Ferramenta online para pareamento de questões de programação

Paulo Henrique Dias de Oliveira[1] Igor de Oliveira Knop[1]

[1]Universidade Federal de Juiz de Fora, Departamento de Ciência da computação, Brasil

RESUMO

As turmas de programação no início dos cursos de exatas e engenharias exibem um alto índice de reprovação. Entre os possíveis motivos, tem-se a necessidade de se desenvolver um pensamento lógico e a abstração na resolução de problemas. Diferente das disciplinas tradicionais, vistas no ensino fundamental e médio: é exigido constantemente que o aluno mude a visão entre a análise e síntese, criando e evoluindo soluções para cada questão ao interagir com o compilador ou interpretador. Entre as propostas para facilitar o processo de aprendizagem, está o uso de ambientes *online* de treinamento ou competição de programação. Esses ambientes permitem que os alunos submetam suas soluções para um conjunto de questões e tenham, de imediato, uma correção automática ou uma avaliação com os erros encontrados. Entretanto, o grande número de questões, de diferentes níveis de dificuldade, podem se tornar um problema para os alunos que não conseguem ter um progresso contínuo na resolução das questões. Este trabalho propõe uma ferramenta *online* chamada Code Training para auxiliar no processo de aprendizagem em programação. Foi criado um protótipo que permite que os alunos enviem suas soluções e obtenham uma resposta imediata, sobre quais condições foram atingidas corretamente. A cada envio, a ferramenta analisa e estima um grau de habilidade para o aluno. Esse grau de habilidade é utilizado para realizar uma recomendação automática de questões, fazendo um pareamento entre aluno e questão. Um estudo de caso é realizado no qual o protótipo é colocado em funcionamento com um conjunto de questões. Os dados colhidos evidenciaram que a ferramenta consegue estimar a habilidade e a dificuldade das questões, possibilitando a proposta aplicações subsequentes em sala de aula.

Palavras-chave: Ambiente virtual de aprendizagem, avaliação de *software*, pareamento.

1 INTRODUÇÃO

Iniciar os estudos em programação tem sido uma barreira, mesmo para os alunos dos cursos na área de ciência da computação. Mundialmente, as taxas de reprovação ultrapassam 30% [17] mas estima-se que esse número seja maior. Tais cursos possuem, logo no primeiro semestre, as disciplinas baseadas na análise e síntese de algoritmos, que possuem uma dinâmica bem diferente das tradicionais vistas no ensino fundamental e médio. Nessas disciplinas é exigido uma visão analítica e abstrata, conceitos matemáticos e raciocínio lógico, elementos aos quais os alunos podem não ter sido apresentados antes de ingressar na faculdade ou em um curso técnico [3].

Um outro problema, comumente observado, é o distanciamento entre alunos e professores, visto que em geral, um professor deve dividir sua atenção com vários alunos, já que essas disciplinas podem ser ofertadas para uma série de outros cursos da área de exatas. Dessa forma, fica difícil para o professor acompanhar pessoalmente o desempenho e rendimento de todos os alunos da turma até o momento da avaliação.

Além disso, muitas vezes as turmas são desniveladas, ou seja, existem alunos que já são proficientes na disciplina, provenientes de cursos técnicos no ensino médio ou cursaram previamente a disciplina em anos anteriores, enquanto outros nunca tiveram contato ou possuem uma grande dificuldade em acompanhar o andamento do curso. Com isso, é necessário por parte do professor definir um ponto comum para dar sequência e ritmo no andamento no curso com base na observação da turma. Entretanto, ao definir um ponto em comum para uma turma, da mesma forma que alguns alunos podem se sentir desmotivados por não ser fornecido desafio suficiente e outros perdem o interesse por não conseguir acompanhar o ritmo da turma.

As formas de avaliação também apresentam certas dificuldades. Durante uma prova ou em exercícios propostos, alunos recebem questões que são arbitrariamente rotuladas com um certo nível de dificuldade (quando o são). Entretanto, uma questão que seja fácil para um aluno, não necessariamente, é fácil para outro. Propor uma questão muito difícil para um aluno pode desmotivá-lo, por nem sequer conseguir começar a resolvê-la. O contrário, propor apenas questões fáceis, pode fazer com que o aluno se sinta entediado ou pense que não há mais necessidade de estudar.

