UM PANORAMA SOBRE O DESEMPENHO DE UMA DISCIPLINA INICIAL DE PROGRAMAÇÃO EM UM CURSO DE GRADUAÇÃO

Emanuel Ferreira Coutinho[1]

Ernesto Trajano de Lima[2]

Clemilson Costa Santos[3]

RESUMO

Aprender a programar por meio de uma linguagem de programação computacional não é uma tarefa fácil para iniciantes principalmente para quem não possui uma base matemática e lógica. Diversos autores na literatura já alertaram programar é uma tarefa complexa e que frequentemente está associado a uma alta taxa de reprovação e evasão. Diversos fatores prejudicam o ensino de programação, tais como: falta de conhecimentos básicos do ensino médio, desmotivação, falta de maturidade do aluno e professores despreparados. Além disso, programar requer esforço e dedicação. Comumente em cursos de graduação mais ligados às ciências exatas, os alunos têm contato com disciplinas de programação, e uma base em raciocínio lógico e matemático é essencial para o pleno entendimento das estruturas fundamentais de programação, além das linguagens de programação em si. Este artigo tem como objetivo apresentar e analisar experiências obtidas durante cinco turmas da disciplina Programação I do curso de Sistemas e Mídias Digitais da Universidade Federal do Ceará. Esta disciplina tem como objetivo prover ao aluno conceitos de lógica de programação para o desenvolvimento de soluções de problemas, através do uso das estruturas básicas. Neste trabalho verificou-se o desempenho acadêmico dos alunos por meio de uma análise da média final dos alunos e a quantidade de faltas.

Palavras chave: **lógica de programação; algoritmos; linguagens de programação.**

1. INTRODUÇÃO

[1] Prof.Adjunto-Universidade Federal do Ceará
[2] Prof.Adjunto-Universidade Federal do Ceará
[3] Prof.Adjunto-Universidade Federal do Ceará

Aprender a escrever programas de computador, seja utilizando componentes que se encaixam formando uma sentença lógica ou codificando de maneira textual, não é uma tarefa fácil para quem está iniciando e, principalmente, para quem não possui uma base consistente em Matemática e Lógica. A programação, entretanto, integra cada vez mais currículos de cursos de graduação que, tradicionalmente, não a teriam como disciplina do curso, como é o caso das Engenharias e cursos de Computação. Muito disso se deve ao caráter cada vez mais interdisciplinar das áreas de conhecimento, o que faz com que cursos com objetivos significativamente diferentes passem a colaborar entre si. Sendo assim, a programação de aplicações e serviços se torna um elo entre diferentes áreas, possibilitando que novas pesquisas e atividades profissionais emerjam.

Alguns autores na literatura já sinalizaram que aprender a programar é uma atividade considerada complexa (KOULOURI, 2014; KUNKLE, 2016). Outros, já indicaram também que está associado às disciplinas introdutórias de programação uma alta taxa de reprovação e evasão (BENNEDSEN; CASPERSEN, 2007; WATSON; LI, 2014).

Na literatura sobre ensino introdutório de programação existe uma longa lista de motivos que impactam no aprendizado dos conceitos ligados não só à programação de computadores, mas de dispositivos computacionais de maneira geral, tais como: falta de conhecimentos básicos do ensino médio, falta de motivação, falta de maturidade do aluno, professores despreparados e exercícios teóricos ou inadequados, entre outros.

Ainda no que tange a dificuldade de aprender a programar, Barbosa, Couto e Terra (2016) pesquisaram que no início do ensino superior é comum que estudantes da área de Ciência da Computação e áreas afins tenham dificuldade no raciocínio lógico, conteúdo necessário e essencial para a o projeto e criação de soluções, uma vez que, no ensino médio, tais estudantes estavam acostumados a apenas aplicar fórmulas previamente existentes na solução de problemas.

