Dificuldades no Processo de Aprendizagem de Programação de Computadores: um Survey com Estudantes de Cursos da Área de Computação

Maurício Massaru Arimoto[1], **Weldrey Toneto de Oliveira**[1]

[1]Centro de Ciências Tecnológicas (CCT)
Universidade Estadual do Norte do Paraná (UENP)
Caixa Postal 261 – 86360-000 – Bandeirantes – PR – Brasil

{mauricioarimoto,weldreyoliveira}@gmail.com

Abstract. *Computer programming is an important activity that allows for the development of logical thinking and creativity, increasing the problem-solving ability of those who know how to program. However, programming learning requires a high degree of abstraction, as well as time and effort from the apprentice. In this paper we conduct a survey with students of Computer courses, trying to identify the barriers and difficulties related to programming learning. The results indicate the need for new approaches (combined with current teaching practices) that make programming learning more attractive, interactive and playful, instigating the interest and engagement of the students.*

Resumo. *A programação de computadores é importante para o desenvolvimento do raciocínio lógico e da criatividade do indivíduo, aumentando sua capacidade de resolução de problemas. Entretanto, a aprendizagem de programação exige alto grau de abstração, além de tempo e esforço de aprendizagem. Neste trabalho um survey é conduzido com estudantes de cursos da área de Computação visando identificar as barreiras e dificuldades relacionadas à aprendizagem de programação. Os resultados obtidos apontam para a necessidade de abordagens (aliadas às práticas atuais de ensino) que tornem a aprendizagem de programação mais atrativa, interativa e lúdica e que instiguem o interesse e o engajamento dos estudantes.*

1. Introdução

Disciplinas de programação de computadores são importantes para a formação dos alunos, pois abordam princípios de lógica e programação que visam desenvolver a capacidade de análise e de resolução de problemas [Rapkiewicz et al. 2006]. No entanto, o processo de aprendizagem de programação não é considerado trivial, uma vez que os conceitos abordados e a carga de conhecimentos agregados não são simples e de fácil compreensão, especialmente para os alunos iniciantes. Por consequência disso, a programação de computadores tem sido uma das principais disciplinas responsáveis pelo aumento da taxa de reprovação e evasão em cursos da área de Computação [Silva et al. 2015, Santiago 2016].

Santos e Costas [Santos 2006] destacam alguns problemas no processo de ensino e aprendizagem de programação, envolvendo desde a falta de motivação dos estudantes, que pode ocorrer devido às suas dificuldades no aprendizado, até a metodologia de ensino

adotada pelo professor em sala de aula. Tais problemas são uma das causas de desinteresse dos alunos pela programação e contribuem com o aumento das taxas de reprovação na disciplina. As reprovações causam atraso nos estudos, aumentando os índices de desistências nos cursos da área [Santiago 2016].

Apesar de existirem diversos estudos que tratam da problemática no ensino e aprendizagem de programação de computadores [Gomes et al. 2008, Campos 2009, Viegas et al. 2015, Souza et al. 2016], ainda não há clareza sobre quais são os reais problemas e obstáculos que dificultam sua aprendizagem.

Diante do exposto, o principal objetivo deste trabalho é identificar e compreender quais são os problemas e dificuldades enfrentados pelos alunos na aprendizagem de disciplinas relacionadas à programação de computadores. Um survey é conduzido junto aos alunos e recém-formados em cursos da área de Computação. Nesse contexto, a ideia é evidenciar quais são as necessidades que precisam ser tratadas de modo que medidas possam ser tomadas para auxiliar na aprendizagem de programação, e consequentemente contribuir com a diminuição do alto índice de reprovação e evasão.

O trabalho está organizado da seguinte forma: na Seção 2 apresentam-se os métodos e procedimentos adotados para o desenvolvimento do trabalho; na Seção 3 apresentam-se as etapas de planejamento e execução do survey; na Seção 4 apresenta-se uma síntese dos principais resultados obtidos; por fim, na Seção 5, apresentam-se as considerações finais sobre o trabalho realizado.

2. Metodologia

A pesquisa classifica-se como um estudo exploratório, pois tem como objetivo proporcionar maior familiaridade com a problemática e dificuldades dos estudantes na aprendizagem de programação de computadores, em que o foco principal deste tipo de pesquisa está no aprimoramento de ideias ou de descoberta de intuições [Gil 2002].

