poderia imaginar.

Sou grato aos membros da banca, os professores Marcos Silvano Almeida e Radames Juliano Halmeman, por aceitarem de bom grado este convite especial e contribuírem com comentários e sugestões.

Por fim, sou grato aos meus amigos, em especial a Mairieli Wessel, e a toda minha família: minha mãe Vilma Mendes, ao meu pai Jose Marcos, e aos meus irmãos Patricia Reiffer e Rodrigo Ben Hur, por todo o apoio que me deram.

Agradeço também a Fundação Araucária pelo apoio financeiro na realização de uma Iniciação em Desenvolvimento Tecnológico e Inovação que fez parte desta pesquisa.

Resumo

Mendes de Souza, Bruno. Investigação sobre gamificação em disciplinas introdutórias de programação. 2017. 53. f. Monografia (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2017.

Contexto: Disciplinas de introdução a programação são consideradas difíceis e desafiadoras por alunos ingressantes. O *feedback* e a motivação para estudar são fatores importantes para que os alunos se envolvam com estas disciplinas. Diversos cursos são implementados em ambientes virtuais de aprendizagem, sendo um desses tipos os cursos MOOC, do inglês *massive open online courses*. Normalmente cursos MOOCs são ofertados em plataformas de MOOCs, sendo estes cursos em qualquer área, inclusive em Computação. Uma técnica utilizada para melhorar a aprendizagem de alunos é a gamificação, em que se utilizam elementos de *design* de jogos em contexto não-jogos.
Objetivo: O objetivo deste trabalho foi propor e implementar elementos de *design* de jogos como gamificação para disciplinas introdutórias de programação, respondendo três questões de pesquisa, de tal maneira que estes elementos motivassem os alunos a realizarem códigos com qualidade e corretude em atividades avaliativas e, que a gamificação permiti-se a ausência de professores nas avaliações.
Método: O primeiro passo para o método foi a escolha de uma plataforma de MOOCs para a implementação da gamificação proposta. Assim que definida a plataforma de MOOCs, foi necessário propor e escolher elementos de *design* de jogos que poderiam ser utilizados, como o elemento de competição entre os alunos. A implementação da gamificação foi junto de plataforma de MOOCs de código aberto Open edX. Para avaliar a gamificação junto à plataforma de MOOCs, foi preparado e ofertado um curso introdutório à programação na plataforma. No período que o curso foi oferecido, foram realizadas atividades de avaliação de desempenho dos alunos participantes. Utilizando a técnica de testes A/B, a cada atividade na plataforma de MOOCs, um grupo diferente de alunos resolveu a atividade com gamificação e a outra parte sem a gamificação, desta forma oferecendo uma maior confiabilidade para os resultados. Por fim, ao final do período da disciplina, foi fornecido um questionário aos alunos participantes para caracterizar a motivação, e obter um *feedback* sobre os elementos de *design* de jogos selecionados e o modo como a gamificação foi proposta.
Resultados: Para a primeira questão de pesquisa, no qual refere-se a diferença entre a realização de atividades com gamificação e sem gamificação, utilizando um método estatístico, não foi

observado tal diferença, logo, não foi possível obter uma resposta satisfatória. Na segunda questão de pesquisa, analisando as submissões dos alunos e suas posições no *ranking*, observamos que a gamificação pode ser um fator auxiliar na correção das soluções. Finalmente na terceira questão, obtendo a opinião dos alunos no experimento, a gamificação, com a competitividade, motivou a entrega das soluções das atividades na plataforma.
Conclusão: Neste trabalho foi implementado um MOOC gamificado com um mecanismo de competição que foi capaz de motivar os alunos a realizarem as atividades avaliativas. Também foi possível notar que os elementos utilizados podem facilitar a correção das soluções por parte do professor. Ainda que, diante os resultados apresentados de acordo com o experimento realizado, faz-se necessário a adição de novos elementos à gamificação, principalmente uma maneira de guiar o aluno ao entendimento do MOOC gamificado.

Palavras-chaves: Gamificação. Educação. Computação. Programação. MOOC. Open edX.

Abstract

Mendes de Souza, Bruno. Investigation about gamification in introductory programming courses. 2017. 53. f. Monograph (Undergradute Program in Computer Science), Federal University of Technology – Paraná. Campo Mourão, PR, Brazil, 2017.

Context: Programming introduction courses are considered difficult and challenging by newcomer students. Feedback and motivation to study are important factors for students to engage with these courses. Several courses are implemented in virtual learning environments, one of these types are MOOC, massive open online courses. Usually, MOOC are offered in MOOC platform, being these courses in any area, including in Computing. A technique used to improve student learning is gamification, consisting of using game design elements in non-game context.
Objective: The objective of this work was to propose and implement game design elements as gamification for introductory programming courses, motivating students to develop code with quality and correctness in evaluative activities. The implementation of gamification was coupled with a MOOC platform, more specifically, the Open edX open source platform.
Method: The first step to the method was the choice of a MOOC platform, in which the proposed gamification was implemented. Once the MOOC platform was defined, it was necessary to propose and choose game design elements that could be used as the element of competition among students. To evaluate the gamification along the MOOC platform, an introductory course of programming was prepared and offered. Using the A/B test technique, for each activity in the MOOC platform, a different group of students solved the activity with gamification and the other part without the gamification, thus offering a greater reliability for the results. Finally, at the end of the course, a questionnaire was given to the participating students to characterize the motivation, and to obtain feedback on the selected game design elements and how the gamification was proposed.
Results: For the first research question, which refers to the difference between performing activities with gamification and without gamification, it was not possible to obtain a satisfactory answer. In the second research question, analyzing the students' submissions and their positions in the ranking, we observed that the gamification can be an auxiliary factor in the correction of the solutions. And in the third question, obtaining the opinion of the students in the experiment, the gamification, with the competitiveness, motivated the delivery of the solutions of the activities in the platform.
Conclusions: In this work was implemented a gamified MOOC with a competition mechanism

that was able to motivate students to perform code with quality and correctness. It was also possible to notice that the elements used can facilitate the correction of the solutions by the teacher. Although, given the results presented according to the experiment carried out, it is necessary to add new elements to the gamification, mainly a way to guide the student to the understanding of the gamified MOOC.

Keywords: Gamification. Education. Computation. Programming. MOOC. Open edX.

Lista de figuras

Lista de tabelas

Sumário

Capítulo 1

Introdução

Diversos cursos são implementados em ambientes virtuais de aprendizagem. Cursos elaborados por professores com a ajuda de tais ambientes permitem a montagem e gerenciamento de conteúdo, administração dos cursos e o acompanhamento das turmas de estudantes.

Cursos *online* abertos e massivos, MOOC, do inglês *massive open online courses*, são cursos abertos e normalmente disponibilizados para uma grande quantidade de alunos através da *Internet* (FASSFINDER et al., 2014). O ambiente e a estrutura dos cursos são projetados para permitir aos alunos uma aprendizagem exploratória e para possibilitar o aprendizado sem o auxílio do professor (PONTI, 2014).

Utilizando as mais variadas plataformas de MOOCs, como o Udemy ou Coursera, é possível ofertar MOOCs em diversas áreas, por exemplo: negócios, *marketing*, fotografia, ciências, músicas, idiomas e muitas outras. O foco desse trabalho está em cursos relacionados a conteúdos introdutórios de Computação e envolvendo programação.

Disciplinas de programação geralmente são cursadas por estudantes que nunca tiveram contato prévio com programação de computadores. Tais disciplinas são consideradas difíceis e desafiadoras por alunos ingressantes. O *feedback* e a motivação são fatores importantes para que os alunos se envolvam com estas disciplinas (IZEKI et al., 2016).

Considerando-se MOOCs introdutórios a computação e a grande quantidade de alunos nestes cursos, observa-se a presença de instrutores, professores e assistentes de ensino em proporção muito inferior àquela de cursos tradicionais. O MOOC tende a se tornar um ambiente que une a necessidade de emprego de métodos ativos de aprendizagem (necessários em qualquer modalidade de ensino, não se restringindo à MOOC) com métodos que escalam, ou seja, que são viáveis quanto a quantidade, custo, tempo e qualidade.

Em disciplinas de programação, uma das formas de garantir a qualidade nos programas e casos de testes dos alunos é a utilização de mecanismos de motivação. Outro método é permitir ao conjunto de alunos a avaliarem os trabalhos de outros alunos ao longo do curso ou utilizar

o método de instrução entre pares (SIMON; CUTTS, 2012). De certa forma, testes de software também contribuem para o aprendizado, incentivando alunos a criarem testes de software no início do ensino de programação (EDWARDS, 2004). Esta também é uma forma para que os próprios alunos avaliem seus códigos e, ao invés da abordagem de tentativa e erro, utilizem uma pratica de reflexão em ação. Outro elemento que pode ser utilizado são jogos, no qual é uma forma de melhorar o aprendizado do aluno em disciplinas de programação (EAGLE; BARNES, 2008). Utilizar elementos de *design* de jogos promove uma melhora no aprendizado dos alunos (SINGER; SCHNEIDER, 2012). Logo, é possível utilizar estes mecanismos de *design* de jogos como uma gamificação afim de motivar alunos realizarem códigos com qualidade.

A gamificação é uma técnica para utilizar *design* de jogos em contexto não-jogos, com o objetivo de reter pessoas e engajá-las a realizarem determinadas tarefas (DETERDING et al., 2011a). Utilizando elementos de jogos no ambiente virtual de aprendizado, pode ser possível motivar alunos a realizar os exercícios dos cursos com algoritmos e casos de teste com qualidade.

O objetivo deste trabalho foi propor e implementar elementos de *design* de jogos como gamificação para disciplinas introdutórias de programação, respondendo três questões de pesquisa, em que estes elementos motivassem os alunos a realizarem códigos com qualidade e corretude em atividades avaliativas e, que a gamificação permiti-se a ausência de professores nas avaliações.

Para a execução de trabalho, o primeiro passo foi a escolha de uma plataforma de MOOCs, na qual foi implementada a gamificação proposta. Assim que definida a plataforma de MOOCs, foi necessário propor e escolher elementos de *design* de jogos para serem utilizados, no caso deteste trabalho, um elemento de competição entre os alunos. Para avaliar a gamificação junto à plataforma de MOOCs, foi realizado um experimento ofertando um curso introdutório à programação na plataforma. Durante o período que o curso foi oferecido, foram realizadas atividades de avaliação de desempenho dos alunos participantes, no qual cada atividade na plataforma de MOOCs, um grupo diferente de alunos resolveu a atividade com gamificação e a outra parte sem a gamificação. Por fim, ao final do oferecimento da disciplina, foi fornecido um questionário aos alunos participantes para caracterizar a motivação, e obter um *feedback* sobre os elementos de *design* de jogos selecionados e o modo como a gamificação foi proposta.

No geral para uma diferença entre a realização de atividades com gamificação e sem gamificação não foi possível obter uma resposta satisfatória, mas analisando as submissões dos alunos e suas posições no *ranking*, observamos que a gamificação pode ser um fator auxiliar aos professores na correção das soluções. Ao final, obtendo a opinião dos alunos no experimento através de um questionário, observamos que a gamificação, com a competitividade, motivou a entrega das soluções das atividades na plataforma.

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 apresenta os fundamentos utilizados neste trabalho, MOOCs e gamificação, bem como os trabalhos relacionados e suas limitações. O MOOC gamificado, resultado deste trabalho de conclusão de curso, e detalhes de sua implementação, é apresentado no Capítulo 3. O Capítulo 4 apresenta a avaliação do MOOC

gamificado, descrevendo o experimento realizado, e os resultados e discussões obtidos. Por fim, o Capítulo 5 apresenta as conclusões referentes a este trabalho.

Capítulo 2

Referencial Teórico

Neste capítulo são apresentados os fundamentos utilizados neste trabalho, bem como os trabalhos relacionados. A Seção 2.1 apresenta os conceitos de MOOC e suas principais características, e a definição de gamificação, junto aos aspectos importantes para sua utilização e entendimento deste trabalho. A Seção 2.2 apresenta os trabalhos relacionados a este estudo juntamente com suas limitações. Por fim, a Seção 2.3 apresenta as considerações finais referentes a este capítulo.

2.1. Fundamentação Teórica

Esta monografia aborda dois conceitos para sua proposta, sendo eles MOOCs e gamificação. O uso de ambientes virtuais de aprendizagem são comuns nos dias de hoje (RUIZ et al., 2014). Os MOOCs são cursos ofertados através da *Internet* e podem envolver diferentes áreas. Para esse trabalho foi considerado a área de programação. Neste contexto, elementos de gamificação podem ser utilizados para melhorar o aprendizado dos alunos em disciplinas de programação (IZEKI et al., 2016).

2.1.1. MOOC

MOOC é o uso da tecnologia para fornecer oportunidades de aprendizado *online* para um grande número de alunos (DARADOUMIS et al., 2013). Basicamente, um curso aberto, *online* e massivo segue as seguintes características. É disponibilizado na *internet* e qualquer pessoa pode participar, sem limite de idade ou formação acadêmica. Escalável em termos do número de estudantes participantes, que pode variar de dezenas a milhares. Normalmente gratuito, podendo ser cobrado uma taxa por certos serviços. Por fim, envolve o uso de fóruns de discussão, questionários, exercícios, vídeos didáticos, leitura e conjuntos de problemas para avaliar a compreensão do aluno (SIVAMUNI; BHATTACHARYA, 2013).

Há uma diferença de nomenclatura para MOOCs: os cursos em si, as plataformas que permitem a um indivíduo criar um MOOC, proporcionando-lhe as ferramentas necessárias, e os provedores. As plataformas são denominadas *Massive Open Online Education Platform* (MOOEP) (SUBBIAN, 2013). As principais plataformas de MOOCs são: Google Course Builder, edX Platform, OpenMOOC e openHPI. Há também os provedores de MOOCs, no inglês MOOC *Provider*, no qual é um ambiente onde são somente disponibilizados MOOCs para os estudantes (FASSFINDER et al., 2014).

A Figura 2.1 demonstra os elementos básicos para a construção de um curso de MOOCs. A essência básica do componente MOOC é possuir cursos e alunos, apresentado na Figura 2.1 como Aprendiz. Uma vez construído os cursos, estes podem ser replicados em vários semestres para diferentes turmas. Os estudantes podem ingressar em diferentes turmas para cursar os cursos disponíveis na plataforma. Elementos instrucionais, no qual são elementos de aprendizagem, são partes importantes para o aprendizado do aluno. Um desses elementos são as atividades.

