Ensino de Algoritmos e Lógica de Programação para os Diferentes Cursos: Um Mapeamento Sistemático da Literatura

Luiza Engler Stadelhofer[1], Isabela Gasparini[1,2,3]

[1]Departamento de Ciência da Computação, [2]PPGECMT, [3]PPGCA
Universidade do Estado de Santa Catarina (UDESC), Joinville, Brasil

luiza.engler@gmail.com, isabela.gasparini@udesc.br

Abstract. *The teaching-learning process of algorithms and programming logic can be a difficult task, especially for non-CS majors, e.g. engineering, mathematics and physics' courses. In addition to coping with high rates of failure and dropout, most students have negative views on programming, viewing it as unnecessary and irrelevant. A proposed solution to this problem is the use of a teaching approach that relates the students major to the contents learned, aiming to increase their motivation. In this study a Systematic Literature Mapping was conducted, whose results showed how these approaches are developed and how they exhibit promising repercussions.*

Resumo. *O processo de ensino-aprendizagem de algoritmos e lógica de programação pode ser uma tarefa difícil, especialmente para estudantes de cursos superiores não pertencentes à área da computação, como por exemplo das engenharias, matemática, física, etc. Além de lidar com altas taxas de reprovação e evasão, a maioria dos estudantes apresentam opiniões negativas em relação à programação, a vendo como desnecessária e irrelevante. Uma solução proposta para esta problemática é a utilização de uma abordagem de ensino que relacione o curso dos estudantes com os conteúdos aprendidos, visando aumentar a motivação dos mesmos. Neste estudo foi conduzido um Mapeamento Sistemático da Literatura, cujos resultados apresentaram como essas abordagens são desenvolvidas e como elas exibem repercussões promissoras.*

1. Introdução

O processo de ensino e de aprendizagem de algoritmos e lógica de programação pode ser, em muitos casos, uma tarefa difícil. As disciplinas introdutórias dessa área apresentam altas taxas de reprovação e evasão, algo que não é surpreendente se levarmos em consideração as dificuldades em aprender certos conteúdos, e a falta de motivação e interesse demonstrada por muitos estudantes [Souza, Batista e Barbosa 2016]. Para estudantes de cursos superiores não pertencentes à área da computação, essa tarefa se torna ainda mais complexa. Além das dificuldades citadas anteriormente, estudantes de diferentes cursos são mais suscetíveis a descrever programação como algo desnecessário e irrelevante, assim também como uma tarefa tediosa [Simon *et al.* 2009].

Uma das soluções propostas visando esta problemática, é relacionar os conteúdos aprendidos em sala de aula com os respectivos cursos de formação dos estudantes. Entretanto, essa abordagem ainda não é muito conhecida e utilizada, e não existe nenhum estudo na área que analise como essas abordagens estão sendo desenvolvidas e se elas

DOI: 10.5753/cbie.sbie.2018.108

são eficientes ou não para o ensino de programação para os diferentes cursos. Sendo assim, este estudo apresenta um Mapeamento Sistemático da Literatura – definido como um estudo secundário projetado para estruturar e proporcionar uma visão geral de uma área de pesquisa [Petersen *et al.* 2015] – com o objetivo de realizar esta análise. Este artigo está estruturado como segue. A seção 2 apresenta a fundamentação teórica, explorando os aspectos relacionados ao ensino de programação. A seção 3 detalha o processo do Mapeamento Sistemático conduzido neste trabalho. A seção 4 explora os resultados obtidos e a seção 5 as considerações finais.

2. Fundamentação Teórica

A motivação é um aspecto importante para o processo de aprendizagem. Entretanto, motivar estudantes de outras áreas pode ser um desafio, especialmente quando os mesmos sentem que não há uma ligação da disciplina com os seus cursos ou que os conteúdos aprendidos em sala de aula não serão usados ou aplicados em suas vidas [Shoufan 2016].

A questão da motivação se torna crucial no ensino de algoritmos e lógica de programação, se levarmos em consideração que acadêmicos de diferentes cursos (e.g. engenharias, matemática, física) tendem a apresentar mais opiniões negativas do que estudantes da área (computação e afins), em relação às disciplinas introdutórias de programação, argumentando que as mesmas são sem sentido e desnecessárias [Simon *et al.* 2009].