Este trabalho busca tratar essas questões por criar um método e uma ferramenta (Code Training) que forneça aos alunos um ambiente para estudo e treinamento, com resposta imediata, utilizando elementos de jogos na forma de pareamento por habilidade. Com base nas respostas dos alunos, um nível de habilidade é associado a eles e um nível de dificuldade é associado à questão. Ao comparar os dois níveis, uma probabilidade de acerto é estimada e as questões são organizadas de forma que os alunos não se sintam nem frustrados nem entediados ao responderem os exercícios.

2 FUNDAMENTAÇÃO E TRABALHOS RELACIONADOS

Parte deste trabalho consiste em construir um método que identifica a chance de um determinado aluno em responder uma questão utilizando a Teoria de Resposta ao Item (TRI) utilizando como ferramentas para estimativa de habilidade e dificuldade das questões o *ELO Rating System*. Esta seção apresenta esses métodos e faz uma revisão de trabalhos relacionados assim como um breve revisão das ferramentas *online* de prática de programação e ambientes *online* de educação será apresentada.

2.1 Gamificação

Gamificação é vista como o processo de pensar em jogos e seus mecanismos para incentivar os usuários a resolverem problemas [18], tendo como base o ato de pensar como se estivessem em um jogo, interagindo com um sistema e respeitando a dinâmica de jogos, sem necessariamente estar jogando [5].

Nesse aspecto, elementos de jogos podem ser utilizados para estimular os alunos a estudar através de um ambiente virtual, visto que um conjunto de atividades podem ser entendidas como um desafio a ser completado e cada atividade ser comparada a uma fase de um jogo, inclusive aumentando o nível de dificuldade a medida que se progride nas tarefas [18].

Além disso, para que um usuário fique imerso na atividade, a gamificação utiliza frequentemente a chamada teoria do *Flow*, para

tentar manter o foco do usuário, prendendo sua atenção na atividade que está realizando para que não se desconcentre com o que acontece ao seu redor [5]. A teoria do *Flow*, representa o estado de emoção de um usuário em função da sua habilidade e da dificuldade proposta na atividade.

Se o nível de dificuldade de uma atividade é muito alto em relação à habilidade do usuário, seu nível de ansiedade fica alto, possivelmente conduzindo à frustração e consequente desmotivação. Por outro lado, se o nível de dificuldade da atividade é baixa demais em relação à sua habilidade, ele pode ficar entediado e acabar desistindo por considerar trivial demais. Com isso, a teoria do *Flow*, sugere equilibrar a experiência do usuário controlando os níveis de dificuldade em função do progresso de habilidade para mantê-lo constantemente engajado na atividade [18].

Ao combinar esses aspectos de jogos em um ambiente virtual de aprendizagem (AVA), é construído um cenário no qual os alunos são estimulados a participar, socializar com os colegas de turma e professores. Entretanto é preciso planejar bem a forma na qual as atividades serão apresentadas para que o conteúdo seja consumido de uma forma ativa pelo usuário, que os desafios sejam páreos para as suas habilidades.

2.2 Teoria de resposta ao item

A teoria de resposta ao item (TRI) consiste em modelos matemáticos que visam determinar a probabilidade de uma pessoa responder a uma questão corretamente de acordo com parâmetros relacionados ao item e as habilidades da pessoa. Dessa forma, quanto maior a habilidade de um indivíduo, maior a probabilidade dele acertar uma questão [1].

De acordo com [1] a TRI pode ser calculada de três formas diferentes e a equação 1 apresenta uma delas.

$$P(U_{ij} = 1|\theta_j) = \frac{1}{1 + e^{-D(\theta_j - b_i)}} \quad (1)$$

Nessa equação, i é um número que varia de 1 a n, o número de itens no teste. U_{ij} é uma variável dicotômica que assume os valores 1, quando o indivíduo j responde corretamente o item i, ou 0 quando o indivíduo j não responde corretamente o item i. O termo θ_j é o valor do traço latente de um indivíduo j. Já $P(U_{ij} = 1|\theta_j)$ é a probabilidade de um usuário j com habilidade θ responder corretamente à questão i. Já e é um número constante cujo valor é $2,718$, base natural dos logaritmos neperianos e D é um número constante cujo valor é $1,7$ para tornar a função o mais próximo de uma função normal. Por fim, b_i é a dificuldade relativa ao item.

Com este cálculo é possível determinar qual a probabilidade de um usuário, com certo nível de habilidade, acertar a uma questão com determinado nível de dificuldade.