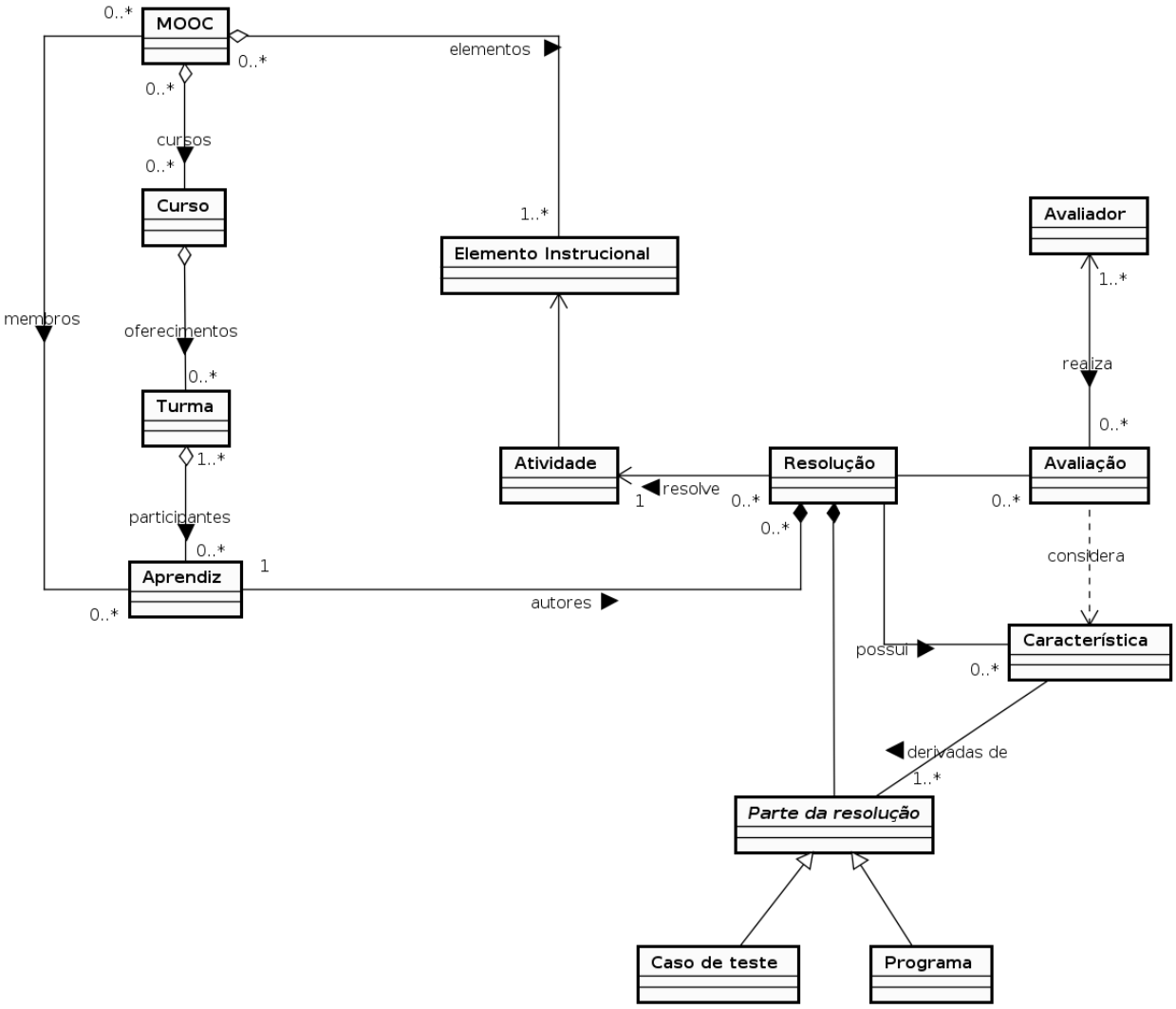


Figura 2.1. Componentes básicos de uma plataforma de MOOCs
Fonte: Autoria própria.

Os alunos produzem resoluções para as atividades ao decorrer dos cursos, resoluções que, por exemplo, em uma disciplina de programação podem ser divididas em partes, como o programa e os casos de testes. Tais resoluções necessitam de uma avaliação, podendo ser feita por um avaliador automático ou por um professor. Neste trabalho, os componentes de resolução e avaliador são aspectos importantes para a implementação dos elementos de gamificação.

Como qualquer MOOC pode chegar a um número elevado de alunos, por exemplo 10.000, não é possível realizar uma avaliação manual do desempenho de cada estudante. Neste caso, uma MOOEP pode oferecer ferramentas que facilitam a avaliação de exercícios de múltipla escolha. No entanto, há algumas atividades que não são triviais para um avaliador automatizado por haver certas nuanças e que, portanto, necessita de correção por um ser humano. Uma forma de avaliar estes tipos de exercícios é utilizando *crowdsourcing*, ou avaliação em pares (SUBBIAN, 2013). Cada atividade de um aluno é avaliada por múltiplos alunos, considerando um guia ou gabarito, e a nota final desta atividade é uma média das avaliações.

Em ambientes MOOCs, alguns problemas ainda devem ser tratados. Problemas que não são exclusivos no contexto de MOOCs, mas que se sobressaem, por exemplo, a autenticidade das submissões dos alunos, em que é difícil saber se um aluno fez suas atividades por conta própria ou alguém fez por ele (SIVAMUNI; BHATTACHARYA, 2013). A automotivação também é uma questão para MOOCs, pois é essencial para participação e a conclusão de cursos *online*. Os MOOCs têm fracas taxas de conclusão em comparação com cursos universitários tradicionais (SIEMENS, 2013). Por exemplo, em um curso MOOC do MIT de circuitos e eletrônica, de 155.000 alunos cadastrados, 7.157 concluíram o curso, e apenas 340 conseguiram a pontuação máxima no exame final (DANIEL, 2012). No quesito motivação, o mecanismo de gamificação é uma técnica que pode ser utilizada para reter alunos e motiva-los a realizarem as atividades avaliativas.

2.1.2. Gamificação

Gamificação, do inglês *gamification*, é o uso de elementos de *design* de jogos em contexto não-jogos (DETERDING et al., 2011b). O termo gamificação é originado da indústria de mídias digitais. Seu primeiro uso documentado remete a 2008, mas somente a partir de 2010 que teve uma adoção mais generalizada (DETERDING et al., 2011a).

Jogos atraem milhões de usuários em todo o mundo para gastar horas realizando tarefas puramente de entretenimento (DETERDING, 2012). Logo, a gamificação trata da utilização desse poder de motivação para a realização de tarefas específicas em diferentes contextos. Exemplos de comportamento de jogadores encontrados em jogos incluem: engajamento, interação, vício, competição, colaboração, conscientização, aprendizado e alcance de objetivos (DORLING; MCCAFFERY, 2012). Na gamificação, utilizam-se elementos que são característicos de jogos, ou seja, elementos que são encontrados na maioria dos jogos e que desempenham um papel significativo na jogabilidade.

2.1.3. Elementos de *design* de jogos

Aplicações utilizam elementos de *design* de jogos para motivar sua base de usuários, variando entre aplicações de produtividade, finanças, saúde, educação, sustentabilidade, bem como mídias de notícias e entretenimento (DETERDING et al., 2011a). Normalmente, oferecem a gamificação como uma camada de software que fornece sistemas de recompensa e reputação, como pontos, emblemas, níveis e placares de liderança.

Elementos de *design* de jogos são os princípios, regras ou mecanismos que geram um comportamento com resultados razoavelmente previsíveis no usuário. Estes elementos são blocos que podem ser aplicados e combinados para gamificar qualquer aplicação não-jogo (DORLING; MCCAFFERY, 2012). Exemplos de elementos podem ser: pontos, utilizado para situar o usuário que uma tarefa foi concluída, e encoraja-lo a realizar as tarefas; emblemas, fornecem a capacidade de criar recompensas sociais demonstráveis para atividades específicas; níveis, uma solução para criar uma sensação constante de avanço e; placares de liderança, uma maneira universal de transmitir o sucesso.

Outra técnica que utiliza jogos para engajar o usuário a atividades especificas são os jogos sérios. A diferença entre a gamificação é que jogos sérios são totalmente jogos e, em gamificação, somente algumas partes remetem a jogos (DETERDING et al., 2011a).

Muito mais do que apenas adicionar elementos, para criar um ambiente gamificado que aumente a motivação dos alunos é necessário focar nos conceitos fundamentais que tornam os jogos atraentes para os seus jogadores. Os jogadores são motivados por conta do impacto nas áreas emocionais, sociais e cognitivas. Assim, a gamificação na educação também deve se concentrar nessas três áreas (DOMíNGUEZ et al., 2013). O impacto na área emocional funciona principalmente em torno do conceito de sucesso e fracasso, como arrecadação de recompensas por uma tarefa concluída. Quando vários jogadores interagem através do jogo, essas interações têm impacto na área social dos jogadores, por exemplo, em uma competição. Por fim, na área cognitiva, o jogo pode fornecer um sistema complexo de regras junto a uma série tarefas que guiam o jogador a um processo em que ele domine estas regras.

2.1.4. Limitações da gamificação

Um problema aflige as atuais implementações de gamificação. Tais propostas reduzem a complexidade de um jogo, ou de uma aplicação apropriadamente gamificada e equilibrada, a apenas adição de componentes, tais como simples pontos e níveis (DETERDING, 2012). Estas aplicações não apenas deixam de envolver os jogadores, mas também podem danificar o interesse existente ou engajamento de um usuário com o serviço ou produto.

Outro levantamento, utilizando um *framework* para avaliar jogos sérios, mostra que a gamificação atual é predominantemente simplória, centrada quase exclusivamente na mecânica baseada em recompensas (BARIK et al., 2016). Este *framework* contém seis conceitos destinados

a serem utilizados de forma coesa para proporcionar uma boa experiência de jogo: propósito, conteúdo, mecânicas, narrativa, estética e suporte para evolução no ambiente.

A forma de como a gamificação é aplicada também influencia na sua efetividade e aprovação dos usuários. Por exemplo, um sistema de gamificação foi aplicado em sala de aula a um curso de engenharia de software na Universidade de Baden Württemberg, na Alemanha (BERKLING; THOMAS, 2013). Em geral, os estudantes não receberam a ideia de gamificação, como foi empregada, de uma forma positiva. Os resultados da pesquisa sugerem que os benefícios de um ambiente gamificado para a sala de aula não foram evidentes para os alunos e indicam que eles tiveram uma baixa expectativa ao usar ao termo gamificação no contexto da sala de aula. O termo gamificação gerou uma conotação de jogo como um desperdício de tempo em um ambiente focado em resultados, no caso de uma Universidade.

2.2. Gamificação em educação de Computação

Disciplinas de introdução a programação são consideradas difíceis e desafiadoras por alunos ingressantes. O *feedback* e a motivação para estudar são fatores importantes para que os alunos se envolvam com as disciplinas (IZEKI et al., 2016). Este é um problema a ser resolvido, já que disciplinas de programação são essenciais na formação de discentes de computação. Ambientes de aprendizagem gamificados podem funcionar como uma estratégia no engajamento de estudantes no aprendizado de programação (SILVA et al., 2016).

2.2.1. Pex4Fun e Code Hunt

Pex4Fun (TILLMANN et al., 2011) é uma plataforma *web* de código fechado para ensino de programação em C# e F#. Diferente de outras plataformas de ensino *online*, o professor não cria a especificação do exercício e um conjunto de caso de testes para avaliar a solução dos alunos, criando somente a especificação na forma de um programa, e os alunos implementam a solução iterativamente para ter o mesmo comportamento do programa do professor, analisando casos de teste gerados a partir do código do professor, considerando uma técnica baseada em execução simbólica dinâmica.

O principal elemento de gamificação do Pex4Fun são os desafios, também chamados de duelos (TILLMANN et al., 2014b). A Figura 2.2 demonstra a plataforma Pex4Fun e um exemplo de duelo para o aluno. Um duelo consiste no professor implementar um código que deve ser o objetivo dos alunos, contemplando os conceitos e tópicos que devem ser ensinados. Este código é secreto, ou seja, não é visível para os alunos. Em seguida cria o código base, contendo as mesmas assinaturas dos métodos e os mesmos parâmetros do código secreto. Este código pode estar vazio, parcialmente preenchido corretamente ou incorretamente. Os alunos então utilizam este código base para implementar um comportamento semelhante ao do código secreto, analisando os resultados da ferramenta de geração de testes automáticos baseado em execução simbólica

dinâmica conforme implementado pela técnica Pex, indicando se o aluno está no caminho correto ou não de sua implementação.

Na plataforma Pex4Fun, ao aluno submeter a solução, o Pex gera alguns valores de entrada e executa o código secreto e o código do aluno, exibindo em seguida quais os valores de entrada e os valores de saída para os dois códigos. Com esta informação, o aluno observa quais alterações deve fazer para obter os mesmos valores de saída do código secreto. Desta forma, iterativamente ele escreve o programa para ter o mesmo comportamento que o programa do professor utilizando as informações do Pex.

A plataforma também conta com um sistema de pontuação para motivar os alunos a continuar interagindo com a plataforma, no qual pode-se ganha-los resolvendo duelos, inscrevendo-se em cursos ou a cada outro aluno que ganha um duelo criado por você.

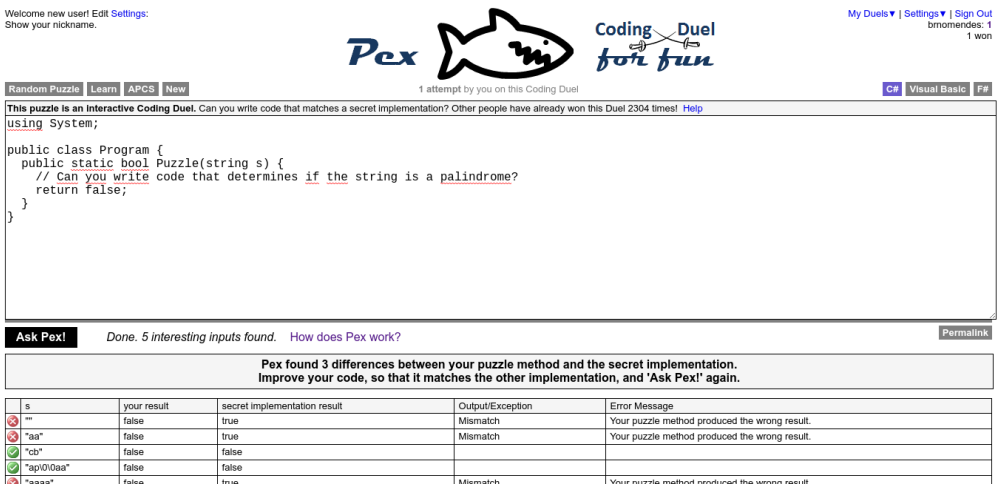


Figura 2.2. Exemplo de duelo no Pex4Fun
Fonte: Página do Pex4Fun - pexforfun.com.

O Pex4Fun contém um ambiente de cursos em que qualquer professor pode criar um curso e compartilhar com seus alunos. Aos cursos podem agregar duelos para que os alunos exercitem o conteúdo proposto. O aluno passa no curso se conseguir completar os duelos estabelecidos pelo professor.

Implicitamente, a plataforma ensina aos alunos como criar e usar casos de testes (BISHOP et al., 2015), em que o aluno deve implementar uma solução pensando na execução do conjunto de casos de testes, o comportamento e os valores resultantes. Outra prova disso é que após a submissão da solução, aparece um novo botão com "*How does Pex work?*" (Como o Pex trabalha?), que leva a uma documentação do Pex4Fun detalhando como a ferramenta Pex cria e executa os testes.