Além disso, Elarde e Fatt-Fei [2011] abordam a questão de que, na maioria dos casos, estudantes de diferentes áreas só possuem uma disciplina introdutória relacionada à computação. Sendo assim, é essencial que além do ensino de conceitos fundamentais, essas disciplinas demonstrem o poder e o valor da programação de modo geral e, como ela se relaciona com o curso do estudante [Elarde e Fatt-Fei 2011].

Finalmente, outro argumento que apoia o desenvolvimento de uma maior conexão entre o curso do acadêmico e os conteúdos/atividades da disciplina, é o fato de que muitos estudantes já verificam a importância da disciplina para seus currículos – aumentando a possibilidade de futuros estágios e empregos [Chilana *et al.* 2015]. Sendo assim, é importante apresentar aos estudantes como aplicar os conteúdos abordados em suas áreas, mostrando como a programação pode ser uma habilidade útil em suas carreiras.

3. Mapeamento Sistemático

Nesta seção é apresentado o processo metodológico utilizado no desenvolvimento do Mapeamento Sistemático da Literatura (MSL). A metodologia aplicada foi orientada pelas diretrizes definidas por [Petersen *et al.* 2015].

3.1. Questões de Pesquisa

O objetivo desse estudo é compreender as abordagens de ensino com contexto específico para cada curso, utilizadas nas disciplinas introdutórias de algoritmos e lógica de programação para os cursos de graduação de outras áreas; além de verificar a eficiência dessas abordagens propostas. Para atingir este objetivo, foram definidas cinco questões de pesquisa:

- **QP1**: Quais são as metodologias de ensino utilizadas nas disciplinas introdutórias de programação de cursos não pertencentes à área da computação?

- **QP2**: Quais são os recursos tecnológicos utilizados nessas disciplinas?
- **QP3**: Os conteúdos abordados nelas são os mesmos que os normalmente abordados em disciplinas de introdução à programação da área da computação?
- **QP4**: De que forma é feita a contextualização do ensino, relacionando o curso do estudante com o que está sendo ensinado?
- **QP5**: O quão eficiente é utilizar uma abordagem de ensino onde o curso de formação do estudante é levado em consideração?

3.2. Método de Busca

Para a execução da busca foi definida uma *string* de busca, apresentada na Tabela 1. Inicialmente, foi feita uma pesquisa buscando palavras-chave que representariam de forma eficaz o tipo de artigo procurado para o mapeamento. Foram identificados três termos de interesse, utilizados para montar a *string* final: *teaching*, para representar o ensino; *algorithms* e *programming*, para representar as disciplinas que ensinam algoritmos, lógica de programação ou programação; e as palavras *non major*, *non-major*, *non-CS major* e seus respectivos plurais, que são diversas variações de um termo usado para representar estudantes de cursos superiores e que não são da área da computação. Para a aplicação da *string*, foi considerado apenas a presença dos termos no título, palavras-chave e *abstract* dos artigos.

A *string* definida foi executada em quatro mecanismos de busca acadêmicos diferentes, que foram escolhidos através da análise de [Buchinger *et al.* 2014]. Os mecanismos selecionados foram: ACM Digital Library, IEEE Xplore, Engineering Village e Scopus. Ao fazer a busca, reunindo o resultado de todas as bases, foi encontrado um total de 218 artigos, como mostrado na Tabela 2, de onde desse total apenas 151 estavam disponíveis na íntegra.

String Final
(algorithms OR programming) AND ("non major" OR "non-major" OR "non-CS major" OR "non majors" OR "non-majors" OR "non-CS majors") AND teaching

Tabela 1. *String* de Busca

3.3. Seleção dos Estudos Primários

Após a realização da busca, os 151 artigos restantes passaram por um processo de seleção, com o intuito de fazer a identificação dos estudos primários. Neste processo, foram analisados o título, as palavras-chave, o *abstract* e quando necessário, caso existisse dúvidas em relação a inclusão ou exclusão, a introdução e a conclusão dos artigos. Foram aplicados os seguintes critérios de inclusão:

- Artigos completos, que possuam quatro páginas ou mais;
- Artigos que focam no ensino de algoritmos/programação para cursos superiores que não pertencem a área da computação.

Os critérios de exclusão aplicados foram:

- Artigos publicados fora do período de 2007 até 2017;

- Artigos duplicados;
- Artigos que não apresentam estratégias ou abordagens de ensino relacionadas aos cursos de formação dos estudantes.