O trabalho também é classificado como uma pesquisa não-experimental, consistindo de uma pesquisa de levantamento [Wazlawick 2014] no qual os dados são adquiridos diretamente do ambiente por meio de um questionário. Nesse contexto, a pesquisa é caracterizada como um survey com o propósito de obter dados e informações relevantes por meio de uma amostra representativa de uma população.

A escolha pela condução de um estudo baseado em survey justifica-se pelo fato de envolver uma pesquisa em larga escala, abrangendo estudantes de diversos cursos da área de Computação dispersos em várias regiões do Brasil, sendo a amostra do estudo composta por alunos e recém-formados nesses cursos.

Os dados coletados por meio do survey são analisados de forma qualitativa e quantitativa. A análise dos dados é apoiada pelo software livre Weka[1] que dispõe de uma variedade de algoritmos de aprendizagem de máquina para a mineração de dados. Utiliza-se o algoritmo de classificação que separa os dados minerados em classes distintas para serem analisados de forma mais apropriada.

[1] cs.waikato.ac.nz/ml/weka/

3. Planejamento e Execução do Survey

Para a delimitação do público-alvo foram definidos apenas os cursos técnicos/superiores da área de Computação que possuíam em sua grade curricular disciplinas relacionadas à programação de computadores.

Como instrumento de coleta de dados foi proposto um questionário on-line com 38 questões estruturadas. O questionário foi composto por questões "fechadas"e "abertas". A maioria das questões fechadas foram definidas de acordo com a escala de *Likert* no qual cada participante do survey específica o grau de dificuldade ou nível de concordância em relação a uma determinada questão ou item investigado.

As questões abertas foram utilizadas para coletar dados adicionais para a pesquisa, tais como as linguagens de programação mais conhecidas pelos participantes, linguagens de programação que apresentam mais dificuldade de aprendizado, etc. Além disso, tais questões também foram utilizadas para sugestões e possíveis mudanças no processo de ensino e aprendizagem de programação.

Para apoiar na elaboração do questionário foi utilizado o *Google Forms*, uma ferramenta gratuita para criação de questionários on-line. Utilizou-se dessa ferramenta como forma de facilitar a distribuição do questionário e alcançar o maior número de participantes nas diversas regiões do Brasil.

Um estudo piloto foi conduzido envolvendo a aplicação do questionário com três turmas de cursos de Ciência da Computação e Sistemas de Informação, totalizando 55 participantes. Além disso, o questionário foi enviado para professores e especialistas da área para análise e validação. Correções e melhorias foram feitas no questionário, antes de sua submissão aos participantes da pesquisa.

Para a aplicação do survey, foi disponibilizado um link com o questionário em listas de e-mails, páginas e grupos de discussão nas redes sociais (como o Facebook) relacionados à área de pesquisa. Além disso, foram enviados e-mails para professores de instituições de ensino técnico/superior com informações da pesquisa, solicitando a participação de seus alunos. O questionário foi aplicado no período de Novembro/2017 a Março/2018, e ao final da pesquisa, foram obtidas 284 respostas.

Na Figura 1 sintetiza-se a amostra do survey de acordo com as regiões/estados dos participantes. A maior taxa de respostas são oriundas dos estados do Paraná (33%), São Paulo (28%), Minas gerais (16%), Santa Catarina (6%) e Alagoas (5%). O campo "Outros" inclui um número pequeno de respondentes de outros estados como Rio de Janeiro, Espírito Santo, Amazonas, Brasília, Rio Grande do Sul e Bahia (12%).

Na Tabela 1 apresenta-se um panorama geral do perfil dos respondentes, incluindo a faixa etária, o sexo, o tipo de instituição (pública ou privada) e o período do curso (noturno, diurno ou integral). Informações como faixa etária, período do curso e sexo foram cruzadas no intuito de identificar padrões e relacionamentos entre os dados e auxiliar na avaliação dos resultados.