2.3 Elo rating

O *Elo rating* é um método estatístico criado inicialmente com o intuito de calcular, para cada jogador, uma classificação numérica em partidas de xadrez. Esta classificação, geralmente entre os valores 0 e 3000, é atribuída a cada jogador e varia com o tempo, de acordo com o seu desempenho [9].

Ainda de acordo com [9], em uma partida entre dois jogadores A e B, a pontuação esperada para o jogador A é dada por:

$$E_a = \frac{1}{1 + 10^{-(R_a - R_b)/400}} \quad (2)$$

Onde E_a é o valor da pontuação e R_a e R_b são, respectivamente, as classificações dos jogadores A e B. Assim, a fórmula para atualizar a classificação do jogador A é dada por:

$$R'_a = R_a + K(S_a - E_a) \quad (3)$$

Onde, R'_a é o valor atualizado de sua classificação, R_a é a classificação antes do jogo, K, é uma constante que em geral, possui o valor 16, S_a é a pontuação obtida na partida e E_a é o valor esperado de pontuação, obtido conforme Equação 2.

Neste trabalho, considera-se que um usuário está competindo com uma questão, ou seja, a pontuação do adversário é vista como o nível de dificuldade da questão, e assim, a pontuação dos usuários e das questões podem ser calculados de acordo com equação 3.

2.4 Algoritmos e linguagens de programação

O princípio básico da computação é o desenvolvimento de algoritmos. Um algoritmo é um conjunto finito de instruções para alcançar determinado objetivo. Desenvolver um algoritmo é estabelecer uma norma de ações simples, sem ambiguidade e ordenadas, que em um tempo finito produz uma solução bem definida para um problema. Seguindo essa norma, sempre que executado, sobre as mesmas condições, um algoritmo produz o mesmo resultado [7, 4].

Para que se possa escrever um algoritmo para uso prático em um computador é necessário utilizar-se de um conjunto de notações formais para descrever as ações ou operações que serão realizadas pela máquina [10]. Esse conjunto de notações é composto por regras sintáticas, que dizem respeito à forma de escrita e regras semânticas que se referem ao significado. Seguindo essas regras, o computador é capaz de executar o algoritmo e gerar uma saída [11].

2.5 Teste de software

Testar um *software* é avaliar se um código ou modelo, seja de um sistema completo ou parte dele, se comporta da maneira esperada e identifica quais os motivos caso sua execução não atinja os resultados esperados [6]. Existem vários níveis de teste de *software*, como os testes unitários, de integração, de sistema, de aceitação e de regressão[15].

Neste trabalho, o propósito é verificar se um algoritmo simples, implementado em uma linguagem de programação, apresenta defeitos ou erros. O modelo de teste a ser aplicado é uma adaptação do teste unitário, o qual verifica de forma isolada se aquele algoritmo atende ou não ao objetivo especificado em uma questão. Para tal, além da compilação ou interpretação que vai capturar erros de compilação ou interpretação (devido a defeitos de sintaxe) ou erros de execução. As questões serão construídas com alguns dados válidos e inválidos de entrada e feitas algumas afirmativas sobre os valores finais das variáveis e retornos de funções. As negativas em tais afirmações serão capturadas como defeitos no código submetido pelo aluno.

2.6 Educação a distância e ambientes virtuais de aprendizagem

Educação a distância é uma forma de aprendizado que ocorre em um lugar diferente do local do ensino. Exige modelos diferentes de comunicação geralmente por meio da tecnologia de informação e tem por vantagens melhorar o sistema educacional, proporcionando aos alunos a chance de ampliarem suas aptidões, reduzir custos educacionais, direcionar a informação para público-alvos específicos, nivelar as desigualdades dos grupos etários [13].

Os AVAs são ferramentas da tecnologia de comunicação e informação que se utilizam do meio eletrônico para disseminar informação e permitir uma maior interação entre discentes e docentes [14]. Para que o aprendizado seja controlado e o conteúdo disponibilizado corretamente, os ambientes devem oferecer ferramentas como controles de acesso, tempo e progresso dos alunos, avaliação, comunicação, ajuda e manutenção [12]. Com isso, os AVA's utilizam a internet para aproximar os alunos dos professores e disponibilizarem o conteúdo e materiais de estudo de forma acessível a qualquer lugar e horário.