Outro fator estreitamente ligado à dificuldade de aprendizado de conteúdos relacionados a programação (como lógica e matemática) é o aspecto motivacional. O fato de os professores do ensino superior simplesmente transmitirem o conhecimento na sala de aula de forma convencional, geralmente por meio de aulas expositivas, é um problema a ser resolvido, pois faz com que os estudantes sejam atores passivos no processo de aprendizado (DA SILVA, 2003). E muitas vezes aulas expositivas por meio de *slides* não são adequadas

tanto para repassar a mensagem quanto para o aluno praticar, dada as características inerentemente práticas da atividade de programar.

Disciplinas de introdução à programação costumam fazer parte do ciclo básico de formação nos currículos de cursos de graduação em Engenharia e Ciências Exatas (CARVALHO et al, 2016). Pela falta de maturidade em perceber a importância da disciplina no exercício da sua atividade profissional, muitos alunos desses cursos normalmente não possuem motivação suficiente para a disciplina, o que resulta em altos índices de reprovação. Além disso, também é comum que disciplinas de programação sejam pré-requisitos de diversas outras disciplinas e sua reprovação implica em uma quantidade considerável de alunos represados que não conseguem avançar no curso.

Aureliano et al. (2016) apontaram que ser professor de uma disciplina introdutória de programação é uma tarefa tão complexa quanto ser um estudante dela. Em outras palavras, o desafio de ensinar é proporcional ao de aprender. É por esta razão que muitas universidades escolhem como docentes destas disciplinas os professores mais experientes e com mais vivência em assuntos relacionados ao ensino de programação. Programar vai requerer habilidades que são diferentes daquelas necessárias para ensinar a programar. Além disso, a experiência profissional em programar colabora muito para o ensino, o que infelizmente não é tão comum.

Outro aspecto que vale a pena destacar é que aprender a programar é uma atividade que requer esforço e dedicação. Porém, é comum, no ensino de programação, a utilização de exercícios teóricos e, muitas vezes, de pouca ou nenhuma aplicação aos conceitos que serão aprendidos no restante do curso (GUZDIAL; GUO, 2014). Isso prejudica tanto o aprendizado quanto a motivação. A ligação entre a teoria da lógica de programação com a programação em si deve ser bem fundamentada e bem conectada, de forma que os alunos possam entender a verdadeira relação e necessidade da lógica de programação como base para a disciplina e área.

Astolfi e Lopes Junior (2016) sinalizaram que no contexto do ensino de linguagem de programação, principalmente nos últimos semestres de um curso, os conhecimentos prévios de um estudante são fomentados por disciplinas pré-requisitos de semestres anteriores. Em muitos casos, percebe-se que os estudantes chegam com deficiências conceituais e

despreparados para adquirir o novo conhecimento que lhe será transmitido. Como resultado têm-se muita reprovação e desistência no curso.

Considerando o exposto, o presente artigo tem como objetivo apresentar e analisar experiências obtidas durante cinco turmas da disciplina Programação I do curso de Sistemas e Mídias Digitais (SMD) da Universidade Federal do Ceará (UFC). 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, o curso de graduação e a disciplina 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

2.1 LÓGICA DE PROGRAMAÇÃO

A Lógica e seus formalismos são aplicados às mais diversas áreas, tais como Psicologia (STENNING; VAN LAMBALGEN, 2011), Matemática, Física e Computação (BAEZ; STAY, 2011). Na Computação, normalmente, ela é aplicada a todas as suas áreas (CHANG; 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. Já na construção do software, é por meio do raciocínio lógico que se projetam algoritmos, que, por sua vez, são transformados em programas capazes de solucionar problemas. 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.

Um conceito básico na programação é o algoritmo, que pode ser definido como uma sequência lógica de instruções que devem ser seguidas para a resolução de um problema ou para a execução de uma tarefa (CORMEN, 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, ainda que, em grande medida, as realizemos de maneira intuitiva.

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 forma: é necessário fazer isto de maneira precisa, evitando ambiguidades. Tal maneira é a linguagem de programação, que pode se basear em diferentes paradigmas, possuindo, desta forma, cada uma, características particulares e específicas (SEBESTA, 2012). Exemplos de linguagens de programação são Java, C++, FORTRAN, Python e Ruby.

