Uma Análise das Respostas Abertas da Avaliação Docente em Disciplinas de Ensino de Lógica e Fundamentos de Programação

Emanuel F. Coutinho[1,2,4], **Leonardo O. Moreira**[1,3,4], **Maurício Moreira Neto**[1,4]

[1]IBITURUNA - Grupo de Estudos em Computação em Nuvem
[2] Universidade Federal do Ceará (UFC) – Quixadá, Brazil
[3] Instituto Universidade Virtual (UFC Virtual)
[4] Universidade Federal do Ceará (UFC)

emanuel@ufc.br, leoomoreira@virtual.ufc.br, maumneto@alu.ufc.br

Abstract. *A beginner programming discipline usually includes logic concepts and programming fundamentals, challenging both learners and teachers. Teaching, learning and evaluation processes in programming have been the subject of several discussions in the literature. In addition to the technical aspects required for programming learning, there are also social and human aspects that impact on learning, such as communication, motivation and profile. This article aims to analyze open answers obtained from the evaluation of teaching by students for programming classes in an undergraduate course. As a consequence of this analysis, a conceptual map was generated identifying large categories of student feedback.*

Resumo. *Uma disciplina de programação para iniciantes normalmente inclui conceitos de lógica e fundamentos de programação, sendo um desafio tanto para discentes quanto para docentes. Processos de ensino, aprendizagem e avaliação em programação têm sido motivo de diversas discussões na literatura. Além dos aspectos técnicos necessários ao aprendizado de programação, há também aspectos sociais e humanos que impactam no aprendizado, como comunicação, motivação e perfil. Este artigo visa analisar respostas abertas obtidas da avaliação docente realizada pelos alunos para turmas de programação em um curso de graduação. Como consequência dessa análise, um mapa conceitual foi gerado, identificando grandes categorias de comentários dos alunos.*

1. Introdução

O ensino de qualquer disciplina em áreas técnicas constitui um conjunto de atividades desafiantes para o docente, tanto pelo constante surgimento de novas tecnologias e ferramentas, como pela busca por estratégias na tentativa de se alinhar às novas práticas de aquisição de conhecimento adotadas pelos aprendizes atuais [Coutinho et al. 2018]. Em particular, o ensino de lógica de programação e consequentemente a codificação são tarefas complexas que requerem muito esforço por parte dos docentes. Além disso, um desafio é auxiliar o aluno a associar a teoria ministrada com a prática, pois os alunos geralmente ingressam nos cursos de graduação com os mais variados perfis.

Disciplinas com a intenção de adequar teoria com prática são normalmente muito difíceis de se conduzir devido à necessidade de se tratar fatores técnicos, como linguagens de programação e ferramentas, e fatores humanos como comunicação e gestão, tudo

ao mesmo tempo [Coutinho et al. 2016]. Além disso, a tarefa de motivar os alunos e os próprios professores não é uma atividade fácil. Isto se verifica, em especial, em disciplinas de primeiro ano de cursos de graduação, pois os conteúdos a serem abordados são, na maioria das vezes, novidade para os alunos, constituindo, assim, um desafio para eles. A exposição a conteúdos inteiramente novos normalmente resultam em um rendimento acadêmico aquém daquele que o estudante estava acostumado a apresentar antes de ingressar em uma universidade. Esse baixo rendimento pode levar o aluno a se sentir desmotivado.

Os processos de ensino, aprendizagem e avaliação de programação têm sido motivo de diversas discussões nas áreas de Educação e Computação. Muito disso se deve às dificuldades do domínio da programação [de Oliveira et al. 2017]. O aprendizado em linguagens de programação necessita de conhecimentos prévios em lógica e matemática [Ribas et al. 2016]. Mesmo que o raciocínio lógico não seja trabalhado exclusivamente em disciplinas de Matemática, é nessa área que geralmente esses conhecimentos são mais explorados nas escolas. Nesse sentido, poderíamos visualizar que cada vez mais os estudantes chegam com menor conhecimento prévio sobre matemática e lógica nos cursos superiores.