A partir do Pex4Fun originou-se o Code Hunt (TILLMANN et al., 2014), no qual ampliou-se a capacidade como plataforma de jogo. O Code Hunt é um jogo *web* sério para ensino de programação, apresentado na Figura 2.3. No Code Hunt, os exercícios (duelos) são divididos em setores (TILLMANN et al., 2014a). Cada setor tem um tema, por exemplo, aritmética, laços de

repetição ou condicionais. Em cada setor, há diferentes níveis, que aumentam a dificuldade de forma crescente. Os níveis são desafios semelhantes aos duelos do Pex4Fun, onde há um código base e o aluno deve implementar iterativamente analisando as informações do Pex.

Figura 2.3. Ambiente Code Hunt
Fonte: Página do Code Hunt - codehunt.com.

Há outras diferenças do Code Hunt em relação ao Pex4Fun além dos setores e níveis. O Code Hunt não possui um ambiente para cursos e os duelos não podem ser adicionados por outros alunos ou professores. Outra diferença é o suporte a linguagens para implementar os duelos: no Code Hunt é possível utilizar C# e Java enquanto no Pex4Fun temos C# e F#.

O ambiente do Code Hunt é mais lúdico, pois além de possuir um *design* baseado em jogos, possui animações, sons e analogia a jogos de caça, em que ao resolver um duelo o aluno "captura o código", conforme apresentado na Figura 2.4. A plataforma fornece um tutorial inicial e, além dos pontos, possui uma tabela global de liderança, no qual os usuários com uma maior pontuação são exibidos em ordem decrescente pela quantidade de pontos. A pontuação é obtida concluindo-se níveis, podendo ganhar 10, 20 ou 30 pontos por nível, dependendo da qualidade do código. O objetivo de todos estes elementos é motivar o aluno a concluir os duelos para melhorar suas habilidades em programação e engenharia de software. Os duelos são desafiantes, isto motiva os alunos. Em jogos, é comum os jogadores serem motivados a pontuarem, serem melhores e completarem o jogo. O sentimento de completar objetivos e de ganhar é familiar a todos. Logo ganhar é divertido, e tornar o ambiente divertido facilita o aprendizado do aluno.

O fato do ambiente ser gamificado e motivar os usuários a resolverem os desafios faz com que a plataforma seja utilizada para outros fins como, por exemplo, para *crowdsourcing* (XIE et al., 2015a). Pode-se projetar um duelo o qual o criador disponibiliza na plataforma Code Hunt e, quando os outros usuários resolverem o duelo, surgirão implementações alternativas que compartilham os mesmos comportamentos funcionais que o código secreto. Depois que os usuários contribuíram resolvendo o duelo, o criador pode aproveitar ferramentas de automação para analisar essas soluções e selecionar as mais desejáveis para usar de acordo com algumas

Figura 2.4. Exemplo de tela ao concluir um desafio no Code Hunt
Fonte: Página do Code Hunt - codehunt.com.

métricas. Por exemplo, seria possível selecionar implementações alternativas que sejam mais eficiência ou que possuam uma melhor qualidade de código.

O Code Hunt pode ser empregado para ensinar outras habilidades além de programação geral, testes e engenharia de software. Por exemplo, pode-se treinar habilidades de programação voltada a segurança (XIE et al., 2015b). Nesse caso, os setores, níveis e duelos são construídos com base em conceitos que devem ser aprendidos pelos alunos, como tópicos de validação de entrada e controle de acesso.

Entre março de 2014 e fevereiro de 2015, mais de 65.000 usuários começaram a jogar no Code Hunt, e 188 completaram os 130 desafios (BISHOP et al., 2015). Os resultados obtidos da análise neste período mostram que há uma taxa constante de 15% de desistência, e esta taxa diminui conforme os jogadores se tornam mais especialistas e querem terminar todos os desafios. Estes resultados mostram que a dificuldade dos duelos influencia na motivação dos usuários a continuarem utilizando a plataforma e concluir todos os duelos. Uma questão importante é que o número de desistentes é sempre maior quando um novo conceito é introduzido, ou seja, entre os setores do jogo.

Entretanto a plataforma possui uma deficiência: não contar com materiais didáticos e de apoio (FOUCHé; MANGLE, 2015). A implementação existente funciona bem quando o aluno tem acompanhamento tradicional como em sala de aula, mas não fornece suporte suficiente para outros tipos de usuários considerando a disponibilidade do Code Hunt, em que só é necessário conexão com *Internet*. Deve-se notar que o predecessor do Code Hunt, o Pex4Fun, permite a inclusão de textos didáticos.

2.2.2. Code Defenders

Outra plataforma que implementa gamificação junto a programação é o Code Defenders. Code Defenders é uma plataforma *web* de código fechado para ensino de testes de mutantes em Java

(ROJAS; FRASER, 2016b). A principal abordagem de jogabilidade é a utilização de uma unidade em teste (uma classe Java) como o elemento central e os jogadores assumem os papéis de atacantes e defensores.

O Code Defenders utiliza a gamificação para resolver o seguinte problema (ROJAS; FRASER, 2016c): teste de software é crucial em um mundo dominado por software, mas difícil de ser aprendido e requer experiência teórica e prática. No entanto, as técnicas de teste são muitas vezes percebidas como chatas pelos desenvolvedores e, portanto, difíceis de serem aprendidas na educação de programação.

Teste de mutantes é uma técnica de engenharia de software que visa orientar o desenvolvimento do código de teste e avaliar sua qualidade (ROJAS; FRASER, 2016c). Dado um programa em teste, o teste de mutante envolve a criação de um conjunto de variações do programa, cada um chamado de mutante, em que diferem do programa original em pequenas mudanças, por exemplo, um sinal de maior para maior igual. A premissa é que os desenvolvedores tendem a escrever programas quase corretos, mas que podem pecar em pequenos erros. Esta premissa é conhecida como hipótese do programador competente. O conjunto de programas mutantes podem ser usados para verificar a qualidade dos testes existentes, medindo quantos dos mutantes são detectados. Um teste mata um mutante quando o resultado da sua execução no programa original é diferente do resultado de sua execução no mutante. Mutantes que não são mortos podem ser utilizados para guiar a criação de mais testes. É possível que um mutante tenha a execução equivalente a do programa original, caso em que não existe nenhum teste que possa distingui-lo do programa original. Detectar mutantes equivalentes é um problema difícil e normalmente requer uma profunda análise dos programas por parte do engenheiro de software.

O conceito de testes de mutante pode ser confuso para desenvolvedores a primeira vista. Isto pode ser influenciado por testes de mutantes não serem componentes bem estabelecidos no ensino de programação e que, por sua vez, pode ser causado pela falta de ferramentas de suporte educacional para testes de mutantes.

Além dos conceitos de teste de mutantes, a plataforma do Code Defenders fornece aprendizado a outros conteúdos, como, por exemplo cobertura de testes, em que o aluno deve analisar o programa, e criar testes que eliminem todos os mutantes, ou seja, cobrir o máximo possível do código (CLEGG et al., 2017).

No Code Defenders, o jogo é jogado por dois jogadores, no qual jogam utilizando a linguagem Java e o *framework* de testes JUnit (ROJAS; FRASER, 2016c). O principal elemento do jogo é o ataque e defesa. Um jogador assume o papel de atacante e o outro o papel de defensor. Inicia-se com um programa no início do jogo (uma classe Java) que pode ser escolhido da plataforma ou enviado pelos jogadores. O atacante deve criar mutantes sutis desta classe, pensando em que estes mutantes devem ser difíceis de serem mortos, conforme mostrado na Figura 2.5. O defensor cria casos de teste com JUnit que mata os mutantes criados pelo atacante, demonstrado na Figura 2.6. O jogo é jogado em turnos de ataque e defesa, sendo que o primeiro turno é do atacante. Uma vez

que o atacante submeteu um mutante, o turno é passado para o defensor, que então tenta criar um novo teste para detectar os mutantes. Uma rodada é completada quando o atacante submeteu um novo mutante e o defensor defendeu com um novo teste. O jogo prossegue até que o número de rodadas escolhidas no início do jogo tenha sido jogadas. Após a conclusão de cada rodada, uma avaliação de mutação é realizada para determinar se o testes mataram os mutantes vivos. Um cenário especial surge quando o defensor suspeita que um mutante é equivalente. Nesse caso, em vez de fornecer um novo teste, o defensor pode requisitar um duelo de equivalência, desafiando o atacante a aceitar que o mutante é equivalente, ou a produzir um teste para provar que não são equivalentes. Ambos os jogadores são constantemente informados sobre o estado de todos os mutantes no jogo, se eles estão vivos, foram mortos, ou foram identificados como equivalentes.

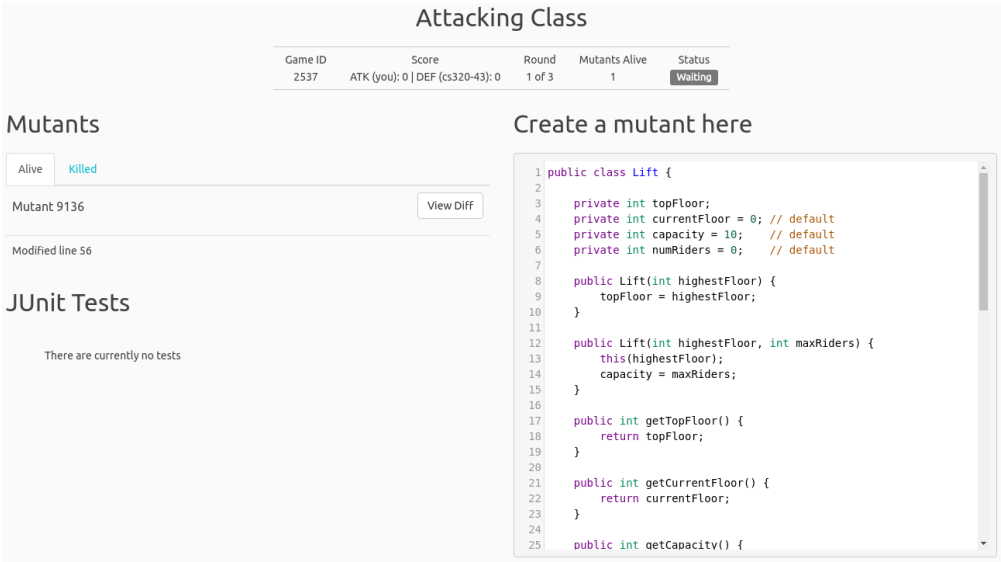


Figura 2.5. Visão do atacante em Code Defenders
Fonte: Página do Code Defenders - code-defenders.org.

O Code Defenders baseia-se na competitividade como um incentivo para envolver os jogadores (CLEGG et al., 2017). Para este efeito, o jogo implementa um sistema de pontuação que visa proporcionar incentivo para ambos jogadores, atacantes e defensores, e para decidir o vencedor de um jogo. O sistema de pontos baseia-se nas seguintes regras:

- Quando um mutante é criado, se este passar pelos casos de testes existentes, recebe um ponto. Se o mutante é morto por um teste existente logo quando é submetido, então ele não recebe pontos.
- Se um mutante é criado e não é morto por qualquer teste existente, além de receber um ponto, pode receber pontos extras obtidos a partir de testes que cobrem a mutação, mas falham. Uma vez que um mutante é morto, sua pontuação não é mais aumentada.
- Para cada mutante que um teste mata, o teste ganha pontos iguais à pontuação do mutante mais um. Isso se aplica a mutantes que já existiam no momento em que o teste foi criado, bem como mutantes adicionados ao jogo mais tarde. Quando um mutante é submetido,

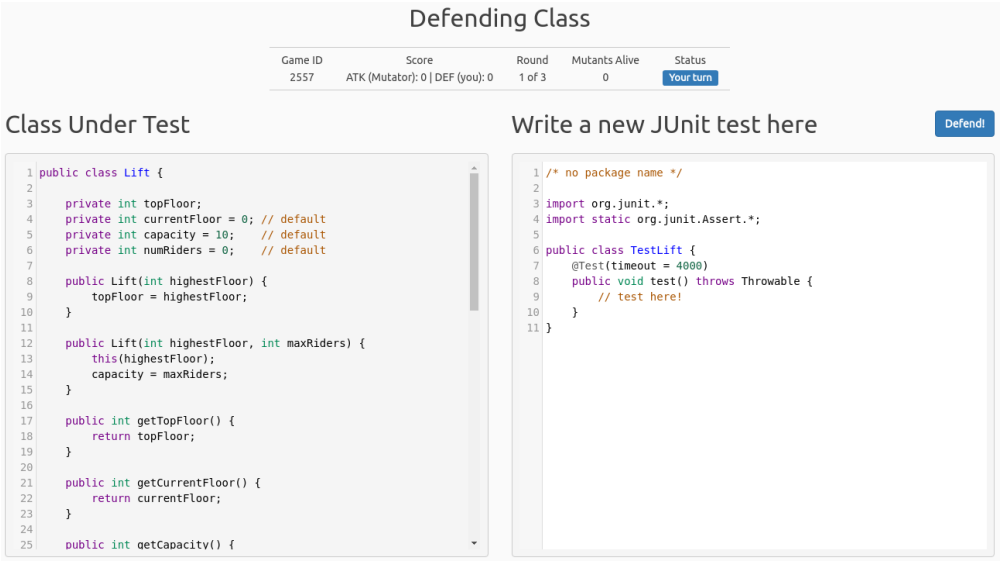


Figura 2.6. Visão do defensor em Code Defenders
Fonte: Página do Code Defenders - code-defenders.org.

os testes são executados na ordem em que foram criados. Assim, o teste mais antigo que matar um mutante receberá os pontos, e nenhum outro teste receberá pontos para o mesmo mutante.

Em suma, os atacantes ganham se eles produzem mutantes que sobrevivem a testes de unidade dos defensores por mais tempo. Alternativamente, os defensores ganham se produzirem testes de unidade que matam a maioria dos mutantes logo após serem submetidos.

No Code Defenders, existem dois níveis de dificuldade, fácil e difícil (ROJAS; FRASER, 2016a). A diferença entre os dois é que no modo fácil, o código de todos os mutantes é revelado ao defensor na forma de um relatório de diferenças, enquanto que no modo difícil apenas uma breve descrição do mutante é apresentada ao defensor, por exemplo, houve uma alteração na linha 18. Intuitivamente, o modo fácil permite uma estratégia de teste mais reativa, onde o defensor escreve testes direcionados diretamente para matar mutantes de acordo com as alterações que diferem do programa original. Já o modo difícil promove uma estratégia de testes mais proativa, onde os defensores tentariam se defender contra todos os ataques possíveis correspondentes à descrição mutante.

Há formas diferentes de se jogar no Code Defenders. Uma delas é de forma assíncrona, no qual os jogadores não necessitam estar ao mesmo tempo na plataforma. Outro modo de jogo é o modo sozinho (CLEGG et al., 2017), em que continua o conceito de ataque e defesa, mas há a troca do adversário por um jogador automatizado. Quando o jogador escolhe o papel de um atacante, os testes são gerados por uma ferramenta automatizada, o EvoSuite. Quando o jogador escolhe o papel de um defensor, então os mutantes são gerados por uma ferramenta automatizada de análise de mutação, o Major.