Inicialmente 3 artigos foram excluídos por não estarem dentro do período de anos estabelecido; outros 12 foram excluídos por possuírem menos de quatro páginas e um total de 73 foram removidos por serem repetidos. Além destes, 58 artigos não entraram no mapeamento pelo motivo de não focarem no ensino de programação para cursos de outras áreas ou por não apresentarem técnicas de ensino relacionadas com o curso do estudante.

Portanto, após este processo restaram um total de 5 artigos, que foram selecionados como estudos primários. A Tabela 3 apresenta a lista de artigos que foram inclusos.

Mecanismo de Busca	Quantidade
ACM Digital Library	48
IEEE Xplore	13
Engineering Village	77
Scopus	80
Total	218
Disponíveis	151
Inclusos	5

Tabela 2. Resultados da *String* por Mecanismo de Busca

ID	Título	Autor(es)
1	Transforming the Instruction of Introductory Computing to Engineering Students (2010)	Hurson, A. L. e Sedigh, S.
2	Increasing student commitment in introductory programming learning (2012)	Mendes, A. J.; Paquete, L.; Cardoso, A. e Gomes, A.
3	Computing For STEM Majors: Enhancing Non CS Majors' Computing Skills (2012)	Adams, J. C. E Pruim, R. J.
4	Computing for Medicine: An Experience Report (2017)	Campbell, J.; Craig, M. e Law, M.
5	Evaluation and Impact of a Required Computational Thinking Course for Architecture Students (2017)	Senske, N.

Tabela 3. Lista de Artigos Inclusos

4. Resultados e Análises

Após a seleção desses 5 artigos, foi realizado um processo de extração e análise dos dados coletados. Em seguida, apresentamos os resultados obtidos a partir deste processo, incluindo uma visão geral das características dos artigos e os dados relacionados às questões de pesquisa.

4.1. Visão Geral

O primeiro tópico analisado foi o ano de publicação dos artigos. Como pode-se observar no Gráfico 1, houve um aumento no número de produções do ano de 2010 para 2012, enquanto em 2017 este número permaneceu o mesmo. Levando em consideração a quantidade pequena de artigos, é difícil concluir se ocorreu ou não um crescimento na discussão do ensino de programação para os diferentes cursos ao longo dos anos. Entretanto, podemos identificar que após alguns anos sem nenhum trabalho abordando este tema, sua discussão foi retomada em 2017.

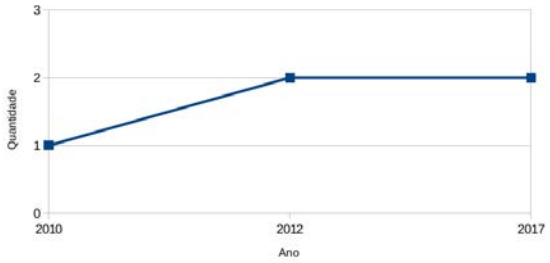

Gráfico 1. Quantidade de Artigos Publicados por Ano

Ao analisar os países das publicações, levando em conta os países de origem das universidades vinculadas aos autores, pode-se observar que a maioria se encontra presente na América do Norte, mais especificamente nos Estados Unidos e no Canadá, com apenas um outro artigo sendo de Portugal, como mostra o Gráfico 2.

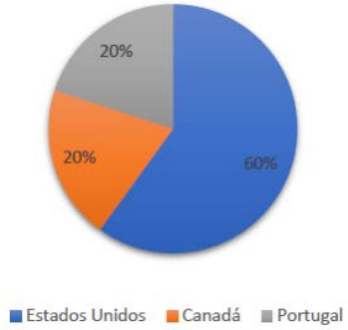

Gráfico 2. Países de Origem das Publicações

Conferência	Quantidade
Frontiers in Education (FIE)	1
Innovation and Technology in Computer Science Education (ITiCSE)	1
Special Interest Group on Computer Science Education (SIGCSE)	2
Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments (TEE)	1

Além disso, todos os artigos selecionados foram publicados em conferências, sendo três delas na área da Computação – as quais todas possuem um *Qualis* restrito: B1 (FIE), A2 (ITiCSE) e A1 (SIGSE) – e uma na área da Engenharia (TEE). As respectivas conferências são apresentadas na Tabela 4.