Além dos aspectos técnicos necessários ao aprendizado de programação, há também aspectos mais sociais e humanos. Em um trabalho realizado por Mora e Giraffa (2013) foi constatado que a maioria dos alunos possuem dificuldade em compreender o enunciado e o que é solicitado, mas ao mesmo tempo reconhecem que não fazem perguntas para resolução de dúvidas [Mora e Giraffa 2013]. Outra dificuldade comumente discutida é como ensinar lógica de programação de maneira que o aluno se motive a aprender, ao mesmo tempo em que eles se preparam para a utilização de ferramentas de codificação mais avançadas e aplicar os conceitos de lógica de programação na prática [Marinho et al. 2016][Coutinho et al. 2018]. Por fim, há também a aptidão do aluno para trabalhar com programação, que não necessariamente existe em todos, ou seja, o perfil para programar.

Considerando o exposto, o presente artigo tem como objetivo analisar respostas abertas obtidas a partir de avaliações docentes realizadas pelos alunos para turmas de Programação 1, do curso de graduação em Sistemas e Mídias Digitais (SMD) da Universidade Federal do Ceará. Esta disciplina tem como objetivo possibilitar ao aluno aprender conceitos de lógica de programação para o desenvolvimento de soluções de problemas, através do uso das estruturas básicas comuns às linguagens de programação, atividade importante para diversas áreas do conhecimento e para o desenvolvimento de aplicações e serviços. O restante do artigo está estruturado da seguinte maneira: na Seção 2 apresentaremos um breve embasamento teórico sobre lógica de programação; na Seção 3 apresentaremos a metodologia aplicada ao trabalho; na Seção 4 os resultados obtidos e análises serão expostos; e por fim, a Seção 5 apresentaremos as considerações finais deste trabalho.

2. Embasamento Teórico

Muitas disciplinas em cursos de graduação possuem como parte de suas ementas disciplinas de lógica de programação e fundamentos de programação. Tais disciplinas se fundamentam em conceitos de raciocínio lógico e matemática, sendo essenciais para um bom

aprendizado. Exemplos de assuntos comumente ensinados em lógica de programação são estruturas de atribuição, condição e seleção (ou repetição). Além disso, alguns mecanismos matemáticos também são ministrados, tais como operadores aritméticos e relacionais, além de estruturas de dados como vetores e matrizes.

Vários trabalhos na literatura discutem diferentes formas e relatos do ensino de lógica de programação e codificação em semestres iniciais de cursos de graduação [Ferreira et al. 2016][Marinho et al. 2016][Coutinho et al. 2017] [Coutinho et al. 2018]. Normalmente sempre relatam problemas associados ao aprendizado.

Em Computação, normalmente a lógica é aplicada a todas suas subáreas [Chang e Lee 1973], incluindo a construção de software e hardware. Por exemplo, para o projeto e construção de um teclado, seus circuitos integrados utilizarão conceitos da lógica, implementados através de portas lógicas, que permitem ou não a passagem de pulsos elétricos. Na construção do software, por meio do raciocínio lógico se projetam algoritmos, que, por sua vez, propiciam a geração de programas capazes de solucionar problemas.

Um conceito básico na programação é o algoritmo, que pode ser definido como uma sequência lógica de instruções a serem seguidas para a resolução de um problema ou execução de uma tarefa [Cormen et al. 2001]. Muitas vezes, para se escrever um algoritmo é necessário tomar decisões, definir estratégias e comparar dados. Todas essas situações se utilizam de fundamentos de raciocínio lógico e, muitas vezes, matemático.

Um programa de computador pode ser considerado como um conjunto de instruções que informam à máquina o que deve ser feito [Farrell 2015]. Entretanto, não se pode instruir a máquina de qualquer maneira. É necessário fazer isto de maneira precisa, evitando ambiguidades. Tal maneira é a linguagem de programação, que pode se basear em diferentes paradigmas, onde cada um possui características particulares e específicas [Sebesta 2012]. Exemplos de linguagens de programação são Java, C++, Python e Ruby. Um programa de computador é escrito através de sentenças nestas linguagens, que são formadas por uma série de instruções ou comandos. Ainda que tais comandos possam diferir entre linguagens, o conhecimento de lógica de programação é imprescindível para a construção de programas funcionais e corretos, independente da linguagem de programação utilizada. O raciocínio lógico é, portanto, uma das bases da programação.

Apesar de muitas linguagens de programação serem consideradas de alto nível, ou seja, próximas da linguagem natural e humana, este objetivo ainda está longe de ser alcançado [Puga 2003]. Isto torna o desafio em ensinar e aprender uma linguagem de programação ainda maior. E a diversidade de linguagens de programação é cada vez maior, com tecnologias e sintaxes variadas. Assim, o pleno entendimento dos fundamentos da lógica de programação, base para as linguagens de programação, se torna essencial.