Assim como mencionado sobre o Code Hunt, a motivação exercida aos usuários na plataforma do Code Defenders pode fazer com que ela seja utilizado para *crowdsourcing* (ROJAS

et al., 2017). Experimentos em cenários controlados e de *crowdsourcing* revelam que escrever testes como parte de um jogo é mais agradável e que, ao jogar na plataforma Code Defenders, resulta em um conjunto de testes e mutantes melhores do que quando produzidos por ferramentas automatizadas. A cobertura de código e a detecção de mutantes são mais elevadas em comparação com testes escritos fora do jogo e gerados por ferramentas automatizadas: em média, 28% mais alto que a ferramenta Randoop e 25% mais elevado que o EvoSuite. Mutantes criados por atacantes são significativamente mais difíceis de serem mortos do que aqueles criados pela ferramenta de mutação Major. Em uma avaliação com 20 classes Java de código aberto, mostra que o Code Defenders alcançou uma maior cobertura e detecção de mutantes do que as ferramentas de geração de teste de última geração (ROJAS et al., 2017).

Há alguns problemas na plataforma Code Defenders como, por exemplo, o suporte a apenas uma classe Java por jogo (ROJAS; FRASER, 2016a). Logo, testar programas mais complexos não é possível. Considerando que programas no mundo real não são compostos de apenas uma única classe, a gamificação no Code Defenders permite somente o aprendizado em unidades particulares com testes de mutantes. Outra questão, é o requisito que os jogadores devem conhecer previamente a linguagem Java e o *framework* JUnit. A plataforma não oferece suporte a qualquer estudante que queira aprender e aperfeiçoar testes de mutantes, mas sim a aqueles que possuam conhecimento destas tecnologias, pois não oferece material didático ou de suporte para Java e JUnit.

2.2.3. Modelo gamificado de avaliação em pares

Tenório et al. (2016) descreve um modelo gamificado de avaliação em pares para ambientes de aprendizagem *online* em um contexto competitivo.

Avaliação em pares oferece uma solução poderosa para ajudar professores em ambientes de aprendizagem *online* a fazer a correção de atividades distribuidamente entre os estudantes (TENÓRIO et al., 2016). A qualidade da avaliação em pares depende de boas avaliações dos avaliadores. Assim há aspectos desvantajosos para escalar o uso de avaliação em pares, como a presença de comportamentos inadequados, por exemplo, ser muito rígido ou muito suave na avaliação, ou mesmo não oferecer um *feedback* útil. Isso geralmente ocorre devido à falta de motivação e engajamento dos alunos no processo avaliação em pares. Para lidar com este problema, o trabalho propõe um modelo de avaliação de pares gamificado, onde os elementos de gamificação são utilizados para engajar os alunos nas atividades de avaliação em pares.

Inicialmente é proposto um modelo para avaliação em pares em ambientes de aprendizagem *online* e depois são agregados elementos de gamificação. O modelo é apresentado na Figura 2.7, sendo ele dividido em 6 etapas.

Na primeira etapa, o professor cria as atividades que deverão ser resolvidas pelos estudantes. Junto a cada atividade, o professor deve criar um formulário de avaliação para a

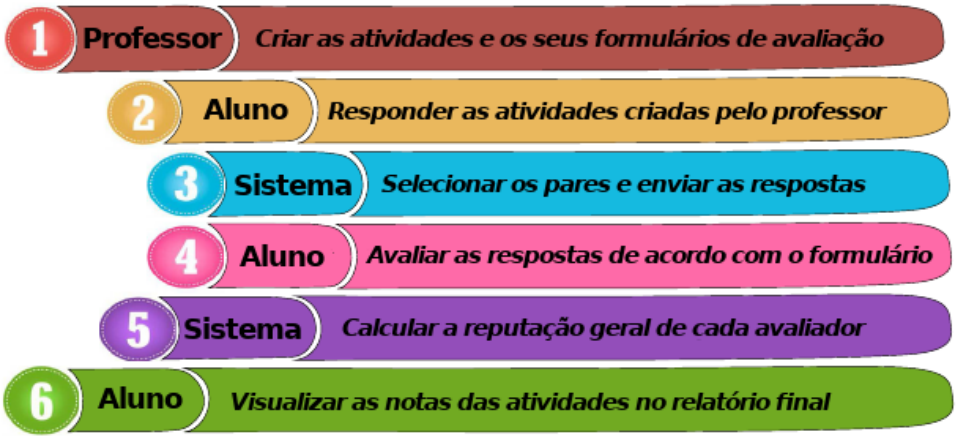


Figura 2.7. Modelo de avaliação em pares
Fonte: Autoria própria. Adaptada de Tenório et al. (2016).

atividade. No formulário de avaliação o professor pode empregar campos de verdadeiro ou falso, seleção de uma única opção, intervalo numerado ou texto aberto.

A segunda etapa consiste na resolução das atividades pelos alunos. Tais resoluções serão utilizadas na terceira etapa, no qual o sistema seleciona um outro estudante, e o envia uma resolução para que ele avalie utilizando o formulário da atividade na etapa quatro.

Para ter uma melhor confiança na avaliação e consequentemente, na nota final na etapa cinco, o sistema avalia a reputação e o nível de competência de cada avaliador. Alguns aspectos para o cálculo da reputação são: acesso frequente ao sistema, número de atividades realizadas e número de correções realizadas.

Na sexta e última etapa, é apresentado o relatório contendo os resultados da avaliação ao aluno que resolveu uma atividade. Este relatório deve ser detalhado com os critérios, bem como o resultado final. É possível a inclusão de comentários e possíveis mudanças nas notas por conta do professor antes da apresentação ao aluno.

Neste modelo de avaliação em pares, são incluídos elementos de gamificação. O principal objetivo da gamificação é garantir que os alunos se sintam motivados a participar em atividades resolvendo e corrigindo-as. Os elementos utilizados são recompensas, como medalha e troféus, e uma tabela de liderança dos que realizaram mais atividades. Tais elementos trazem um aspecto motivacional para aqueles estudantes que se desmotivam, inclusive aqueles que, por algum motivo, desistiram de realizar as atividades.

Algumas regras utilizadas junto a implementação dos elementos de gamificação são: receber pontos ao resolver três atividades, corrigir três atividades e login por três dias consecutivos. Já as medalhas podem ser conquistadas, por exemplo, corrigindo cinco atividades ou estando entre os dez melhores na tabela de liderança.

O modelo gamificado de avaliação em pares foi aplicado em um ambiente educacional, o MeuTutor. A versão escolhida para os experimentos foi o MeuTutor-ENEM, que visa ajudar estudantes do ensino médio a se prepararem para o Exame Nacional do Ensino Médio (ENEM).

Assim, o ambiente oferece cursos relacionados a disciplinas do ensino médio, como português, matemática, física, entre outros. Foram realizados dois experimentos utilizando o MeuTutor-ENEM junto ao modelo gamificado. Os resultados se mostram satisfatórios, uma vez que foi comprovado estatisticamente que o modelo é equivalente ao modelo tradicional de acordo com os resultados das soluções (correção por especialistas), tendo vantagens em relação a tempo, pois é necessário um menor tempo. Além disso, foi demonstrado nos experimentos que a gamificação influenciou positivamente em um contexto geral o modelo de avaliação em pares. Os resultados indicam que a gamificação incentivou os alunos a usarem a plataforma. De acordo com os resultados, houve aproximadamente 10,53% mais resoluções realizadas e cerca de 20% mais correções.

Como os experimentos realizados são relacionados a disciplinas do ensino médio, não é possível afirmar que sua eficácia continue em disciplinas de programação. Outro ponto é que o modelo foi implementado no ambiente MeuTutor-ENEM, o que pode ter afetado a eficiência do modelo e dos elementos de gamificação. Neste trabalho não é considerado o nível de aprendizagem dos alunos nos resultados do experimento, sendo este um aspecto importante a ser avaliado, mas uma limitação evidente neste modelo é a necessidade que o professor tem de criar um formulário de avaliação para cada atividade proposta.

2.3. Considerações finais

O objetivo deste capítulo foi definir os conceitos de MOOCs e os aspectos importantes sobre gamificação. Como Barik et al. (2016) apresentam, a gamificação atual é predominantemente simplória, centrada quase exclusivamente na mecânica baseada em recompensa. Os trabalhos relacionados apresentados fogem deste paradigma, utilizando não somente recompensas, mas vários mecanismos de *design* de jogos.

Nas pesquisas apresentadas nos trabalhos relacionados, mostra-se que os mecanismos de gamificação podem ser utilizados como ferramentas para engajar alunos na educação. O Code Hunt, Pex4Fun e a plataforma Code Defenders utilizam mecanismos competitivos junto a programação. No modelo gamificado de avaliação em pares, é possível notar que, de alguma forma, os próprios alunos podem definir suas notas e realizar as avaliações, aspecto que pode ser utilizado na implementação desta monografia.

Como discutido, o Code Hunt tem uma taxa de desistência maior quando novos assuntos de programação são apresentados aos usuários, mostrando a importância de um material didático ou alguma forma de tutoria durante as resoluções das atividades. Como na proposta deste trabalho, pretende-se aplicar a gamificação em plataformas de MOOCs, em que se oferecem conteúdos necessários para que os alunos os aprendam e depois os exercitem junto às atividades gamificadas. A mesma falta de conteúdo ou de tutoria acontece na plataforma Code Defenders, em que não há informações sobre o funcionamento do *framework* JUnit ou os conceitos básicos sobre testes de mutantes.

Como o Pex4Fun e o Code Hunt utilizam a ferramenta Pex para gerar as informações aos alunos, tais plataformas mantêm uma limitação das linguagens que podem ser utilizadas no ambiente, sendo as linguagens C#, Visual Basic e F# suportadas pelo Pex. No caso do Code Hunt, em oferecer suporte a Java, teve que haver um esforço significativo para tal, no qual todo código Java submetido é convertido em código C#, conversão que não suporta, por exemplo, classes e métodos genéricos e o tipo *enum* (TILLMANN et al., 2014).

Um problema na plataforma Code Defenders, no qual tem o objetivo de ensinar testes de mutantes utilizando a linguagem Java e o *framework* JUnit, é o suporte a apenas uma classe Java por jogo. Logo, realizar testes de mutantes, por exemplo, em uma classe que utiliza outras classes em suas instruções, não é possível. Como comumente as implementações utilizando Java tem um foco no paradigma orientado a objetos, no qual se tem diversas classes, esta limitação no Code Defenders é relativamente significativa.

Outro ponto é a participação do professor para que os mecanismos de avaliação funcionem. No caso do Code Hunt e Pex4Fun, é necessário que o professor implemente o programa gabarito. Já no modelo de avaliação em pares, há a necessidade que o professor crie um formulário de avaliação para cada atividade proposta.

Desta forma, considerando os resultados positivo dos trabalhos relacionados, neste trabalho empregos um mecanismo de competição como gamificação, conciliado introdução a programação com testes de software. Ademais, disponibilizamos curso MOOC com conteúdo para auxiliar na resolução das atividades, tratando uma limitação nos trabalhos relacionados.

Capítulo 3

MOOC Gamificado

Após a revisão da literatura, foi possível verificar que a gamificação pode ser utilizada na educação em esferas diferentes. A gamificação pode ser subsídio para motivar os alunos na aprendizagem, seja motivando-os a cursar os cursos ou realizar diversas tarefas. Não exclusivamente, mas a gamificação pode ser implementada em disciplinas introdutórias de programação que, como Izeki et al. (2016) apresentam, são disciplinas desafiadoras para alunos ingressantes.

Com o objetivo de utilizar a gamificação em uma disciplina introdutória de programação, necessita-se de uma plataforma de MOOCs que permita implementação de funcionalidades extras, e preferencialmente de código aberto. Este capítulo trata do desenvolvimento de um MOOC gamificado, tratando a escolha da plataforma, definição do mecanismo de gamificação a serem empregados e implementação desses na plataforma escolhida.

O primeiro passo para o método foi a escolha de uma plataforma de MOOCs. Assim que definida a plataforma de MOOCs, é necessário propor e escolher elementos de *design* de jogos que poderão ser implementados junto à plataforma. Alguns dos possíveis elementos para a gamificação são apresentados a seguir.

Alguns jogos engajam seus usuários através de competições. O principal elemento da gamificação proposta utiliza-se deste aspecto. As submissões dos alunos para as atividades avaliativas do professor, em uma disciplina de programação, podem ser programas, escritos em alguma linguagem, e casos de testes. Utilizando estes dois recursos, a gamificação proposta realiza uma competição entre os alunos. Para cada atividade, o aluno submete o seu programa e o caso de teste. Seu programa então é testado pelos casos de testes de todos os outros alunos, assim recebendo um *feedback* sobre qualidade de sua solução. Ainda o aluno pode refazer e ressubmeter sua solução para atividade caso não esteja satisfeito com os resultados. De certa forma, o aperfeiçoamento dos programas enviados e a criação de novos casos de teste são uma forma dos alunos expressarem opiniões e questionar os programas dos colegas.

Para agregar ao elemento de competição, mais elementos de *design* de jogos são necessários, como pontuações e uma tabela de liderança (no inglês *ranking*). A pontuação pode ser obtida seguindo algumas regras, por exemplo: programa ser aprovado em todos os casos de testes dos outros alunos; caso de teste submetido encontrar falhas em N programas de outros alunos, mas também, pode haver uma perca de pontos seguindo alguns critérios, por exemplo: programa não ser aceito por N outros casos de testes; programa não ser aceito pelos próprios casos de testes; ressubmeter a solução para a mesma atividade N vezes. Utilizando o sistema de pontuações, uma tabela de liderança pode ser utilizada como parte da gamificação. Como Dorling e McCaffery (2012) apresentam, a tabela de liderança é uma maneira de mostrar a sua pontuação para os outros usuários. A tabela de liderança consiste em uma classificação dos alunos por ordem decrescente de pontos no ambiente gamificado.

O restante deste capítulo está organizado da seguinte forma. A Seção 3.1 apresenta a plataforma de MOOCs Open edX. Os mecanismos de avaliação disponíveis nesta plataforma são descritos na Seção 3.2. Na Seção 3.3 é descrito, a implementação de mecanismos de avaliação com elementos de *design* de jogos. Por fim, a Seção 3.5 apresenta as considerações finais referentes a este capítulo.

3.1. Open edX

Para esta implementação, é necessário uma plataforma de MOOCs no qual seja possível a extensão de funcionalidades. Plataformas como Coursera e Udemy mantêm seu código fechado, o que impossibilita utilizá-las para a implementação. Uma plataforma de MOOCs de código aberto é o Open edX. O Open edX foi criado pela edX, uma iniciativa *online* sem fins lucrativos composto por parceiros da Universidade Harvard e o Instituto de Tecnologia de Massachusetts.

O Open edX é considerado a principal aplicação de código aberto para suportar MOOCs (DíAZ et al., 2016). Em 2013, a edX abriu o código de sua plataforma para o público. Isto permitiu a desenvolvedores de todo mundo a criarem melhorias e extensões para a plataforma. Várias organizações, como universidades, utilizam o Open edX para lançar seus próprios MOOCs (GORDóN; MORA, 2015).