Tabela 4. Conferências nas quais os Artigos foram publicados

4.2. Criação das Disciplinas

Analisando os estudos primários, percebe-se diversas motivações que encadearam a criação de uma disciplina de ensino de algoritmos e lógica de programação específica apenas para um curso ou uma área. Hurson e Sedigh [2010] citaram que essa decisão foi tomada devido à falta de uma conexão entre as disciplinas que ensinam computação e das que apresentam problemas que podem ser solucionados a partir de suas aplicações. Já Campbell *et al*. [2017] relatam que houve um grande interesse por parte dos estudantes em aprender a programar. Senske [2017] argumenta que a criação da disciplina foi necessária pois a programação é uma das ferramentas tecnológicas mais importantes para

a área dos estudantes, entretanto, a competência deles nela ainda é muito baixa. Adams e Pruim [2012] citaram como motivação o declínio no número de graduados em computação ao mesmo tempo que há um aumento na demanda por profissionais da área. Por fim, Mendes *et al.* [2012] tiveram como motivação o alto número de reprovações e desistências, além de afirmar que estudantes de cursos distintos têm diferentes origens, necessidades e interesses, sendo assim, uma disciplina deve ser adaptada para cada curso.

Destaca-se a grande variedade dos cursos de graduação nos quais as disciplinas de algoritmos e lógica de programação foram aplicadas. Hurson e Sedigh [2010] desenvolveram a disciplina com estudantes do primeiro ano de Engenharias; Campbell *et al.* [2017] trabalharam com o curso de Medicina; Senske [2017] aplicou sua abordagem com estudantes de Arquitetura; Adams e Pruim [2012] trabalharam com estudantes da área de STEM (Ciência, Tecnologia, Engenharia e Matemática) e Mendes *et al.* [2012] ofereceram a disciplina para graduandos em Design e Multimídia.

Além disso, como apresentado no Gráfico 3, três artigos declararam que o desenvolvimento da disciplina em questão foi um trabalho conjunto, entre docentes do departamento de Computação e do departamento do(s) curso(s) no(s) qual(is) a disciplina foi ofertada. Um outro artigo apresentou que o desenvolvimento da matéria foi realizado apenas por membros do departamento do curso que a oferta, e um outro não apresentou claramente quem foram os responsáveis por este desenvolvimento.

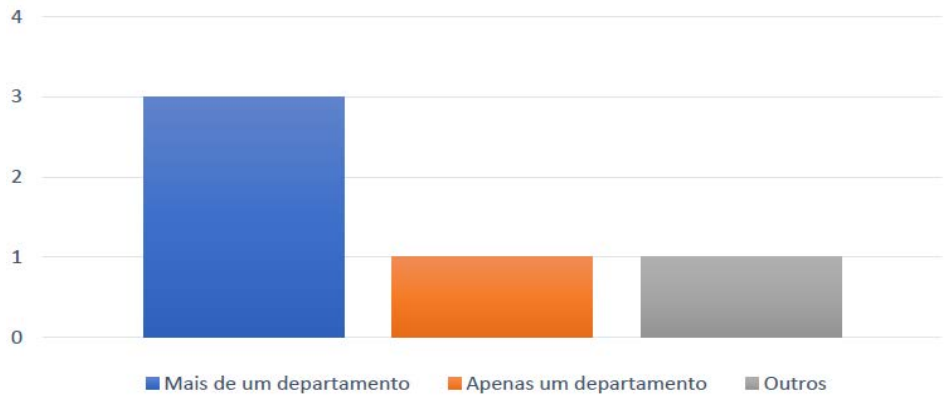

Gráfico 3. Responsáveis pelo Desenvolvimento da Disciplina

4.3. Metodologias de Ensino (QP1)

Para responder a primeira questão de pesquisa, foram analisadas quais as abordagens de ensino foram utilizadas nos artigos selecionados. Todos os estudos apresentaram mais de uma técnica de ensino, sendo que a maioria fez uso de estratégias comuns como aulas teóricas e práticas, e a aplicação de projetos com os estudantes, como apresentado no Gráfico 4. Além disso, um artigo citou o uso da Programação em Pares (*Pair Programming*) e da sala de aula invertida (*Flipped Classroom*), assim como também foi mencionada a aplicação de aulas tutoriais, pelas quais entende-se que são aulas normalmente dadas por um tutor, onde são abordados os conteúdos vistos em sala de aula de forma mais profunda, além de abrir discussões sobre diversos problemas e como os mesmos podem ser resolvidos. A substituição de aulas por *workshops* foi outro método apresentado, com o objetivo de proporcionar aos estudantes uma aprendizagem ativa.