Por fim, o ensino tradicional em programação apresenta duas situações: alunos habituados a serem indivíduos passivos dentro do ambiente escolar e alunos ou professores limitados pelo tempo [Amaral et al. 2017]. Nesse caso, é preciso emancipar o aluno de maneira que ele possa “prender a aprender”e valide seus próprios saberes [Wang e Prado 2015].

3. Metodologia

O objetivo desta pesquisa é analisar os comentários das questões abertas dos alunos, realizados durante as avaliações docentes que ocorrem semestralmente. Essas avaliações são compostas por questões objetivas relacionadas a diversos critérios, com notas de 1 a 5, e as questões abertas subjetivas, onde o aluno pode escrever o que desejar. Todas as avaliações foram cedidas com a permissão dos professores e todos os comentários dos alunos são anônimos. A partir dos comentários dos alunos, alguns conceitos foram identificados, para posterior análise. Esses conceitos foram destacados em um mapa conceitual. Após a elaboração do mapa conceitual, algumas análises foram feitas, com os assuntos mais comentados pelos alunos, possibilitando assim identificar aspectos que se destacaram e possíveis propostas de melhoria para novas edições da disciplina. Os participantes do estudo foram alunos da disciplina de Programação 1, de 13 turmas, de professores variados, do período de 2016 a 2018. A pesquisa é de caráter descritivo, que segundo Gil [Gil 2008], pesquisas descritivas possuem como objetivo a descrição das características de uma população ou fenômeno, ou o estabelecimento de relações entre variáveis. Também existe a possibilidade de algumas pesquisas descritivas contribuam mais para proporcionar uma nova visão de um problema. Quanto à análise dos dados, esta pesquisa será qualitativa.

Para a análise dos comentários dos alunos obtidos a partir das avaliações de cada professor, utilizaremos a análise de conteúdo, dentro da proposta de Bardin [Bardin 2011]. Segundo Bardin (2011), as diferentes fases da análise de conteúdo organizam-se em torno de três itens: (i) pré-análise; (ii) exploração do material; e (iii) tratamento dos resultados, inferência e interpretação. A fase de pré-análise é a organização propriamente dita, correspondendo a um momento de intuição, com o objetivo de tornar operacional e sistematizar ideias iniciais. Normalmente essa fase possui três missões: seleção dos documentos a serem submetidos à análise, formulação das hipóteses e objetivos e elaboração de indicadores que fundamentem a interpretação final. Na fase de exploração se realiza a análise propriamente dita (administração sistemática das decisões tomadas), consistindo essencialmente de operações de codificação, desconto ou enumeração. Por fim, na fase de tratamento, os resultados brutos são tratados de maneira a se tornarem significativos e válidos. Adicionalmente, operações estatísticas possibilitam o estabelecimento de resultados, diagramas, figuras e modelos, os quais consolidam e destacam as informações fornecidas pela análise.

4. Análise e Discussão dos Dados

A Figura 1 exibe um mapa conceitual resultante dos comentários apresentados. Os conceitos gerados em torno das questões abertas foram: Mensagens positivas, Didática, Metodologia, Avaliações, Infraestrutura, Alunos e Extras. A maioria dos conceitos eram relacionados de certa forma com aspectos comuns da gestão acadêmica de uma disciplina de um curso de graduação. Um conceito adicional denominado extra foi criado só para destacar aspectos particulares, que não se encaixaram nos demais conceitos. Ressalta-se que a quantidade de comentários foi bem maior que a quantidade de conceitos gerados.