Um programa de computador é escrito através de sentenças nestas linguagens, sentenças estas que são formadas por uma série de instruções ou comandos. Ainda que tais comandos possam diferir entre linguagens, o conhecimento de lógica de programação, porém, 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, mais próximas da linguagem natural e humana, tal objetivo ainda está longe de ser alcançado (PUGA; RISSETI, 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 sintaxes diferentes e tecnologias variadas. Sendo assim, o pleno entendimento dos fundamentos, da lógica de programação, que é a base para as linguagens de programação, se torna essencial.

2.2 CURSO DE GRADUAÇÃO E DISCIPLINA

O curso de graduação Sistemas e Mídias Digitais (SMD) da Universidade Federal do Ceará tem como objetivo formar profissionais com conhecimentos especializados em duas grandes áreas principais: Sistemas Multimídia e Mídias Digitais. Desta forma, contribui-se

para o desenvolvimento de novos perfis profissionais, viabilizando atividades produtivas nas áreas de geração de mídias digitais e desenvolvimento de sistemas multimídia, tais como: sistemas web, dispositivos móveis, jogos digitais e animações gráficas.

Conforme seu projeto pedagógico, e focando nas competências em Sistemas Multimídia, tem-se os seguintes itens: análise e projeto de sistemas multimídia; design de jogos digitais; programação para web; programação para sistemas embarcados (TV digital interativa, dispositivos móveis etc.); programação de computadores e de consoles voltada ao entretenimento digital; desenvolvimento de aplicações multimídia distribuídas; criação e manutenção de repositórios multimídia; desenvolvimento de tecnologias específicas para jogos eletrônicos (computação gráfica, inteligência artificial, banco de dados, redes de computadores, etc). Todas essas competências estão diretamente relacionadas com programação. Sendo assim, espera-se que o aluno egresso tenha uma habilidade pelo menos razoável na programação e codificação de computadores, e em ambientes diferenciados.

Com um forte apelo tecnológico e prático, uma das disciplinas iniciais do curso é Programação I. Esta disciplina possui 64 horas-aula de carga horária, sendo na maior parte do tempo prática. Sua justificativa e objetivos são: apresentar e demonstrar a importância da programação no curso de graduação, assim como familiarizar o aluno com a lógica para o desenvolvimento de soluções de problemas. Isto é feito através do uso das estruturas básicas para a construção de algoritmos, utilizando pseudocódigo e ferramentas multimídia que permitam o aluno desenvolver aplicações voltadas para a área de mídias digitais.

Nas aulas, via de regra, utilizam-se alguns slides como referência, mas grande parte da disciplina ocorre de maneira prática aplicada, diante do computador. Compõem sua forma de avaliação provas escritas, provas na máquina e trabalhos práticos. A disciplina conta, ainda, com o apoio de bolsistas monitores que auxiliam tanto os professores quanto os alunos.

Essa disciplina possui um importante papel ao longo da grade curricular do curso, pois ela é pré-requisito praticamente de todas as disciplinas de programação. Ao se considerar que ela é a base em raciocínio lógico e matemático, com a iniciação em linguagem de programação, essa disciplina é fundamental para o bom prosseguimento no curso.

Além disso, como o curso possui características interdisciplinares, é comum que o perfil dos alunos ingressantes também é bem variado. Isso implica que na disciplina de Programação I sempre há alunos naturalmente de diversas áreas focais do curso de graduação,

tais como: *design* digital, comunicação, jogos digitais, modelagem 2D e 3D, audiovisual e tecnologias.

3. METODOLOGIA DE TRABALHO

O objetivo desta pesquisa foi analisar os dados de aprovação e frequência dos alunos, identificando pontos de destaque para propor melhorias para a disciplina. Os dados foram coletados a partir da ferramenta de controle acadêmico da universidade, com a autorização dos professores das respectivas turmas. Esses dados foram: notas finais (0,0 a 10,0), situação (aprovado, reprovado, reprovado por falta e disciplina trancada) e frequência (0 a 64 horas aula) de cada aluno alunos de cada turma.