Mais de 500 cursos MOOCs foram ofertados na plataforma edX, totalizando mais de 5 milhões de estudantes. O Open edX é utilizado por mais de 150 instituições externas. Construído utilizando a linguagem Python e o *framework web* Django, o Open edX é dividido em dois projetos: o sistema de gerenciamento de cursos, no qual os professores criam e gerenciam os cursos e seus conteúdos; e o sistema de gerenciamento de aprendizagem, a parte onde os alunos acessam para cursar os cursos. Os bancos de dados MySQL e MongoDB são utilizados para a persistência de dados (RUIPéREZ-VALIENTE et al., 2017).

A plataforma Open edX permite a criação e gerenciamento de cursos MOOCs. Cada curso é composto de diversos componentes, organizado em uma estrutura no qual podem ser modificados.

Os cursos possuem capítulos, tais capítulos são subdivididos em seções e, em cada seção, diversas subseções, no qual se encontra o conteúdo de fato. Por exemplo, o curso Programação em Python pode possuir o capítulo Operações Aritméticas. Este capítulo é dividido nas seções Soma, Subtração, Multiplicação e Divisão. A seção Soma é composta por subseções, como um texto explicativo, um vídeo didático, uma atividade de múltipla escolha e um exame avaliativo.

3.2. Mecanismos de avaliação no Open edX

O principal recurso para avaliação no Open edX é o suporte a questões de múltipla escolha. O professor define atividades com diversas respostas e define qual é a opção correta, utilizando o sistema de gerenciamento de cursos. O aluno acessa as atividades por meio do sistema de gerenciamento de aprendizagem, soluciona as atividades de múltipla escolha e, de acordo com o gabarito pré-estabelecido pelo professor, ao submeter a resposta, pode visualizar os acertos, erros e a nota final.

O Open edX permite outros tipos de atividades além de múltipla escolha, mas o mecanismo é o mesmo: o professor previamente estabelece o gabarito conforme os tipos de entrada na atividade, o aluno soluciona e recebe a nota no momento em que submete para a plataforma. Os tipos de entradas suportados para as respostas dos exercícios são: lista de opções com apenas uma escolha, lista de opções com mais de uma escolha, campo para texto e campo numérico.

Considerando que, para um mesmo problema de programação, a solução correta pode ter diferentes implementações (como utilizar diferentes nomes para variáveis ou diferentes instruções ao decorrer do código), não é viável utilizar o mecanismo citado anteriormente para uma atividade de programação, no qual os alunos podem criar soluções corretas diferentes de um programa gabarito pré-estabelecido pelo professor. Desta forma, o Open edX oferece recursos para criar o próprio avaliador de exercícios externo à plataforma.

O avaliador externo é adequado para professores e instituições que querem avaliar códigos de programação. Uma utilização possível, por exemplo, é o professor estabelecer um conjunto de casos de testes no avaliador externo e, ao aluno submeter seu programa, este é testado pelos testes do professor, assim oferecendo uma nota final de acordo com os requisitos.

O funcionamento do avaliador externo ocorre da seguinte forma. O professor estabelece um exercício no curso da plataforma Open edX. Este exercício pode possuir um campo de texto ou um campo para submeter arquivos, em que será o meio para o aluno enviar a solução. Após o aluno submeter a solução, esta vai para uma fila de espera, no qual é o mecanismo de comunicação com o avaliador externo. A fila envia para o avaliador externo as soluções e aguarda por uma resposta. O avaliador externo pode ser implementado utilizando qualquer tecnologia ou linguagem e a fila de espera oferece uma interface RESTful para a comunicação.

Além do conteúdo das soluções, o avaliador externo recebe alguns metadados, como o tempo em que foi submetido a solução e um identificador único anônimo do aluno que submeteu

a resposta. Todos estes dados são obtidos por meio de um JSON através da fila de espera. O avaliador, após suas execuções, também deve enviar uma resposta à fila de espera através de um JSON. Para cada submissão, o avaliador externo deve fornecer um campo com verdadeiro ou falso, indicando se solução está correta, outro campo informando a pontuação, e um último campo com uma mensagem em texto. Todos estes dados são mostrados ao aluno após o recebimento na fila de espera.

3.3. Implementação de mecanismo de avaliação com elementos de *design* de jogos

Após a escolha da plataforma de MOOCs, do Open edX e de um estudo preliminar sobre seus mecanismos de avaliação, principalmente o avaliador externo, foi implementado o sistema proposto nesta monografia.

A Figura 3.1 apresenta o projeto do mecanismo de avaliação, junto ao fluxo entre a resolução de atividades pelos alunos e a execução do sistema com os elementos de *design* de jogos utilizando o Open edX. A implementação, utilizando a linguagem Python, consiste em um sistema com uma interface de comunicação com a plataforma Open edX, um banco de dados, o MySQL, e um mecanismo competitivo de *design* de jogos.

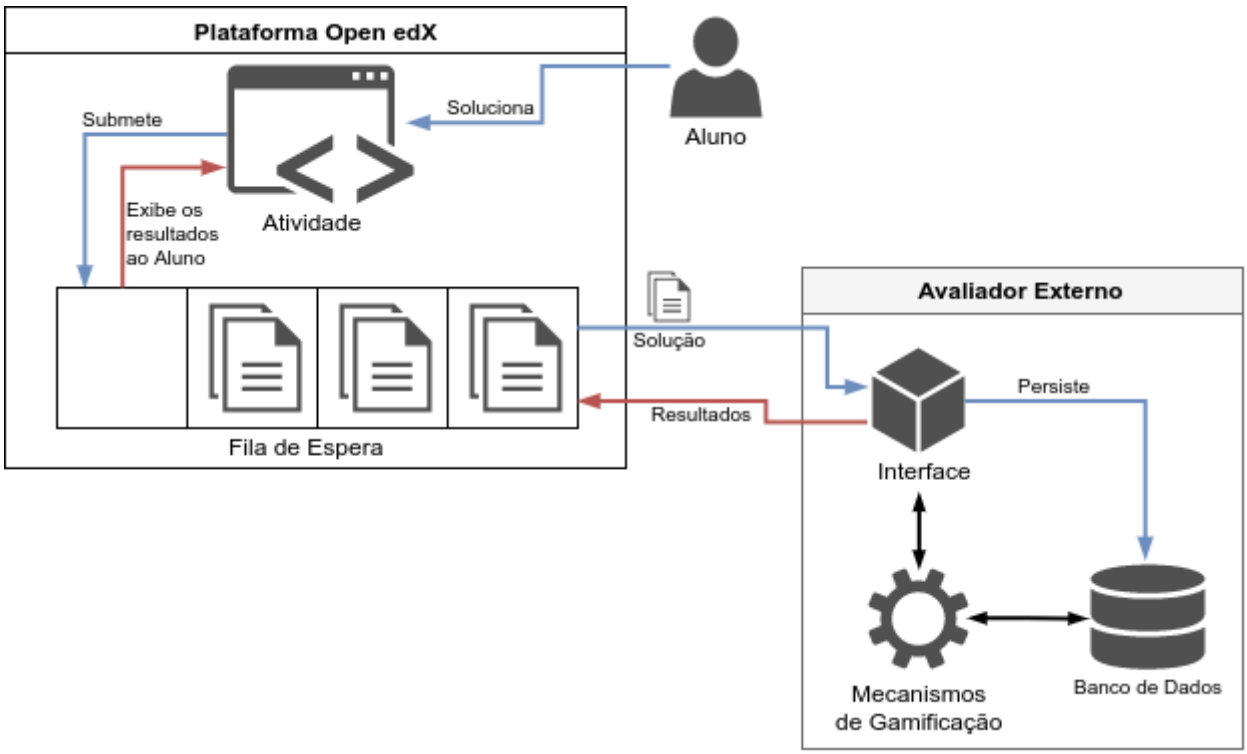
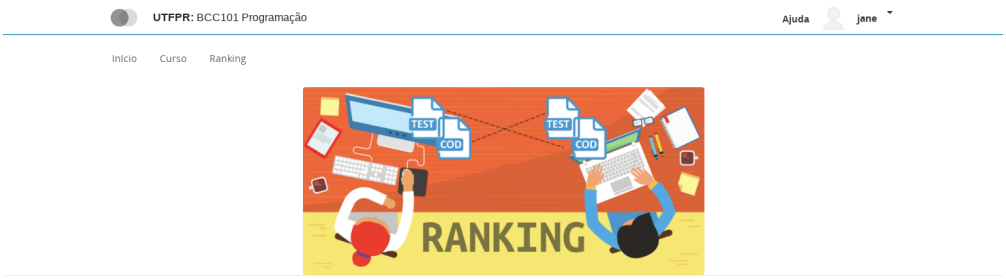


Figura 3.1. Implementação de mecanismo de avaliação com elementos de *design* de jogos
Fonte: Autoria própria.

Para a utilização, o professor deve criar uma atividade com campo para submissão de um programa e casos de testes. Tal atividade deve ser configurada para utilizar o avaliador externo implementado. O aluno pode resolver localmente a atividade e enviar sua solução na plataforma. Ao submeter, esta solução entra em uma fila de espera e é enviada para o avaliador externo, que recebe na sua interface, persiste no banco de dados o programa e os casos de testes, aciona o mecanismo de gamificação baseado em competitividade, e retorna ao Open edX um resultado que será apresentado ao aluno. O mecanismo de gamificação será apresentado a seguir.

O mecanismo de *design* de jogos implementado se baseia na competitividade entre os alunos. Os alunos devem enviar um programa e casos de testes para as atividades de um curso MOOC. O programa submetido é avaliado pelos casos de testes de todos os outros alunos submetidos para a mesma atividade. Desta forma, diversas regras podem ser estabelecidas. Tais regras serão apresentadas na Seção 3.4. Este é o principal mecanismo, mas para criar um ambiente gamificado, outros elementos de *design* de jogos foram utilizados, como os pontos, que são atribuídos aos alunos de acordo com as regras, e o *ranking*, que consiste em uma visualização global da competição entre os alunos, no qual exibe todos os alunos em ordem decrescente por pontuação. Na plataforma foi implementada uma aba com uma página de *ranking* que pode ser acessada por todos os alunos cadastrados na plataforma. Tal página inicia com um elemento visual, para que relembre aos alunos que as atividades são competições entre suas soluções. A página na plataforma que contém o *ranking* é apresentada na Figura 3.2.



Posição	Aluno	Quantidade de Submissões	Pontuação
1	[illegible]	2	223.045
2	[illegible]	3	218.622
3	[illegible]	3	137.45
4	[illegible]	3	124.167
5	[illegible]	1	124.0
6	[illegible]	3	119.881
7	[illegible]	1	99.667
8	[illegible]	3	61.833
9	[illegible]	3	60.81
10	[illegible]	4	48.62

Figura 3.2. Página do *ranking*
Fonte: Autoria própria.

Dada a visão geral do fluxo de submissão entre o avaliador externo e a plataforma, a Figura 3.3 apresenta uma visão detalhada da implementação do avaliador exibindo o diagrama de classes.

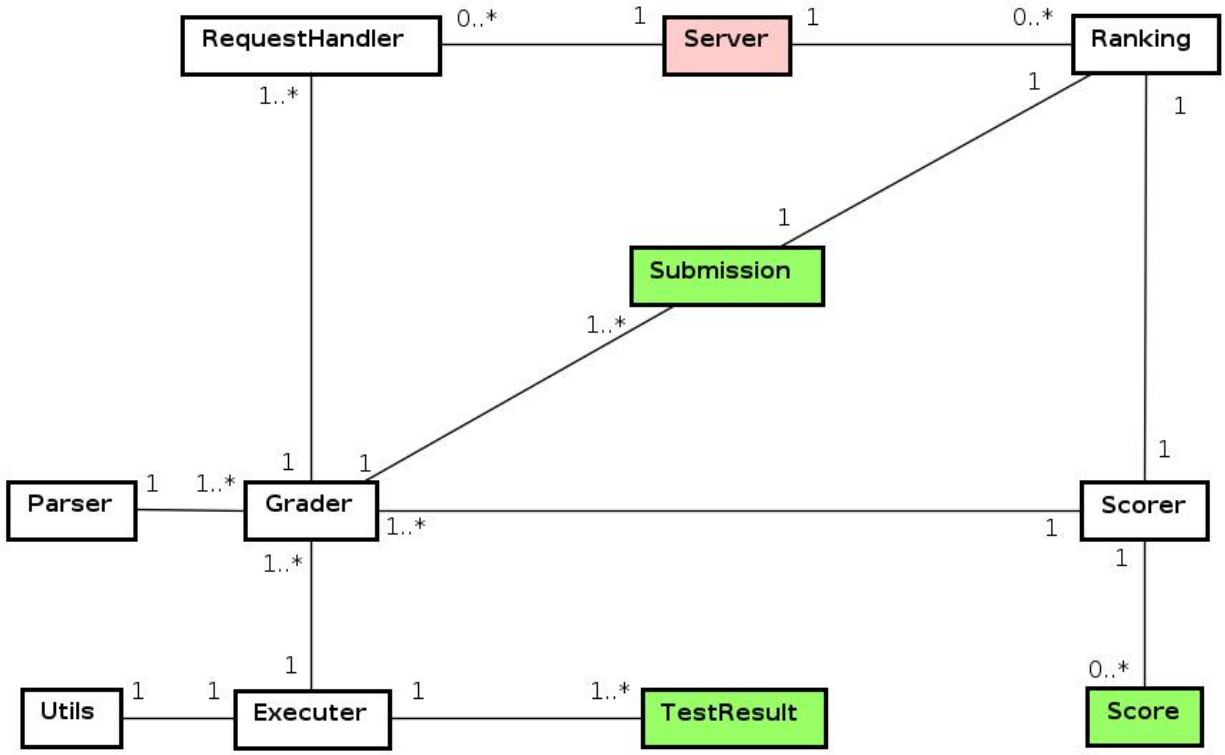


Figura 3.3. Diagrama de classes do avaliador externo
Fonte: Autoria própria.

A comunicação entre a plataforma e o avaliador inicia-se com a classe **Server**, utilizando o *microframework* Flask, em que recebe o conteúdo da submissão através do método http POST. Em seguida aciona a classe **RequestHandler** para que processe a submissão e retorne um resultado em formato JSON.

O protocolo para a submissão de uma solução é pré-estabelecido pela plataforma Open edX, mas nada impede que outros sistemas, utilizando o mesmo protocolo, realizem a submissão de soluções para o avaliador externo gamificado. A Figura 3.4 apresentada o JSON utilizado para a submissão de soluções.

Ao receber os dados em formato texto, no caso o JSON, o **RequestHandler** processa este texto puro para que se torne uma estrutura do Python e obtenha-se os metadados, como o id do estudante e o id do problema. Em seguida executa a classe **Grader**, para o processamento de fato da submissão.

A classe **Grader** executa tarefas como: utilizar o **Parser** para separar o código de testes do código do programa, utilizar o modelo **Submission** para salvar a submissão no banco de dados

```
{
    "xqueue_body":
    "{
        "student_info": { "anonymous_student_id": "id_do_estudante" },
        "student_response": "codigo_do_programa_e_casos_de_testes",
        "grader_payload": "id_do_problema"
    }"
}
```

Figura 3.4. Protocolo de submissão para o avaliador externo
Fonte: Autoria própria.

e, o mais importante, realizar a execução cruzada entre a nova submissão recebida e as submissões que os outros alunos já enviaram para a mesma atividade.