2.7 Trabalhos relacionados

Os autores de [16] desenvolveram uma ferramenta integrada a um AVA, se baseando no Sistema Personalizado de Ensino. Além disso, realizaram uma análise de resultados de sua implantação em uma instituição de ensino e constataram que com o uso da ferramenta houve um aumento significativo na aprovação dos alunos e uma redução no abandono e desistência da disciplina. Como trabalhos futuros, os autores indicam a aplicação dessa metodologia em outras disciplinas de programação; o uso dessa e de outras ferramentas semelhantes em outras instituições de ensino para que seja feita uma análise mais profunda e a implementação de novas funcionalidades na ferramenta, como correção automática das submissões.

Em [8] os autores desenvolveram uma ferramenta que ajuda a identificar as dificuldades de alunos iniciantes em programação e fornece material didático de suporte para ensino da linguagem de programação C. Tal ferramenta funciona analisando *logs* de leitura do material feita pelos alunos e erros de compilação nos exercícios que os alunos respondem. Ao localizar um erro de sintaxe a ferramenta recomenda um trecho do material para leitura. Com isso, concluem seus resultados ressaltando que a ferramenta utiliza os dados coletados dos próprios usuários para melhorar a sua utilização, e assim, aprimorar o aprendizado dos alunos.

Já em [2] foi desenvolvido um ambiente de apoio ao ensino de lógica de programação que combina os elementos da disciplina com jogos eletrônicos. Além disso, esse ambiente utiliza técnicas de mineração de dados para realizar um monitoramento *online* da etapas concluídas por cada aluno. Um teste de aceitação foi realizado, e com dados colhidos através de questionários, os autores concluem que o ambiente foi bem aceito pelos alunos que se sentiram motivados a continuas seus estudos em lógica de programação.

3 Método

Este trabalho é desenvolvido de forma a capturar o processo de aprendizado seguido pelo aluno. Para tal, é formalizado um modelo do processo de resolução de questões para levantar quais dados serão colhidos. Uma implementação de ambiente de resolução de questões chamado Code Training é feita e os dados de interesse são colhidos. O Processo de Resolução se inicia no momento em que um usuário inicia a solução de uma questão e termina com a avaliação de sua solução.

As Medidas são observações realizadas durante todo o Processo de Resolução. Elas servem para que o sistema possa, posteriormente, definir métricas sobre as questões e sobre o progresso dos usuários.

Inicialmente, as medidas realizadas durante o Processo de Resolução são o tempo para solução (TPS), quantidade de tentativas (QDT), quantidade de enganos cometidos (QEC), quantidade de respostas corretas (QRC), quantidade de processos de solução interrompidos (QPI), a lista de códigos dos enganos cometidos (LEC) e as soluções, tanto intermediárias (SI), que são as soluções em cada tentativa, quanto a solução final (SF). As datas de início (DI) e fim (DF) da solução, o valor esperado (E_a), valor de medidas (P_m) e pontuação final (P_f) também são registrados.

4 Ambiente de treinamento Code Training

O Code Training apresenta uma interface com usuário que é acessada através de um navegador de internet, em qualquer sistema operacional, inclusive em dispositivos móveis. Uma conta deverá ser criada, com identificação e senha, que irá garantir o acesso seguro do usuário à plataforma. Através da sua conta, o usuário poderá ter acesso aos exercícios propostos e ao acompanhamento das questões já respondidas. Em um primeiro momento, a plataforma irá oferecer apenas atividades em uma única linguagem de programação, o JavaScript.

Cada usuário vê a lista de atividades propostas, previamente cadastradas no Code Training e envia a solução para cada uma delas. Todo o processo é registrado e medido. Uma avaliação sobre a execução de um código submetido será fornecida ao usuário, que poderá seguir ao próximo exercício ou refazer uma solução, caso algum erro ou falha tenham sido encontrados.

O Code Training foi implantado no Heroku[1], uma plataforma na nuvem que fornece *conteiners* para aplicações como serviço. A plataforma pode ser acessada `https://guarded-refuge-80906.herokuapp.com`, onde é possível se cadastrar e responder as questões.

5 Coleta e análise de dados reais

Para que fosse possível coletar dados reais, 18 questões foram disponibilizadas na plataforma e alunos das disciplinas Laboratório de Programação Web (segundo período, segunda disciplina com programação mas com JavaScript), DCC192 - Laboratório de Programação de Sistemas Web (sétimo período, alunos já com conhecimento em várias linguagens) e Seminários em Computação (eletiva, períodos mistos e altamente focada em JavaScript), foram convidados a utilizar a plataforma no período de 22/06/2018 à 03/07/2018. Nesse intervalo, 28 usuários se cadastraram, 158 respostas corretas e 513 tentativas foram registradas.