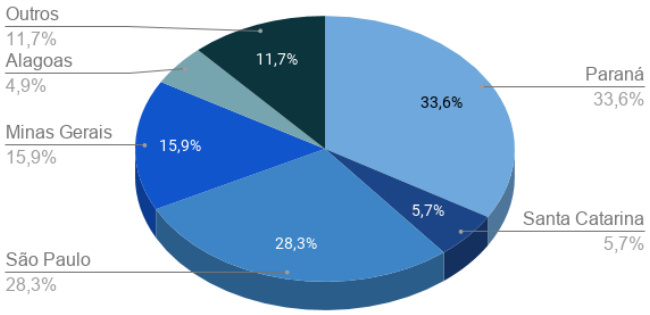

Figura 1. Estado/regiões dos participantes da pesquisa

Tabela 1. Panorama geral do perfil dos respondentes

Faixa Etária	**Respondentes**	**(%)**
17 a 20 anos	108	38%
21 a 24 anos	99	35%
25 a 28 anos	34	11%
Acima de 29 anos	43	15%
Sexo		
Masculino	218	77%
Feminino	66	23%
Instituição		
Pública	239	84%
Privada	45	16%
Período		
Diurno	20	7%
Noturno	182	64%
Integral	82	29%
Total	**284**	**100%**

4. Análise e Síntese dos Dados

A apresentação dos resultados do survey baseia-se na resposta de todos os participantes, incluindo os alunos e recém-formados nos cursos de Computação. A análise e os resultados obtidos são apresentados em três grandes categorias: (1) conceitos e práticas de programação, (2) ensino e aprendizagem de programação e (3) sugestões e melhorias.

4.1. Conceitos e Práticas de Programação

Na Tabela 2 apresentam-se os itens relacionados às questões sobre conceitos e práticas de programação abordados no questionário e o nível de dificuldade dos estudantes (%) em relação a esses itens. Tais níveis foram classificados segundo a escala de *Likert*, sendo: 1 - nenhum, 2 - muito baixo, 3 - baixo, 4 - alto e 5 - muito alto.

De acordo com a Tabela 2 pode-se constatar um índice maior de dificuldade nas questões que abordam: definição e uso de ponteiros, definição e uso de algoritmos

recursivos e definição e uso de estruturas de dados. Por exemplo, para o item definição e uso de ponteiros, os níveis de dificuldade considerados "muito alto" chegam a 34%, o que passa de 90 estudantes no total.

Na Figura 2 apresenta-se uma análise dos níveis de dificuldade dos estudantes em relação à definição e uso de estrutura de dados, levando-se em consideração (1) a faixa etária dos estudantes, (2) se durante o curso eles só estudam ou também trabalham e (3) o tipo de instituição (pública ou privada). Na análise, foi utilizado o algoritmo de classificação proposto pelo Weka, estruturando os dados na forma de árvores de decisão.

Tabela 2. Dificuldades em relação aos conceitos e práticas de programação

Conceitos e Práticas de Programação	**Nível de Dificuldade**				
	1	2	3	4	5
Lógica de programação	20%	29%	29%	16%	6%
Definição e uso de variáveis	41%	28%	22%	6%	3%
Definição e uso de estrutura de decisão	34%	26%	22%	13%	5%
Definição e uso de estrutura de repetição	30%	30%	23%	13%	4%
Definição e uso de funções/métodos	23%	23%	25%	19%	10%
Passagem de parâmetro (valor/referência)	20%	16%	25%	22%	17%
Definição e uso de arrays	22%	20%	24%	18%	16%
Manipulação de strings	20%	21%	28%	19%	12%
Definição e uso de ponteiros	11%	14%	19%	22%	34%
Definição e uso de algoritmos recursivos	11%	14%	19%	22%	34%
Definição e uso de estrutura de dados	10%	17%	24%	22%	27%

Pode-se observar na Figura 2 que em cada verificação são apresentados três valores. O primeiro valor é o nível de dificuldade mais assinalado pelos estudantes; logo em seguida, são indicados dois valores entre parênteses, o primeiro é o número total de respondentes naquela escala, enquanto o segundo representa o número de respondentes que assinalaram o nível de dificuldade diferente do apresentado.

Na primeira análise da Figura 2 encontra-se a faixa etária dos respondentes, que dentre os estudantes de dezessete e vinte anos (17-20), foi identificado o nível de dificuldade "5 - muito alto", seguido pelos valores 107 que correspondem ao total de estudantes naquela escala e 74 que são os estudantes que possuem um nível de dificuldade diferente de "5 - muito alto" (valores entre parênteses). O nível de dificuldade apresentado é "5" pois é verificado a opção mais assinalada pelos respondentes. Ao subtrair os 74 estudantes que assinalaram outros níveis de dificuldade do número total de estudantes nessa categoria (107), o resultado encontrado é 33, que foi o mais assinalado dentre os outros níveis de dificuldade.