Para cada teste de um aluno e programa de outro aluno, o **Grader** chama a classe **Executer** e, ao obter os resultados do teste, os envia para o **Scorer** realizar a devida pontuação.

O processo de salvar o código do programa e o código dos testes em arquivos, para executar os testes de unidade e em seguida montar o resultado estruturando em um modelo **TestResult**, é desempenhado pela classe **Executer**.

Por fim, ao obter a posse dos resultados, a classe **Grader** aciona a classe **Scorer** para salvar as novas pontuações, de tal forma que esta analise os resultados do teste e utilize as regras pré-estabelecidas para definir a nova pontuação. A classe **Grader** então retorna um JSON ao **Server** que responde ao cliente com o protocolo exibido na Figura 3.5.

```
{
    "correct": true,
    "score": 1,
    "msg": "<p>The code passed all tests.</p>"
}
```

Figura 3.5. Protocolo de resposta do avaliador externo
Fonte: Autoria própria.

Todo processo descrito anteriormente refere-se ao fluxo de uma nova submissão. No caso do fluxo da obtenção do *ranking*, o processo é simples, no qual o **Server** recebe uma requisição GET, e em seguida aciona a classe **Ranking**. A classe **Ranking** simplesmente obtém do banco de dados as informações necessárias para o *ranking*, como o id do usuário, a quantidade de submissões e a pontuação, por fim, os retorna para o **Server**, que responde ao cliente de acordo com o protocolo da Figura 3.6.

Em todos os processos de comunicação com o banco de dados, é utilizado como abstração o *framework* Sqlalchemy para a comunicação com o banco de dados Mysql.

Para toda submissão, o aluno recebe uma resposta da plataforma, provida pelo avaliador externo, a Figura 3.7 apresenta um exemplo de uma avaliação para uma atividade submetida com gamificação.

```
[
    {
        "student_id": "aluno_1",
        "submissions": 3,
        "score": 30
    },
    {
        "student_id": "aluno_2",
        "submissions": 2,
        "score": 15
    }
]
```


Figura 3.6. Protocolo de resposta do *ranking*
Fonte: Autoria própria.

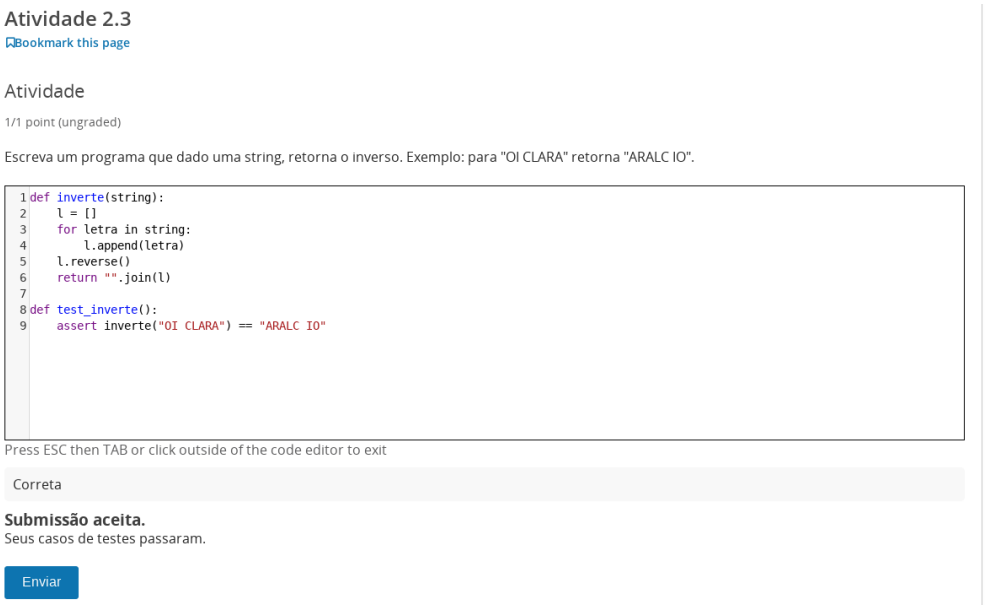


Figura 3.7. Exemplo de avaliação de submissão
Fonte: Autoria própria.

O código do avaliador externo está disponível na plataforma Github, no seguinte endereço: https://github.com/brnomendes/grader-edx, sob a licença MIT.

3.4. Regras para pontuações da gamificação

Uma questão importante para a utilização de elementos de *design* de jogos como gamificação é a forma com que serão utilizados, ou seja, as regras. No caso deste trabalho, tendo como o elemento principal a competição entre os códigos dos alunos, e como resultado pontuações para os mesmos, as regras devem ser estabelecidas para considerar o que realmente gera pontos e regras que retirem pontos para certas punições.

Um conjunto de regras foi estabelecido priorizando certas ações por parte dos alunos em seus programas. Um resumo do conjunto de regras é apresentando a seguir, no qual indica-se os pontos a serem adicionados ou removidos e a condição para tal.

1. **-100** – Envio de uma atividade cuja resolução foi idêntica a um envio anterior.
2. Caso não ocorra erros no seu programa (por exemplo, de sintaxe):
   2.1. **-50** – **Por falha** dos próprios testes.
   2.2. Seu programa **contra** cada conjunto de teste de outros alunos:
      2.2.1. **+10** – Se o seu programar passar sem falhas.
      2.2.2. **-2** – **Por falha** encontrada em seu programa.
   2.3. Seu conjunto de testes **contra** cada programa de outros alunos:
      2.3.1. **+10 * cov** – A cobertura de seus testes (0 <= cov <=1).
      2.3.2. **+2** – **Por falha** encontrada pelo seus testes.

A regra **1** estabelece uma punição para o reenvio de uma mesma atividade. Tal regra é necessário para evitar reenvios desnecessários, seja por falta de intenção ou por malícia. A regra **2** é uma regra condicional para a aplicação das regras seguintes, no qual verifica se o código do aluno contém erros que o impedem de executar, de tal forma que não é possível obter resultados de execuções de conjuntos de testes.

A regra **2.1** define uma punição de 50 pontos por cada falha que o conjunto de testes encontra no próprio programa. Desta forma, evita-se que um aluno crie casos de testes fictícios para prejudicar outros alunos e que realmente criem testes que condizem com o seu próprio programa.

As regras **2.2.1** e **2.2.2** referem-se a o programa que está sendo testado. Logo, ou recebe-se 10 pontos por conjunto de caso de testes de outro aluno que não encontra falhas no programa, ou perde-se 2 pontos a cada falha encontrada. Tais regras ajudam a motivar a criação de programa completos e corretos.

Por fim, as regras **2.3.1** e **2.3.2** referem-se aos conjuntos de casos de testes. Estas regras atribuem pontos para bons conjuntos de casos de testes, no qual a regra **2.3.1** pontua de acordo com a cobertura de código que os testes cobrem, e a regra **2.3.2** pontua a exploração de falhas nos programas dos colegas.

Para um entendimento melhor do mecanismo de competição e da aplicação das regras, a seguir é descrito um cenário com uma simulação da aplicação das regras. Como exemplo, considere o curso com os alunos fictícios João, Maria e Josias. Eles se deparam com o seguinte enunciado: Crie uma função que retorne a soma de dois números. Analisaremos as regras sendo aplicadas na submissão da resposta de Josias, para isto considere anteriormente o seguinte cenário:

O aluno **João** enviou a seguinte resposta:

```
def soma(x, y):
    if x == 2 and y == 2:
        return 4
    else:
        return 0

def test_soma():
    assert soma(2, 2) == 4
```

A aluna **Maria** enviou para a mesma atividade a seguinte resposta:

```
def soma(x, y):
    return x + x

def test_soma():
    assert soma(3, 3) == 6
```

Após a submissão desses alunos, o *ranking* ficou da seguinte forma:

- 1º Maria - 19.5 Pontos
- 2º João - 8 Pontos
- 3º Josias - 0 Pontos

Dado este cenário inicial, verificaremos a aplicação das regras e o resultado do *ranking* após a submissão da resposta de **Josias**:

```
def soma(x, y):
    return x + y

def test_soma():
    assert soma(1, 2) == 3
```

A seguir é aplicado as regras **1**, **2** e **3**. Como será descrito, tais regras não implicarão na pontuação de **Josias**.

- Ao submeter a solução, é verificado se houve outros envios para o mesmo exercício. Caso ocorra, ocorre a perda de 100 pontos. Como **Josias** está enviando pela primeira vez, não perde pontos.
- Em seguida, é verificado se o programa contém erros triviais (por exemplo, de sintaxe). Caso possua algum erro trivial, ele não será executado com os programas dos outros alunos e não gerará pontos para ninguém. Caso não possua erros triviais, continua a avaliação dos pontos.

- Após verificado as duas situações anteriores, o programa é executado com relação aos seus próprios casos de teste. Para cada caso de teste próprio que falhar, ocorre a perda de 50 pontos. No exemplo, o programa de **Josias** não falhou com relação aos seus próprios casos de teste, então não perde pontos.

Neste momento, o programa de **Josias** será testado com os casos de testes dos outros alunos. Além disso, o caso de teste de **Josias** será executado contra os programas dos outros alunos. Iniciaremos com a aplicação das regras **2.2.1** e **2.2.2**, executando o programa de **Josias** contra os casos de teste de **João** e **Maria**.

Conjunto de casos de testes do **João**:

```
def test_soma():
    assert soma(2, 2) == 4
```

Conjunto de casos de testes da **Maria**:

```
def test_soma():
    assert soma(3, 3) == 6
```

- Programa de Josias **contra** casos de teste de João:
  - Neste caso, os casos de testes do João não encontram erros no programa de Josias. Logo, Josias recebe **10 pontos**. Como os casos de testes do João tiveram uma cobertura de 100%, ele recebe 10 * cov, no qual cov é 1, logo recebe também **10 pontos**.
- Programa de Josias **contra** casos de teste de Maria:
  - O mesmo ocorre entre Josias e a Maria, ou seja, os casos de testes dela não encontraram falhas no programa de Josias, mas tiveram uma cobertura de 100%, ambos recebem **10 pontos**.

Atualizando o *ranking*, temos que:

- 1º Maria - 29.5 Pontos
- 2º Josias - 20 Pontos
- 3º João - 18 Pontos

Neste momento, analisaremos os casos de teste de Josias **contra** os programas de João e Maria, considerando as regras **2.3.1** e **2.3.2**.

Programa do João:

```
def soma(x, y):
    if x == 2 and y == 2:
        return 4
else:
    return 0
```

Programa da Maria:

```
def soma(x, y):
    return x + x
```

- Casos de teste de Josias **contra** o programa de João:
  - Os casos de teste de Josias encontram um erro no programa do João. Logo, Josias recebe **2 pontos**. A cobertura dos casos de teste de Josias em relação ao código de João é de 75%, logo Josias recebe mais 10 * 0.75 = **7.5 pontos**. Em contrapartida, João perde **2 pontos** por causa do erro encontrado em seu programa.
- Casos de teste de Josias **contra** o programa de Maria:
  - Os casos de teste de Josias encontram um erro no programa da Maria. Logo, Josias recebe **2 pontos**. A cobertura dos casos de teste de Josias em relação código de Maria é de 100%, logo Josias recebe mais 10 * 1 = **10 pontos**. Em contrapartida, Maria perde **2 pontos** por um erro encontrada em seu programa.

Por fim, o *ranking* final conta com as seguintes pontuações após a submissão de **Josias**:

- 1º Josias - 41.5 Pontos
- 2º Maria - 27.5 Pontos
- 3º João - 16 Pontos

Concluindo, é possível analisar que Josias ficou com a maior quantidade de pontos, devido a corretude de seu programa e seus casos de testes que encontram falhas nos programas dos outros alunos.

3.5. Considerações finais

Neste capítulo foi apresentado o método utilizado para a implementação da gamificação. Também foi apresentada a plataforma de MOOC escolhida, o Open edX. O Open edX permite a extensão de funcionalidades e ferramentas auxiliares para sua utilização. De código aberto, o Open edX possui diversos mecanismos de avaliação, um deles é o avaliador externo.

O mecanismo de avaliação externa no Open edX permite a criação de um sistema para validar as propostas de submissão dos alunos em uma disciplina introdutória de programação.

Utilizando este mecanismo, foi implementado um sistema com um elemento competitivo, em que os programas e casos de testes dos alunos são cruzados para validar as soluções e, com a agregação de elementos de *design* de jogos, como o *ranking* e a definição das regras, houve a implementação de um ambiente ensino gamificado com o objetivo de estimular os alunos à programarem códigos de qualidade e com corretude.

Após a implementação do MOOC gamificado, uma avaliação precisa ser feita para obter os aspectos acertados e o que precisa ser melhorado. Para isto, foi realizado um experimento com uma amostra de alunos, no qual utilizaram o MOOC com a gamificação e, dado os resultados deste experimento, conforme relatado no próximo capítulo.

Capítulo 4

Experimento

Este capítulo apresenta o projeto de experimento referente a este trabalho e os resultados obtidos, além de uma discussão sobre tal. O objetivo do experimento, ao executar a disciplina de programação na plataforma de MOOCs junto a gamificação, foi avaliar a proposta em motivar os alunos a criarem soluções de código com qualidade e corretude para atividades avaliativas com o mínimo de esforço do professor.

O restante deste capítulo esta organizado da seguinte forma, a Seção 4.1 descreve o método utilizado para realizar a avaliação do MOOC gamificado proposto neste trabalho. Na Seção 4.2 é apresentado a organização e como foi executado o experimento, e a Seção 4.3 descreve os resultados obtidos bem como uma discussão sobre estes resultados. Os resultados do questionário aplicado ao fim do experimento aos alunos participantes é apresentado na Seção 4.4, e por fim, a Seção 4.5 descreve as considerações finais referentes a este capítulo.

4.1. Avaliação do MOOC gamificado

O objetivo ao executar a disciplina de programação com os elementos de *design* de jogos é de motivar os alunos a criarem soluções de código com qualidade e corretude para atividades avaliativas através de uma plataforma de MOOCs, com o mínimo de esforço do professor. Assim, foram definidas as seguintes questões de pesquisa (QP):

- **QP**$_1$: os mecanismos de *design* de jogos escolhidos permitem os alunos a implementarem código com qualidade?
- **QP**$_2$: a proposta de gamificação permite que os próprios alunos possam realizar as avaliações sem esforços por parte do professor?
- **QP**$_3$: a proposta de gamificação motiva a entrega das soluções pelos alunos nas atividades avaliativas?

Para avaliar os elementos de *design* de jogos escolhidos, durante o período da disciplina de programação, houve uma avaliação de desempenho dos alunos participantes. A cada atividade, um grupo diferente de alunos realizou a atividade com gamificação e o outro grupo sem a gamificação. Esta separação dos grupos para cada atividade pode ajudar a validar a motivação e a influência da gamificação nas atividades avaliativas.