No presente trabalho, foram utilizados os dados de cinco turmas da disciplina de Programação I, do curso SMD. Duas turmas foram do primeiro semestre de 2016, e as três turmas restantes, do segundo semestre de 2016. Todas essas turmas tiveram o mesmo perfil de metodologia e didática, pois os dois professores que as ministram possuíam o mesmo perfil e utilizaram o mesmo material. As ferramentas de programação utilizadas foram as mesmas: *Processing*[4] e *Stencyl*[5]. Todas as disciplinas possuíram provas escritas para algoritmos e provas práticas na máquina, com codificação.

Quanto ao perfil dos alunos, quatro turmas eram de alunos ingressantes no curso e uma era uma turma específica para alunos que reprovaram a disciplina.

Para a análise e interpretação dos dados, utilizamos estatística descritiva, analisando os valores por meio de média, mediana, valor mínimo, valor máximo e desvio padrão.

A mediana é o valor que separa a metade maior e a metade menor de uma amostra, uma população ou uma distribuição de probabilidade. O desvio padrão é uma medida de dispersão, ou seja, qual a distância dos dados de uma amostra com relação à média. Um baixo desvio padrão indica que os pontos dos dados tendem a estar próximos da média ou do valor esperado. Um alto desvio padrão indica que os pontos dos dados estão espalhados por uma ampla gama de valores.

Para uma melhor apresentação dos dados, utilizamos três tipos de gráficos: histograma, *boxplot* e gráfico de pizza. Um histograma é um diagrama que representa uma

[4] Disponível em http://www.processing.org

[5] Disponível em http://www.stencyl.com

relação de valores entre duas variáveis. Ele é constituído por retângulos ou linhas, desenhados a partir de uma linha na base, na qual suas posições ao longo dessa linha representam o valor ou a amplitude de uma das variáveis, e a sua altura, o valor correspondente de uma segunda variável.

Um *boxplot*, ou diagrama de caixa, é um gráfico no qual o eixo vertical representa a variável a ser analisada e o eixo horizontal representa um fator de interesse. É uma ferramenta para localizar e analisar como uma variável está variando entre diferentes grupos de dados. É utilizado também para se identificar onde estão localizados 50% dos valores mais prováveis, a mediana e os valores extremos.

O gráfico de pizza é um gráfico circular, divido em setores, onde cada setor exibe o tamanho de parte da amostra analisada. É útil quando se pretende ilustrar os tamanhos relativos nos quais um todo foi dividido.

4. RESULTADOS E ANÁLISES

A Tabela 1 exibe para cada turma de cada ano da disciplina de Programação 1 a quantidade de alunos matriculados e a quantidade de alunos aprovados. De um total de 129 alunos, 65 alunos reprovaram a disciplina, e 64 foram aprovados, o que resulta em 49.6% de aprovação. Esse valor é alarmante e desperta atenção em saber os motivos pelos qual existe um alto índice de reprovação em apenas uma disciplina. Além disso, como essa disciplina é obrigatória e logo do primeiro semestre do curso, estes números impactam seriamente o percurso acadêmico dos alunos.

Tabela 1 – Resumo das disciplinas de programação

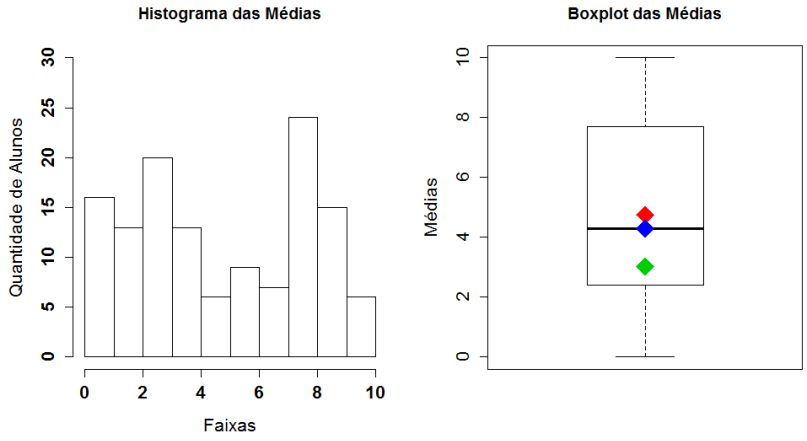
Semestre	**2016.1**	**2016.1**	**2016.2**	**2016.2**	**2016.2**
Turma	T1	T2	T1	T2	T3
Quantidade de alunos matriculados	28	26	27	26	22
Quantidade de alunos aprovados	21	12	9	13	9
Quantidade de alunos reprovados	7	14	18	13	13