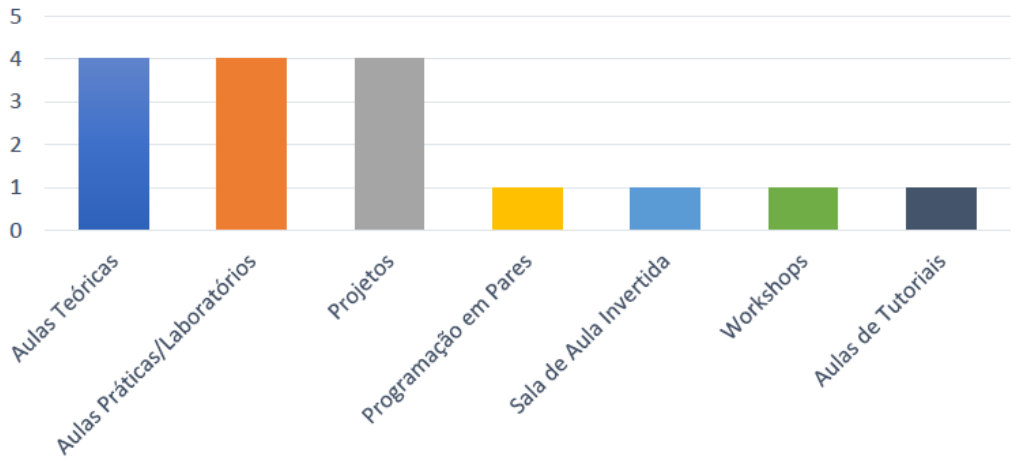

Gráfico 4. Metodologias de Ensino Aplicadas

4.4. Recursos Tecnológicos (QP2)

Como recursos para apoio do ensino-aprendizagem das disciplinas introdutórias de programação dos diferentes cursos, foram citadas: as linguagens de programação Python, C++, Grasshopper e Processing; e as ferramentas Jupyter Notebooks, PCRS – sistema de submissão online de exercícios, e um Ambiente de Desenvolvimento Integrado (IDE) para Python.

4.5. Conteúdos Abordados (QP3)

As respostas da terceira questão de pesquisa mostram que além dos conteúdos básicos, presentes na maioria das disciplinas de introdução à programação (como tipos de dados, estruturas de seleção e repetição, variáveis e estruturas de dados, por exemplo) muitos conceitos adicionais são vistos nas disciplinas voltadas para os cursos de outras áreas. Na disciplina desenvolvida por [Hurson e Sedigh 2010], tópicos como validação, verificação e manutenção de *software* são abordados. Já [Campbell *et al*. 2017] apresentaram conteúdos como SQL, Machine Learning e algoritmos como Dijkstra, assuntos tecnicamente avançados que são apresentados aos estudantes de computação apenas em disciplinas mais à frente no curso, e não em uma disciplina introdutória.

Além disso, conceitos de programação orientada a objetos, como tratamento de exceções, herança e abstração estão presentes nos planos de ensino de algumas disciplinas dos artigos analisados. Portanto, podemos concluir que os assuntos abordados em cada disciplina possuem semelhanças.

4.6. Contextualização do Ensino (QP4)

Como pode-se observar no Gráfico 5, diversas maneiras de relacionar o curso de formação do estudante com aquilo sendo aprendido em sala de aula foram apresentadas. A maioria dos estudos aplicaram com seus estudantes projetos que colocassem em prática os conceitos aprendidos, mas ao mesmo tempo, motivassem os acadêmicos, abordando temas de interesse dos seus respectivos cursos.

Outro modo, utilizado por [Campbell *et al.* 2017], consiste na realização de seminários com *experts* palestrando sobre suas pesquisas e sobre as aplicações da

computação nas suas respectivas áreas. Ainda temos a abordagem de [Senske 2017], onde é feita uma aula introdutória sobre o papel da computação na prática profissional da área; e uma abordagem de [Mendes *et al.* 2012] onde os recursos escolhidos como apoio para a disciplina são específicos para os estudantes de um curso, assim como as atividades desenvolvidas em sala: estudantes de Design e Multimídia, por exemplo, podem trabalhar com ferramentas que facilitam a criação de elementos de interesse dos mesmos, como desenhos e animações.