Em um estudo prévio foi, possível verificar que o Open edX possui um mecanismo para a separação dos grupos em cada atividade, também chamado de teste A/B. No Open edX, para a execução do experimento, cada atividade terá a versão com gamificação e a versão sem gamificação, e o Open edX distribuirá 50% dos alunos para cada versão aleatoriamente.

Por fim, ao final do período da disciplina, foi fornecido um questionário aos alunos para caracterizar a motivação e obter um *feedback* sobre os elementos de *design* de jogos selecionados e o modo como a gamificação foi proposta.

4.2. Organização do experimento

Para este experimento, disponibilizamos a alunos iniciantes em programação acesso a plataforma Open edX com a gamificação. A população de alunos consistes em alunos ingressantes de cursos de Engenharias, no qual possuem uma matéria de programação, geralmente ministrada utilizando a linguagem Python.

O curso oferecido na plataforma consiste em um material introdutório, em que é explicado a dinâmica do curso e da resolução das atividades, bem como a gamificação e as regras para as pontuações. O curso contem também um material de apoio para programação com Python e testes de unidade utilizando o *framework* pytest.

Além da introdução ao curso e o material de apoio, foi disponibilizados aos alunos 8 atividades de níveis diferentes, no qual todas devem ser resolvidas com um programa em Python e um conjunto de casos de testes. O material didático de Python e as atividades selecionadas, foram adaptadas com base de materiais utilizados anteriormente em disciplinas de programação tradicionais. A plataforma esteve disponível no seguinte endereço interno: http://mooc.projetomarco.dacom:8000

A amostra utilizada foram alunos de uma turma do curso de Engenharia Civil da Universidade Tecnológica Federal do Paraná, campus Campo Mourão, e para cada um dos 8 exercícios disponibilizados na plataforma, cada aluno fez ou da maneira tradicional ou da maneira gamificada, em que a plataforma escolhia aleatoriamente para cada exercício de qual forma seria apresentada ao aluno. Ainda, ao final do curso, um questionário foi disponibilizado para que os alunos respondessem.

Neste caso especifico, antes da execução do experimento, a disponibilização da plataforma para os alunos, foi ministrado um treinamento prévio sobre pytest, uma visão geral da instalação e execução, pois tais alunos nunca tiveram contato com testes de software. Para este treinamento

foi utilizado o horário de uma das aulas da disciplina de programação da turma do curso de Engenharia Civil.

Logo após o treinamento, foi marcado junto aos alunos vários dias para a utilização da plataforma em laboratórios, pois a infraestrutura da Universidade impediu que fornecêssemos acesso externo ao Open edX. Nos laboratórios os alunos poderiam escolher um computador e obter acesso ao Open edX. Propositalmente não foram dadas orientação aos alunos sobre a plataforma e o curso, para que obtivéssemos resultados parecidos caso os alunos acessassem externamente a Universidade.

No dia 10 de novembro de 2017, 28 alunos se cadastraram na plataforma para o experimento, e no dia 14 de novembro de 2017, mais 2 alunos se cadastraram na plataforma. Logo, os resultados e discussões apresentados neste capítulo são obtidos de uma amostra de 30 alunos. Os resultados apresentados referem-se a estes dois dias, mas a plataforma continuou disponível para que estes e outros alunos tivessem acesso. Na primeira data do experimento, alguns alunos que não são da turma de Engenharia Civil participaram de livre espontânea vontade, mas não houve uma divulgação pública para tal.

4.3. Resultados do experimento

Dado a amostra de 30 alunos e as 8 atividades, obtivemos um total de 135 submissões, no qual cada aluno resolvia uma atividade ou com a gamificação ou da maneira tradicional, tal que a maneira da atividade ser apresentada ao aluno foi escolhida aleatoriamente pela plataforma. Relembrando que os alunos poderiam reenviar a mesma atividade quantas vezes quisessem, mas no caso das atividades com gamificação, sempre recebendo as pontuações de acordo com as regras pré-estabelecidas. A Tabela 4.1 apresenta de uma forma detalhada a quantidade de submissões enviadas a cada atividade, separadas pela forma ao qual foi apresentado aos alunos. Observa-se que na Tabela 4.1, algumas atividades contam com mais submissões em um grupo do que no outro, isso deve-se ao fato de, para cada atividade foi apresentado a 50% dos alunos de forma gamificada e para os outros 50% de forma tradicional, mas os alunos poderiam não resolver a atividade, ou enviar mais de uma submissão.

Tabela 4.1. Quantidade de submissões para as atividades no experimento

Atividade	Gamificado	Tradicional
1	24	25
2	16	19
3	9	7
4	8	6
5	4	9
6	0	3
7	2	1
8	2	0

Para responder a primeira questão de pesquisa, realizamos uma análise das submissões dos alunos observando se as submissões no qual as atividades apresentadas que valiam pontos para o *ranking* têm diferenças na corretude das submissões enviadas as atividades apresentadas sem a recompensa de pontos. Para isto, utilizamos um teste estatístico.

Utilizamos o teste de Mann-Whitney para verificar se houve diferença nas submissões dos dois grupos, com gamificação e sem gamificação. Criamos um conjunto de 10 casos de testes para cada atividade, 7 casos de testes recolhidos dos programas dos alunos e 3 criados pelos autores. Dado esses conjuntos de casos de testes para cada atividade, executamos os testes para todas as submissões enviados pelos alunos, então obtivemos um valor escalar, de 0 a 10, no qual representa o número de falhas encontradas no programas. Quando o programa da submissão havia erros simples, como de sintaxe, o que impede a execução de testes, definimos o valor para 10. Por fim, para cada atividade, selecionamos o valor da melhor submissão de cada aluno.

A escolha do teste de Mann-Whitney deve-se ao fato de utilizarmos apenas uma variável, tendo somente duas possibilidades, os dois grupos. De posse dos valores, por grupo, da melhor submissão de cada aluno por atividade, executamos o teste de Mann-Whitney. Dado uma hipótese nula, no qual as submissões das atividades com gamificação e sem gamificação não contem diferenças no quesito de corretude, o teste nos indicou que não foi possível rejeitar esta hipótese. Logo, a princípio, não foi possível responder a primeira questão de pesquisa.

Relatando a experiência observada no primeiro dia do experimento ajuda a entender os resultados obtidos pelo teste estatístico. Observou-se que a quantidade de alunos que leram as instruções sobre o funcionamento do curso, programação com Python, da gamificação e das regras para a pontuação, foram mínimas. A maior parte dos alunos que participaram do experimento foram direto para as atividades. A partir deste fato, observa-se também, analisando a saída do registro de eventos do avaliador, que o primeiro acesso a página de *ranking* aconteceu horas após o início do experimento. Logo, muitos alunos não se atentaram ao fato de estarem realizando atividades com gamificação e competindo entre si. Ponto importante para ser discutido e levado a trabalhos futuros.

Outra questão é a quantidade de submissões com erros simples, como de sintaxe, na qual impedem a execução de testes, consequentemente não gerando pontuações. Das 135 submissões, 40 submissões continham no mínimo um erro simples, cerca de 29.63%. Este é um número expressivo, no qual não foi o primeiro contato da amostra de alunos com Python, dado que estão cursando uma disciplina de programação no curso de Engenharia Civil, logo, não só no experimento os alunos não se interessaram em ler os textos introdutórios, como houve falta de atenção no momento da resolução das atividades.

Para responder a segunda questão de pesquisa, no qual questiona-se se a gamificação permite a ausência do professor ou um suporte na avaliação dos programas dos alunos, uma análise das submissões enviadas pelos alunos foi realizada. Observando as submissões dos alunos que ficaram no topo do *ranking*, ou seja, com maior pontuação, é possível notar que os programas e

casos de testes estão corretos, e que foram realizadas no máximo em duas submissões por atividade. Analisando as submissões dos últimos alunos na posição do *ranking*, observa-se que há erros no algoritmo, especificamente na lógica desenvolvida, que não condiz com os resultados esperados como uma solução, além de múltiplos reenvios para uma mesma atividade. Dado está analise manual das submissões dos alunos, observando suas posições no *ranking*, é possível afirmar que a gamificação permite que os próprios alunos possam realizar as avaliações sem esforços por parte do professor.

4.4. Resultados do questionário

Ao final do experimento, na própria plataforma do Open edX, um questionário foi disponibilizado aos alunos, contendo 8 perguntas objetivas e 1 questão dissertativa e aberta. O objetivo da aplicação do questionário foi obter resultados para responder a terceira e última questão de pesquisa, no qual questiona-se se a proposta de gamificação motiva a entrega das soluções pelos alunos nas atividades avaliativas. Dos 30 alunos participantes do experimento, 28 responderam o questionário.

A primeira questão do questionário refere-se ao curso que o aluno cursa no momento, para obter-se uma visão geral da amostra de alunos que participaram do experimento: 21 alunos responderam que cursam Engenharia Civil, 4 cursam Ciência da Computação, 2 cursam Engenharia Ambiental e 1 aluno cursa Engenharia de Alimentos.

As questões 2, 3 e 4 fornecem como opções uma escala Likert com Péssimo, Ruim, Razoável, Bom e Ótimo, no qual pretende-se obter resultados sobre o conhecimento de programação dos alunos. Utilizando a mediana das respostas de cada questão observamos que, para a segunda questão, "**Qual a sua experiência anterior com programação Python?**", temos como mediana a opção Razoável. Para a terceira questão, "**Como você avalia seu aprendizado sobre programação com Python após o término do curso na plataforma edX?**", também temos como mediana a opção Razoável. Por fim, para a quarta questão, "**Como você avalia seu aprendizado sobre testes de unidade com Python após o término do curso na plataforma edX?**", obtivemos o mesmo resultado, Razoável como mediana para esta questão, mas, observa-se que para as três questões, mais de 75% dos alunos responderam ou Razoável ou Bom.

As questões 5, 6, 7 e 8 também apresentam como opções de respostas uma escala Likert, mas com: Discordo totalmente, Discordo parcialmente, Indiferente, Concordo parcialmente e Concordo totalmente. A opção Concordo parcialmente foi o resultado da mediana das respostas destas 4 questões: **(5) As atividades que valiam pontos para o *ranking* motivou mais a resolve-las do que as que não valiam? (6) A competição entre suas submissões com a dos outros alunos motivou para que você criasse programas e testes melhores? (7) O *ranking* motivou para que você criasse programas e testes melhores? (8) Você utilizaria um ambiente gamificado novamente?** Ressalta-se que as opções Indiferente e Concordo parcialmente obtiveram no mínimo 75% das escolhas dos alunos em nestas questões. Com o

resultado destas questões obtemos a experiência dos alunos quanto a gamificação implementada junto a plataforma. Na Tabela 4.2 é apresentado detalhadamente a quantidade de respostas em cada questão.

Tabela 4.2. Resultado das questões do questionário

Questão	Péssimo ou Discordo totalmente	Ruim ou Discordo parcialmente	Razoável ou Indiferente	Bom ou Concordo parcialmente	Ótimo ou Concordo totalmente
2	4	2	11	10	1
3	1	2	14	11	0
4	3	1	12	12	0
5	0	1	11	11	5
6	0	1	10	11	6
7	0	0	11	13	4
8	0	0	9	13	6

Para a última questão, tendo a seguinte questão aberta: Comente sobre sua experiência na realização do curso com a gamificação. Realizamos uma sumarização das respostas dos alunos utilizando 5 categorias obtidas realizando uma análise prévia das respostas:

- Gostou da realização do curso.
- Não gostou da realização do curso.
- A competição motivou.
- Obteve novos conhecimentos.
- Não tinha capacidade para resolver as atividades.

Dentre as 28 respostas obtidas, 17 respostas citam que o aluno gostou da realização do curso, 2 respostas que o aluno não gostou da realização do curso, 4 respostas apontam que a competição motivou, em 3 respostas temos que o aluno obteve novos conhecimento e em 6 respostas que o aluno não tinha capacidade para resolver as atividades, devido à falta de conhecimento. Respostas como "*Eu gosto de pensar em um teste que quebre os dos colegas, incita a competição*" e "*Me senti mais motivado e desafiado, tentando subir no ranking*", mostra que, no geral, os alunos gostaram de realizar o curso gamificado, e aprender um novo conceito que nunca tinha visto antes, o de testes de software, conforme atesta resposta de um aluno: "*Aprendi uma forma mais eficiente de testar minhas funções para garantir que estão corretas*".

Concluímos, após apresentado as respostas da pergunta aberta e das perguntas objetivas, que os alunos consideraram satisfatória a utilização da gamificação com a competição, embora a diferença entre as submissões para atividades com gamificação e sem gamificação não foram significativas conforme observado na Seção 4.3. Logo, de acordo com a opinião dos alunos, é possível responder a terceira questão de pesquisa, no qual a proposta de gamificação motiva a entrega das soluções pelos alunos nas atividades avaliativas.

4.5. Considerações finais

Após a implementação do avaliador externo junto a plataforma Open edX, foi aplicado um experimento como descrito neste capítulo, com o objetivo de obter respostas para as questões de pesquisa. O experimento contou com participação de alunos em sua maioria da disciplina de programação do curso de Engenharia Civil, e em um total de 30 alunos. Estes alunos realizaram atividades na plataforma, no qual aleatoriamente para cada atividade, era apresentado ao aluno na forma gamificada ou sem a gamificação. Após a conclusão do curso, os alunos responderam um questionário contendo 8 perguntas objetivas, e uma pergunta aberta, no qual descreveram suas experiências na realização do curso.

Para a primeira questão de pesquisa, a qual refere-se a diferença entre realizar atividades com gamificação e sem gamificação, não foi possível obter uma resposta satisfatória. Na segunda questão de pesquisa, analisando as submissões dos alunos e suas posições no *ranking*, observamos que a pontuação pode ser um fator auxiliar na correção das soluções. Na terceira questão, obtendo a opinião dos alunos, verificamos que a gamificação, com a competitividade, motivou a entrega das soluções das atividades na plataforma.

Dado a análise das submissões dos alunos, e do relato da experiência no primeiro dia do experimento, observa-se que para trabalhos futuros, a gamificação deve ser mais restritiva e guiar o aluno no entendimento sobre a plataforma, como dicas, imagens, etc. Pois, além de muitos alunos notarem a gamificação somente após algum tempo de utilização da plataforma, e devido ao fato de não terem realizado a leitura dos textos introdutórios, muitos casos de reenvios de submissões (que gera a punição de pontos) e submissões com erros de sintaxe aconteceram. Assim, sugere-se um passo a passo que guie o aluno intuitivamente a compreender toda a plataforma, sua utilização e o aspecto de gamificação.

Uma ameaça a validade ocorrida, e que também se caracteriza para trabalhos futuros, é a disponibilidade da plataforma para acesso externo aos laboratórios da Universidade. Cursos online, como MOOC, geralmente são cursados sem a presença do ambiente escolar, como uma sala de aula ou laboratórios, e que podem influenciar nos resultados. Neste trabalho houveram tentativas para realizar tal feito, mas a infraestrutura da Universidade impediu a disponibilização deste recurso.

Capítulo 5

Conclusões

Disciplinas de introdução a programação são consideradas difíceis e desafiadoras por alunos ingressantes. O *feedback* e a motivação para estudar são fatores importantes para que os alunos se envolvam com as disciplinas (IZEKI et al., 2016). Este é um problema a ser resolvido, já que disciplinas de programação são essenciais na formação de discentes de computação. Ambientes de aprendizagem gamificados podem funcionar como uma estratégia no engajamento de estudantes no aprendizado de programação (SILVA et al., 2016).