Fonte: Ferramenta de gestão acadêmica da universidade.

A Figura 1 exibe o histograma e o *boxplot* para as médias finais dos alunos das cinco turmas, e a Figura 2 exibe os mesmos gráficos para as faltas dos alunos nas mesmas turmas. No *boxplot* temos alguns losangos coloridos. O losango vermelha é a média, o azul é a mediana, e o verde é o desvio padrão.

Analisando as classes (categorias) da Figura 1, verificamos que há uma diversidade nas notas, com a maioria dos alunos obtendo média final entre 7.0 e 8.0. Entretanto há muito aluno com notas inferiores a 5.0, o que implica em reprovação. Ao analisar o *boxplot* das médias, verificamos que pela mediana de valor 4.3 (correspondente à linha horizontal dentro do retângulo), que a metade dos alunos se encontram nessa faixa, o que é outro ponto a ser tratado. Quanto à média, esta foi 4.7, reforçando que em geral os alunos precisam elevar suas notas. Por fim, considerando os valores superiores e inferiores, obteve-se notas de 0.0 a 10.0. Os alunos com nota final 0.0 na maioria foram desistentes. Por fim, o desvio padrão indica que há uma dispersão dos valores variação em torno da média de 3.0 pontos para valores considerados típicos nesta população.

Figura 1 – Histograma e *boxplot* para as médias finais dos alunos



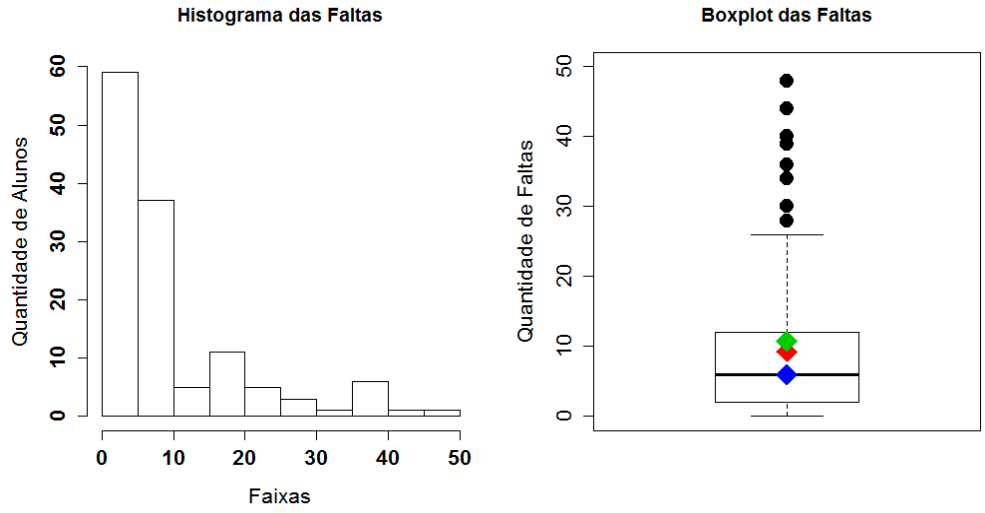
Ao analisar as classes da Figura 2, identificamos que apesar de haver alunos com diferentes quantidades de faltas, há uma grande quantidade de alunos com até 10 faltas. Para não ser reprovado por falta, o aluno deve ter no máximo até 16 faltas. Cada falta corresponde a uma hora aula.