Os dados até a data são visualizados nas Figuras 1 e 2 nas quais uma breve análise dos dados coletados mostra que a medida em que os usuários respondem as questões, suas habilidades começam a ser traçadas.

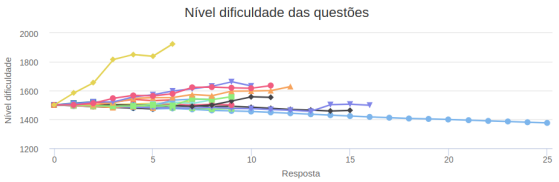

Figura 1: Gráfico dos dados coletados, mostrados por questão.

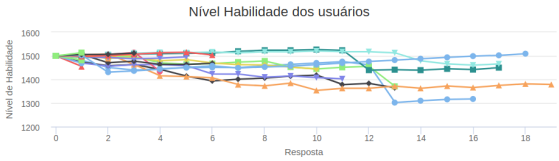

Figura 2: Gráfico dos dados coletados, mostrados por usuário.

Tomando como exemplo dois usuários, a Figura 3 mostra o seu desempenho após responderem mais de 15 questões corretas. A linha na cor azul representa um usuário que inicialmente teve certo grau de dificuldade, mas a medida que foi respondendo mais questões, sua habilidade aumentou. Já a linha na cor laranja, representa um cenário diferente, um usuário que inicialmente conseguiu aumentar um pouco sua habilidade, porém começou a sentir mais dificuldade ao responder o restante das questões.

Como pode ser visto na Figura 4, a questão representada pela linha azul, teve sua dificuldade reduzida quase que constantemente (a questão é apenas uma declaração e iniciação de variáveis). Já a questão representada pela cor lilás, se manteve, aproximadamente com o nível inicial por ter uma mesma frequência de acertos e erros (questão sobre funções, poucas operações internas com os parâmetros). Já a questão representada em amarelo, se destacou

[1] https://www.heroku.com

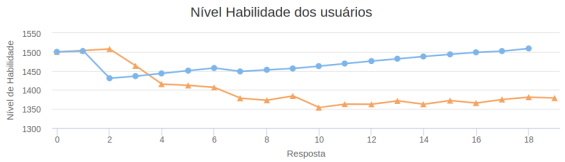

Figura 3: Gráfico que compara o rendimento de dois usuários.

por subir seu nível de dificuldade muito rapidamente pelo baixo número de respostas corretas (cálculo para geração de uma matriz com soma constante em todas linhas, colunas e diagonais conhecida como quadrado mágico).

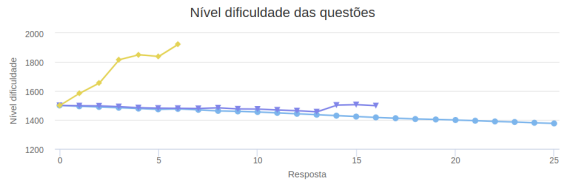

Figura 4: Gráfico que compara o comportamento do nível de dificuldade de algumas questões.

Através desses dados, é possível observar que algumas questões, que exigem muito tempo ou que tiveram várias tentativas de resposta, aumentaram seu nível de dificuldade, enquanto questões que foram resolvidas rapidamente e em poucas tentativas, têm o seu nível de dificuldade reduzido. Dessa forma, é possível ver que na listagem de questões, as informações de dificuldade para cada usuário começam a se estabilizar, ou seja, as questões são classificadas de acordo com o nível mais adequado para cada usuário.

6 Conclusões

As dificuldades encontradas por alunos ao iniciar os estudos em programação, como pensamento lógico e uma abstração dos problemas, distanciamento entre professores e alunos, desnivelamento da turma e dificuldade na forma de avaliação, fazem com que se observe um alto índice de reprovação nas disciplinas iniciais de programação. Entre as propostas para amenizar esses problemas, está o uso ambientes de estudo especializados em programação em conjunto com o ensino tradicional.

Neste trabalho, foi feito um levantamento de conceitos e ferramentas que são utilizadas para aprendizagem de linguagens de programação e foi proposto um modelo para capturar a interação dos alunos com as questões. O método realiza um pareamento automático de questões com probabilidade de acerto dentro de uma faixa de interesse para que o aluno não se sinta, nem frustrado, nem entediado. Foi implementado um ambiente chamado Code Training para evidenciar esta captura e pareamento e um estudo de caso foi conduzido.