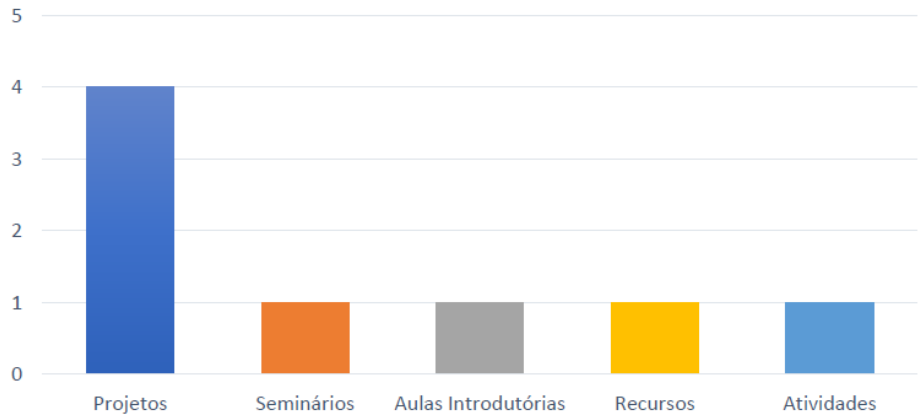

Gráfico 5. Formas Utilizadas para Relacionar a Disciplina com o Curso do Estudante

4.7. Eficiência da Abordagem (QP5)

Para encontrar a resposta da quinta questão de pesquisa, foram examinados os resultados apresentados nos estudos, que foram alcançados a partir da coleta de dados sobre o desempenho dos estudantes nas disciplinas. Esses dados foram obtidos a partir de questionários aplicados com os alunos ou pelas taxas de aprovação e evasão dos cursos.

Como pode-se observar no Gráfico 6, grande parte dos artigos mencionaram que a disciplina ajudou os estudantes a obterem um melhor entendimento de como aplicar a computação nas suas respectivas áreas. Além disso, os acadêmicos relataram que a abordagem utilizada, levando em consideração seus cursos de formação, contribuiu para a aprendizagem dos conteúdos, além de eles se sentirem mais motivados a aprender. Devido a isso, os resultados gerais das disciplinas tiveram uma melhora significativa – a taxa de aprovação subiu e a evasão de estudantes repetentes diminuiu. [Senske 2017] cita que a taxa de aprovação na disciplina desenvolvida passou de 87,5% para 100%, após a adoção da nova abordagem. [Mendes *et al.* 2012] apresenta que a mesma taxa, anteriormente 26,6% e 19,8%, subiu para 57,3%.

Entretanto, [Campbell *et al.* 2017] também menciona que por mais que 78% dos estudantes indicaram que o contexto usado na disciplina contribuiu para a aprendizagem, grande parte deles ainda afirmaram que não foi algo crucial para suas experiências; não haveria uma grande diferença se os projetos e os exemplos não fossem relacionados a área deles.

Analisando os resultados apresentados, temos que a maioria dos estudantes se mostrou positivo, apontando que o uso de uma abordagem de ensino voltada especificamente para seu curso é eficiente.

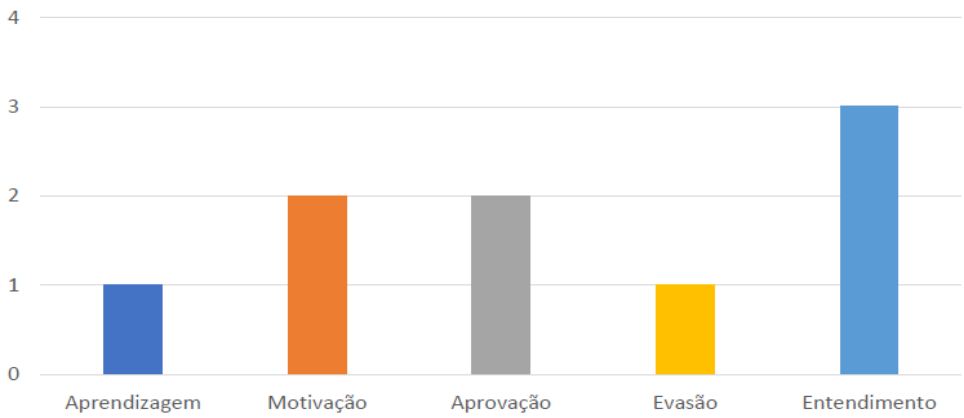

Gráfico 6. Melhorias Apontadas nos Resultados dos Artigos

5. Considerações Finais

Nesse estudo foi conduzido um Mapeamento Sistemático da Literatura com o objetivo de analisar como as abordagens de ensino estão sendo aplicadas nas disciplinas introdutórias de programação no contexto de cursos não pertencentes à área da computação, bem como identificar os recursos, conteúdos e a forma como é realizada a contextualização do ensino e se esta é eficiente para o ensino para os diferentes cursos.