Além dos sete conceitos principais identificados, percebeu-se que a maioria dos demais conceitos gerados pertenceram à Metodologia e Avaliação. Se considerarmos a quantidade de comentários dos alunos, adicionamos o conceito Didática ao grupo dos três

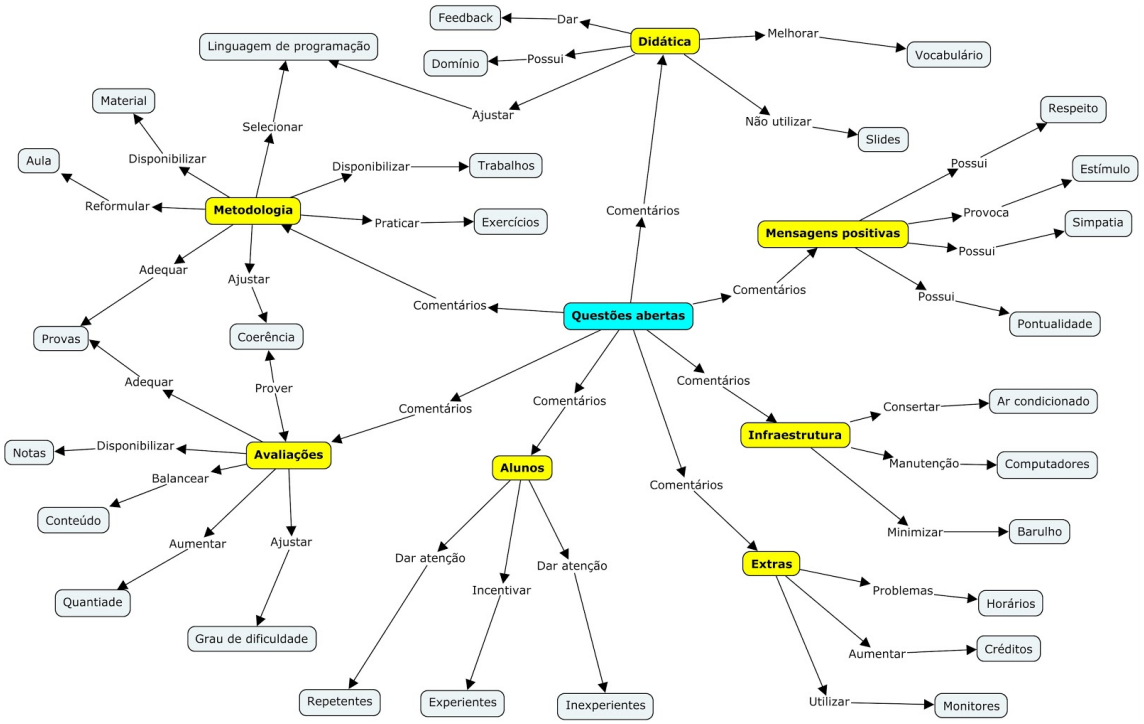

Figura 1. Mapa conceitual com comentários dos alunos agrupados

conceitos que mais se destacaram nos comentários dos alunos. Em geral, nas avaliações docentes esses três aspectos são bastante comentados pelos alunos e também são muitas vezes discutidos em grupos de coordenadores, unidades curriculares, núcleos docentes estruturantes e grupos relacionados, pois impactam diretamente no aprendizado.

4.1. Mensagens Positivas

Alguns dos comentários trataram de mensagens positivas em relação à postura do professor em sala de aula. Aspetos de pontualidade, respeito e simpatia foram citados. O estímulo do professor em sala também foi um ponto forte. Um aluno comentou o seguinte: "*Didática perfeita. Professora prepra, detalha, exemplifica, torna lúdica a explicação ! PARABÉNS !*". Isso reforça que comportamentos dos professores em relação aos alunos podem fazer a diferença.

4.2. Didática

Didática sempre é um aspecto sensível, pois depende de diversos fatores, tais como: experiência do professor, conhecimento técnico, nível dos alunos e flexibilidade das grades curriculares. Diversos alunos comentaram sobre a didática dos professores de programação, seja por comentários de melhorias, seja por observações de más práticas. Pontos fortes também foram destacados, como "*Domínio e clareza ao repassar o conteúdo para a turma ...*". Entretanto a maioria questionou a forma em si da condução das aulas. Por exemplo, a utilização de slides, como "*Não utilizar slides/sempre abordar o conteúdo praticando ...*". Slides em aulas de programação com um caráter prático devem ser planejados com cuidado, já que boa parte das aulas são práticas. O vocabulário utilizado por professores também foi comentado, como "*Falar com um vocabulário menos programador ...*". Nesse sentido, não ser tão técnico nas palavras para quem está

iniciando auxilia no aprendizado. Por fim, necessidades de organizar melhor as aulas e tanto no formato quanto no conteúdo, apontado em "*Reestruturar o formato e didática da aula.*".