Ao analisar os estudantes de faixa etária acima de vinte e nove anos (29-50), nota-se que foram realizadas outras análises, considerando se os estudantes só estudam ou estudam e trabalham, bem como o tipo de instituição. Percebe-se uma grande diferença

em relação às dificuldades dos estudantes que só estudam, para aqueles que precisam estudar e trabalhar ao mesmo tempo. Nesse contexto, para os estudantes que só estudam os níveis de dificuldade apresentados são “1 - nenhum” e “3 - baixo”, enquanto que para os estudantes que estudam e trabalham, os níveis de dificuldade correspondem a “4 - alto” e “5 - muito alto”. Os resultados “sugerem” que o estudantes que trabalham durante o curso acabam enfrentando mais dificuldades no aprendizado por terem menos tempo para se dedicar aos estudos.

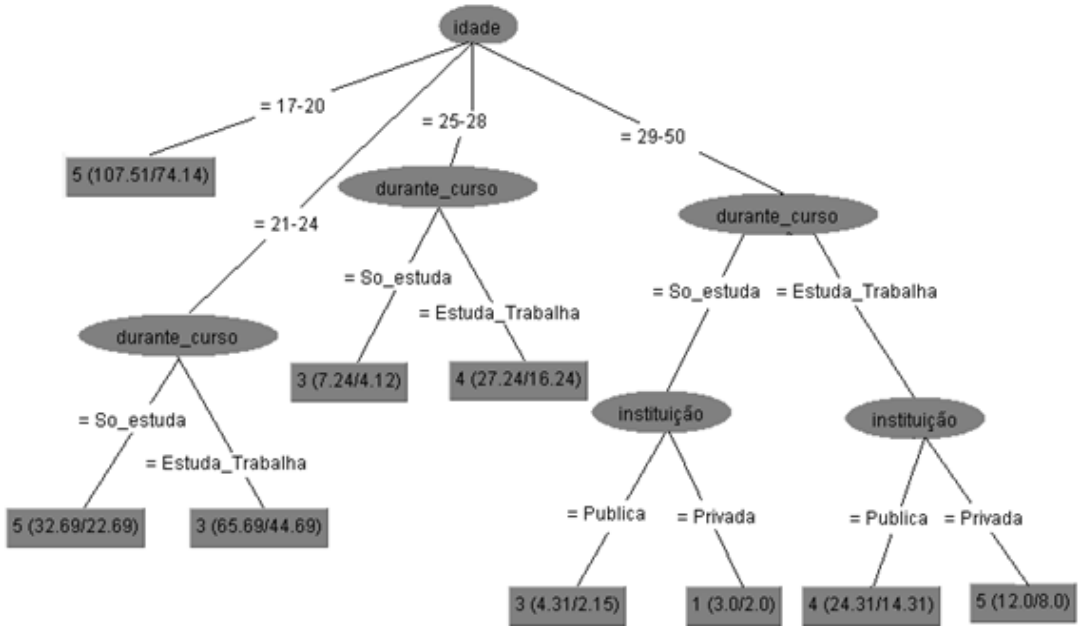

Figura 2. Definição e uso de estrutura de dados

Na Figura 3 apresenta-se uma análise do nível de dificuldade dos estudantes em relação à definição e uso de ponteiros usando o algoritmo de classificação, levando-se em consideração (1) o período do curso, (2) a faixa etária e (3) o sexo dos estudantes. Observa-se que os estudantes do período noturno apresentam um grau de dificuldade maior comparado aos estudantes de outros períodos. Isso também pode ter relação com o tempo dedicado ao estudo, já que a maioria dos estudantes do período noturno trabalham durante o dia.

Considerando o período integral, pode-se observar que os estudantes do sexo feminino possuem maior grau de dificuldade (4 - alto) e (5 - muito alto). O mesmo se aplica aos estudantes acima de 29 anos do sexo masculino, se comparados aos estudantes mais jovens também do mesmo sexo.

4.2. Ensino e Aprendizagem de Programação

No estudo conduzido por Gomes e Mendes [Gomes 2007], para muitos estudantes as dificuldades relacionadas ao aprendizado de programação começam na fase inicial de aprendizagem, quando eles precisam entender e aplicar conceitos abstratos de programação. Neste estudo, cerca de 55% dos respondentes afirmam que as dificuldades são mais frequentes no início da aprendizagem de programação, enquanto 21% dos respondentes são indiferentes quanto a isso.