Observou-se que à medida que os usuários interagem com a plataforma e tentam resolver as questões, a habilidade do aluno e a dificuldade da questão convergem para valores específicos. Utilizando esta informação, o ambiente se adapta à realidade individual de cada aluno, fazendo com que se sintam mais confortáveis em continuar o seu progresso em seu próprio ritmo.

Como sequência direta deste trabalho, espera-se acrescentar outras linguagens de programação para estudo e realizar o pareamento para cada linguagem. Na instituição deste estudo, a primeira linguagem de programação a ser ensinada é a linguagem ANSI C e tê-la implementada na ferramenta permitirá utilizar o sistema em um grupo de 1000 alunos. Isso tornaria possível inclusive montar grupos de controle e observar se a forma como o conteúdo é exposto tem impacto no rendimento dos mesmos.

Referências

[1] J. W. C. ALEXANDRE, D. d. Andrade, A. d. Vasconcelos, A. d. Araújo, and M. J. Batista. Análise do número de categorias da escala de likert aplicada à gestão pela qualidade total através da teoria da resposta ao item. *Encontro Nacional De Engenharia De Produçao*, 23:1–20, 2003.

[2] B. Barbosa, S. Silva, and B. Sousa. Tri-logic proposta lúdica gamificada para o ensino e aprendizagem da lógica de programação com o uso da mineração de dados como ferramenta de auxílio ao professor. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 28, page 1754, 2017.

[3] T. Beaubouef and J. Mason. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin*, 37(2):103–106, 2005.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Algoritmos: teoria e prática. *Editora Campus*, 2, 2002.

[5] A. R. L. da Silva, A. H. Catapan, C. H. da Silva, E. B. Reategui, F. J. Spanhol, I. F. Golfetto, J. B. Diana, L. R. G. Alves, L. M. Fadel, L. H. Lindner, et al. *Gamificação na Educação*. Pimenta Cultural, 2014.

[6] M. Delamaro, M. Jino, and J. Maldonado. *Introdução ao teste de software*. Elsevier Brasil, 2017.

[7] A. L. V. Forbellone and H. F. Eberspächer. *Lógica de programação: a construção de algoritmos e estruturas de dados*, volume 3. Makron Books, 1993.

[8] X. Fu, A. Shimada, H. Ogata, Y. Taniguchi, and D. Suehiro. Real-time learning analytics for c programming language courses. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, pages 280–288. ACM, 2017.

[9] M. E. Glickman and A. C. Jones. Rating the chess rating system. *CHANCE-BERLIN THEN NEW YORK-*, 12:21–28, 1999.

[10] R. R. Gudwin. Linguagens de programação. *Campinas: DCA/FEEC/UNICAMP*, 1997.

[11] K. C. Louden et al. *Programming languages: principles and practices*. Cengage Learning, 2011.

[12] C. Milligan. Delivering staff and professional development using virtual learning environments. *The Role of Virtual Learning Environments in the Online Delivery of Staff Development. Institute for Computer Based Learning, Heriot-Watt University, Riccarton, Edinburgh EH14-4AS*, 1999.

[13] M. Moore, G. Kearsley, and E. a Distância. Uma visão integrada. *Tradução por Roberto Galman. São Paulo: Thomson Learning*, 2007.

[14] A. T. C. Pereira, V. Schmitt, and M. Dias. Ambientes virtuais de aprendizagem. *AVA-Ambientes Virtuais de Aprendizagem em Diferentes Contextos. Rio de Janeiro: Editora Ciência Moderna Ltda*, pages 4–22, 2007.

[15] A. R. C. d. Rocha, J. C. Maldonado, K. C. Weber, et al. Qualidade de software: teoria e prática. *São Paulo: Prenttice Hall*, 2001.

[16] P. S. Rocha, B. Ferreira, D. Monteiro, D. d. S. C. Nunes, and H. C. do Nascimento Góes. Ensino e aprendizagem de programação: análise da aplicação de proposta metodológica baseada no sistema personalizado de ensino. *RENOTE*, 8(3), 2010.

[17] C. Watson and F. W. Li. Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ITiCSE '14, pages 39–44, New York, NY, USA, 2014. ACM.

[18] G. Zichermann and C. Cunningham. *Gamification by design: Implementing game mechanics in web and mobile apps*. "O'Reilly Media, Inc.", 2011.