4.3. Metodologia

Por metodologia consideramos neste trabalho a organização em sala de aula, desde o ensino propriamente dito, exercícios, práticas, ferramentas, utilização dos bolsistas e avaliações. De maneira geral, há uma deficiência constante no ensino de programação, reforçado por "*Acredito que o modelo atual do ensino de programação com um todo seja insuficiente ...*". Isto ocorre em diversos cursos de graduação, tendo muitas abordagens sendo propostas e experimentadas.

Mas não necessariamente todas as turmas possuem uma metodologia ruim. Às vezes é algo pontual que não foi adequado, é a falta de planejamento, ou o desconhecimento do perfil dos alunos. Nesse ponto, um aluno comentou: "*Muito boa a metodologia. Deveria ser aplicada em todas as outras dessa cadeira, principalmente para os alunos novatos.*". Para quem é iniciante em programação, normalmente o impacto é grande caso o aluno não tenha uma base lógica e matemática, reforçado por "*Ensinar melhor para quem nunca viu programação ...*". E a prática é fundamental para a melhor fixação dos conceitos, implicando em mais atividades, destacado em "*Mais atividades em sala, com acompanhamento ...*".

Em aulas de fundamentos de programação é comum utilizar apenas uma linguagem de programação, devido a vários fatores: pouco tempo para aprender a teoria de lógica de programação, necessidade de aprender a sintaxe da linguagem, grande esforço para correção, dúvidas e acompanhamento dos alunos. No caso de se utilizar mais de uma linguagem na disciplina, o que ocorreu em algumas turmas analisadas, todos esses fatores se multiplicam, ampliando a complexidade. Assim, pode-se dizer que é bastante adequado comentários do tipo "*Usar apenas uma linguagem de programação ...*".

Para aulas de programação, material de apoio disponibilizado normalmente são sites, referências da sintaxe da linguagem de programação, ferramentas e livros com a teoria. Apesar disso, muitos professores não utilizam muito tais recursos, o que para alunos iniciantes pode ser prejudicial, provocando comentários como "*Disponibilizar material previamente para desenvolvimento de alunos, pois o material de estudo era disponibilizado só depois das aulas isso dificultou muito o aprendizado.*".

4.4. Avaliações

As avaliações de maneira geral são sempre um ponto sensível, pois para o ensino de programação, o ideal é programar. Entretanto existem avaliações escritas, por meio de jogos, com trabalhos e práticas. Além disso, o impacto de ter que realizar avaliações diante da máquina, codificando exercícios solicitados, juntamente com o tempo, às vezes se torna uma estratégia que não mede o aprendizado, como comentado em "*... elaboração de provas que condizem com o conteúdo dado e com o tempo ofertado ...*".

A quantidade e variedade de avaliações é um comentário comum: "*... como há muito conteúdo na disciplina, defendo a existência de mais avaliações ou atividades valendo nota ...*". A quantidade de avaliações muitas vezes depende da carga horária

e esforço dispendido na disciplina, que normalmente possui muitos alunos. Outro comentário comum é o nível das avaliações, que muitas vezes reflete a didática do professor e o esforço de estudo do aluno, "*... também acho que ela deveria ter um grau de dificuldade crescente, com questões coerentes entre si e com o que será cobrado na prova.*".

4.5. Infraestrutura

Por infraestrutura identificamos elementos do ambiente e sala de aula. A temperatura, ressaltada pelos constantes problemas com ar condicionado foi destaque. O seguinte comentário foi identificado: "*... por todo o semestre o ar condicionado fez muito barulho, atrapalhando na concentração e exigindo um esforço maior do professor para dar o conteúdo de uma forma clara.*". Nesse aspecto, a acústica das salas nem sempre colaborou, pois tanto o ruído do ar condicionado quanto das demais salas atrapalhava a aula, prejudicando a concentração. Por fim, alguns computadores eventualmente davam problemas, além de, em geral para as turmas de novatos com cerca de trinta alunos, não serem suficientes para todos os alunos das aulas.