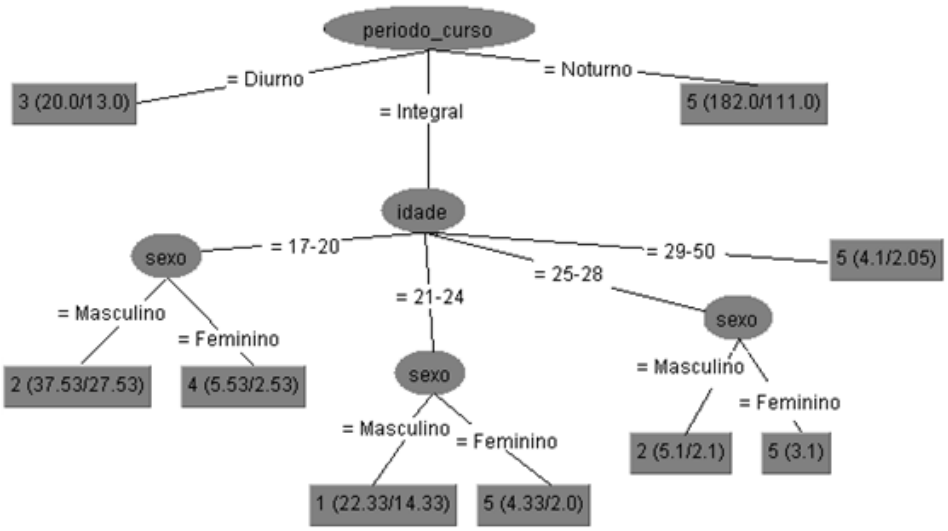

Figura 3. Definição e uso de ponteiross

Os estudantes foram questionados sobre a utilização ou não de ambientes e/ou ferramentas de apoio ao ensino e aprendizagem de programação. Mais da metade dos respondentes (60%) responderam que não era utilizado quaisquer ferramentas/ambientes de aprendizagem além da habitual sala de aula. Por outro lado, foram mencionadas a utilização de ambientes virtuais de aprendizagem como o Moodle, plataformas de cursos on-line como o Udemy, plataformas de envio e correção automática de exercícios de programação como URI e Run.codes, etc.

Em [Ramos et al. 2015] um estudo foi conduzido com o objetivo de verificar artefatos que influenciavam nas taxas de sucesso e reprovação nas disciplinas de programação. Como resultado, os autores relatam que utilização de ambientes/ferramentas de aprendizagem aliadas às metodologias de ensino, teve influência significativa na redução da taxa de reprovação nas disciplinas de programação.

Em relação à efetiva contribuição das ferramentas/ambientes de aprendizagem no ensino e aprendizagem de programação, neste estudo 46% dos respondentes "concordam totalmente" que tais ambientes contribuem de maneira significativa para a melhoria da aprendizagem, enquanto 32% deles "concordam parcialmente" (Figura 4).

No que se refere às dificuldades enfrentadas pelos estudantes na aprendizagem de programação, foi possível observar que a maioria concorda que tais dificuldades contribuem para os altos índices de evasão nos cursos de Computação, sendo que 45% concordam totalmente e 29% concordam parcialmente (Figura 5).

Por fim, os respondentes opinaram sobre as linguagens de programação que eles possuíam mais dificuldades. Dentre as linguagens mencionadas, destacam-se a linguagem Java (31%), seguido da linguagem C (23%) e C++ (13%). Em contrapartida, C# (4%) e Python (4%) foram as linguagens que os respondentes afirmam possuir menos dificuldades de aprendizagem.

Entretanto, alguns dos estudantes respondentes apontaram que sua maior dificuldade não está em entender ou aprender uma determinada linguagem de programação, mas sim em conseguir entender a lógica de programação.