Ao analisar o *boxplot* das faltas, verificamos que pela mediana de valor 6 (correspondente à linha horizontal dentro do retângulo), que a metade dos alunos se encontram nessa faixa, e este valor pode ser considerado interessante para esse contexto,

tendo em vista que implica em muitos alunos com poucas faltas. Quanto à média, esta foi maior que a mediana, sendo 9.2, ainda sendo um valor aceitável pois o limite de faltas é 16. Ao considerar os valores superiores e inferiores, onde o maior valor seria 64 faltas que é a carga horária da disciplina, identificamos diversos alunos com quantidade elevada de faltas. Em geral esses alunos são desistentes. Por fim, o desvio padrão foi de 10.7.

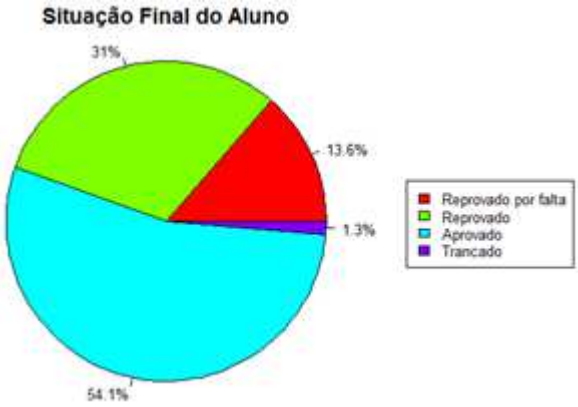
Percebe-se na Figura 2 a presença de alguns pontos na parte superior do gráfico. Esses pontos são conhecidos como *outlier*, ou valores atípicos. Eles são pontos que apresentam um grande afastamento dos demais, sendo considerados bastantes inconsistentes. A existência de *outliers* implica normalmente em prejuízos a interpretação dos resultados dos testes estatísticos aplicados às amostras. No nosso caso, esses valores atípicos são alunos com cerca de mais de 30 faltas por disciplina.

Figura 2 – Histograma e *boxplot* para as faltas dos alunos



A Figura 3 exibe a situação final do aluno para a disciplina. Pelo gráfico podemos visualizar que praticamente metade dos alunos são aprovados. Entretanto, praticamente a outra metade, que são os reprovados, há uma parcela de reprovação por faltas. A reprovação por falta é pior que a reprovação por nota, pois caso o aluno reprove duas vezes a mesma disciplina, ele não consegue se matricular sem autorização da coordenação e assinatura de um termo de compromisso indicando seu comprometimento em não reprovar mais por faltas.

Figura 3 – Gráfico de pizza (setores) com a situação final dos alunos



Ao se tentar identificar alguma razão para quantidade de reprovações, percebemos que muitas vezes a didática e metodologia para determinado assunto não era adequada. Esse é um grande ponto de melhoria a ser considerado. Preferencialmente, antes de se codificar propriamente dito, a aplicação de uma grande revisão ou exercício do raciocínio lógico e matemático vai auxiliar muito nas próximas etapas da disciplina, e assim os alunos passem a ter uma base teórica das estruturas a serem utilizadas na ferramenta de programação e codificação.

Aureliano, Tedesco e Giraffa (2016) discutiram que o papel do professor de programação é buscar orientar o processo de aprendizagem dos estudantes na disciplina (BIGGS; TANG, 2011). Nesse contexto, os autores comentaram alguns dos desafios enfrentados pelos professores ao planejar atividades e na execução, apresentando oportunidades de pesquisa para a educação em programação para iniciantes, que foram: o professor deve propor atividades de ensino e o professor também deve propor atividades de aprendizagem.

Por fim, Astolfi e Lopes Junior (2016) concluíram que é necessário um constante movimento de reflexão, onde professores e alunos compartilham o protagonismo do processo de ensino e aprendizagem em programação, e decidem por um trabalho fora de suas zonas de conforto, instaurando um espaço diferenciado de aprendizagem, diferente dos comumente utilizados.

5. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou um panorama sobre o desempenho de uma disciplina de programação do primeiro semestre de um curso de graduação. Identificou-se a necessidade de se preparar melhor o aluno ingresso nos fundamentos de raciocínio lógico e matemático, pois tais conceitos são essenciais para o melhor aprendizado em programação, principalmente pelas características interdisciplinares do curso, que deve atender a diferentes perfis de profissionais de diversas áreas, como *design* digital, comunicação e sistemas multimídia.

Especificamente para o curso de graduação Sistemas e Mídias Digitais, algumas pesquisas já estão em andamento investigando problemas nas disciplinas de programação (OLIVEIRA et al, 2017) e como melhorar a prática no ensino de programação (EDUARDO; MARINHO; MOREIRA, 2016; MARINHO et al, 2016). Tais iniciativas são específicas para os semestres iniciais e assim espera-se que o ensino em lógica de programação tenha uma qualidade melhor, e também minimize a taxa de evasão e falta de motivação dos alunos.

Como trabalhos futuros pretende-se avaliar novas técnicas de ensino de programação e aplicar às turmas. A utilização de gamificação, por exemplo, pode ser uma boa estratégia para reforçar o aprendizado, principalmente no assunto raciocínio lógico, aspecto fundamental no aprendizado de programação. A utilização de aplicativos para desenvolver o raciocínio lógico e matemático, preferencialmente de software livre, também é uma proposta a ser adotada.

REFERÊNCIAS

ASTOLFI, G.; LOPES JR., D. Ensino de Linguagem de Programação com Ênfase na Aprendizagem Significativa. **XXXVI Congresso da Sociedade Brasileira de Computação - WEI - 24º Workshop sobre Educação em Computação**. 2016.

AURELIANO, V. C. O.; TEDESCO, P. C. A. R.; GIRAFFA, L. M. M. Desafios e oportunidades aos processos de ensino e de aprendizagem de programação para iniciantes. **XXXVI Congresso da Sociedade Brasileira de Computação - WEI - 24º Workshop sobre Educação em Computação**. 2016.

BAEZ, J. C.; STAY, M. Physics, Topology, Logic and Computation: a Rosetta Stone. COECKE, B. (ed) New Structures for Physics. **Lecture Notes in Physics**. Berlim: Springer, v. 813, p. 95-174, 2011.

BARBOSA, L. L.; COUTO, C. M. S.; TERRA, R. PortuCol: uma pseudolinguagem inspirada em C ANSI para o Ensino de Lógica de Programação e Algoritmos. **XXXVI Congresso da Sociedade Brasileira de Computação - WEI - 24º Workshop sobre Educação em Computação**. 2016.

BENNEDSEN, J.; CASPERSEN, M. E. Failure rates in introductory programming. **SIGCSE Bull**., v. 39, n. 2, p. 32-36, 2007.

BIGGS, J.; TANG, C. **Teaching For Quality Learning At University**. McGraw-Hill Education, 2011. ISBN 9780335242757.

CARVALHO, L. S. G. et al. Ensino de Programação para Futuros Não-Programadores: Contextualizando os Exercícios com as Demais Disciplinas de mesmo Período Letivo. **XXXVI Congresso da Sociedade Brasileira de Computação - WEI - 24º Workshop sobre Educação em Computação**. 2016.

CHANG, C.-L.; LEE, R. **Symbolic Logic and Mechanical Theorem Proving**. Nova Iorque: Academic Press, 1973.

CORMEN, T. H. et al. **Introduction to Algorithms**. 2 ed. Nova Iorque: McGraw-Hill, 2001.

DA SILVA, F. M. Aspectos relevantes das novas tecnologias aplicadas a educação e os desafios impostos para a atuação dos docentes. **Akropolis - Revista de Ciências Humanas da UNIPAR**, v. 11, n. 2. 2003. ISSN: 1982-1093.