4.6. Alunos

Em relação aos alunos, algumas observações foram relacionadas à dificuldade na aprovação nas disciplinas de programação. Acontece com frequência turmas dedicadas aos alunos que reprovaram a disciplina, mas mesmo já a maioria tendo visto o conteúdo e praticado, ainda há lacunas a serem preenchidas tanto na metodologia quanto na didática. Um comentário foi "*Posso falar sobre o pouco que participei percebi que por ser uma turma de repetentes o professor considerava que os alunos já sabiam como trabalhar a matéria, mas muitos não tinham muito conhecimento, ou no meu caso, não tinham nenhum contato com o programa, tendo ainda mais dificuldade para entender*". Esse comentário coincide com a necessidade de um maior acompanhamento dos alunos, que pode ser tanto para repetentes quanto para novatos, como em "*Atenção para os iniciantes em programação*".

Outra observação foi relatada como uma dificuldade dos alunos devido ao formato das avaliações, que alternava entre papel, codificação e trabalhos em equipe: "*As avaliações dessa cadeira sempre foram de grande dificuldade até para os mais experientes em programação, seria interessante o professor trazer uma avaliação do nível das atividades trabalhadas em sala de modo que haja um crescimento linear de dificuldade e não exponencial*".

4.7. Extras

Alguns comentários foram mais específicos, não se encaixando nas categorias identificadas. Em relação à carga horária da disciplina, que é atualmente de 4 créditos (64 horas/aula) muitas vezes devido a projetos integrados e a atividades em comum com outras disciplinas, necessita de mais tempo. Um comentário nesse ponto foi "*A disciplina precisaria ter 6 créditos ...*". Caso a disciplina fosse isolada das demais e de projetos, a carga horária de 4 créditos seria suficiente. Entretanto pela natureza interdisciplinar do curso, uma reorganização das disciplinas em conjunto talvez fosse necessária.

Outro aspecto bastante comentado foi a utilização dos monitores da disciplina, normalmente mais de um. "*Aproveitar mais os monitores da cadeira, sendo que é a única*

cadeira que tem monitores efetivos ..."foi destacado por um aluno, o que dá a entender que os alunos veem a importância dos monitores para o aprendizado de programação.

Por fim, alguns comentários foram em relação a problemas de horário. Em geral alunos do turno da noite trabalham durante o dia, e há certa dificuldade em compatibilizar horários de disciplinas nos turnos da manhã e tarde. Além disso, às vezes alunos abandonam disciplinas por motivos de trabalho, seja pela sobrecarga de atividades, seja por pura incompatibilidade de horários. "*Tive problemas com os horários e problemas pessoais. Ia trancar a cadeira mas acabei perdendo o prazo*".

5. Conclusão

Este trabalho procurou apresentar uma análise sobre os comentários dos alunos nas questões abertas, realizados durante as avaliações docentes, em edições da disciplina de Programação 1, uma disciplina inicial no estudo de lógica e fundamentos de programação.

Atualmente, o foco da educação ainda está no professor, que detém a informação e possui a responsabilidade sobre a aprendizagem do aluno [Lopes e Kopsch 2018]. Esse aspecto é um desafio para os docentes, pois existem diversos aspectos que devem ser tratados, tais como metodologias, didática, infraestrutura e tecnologias. Em uma disciplina de programação, com caráter prático e necessidade de tecnologias, esse desafio ainda é maior.

Disciplinas de programação devem trabalhar raciocínio lógico, abstração e sintaxe específica em uma linguagem de programação [Meira et al. 2016][Zanetti e Oliveira 2015]. Alguns dos resultados apontados neste trabalho e destacados no mapa mental foram diretamente relacionados com essa afirmação, e são metodologia, didática e avaliação. Esses três conceitos são em geral dificuldades enfrentadas pelos professores ao lecionarem disciplinas de programação. A infraestrutura, seja laboratórios para as aulas, seja o ambiente em si, são elementos que prejudicam muito o aprendizado caso não estejam adequados durante as aulas de programação (e qualquer tipo de disciplina). Comentários sobre como temperatura, ruídos de maneira geral e problemas nas máquinas foram citados por alguns alunos como elementos que prejudicam tanto a aula quanto o aprendizado.

Para disciplinas de caráter aplicado, e com uma variedade de tecnologias associadas, um grande desafio para os docentes é como alinhar teoria e prática em sala de aula, motivar alunos, e como se atualizar, pois constantemente novas plataformas de desenvolvimento e linguagens de programação são lançadas no mercado, dificultando o trabalho acadêmico.

Como trabalhos futuros, pretende-se analisar cada um dos elementos identificados junto à coordenação do curso e grupos de unidades curriculares de programação, e criar um plano de ação para melhorar as condições de aula e aprendizado. Além disso, trabalhar metodologias variadas e didática com os professores das disciplinas de programação.