Nesta pesquisa, o objetivo foi propor e implementar elementos de *design* de jogos como gamificação para disciplinas introdutórias de programação, respondendo três questões de pesquisa:

- **QP**$_1$: os mecanismos de *design* de jogos escolhidos permitem os alunos a implementarem código com qualidade?
- **QP**$_2$: a proposta de gamificação permite que os próprios alunos possam realizar as avaliações sem esforços por parte do professor?
- **QP**$_3$: a proposta de gamificação motiva a entrega das soluções pelos alunos nas atividades avaliativas?

Então, dadas estas questões, foi implementado um curso MOOC com gamificação utilizando uma plataforma de MOOC de código aberto, o Open edX. O principal elemento da gamificação utilizado foi um mecanismo de competição entre os alunos, no qual, para toda atividade avaliativa, os alunos deveriam submeter um programa e um conjunto de casos de testes. Após isso, esta submissão foi cruzada com as submissões dos outros alunos, resultando em penalidades ou recompensas de pontos, os quais poderiam ser vistos por todos em um *ranking* global na plataforma.

Após a implementação do MOOC gamificado, um experimento foi realizado afim de obter resultados para avaliá-lo. O experimento possuia textos explicativos sobre a plataforma, a gamificação e programação, neste caso utilizando a linguagem Python e conteúdo sobre testes

de software. Ainda, o curso disponibilizado continha exercícios que deveriam ser resolvidos com programas e conjunto de casos de testes. Ao final do curso, um questionário foi disponibilizado para os alunos participantes, a fim de caracterizar a motivação, e obter um *feedback* sobre os elementos de *design* de jogos selecionados e o modo como a gamificação foi proposta. Alunos de uma disciplina introdutória à Computação oferecida para um curso Engenharia Civil foram utilizados como amostra, no qual em dois dias participaram da disciplina.

Analisando os resultados do experimento, para a primeira questão de pesquisa, no qual refere-se a diferença entre a realização de atividades com gamificação e sem gamificação, utilizando um método estatístico, não foi observado tal diferença, logo não foi possível obter uma resposta satisfatória. Na segunda questão de pesquisa, analisando as submissões dos alunos e suas posições no *ranking*, observamos que a pontuação pode ser um fator auxiliar na correção das soluções. Na terceira questão, obtendo a opinião dos alunos, verificamos que a gamificação, com a competitividade, motivou a entrega das soluções das atividades na plataforma.

Como conclusão, neste trabalho foi implementado um MOOC gamificado com um mecanismo de competição que foi capaz de motivar os alunos a realizarem as atividades avaliativas. Também foi possível notar que os elementos utilizados podem facilitar a correção das soluções por parte do professor. Ainda se observa que, diante dos resultados apresentados, de acordo com o experimento realizado, faz-se necessário a adição de novos elementos à gamificação, principalmente uma maneira de guiar o aluno ao entendimento do MOOC gamificado.

Uma ameaça a validade ocorrida, e que também se caracteriza para trabalhos futuros, é a disponibilidade da plataforma para acesso externo aos laboratórios da Universidade, em que, a amostra de alunos nos dois dias de experimento compareceram presencialmente em laboratórios para a realização do curso. Cursos online, como MOOC, geralmente são cursados sem a presença do ambiente escolar, como uma sala de aula ou laboratórios, e que podem influenciar nos resultados.

Referências

BARIK, T.; MURPHY-HILL, E.; ZIMMERMANN, T. A perspective on blending programming environments and games: Beyond points, badges, and leaderboards. In: *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. [S.l.]: IEEE, 2016. p. 134–142. ISBN 978-1-5090-0253-5.

BERKLING, K.; THOMAS, C. Gamification of a software engineering course and a detailed analysis of the factors that lead to it's failure. In: *2013 International Conference on Interactive Collaborative Learning*. [S.l.: s.n.], 2013. p. 525–530.

BISHOP, Judith; HORSPOOL, R. Nigel; XIE, Tao; TILLMANN, Nikolai; HALLEUX, Jonathan de. Code Hunt: Experience with coding contests at scale. In: *37th International Conference on Software Engineering*. Piscataway, NJ, EUA: IEEE, 2015. p. 398–407.

CLEGG, Benjamin; ROJAS, José Miguel; FRASER, Gordon. Teaching software testing concepts using a mutation testing game. In: *International Conference on Software Engineering Companion (ICSE) 2017 – Software Engineering Education and Training (SEET)*. [S.l.: s.n.], 2017.

DANIEL, John. Making sense of MOOCs: Musings in a maze of myth, paradox and possibility. v. 2012, n. 3, p. 1–20, dez. 2012.

DARADOUMIS, T.; BASSI, R.; XHAFA, F.; CABALLé, S. A review on massive e-learning (MOOC) design, delivery and assessment. In: *8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. [S.l.]: IEEE, 2013. p. 208–213. ISBN 978-1-4799-1266-7.

DETERDING, Sebastian. Gamification: designing for motivation. *Interactions*, ACM, New York, NY, EUA, v. 19, n. 4, p. 14–17, jul.–ago. 2012. ISSN 1072-5520.

DETERDING, Sebastian; DIXON, Dan; KHALED, Rilla; NACKE, Lennart. From game design elements to gamefulness: Defining "gamification". In: *15th International Academic MindTrek Conference: Envisioning Future Media Environments*. New York, NY, EUA: ACM, 2011. p. 9–15. ISBN 978-1-4503-0816-8.

DETERDING, Sebastian; SICART, Miguel; NACKE, Lennart; O'HARA, Kenton; DIXON, Dan. Gamification. using game-design elements in non-gaming contexts. In: *CHI EA '11 CHI '11 Extended Abstracts on Human Factors in Computing Systems*. Vancouver, BC, Canada: [s.n.], 2011. p. 2425–2428.

DOMíNGUEZ, Adrián; NAVARRETE, Joseba Saenz de; MARCOS, Luis de; FERNáNDEZ-SANZ, Luis; PAGéS, Carmen; MARTíNEZ-HERRáIZ, José-Javier. Gamifying learning experiences: Practical implications and outcomes. *Computers & Education*, Elsevier, Países Baixos, v. 63, p. 380–392, abr. 2013. ISSN 0360-1315.

DORLING, Alec; MCCAFFERY, Fergal. The gamification of SPICE. In: MAS, Antonia; MESQUIDA, Antoni; ROUT, Terry; O'CONNOR, Rory V.; DORLING, Alec (Ed.). *Software Process Improvement and Capability Determination: 12th International Conference, SPICE 2012, Palma, Spain, May 29-31, 2012. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 295–301. ISBN 978-3-642-30439-2. Disponível em: <http://dx.doi.org/10.1007/978-3-642-30439-2_35>.

DíAZ, Héctor J. Pijeira; RUIZ, Javier Santofimia; RUIPéREZ-VALIENTE, José A.; MUñOZ-MERINO, Pedro J.; KLOOS, Carlos Delgado. A demonstration of ANALYSE: A learning analytics tool for Open edX. In: *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*. New York, NY, USA: ACM, 2016. (L@S '16), p. 329–330. ISBN 978-1-4503-3726-7. Disponível em: <http://doi.acm.org/10.1145/2876034.2893402>.

EAGLE, Michael; BARNES, Tiffany. Wu's castle: teaching arrays and loops in a game. In: *13th Annual Conference on Innovation and Technology in Computer Science Education*. Madrid,: ACM, 2008. p. 245–249. ISBN 978-1-60558-078-4.

EDWARDS, Stephen H. Using software testing to move students from trial-and-error to reflection-in-action. In: *35th SIGCSE Technical Symposium on Computer Science Education*. New York, NY,EUA: ACM, 2004. p. 26–30. ISBN 1-58113-798-2.

FASSFINDER, Aracele; DELAMARO, Márcio Eduardo; BARBOSA, Ellen Francine. Construção e uso de MOOCs: Uma revisão sistemática. In: *XXV Simpósio Brasileiro de Informática na Educação*. Dourados, MS, Brasil: SBC, 2014. p. 332–341.

FOUCHé, Sandro; MANGLE, Andrew H. Code Hunt as platform for gamification of cybersecurity training. In: *1st International Workshop on Code Hunt Workshop on Educational Software Engineering*. New York, NY, EUA: ACM, 2015. p. 9–11. ISBN 978-1-4503-3711-3.

GORDóN, Sandra Sánchez; MORA, Sergio Luján. Adaptive content presentation extension for open edx. enhancing moocs accessibility for users with disabilities. In: *The Eighth International Conference on Advances in Computer-Human Interactions (ACHI 2015)*. [S.l.: s.n.], 2015. p. 181–183.

IZEKI, Claudia Akemi; NAGAI, Walter Aoiama; DIAS, Rodrigo Moreira Carvalho. Experiência no uso de ferramentas online gamificadas na introdução à programação de computadores. In: *XXII Workshop de Informática na Escola*. Porto Alegre, RS, Brasil: SBC, 2016. p. 301–310. ISSN 2316-6541.

PONTI, Marisa. Hei mookie! where do i start? the role of artifacts in an unmanned MOOC. In: *47th Hawaii International Conference on System Science*. [S.l.]: IEEE Computer Society, 2014. p. 1625–1634.

ROJAS, J. M.; FRASER, G. Code defenders: A mutation testing game. In: *9th International Conference on Software Testing, Verification and Validation Workshops*. [S.l.: s.n.], 2016. p. 162–167. ISBN 978-1-5090-3675-2.

ROJAS, José Miguel; FRASER, Gordon. Teaching mutation testing using gamification. In: *European Conference of Software Engineering Education 2016 (ECSEE)*. [S.l.]: Shaker Publishing, 2016. p. 1–5.

ROJAS, José Miguel; FRASER, Gordon. Teaching software testing with a mutation testing game. In: CHURCH, Luke (Ed.). *27th Annual Workshop Psychology of Programming Interest Group 2016 (PPIG)*. [S.l.]: PPIG, 2016. p. 290–293.

ROJAS, José Miguel; WHITE, Thomas; CLEGG, Benjamin; FRASER, Gordon. Code Defenders: Crowdsourcing effective tests and subtle mutants with a mutation testing game. In: *Proc. of the International Conference on Software Engineering (ICSE) 2017*. [S.l.: s.n.], 2017. To appear.

RUIPéREZ-VALIENTE, José A.; MUñOZ-MERINO, Pedro J.; GASCóN-PINEDO, José A. Scaling to massiveness with analyse: A learning analytics tool for open edx. In: *IEEE Transactions on Human-Machine Systems*. [S.l.: s.n.], 2017. PP, n. 99, p. 1–6.

RUIZ, Javier Santofimia; DíAZ, Héctor J. Pijeira; RUIPéREZ-VALIENTE, José A.; MUñOZ-MERINO, Pedro J.; KLOOS, Carlos Delgado. Towards the development of a learning analytics extension in Open edX. In: *2nd International Conference on Technological Ecosystems for Enhancing Multiculturality*. New York, NY, EUA: ACM, 2014. p. 299–306. ISBN 978-1-4503-2896-8.

SIEMENS, George. Massive open online courses: Innovation in education? In: _____. Commonwealth of Learning, Athabasca University, 2013. cap. 1, p. 5–16. ISBN 978-1-894975-62-9. Disponível em: <https://oerknowledgecloud.org/sites/oerknowledgecloud.org/files/pub_PS_OER-IRP_web.pdf\#page=31>.

SILVA, Tatyane; MELO, Jeane; TEDESCO, Patrícia. Um modelo para promover o engajamento estudantil no aprendizado de programação utilizando gamification. In: *Congresso Brasileiro de Informática na Educação – Concurso de Teses e Dissertações - Teses de Doutorado*. Porto Alegre, RS, Brasil: SBC, 2016. p. 71–80. ISSN 2316-8889.

SIMON, Beth; CUTTS, Quintin. Peer instruction: a teaching method to foster deep understanding. *Communications of the ACM*, ACM, New York, NY,EUA, v. 55, n. 2, p. 27–29, fev. 2012. ISSN 0001-0782.

SINGER, Leif; SCHNEIDER, Kurt. It was a bit of a race: Gamification of version control. In: *Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques (GAS)*. [S.l.: s.n.], 2012. p. 5–8.

SIVAMUNI, Kalaimagal; BHATTACHARYA, Sujoy. Assembling pieces of the MOOCs jigsaw puzzle. In: *2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE 2013)*. [S.l.]: IEEE, 2013. p. 393–398. ISBN 978-1-4799-1624-5.

SUBBIAN, V. Role of MOOCs in integrated STEM education: A learning perspective. IEEE, p. 1–4, mar. 2013.

TENóRIO, Thyago; BITTENCOURT, Ig Ibert; ISOTANI, Seiji; PEDRO, Alan; OSPINA, Patrícia. A gamified peer assessment model for on-line learning environments in a competitive context. *Computers in Human Behavior*, Elsevier, Amsterdam, Países Baixos, v. 64, p. 247–263, nov. 2016. ISSN 0747-5632.

TILLMANN, Nikolai; BISHOP, Judith; HORSPOOL, Nigel; PERELMAN, Daniel; XIE, Tao. Code Hunt: Searching for secret code for fun. In: *7th International Workshop on Search-Based Software Testing*. New York, NY, EUA: ACM, 2014. p. 23–26. ISBN 978-1-4503-2852-4.

TILLMANN, N.; HALLEUX, J. De; XIE, Tao. Pex4Fun: Teaching and learning computer science via social gaming. In: *24th Conference on Software Engineering Education and Training*. Waikiki, Hawaii, EUA: IEEE Computer Society, 2011. p. 546–548. ISBN 9781457703492. ISSN 1093-0175.

TILLMANN, Nikolai; HALLEUX, Jonathan de; XIE, Tao; BISHOP, Judith. Code Hunt: Gamifying teaching and learning of computer science at scale. In: *1st ACM Conference on Learning @ Scale Conference*. New York, NY, EUA: ACM, 2014. p. 221–222. ISBN 978-1-4503-2669-8.

TILLMANN, Nikolai; HALLEUX, Jonathan de; XIE, Tao; BISHOP, Judith. Constructing coding duels in Pex4Fun and Code Hunt. In: *2014 International Symposium on Software Testing and Analysis*. New York, NY, EUA: ACM, 2014. p. 445–448. ISBN 978-1-4503-2645-2.

XIE, Tao; BISHOP, Judith; HORSPOOL, R. Nigel; TILLMANN, Nikolai; HALLEUX, Jonathan de. Crowdsourcing code and process via Code Hunt. In: *2nd International Workshop on CrowdSourcing in Software Engineering*. Piscataway, NJ, EUA: IEEE, 2015. p. 15–16.

XIE, Tao; BISHOP, Judith; TILLMANN, Nikolai; HALLEUX, Jonathan de. Gamifying software security education and training via secure coding duels in Code Hunt. In: *2015 Symposium and Bootcamp on the Science of Security*. New York, NY, EUA: ACM, 2015. p. 26:1–26:2. ISBN 978-1-4503-3376-4.