EDUARDO, João Pedro. O. V.; MARINHO, Carlos Sérgio. S.; MOREIRA, Leonardo de Oliveira. Proposta de Coding Dojo para o Bacharelado em Sistemas e Mídias Digitais da Universidade Federal do Ceará**. Revista Sistemas e Mıdias Digitais (RSMD)**, Fortaleza, v. 1, n. 2, ISSN 2525-9555. dez. 2016.

FARRELL, J. **Programming Logic and Design**. 8 ed. Stamford: Cengage, 2015.

GUZDIAL, M.; GUO, P. The difficulty of teaching programming languages, and the benefits of hands-on learning. **Communications of the ACM**. v.57, Issue 7, July 2014.
Pages 10-11.

KOULOURI, T.; LAURIA, S.; MACREDIE, R. D. Teaching introductory programming: A quantitative evaluation of different approaches. **ACM Trans. Comput. Educ**. v.14, n. 4, Dec 2014.

KUNKLE, W. M.; ALLEN, R. B. The impact of different teaching approaches and languages on student learning of introductory programming concepts. **ACM Trans. Comput. Educ**. v. 16, n. 1, Jan 2016.

MARINHEIRO, F.; SILVA, I.; MADEIRA, C.; CORDEIRO, S.; SOUZA, D.; COSTA, P.; FERNANDES, G. Ensinando Crianças do Ensino Fundamental a Programar Computadores com o Auxílio de Jogos Digitais. **Revista Tecnologias na Educação – Edição Temática – Congresso Regional sobre Tecnologias na Educação (Ctrl+E 2016)**, Ano 8, v.16, Set 2016.

MARINHO, Carlos et al. Experiências no Uso da Metodologia Coding Dojo nas Disciplinas Básicas de Programação de Computadores em um Curso Interdisciplinar do Ensino Superior. **Workshops do Congresso Brasileiro de Informática na Educação. II Workshop de Ensino em Pensamento Computacional, Algoritmos e Programação (WAlgProg/2016)**, Uberlândia, 2016.

OLIVEIRA, Bruno Pinheiro de et al. Identificação e Discussão de Problemas nas Disciplinas Iniciais de Programação do Curso de Graduação Sistemas e Mídias Digitais. **Revista Sistemas e Mıdias Digitais (RSMD)**, Fortaleza, v. 2, n. 1, ISSN 2525-9555. abr. 2017.

PUGA, Sandra; RISSETTI, Gerson. **Lógica de Programação e Estruturas de Dados - Com Aplicações em Java**. 1. ed. São Paulo: Pearson Prentice Hall, 2003. ISBN 85-87918-82-6.

RAEDER, Mateus et al. L2PM: relato de uma experiência sobre o ensino integrado de Lógica, Programação e Matemática para Computação. **XXXVI Congresso da Sociedade Brasileira de Computação - WEI - 24º Workshop sobre Educação em Computação**. 2016.

SEBESTA, R. W. **Concepts of Programming Languages**. 10 ed. Boston: Pearson, 2012.

SILVA, V.; SOUZA, A.; MORAIS, D. Pensamento Computacional: Um Relato de Práticas Pedagógicas para o Ensino de Computação em Escolas Públicas. **Revista Tecnologias na Educação – Edição Temática – Congresso Regional sobre Tecnologias na Educação (Ctrl+E 2016)**, Ano 8, v.16, Set 2016.

STENNING, K.; VAN LAMBALGEN, M. Reasoning, logic, and psychology. **Wiley Interdisciplinary Reviews: Cognitive Science**, v. 2, n. 5, p. 555–567, 2011.

VITKUTE-ADZGAUSKIENE, D; VIDZIUNAS, A. Problems in Choosing Tools and Methods for Teaching Programming. **Informatics in Education**, v. 11, n. 2, 271–282, 2012.

WATSON, C.; LI, F. W. B. Failure rates in introductory programming revisited. **Proceedings of the 2014 conference on Innovation & technology in computer science education**. Uppsala, Sweden: ACM: 39-44 p. 2014.

Recebido em abril 2017

Aprovado em junho 2017