Referências

Amaral, E., Camargo, A., Gomes, M., Richa, C., e Becker, L. (2017). Algo+ uma ferramenta para o apoio ao ensino de algoritmos e programação para alunos iniciantes. In *VI Congresso Brasileiro de Informática na Educação (CBIE)*.

Bardin, L. (2011). *Análise de conteúdo*. São Paulo: Edições 70.

Chang, C. e Lee, R. (1973). *Symbolic Logic and Mechanical Theorem Proving*. Nova Iorque: Academic Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2001). *Introduction to Algorithms*. 2 ed. Nova Iorque: McGraw-Hill.

Coutinho, E. F., Bonates, M. F., e Moreira, L. O. (2018). Relato sobre o uso de uma ferramenta de desenvolvimento de jogos para o ensino introdutório de lógica de programação. In *IV Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação (WAlgProg)*.

Coutinho, E. F., de Lima, E. T., e Santos, C. C. (2017). Um panorama sobre o desempenho de uma disciplina inicial de programação em um curso de graduação. *Revista Tecnologias na Educação*, 19(9).

Coutinho, E. F., Gomes, G. A. M., e Leite, A. J. M. (2016). Applying design thinking in disciplines of systems development. In *2016 8th Euro American Conference on Telematics and Information Systems (EATIS)*, pages 1–8.

de Oliveira, B. P., Balan, G. R., Leitao, P. R. M. B., e Coutinho, E. F. (2017). Identificação e discussão de problemas nas disciplinas iniciais de programação do curso de graduação sistemas e mídias digitais. *Revista Sistemas e Mídias Digitais (RSMD)*, 2(1).

Farrell, J. (2015). *Programming Logic and Design*. 8 ed. Stamford: Cengage.

Ferreira, A. C., Santos, J., Silva, R., Oliveira, A. T. R., Zabot, D., Abdalla, D., e Matos, E. (2016). Hello world: relato de experiência de um curso de iniciação à programação. In *II Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação (WAlgProg)*, pages 1306–1315. SBC.

Gil, A. C. (2008). *Como elaborar projetos de pesquisa*. 4. ed. São Paulo: Atlas.

Lopes, M. C. e Kopsch, H. K. (2018). Furbot-web: Umas plataforma adaptativa para o ensino de programação. *Revista Tecnologias na Educação (TECEDU)*, 10(25).

Marinho, C. S. S., Moreira, L. O., Coutinho, E. F., Paillard, G. A. L., e Neto, E. T. L. (2016). Experiências no uso da metodologia coding dojo nas disciplinas básicas de programação de computadores em um curso interdisciplinar do ensino superior. In *II Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação (WAlgProg)*, pages 1097–1106. SBC.

Meira, M. C., Lima, M. S. S., e Borges, M. A. F. (2016). Torneios baseados em robocode para incentivar jovens a aprender programação. In *Anais dos Workshops do XXXVI Congresso da Sociedade Brasileira de Computação (CSBC 2016), Workshop sobre Educação em Computação (WEI)*.

Mora, M. C. e Giraffa, L. M. M. (2013). Evasão na disciplina de algoritmo e programação: Um estudo a partir dos fatores intervenientes na perspectiva do aluno. In *Tercera Conferencia sobre el Abandono en la Educación Superior (III CLABES)*.

Puga, Sandra; Rissetti, G. (2003). *Lógica de Programação e Estruturas de Dados - Com Aplicações em Java*. 1. ed. São Paulo: Pearson Prentice Hall.

Ribas, E., Bianco, G. D., e Lahm, R. A. (2016). Um curso de programação a distância com metodologias ativas e análise de aprendizagem por métricas de software. *RENOTE - Revista Novas Tecnologias na Educação*, 14(2).

Sebesta, R. W. (2012). *Concepts of Programming Languages.* 10 ed. Boston: Pearson.

Wang, M. A. e Prado, E. P. V. (2015). Revisão sistemática sobre alfabetização computacional. In *XI Simpósio Brasileiro de Sistemas de Informação (SBSI)*.

Zanetti, H. A. P. e Oliveira, L. C. V. (2015). Prática de ensino de programação de computadores com robótica pedagógica e aplicação de pensamento computacional. In *Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação (CBIE 2015)*.