Por meio dos resultados obtidos, podemos concluir que as abordagens analisadas utilizam, na maioria dos casos, metodologias de ensino comuns como aulas práticas e de laboratório, e o desenvolvimento de projetos pelos acadêmicos. Além disso, os recursos tecnológicos usados são bem distintos: ferramentas como sistemas de submissão online de exercícios e linguagens de programação como C++, Python, Processing e Grasshopper foram citadas. Os conteúdos abordados em sala de aula possuem a fundamentação básica da programação em comum, mas também acabam tratando de tópicos diferentes como Dijkstra, Machine Learning, SQL, validação e verificação de software, entre outros. Entre os artigos selecionados, a realização de projetos foi a ferramenta mais utilizada para a contextualização do ensino com o curso do acadêmico.

Com base nos resultados dos estudos analisados, foi possível concluir que a utilização de uma abordagem voltada (ou contextualizada) para os cursos de formação dos estudantes é eficiente, visto que foi observado um aumento na taxa de aprovação e uma queda na taxa de evasão nas disciplinas. Com essa abordagem os estudantes também apresentaram um melhor entendimento sobre como aplicar a computação em suas áreas, e se sentiram mais motivados a aprender.

Como trabalhos futuros, sugere-se um estudo detalhando sobre como a adaptação de conteúdos e atividades das disciplinas introdutórias de algoritmos e lógica de programação pode ser feita para os diferentes cursos, especialmente com suporte tecnológico e de forma automática – conectando assim, o que é ensinado em sala de aula com os respectivos cursos dos estudantes.

Agradecimentos

Agradecemos ao apoio financeiro da FAPESC, Edital chamada pública FAPESC/CNPQ Nº 06/2016 apoio a infraestrutura de CTI para jovens pesquisadores, projeto T.O. Nº: 2017TR1755 - Ambientes Inteligentes Educacionais com Integração de Técnicas

Learning Analytics e Gamificação. O presente trabalho também foi realizado com apoio da UDESC.

Referências

Buchinger, D.; Cavalcanti, G. A. de S. e Hounsell, M. da S. (2014) “Mecanismos de busca acadêmica: uma análise quantitativa”, Revista Brasileira de Computação Aplicada, v. 6, n. 1, p. 108-120.

Chilana, P.; Alcock, C.; Dembla, S.; Ho, A.; Armstrong, B. e Guo, P. (2015) “Perceptions of non-CS majors in intro programming: The rise of the conversational programmer”. In IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (Atlanta, Georgia, USA, October 18-22, 2015). 2015.

Elarde, J. e Fatt-Fei, C. (2011) “Introductory computing course content: educator and student perspectives”. In Proceedings of the 2011 conference on Information technology education (SIGITE’11) (West Point, New York, USA, October 20-22, 2011). 2011.

Petersen, K.; Vakkalanka, S. e Kuzniarz, L. (2015) “Guidelines for conducting systematic mapping studies in software engineering: An update”, Information and Software Technology, v. 64, p.1-18.

Shoufan, A. (2016) “ABS Controller: An Introductory Case Study for Motivating Non-Major Students”. In Global Engineering Education Conference (EDUCON’16) (Abu Dhabi, UAE, April 10-13, 2016). 2016.

Simon, B.; Hanks, B.; McCauley, R.; Morrison, B.; Murphy, L. e Zander, C. (2009) “For me, programming is ...”. In Proceedings of the fifth international workshop on Computing education research workshop (ICER’09) (Berkeley, CA, USA, August 10-11, 2009). 2009.

Souza, D.M.; Batista, M. H. da S. e Barbosa, E. F. (2016) “Problemas e Dificuldades no ensino e na Aprendizagem de Programação: Um mapeamento sistemático”. Revista Brasileira de Informática na Educação, v. 24, n. 1, p. 39-52.