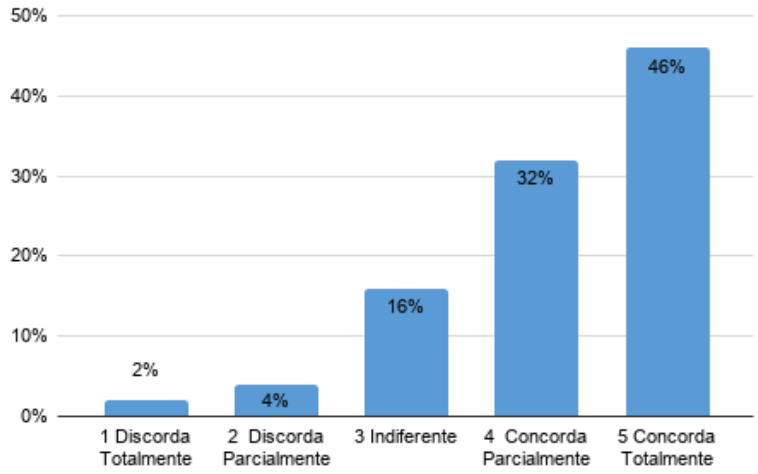

Figura 4. Contribuição dos ambientes/ferramentas de aprendizagem

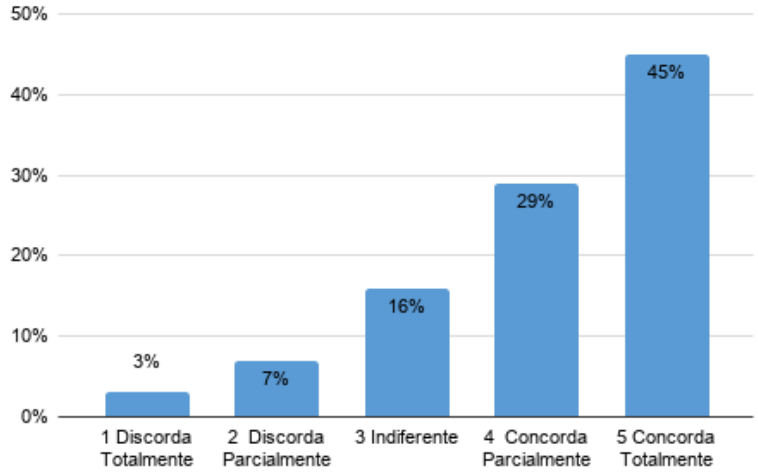

Figura 5. Evasão nos cursos de Computação devido às dificuldades no aprendizado

4.3. Sugestões e Melhorias

Os estudantes foram indagados sobre sugestões para melhorar e facilitar o entendimento dos conceitos abordados na programação. Para a análise dos resultados, foi criada uma categorização com um conjunto de sugestões geradas a partir da análise dos conteúdos das respostas dos estudantes. Na Figura 6, tais sugestões são apresentadas juntamente com a frequência em que foram citadas pelos respondentes. Dentre as sugestões, destacam-se: mudança no método de ensino (29%), aulas mais práticas (11%), uso de abstrações mais lúdicas (exemplos reais (10%), resolução de exercícios juntos com os alunos (10%), uso de linguagem de programação de fácil aprendizagem (no início do curso) (6%) e uso de ferramentas/ambientes de aprendizagem (6%).

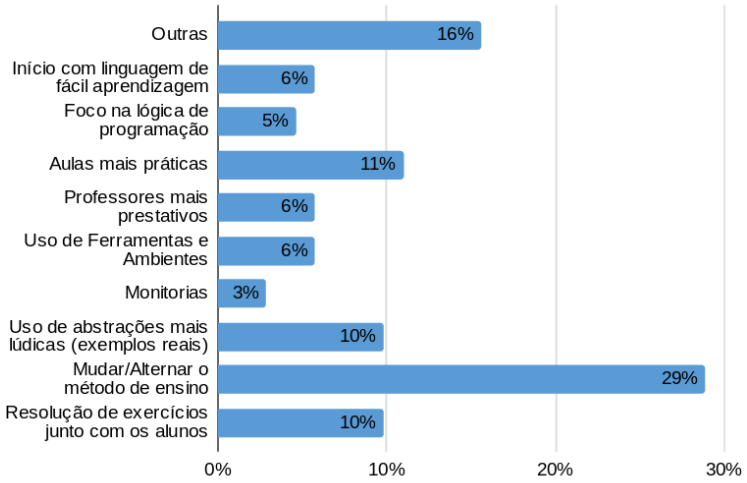

Figura 6. Sugestões e melhorias no ensino e aprendizagem de programação

Sabe-se que é muito difícil para o professor acompanhar individualmente o nível de aprendizado de cada aluno, o que leva muitos a ficarem atrasados e com dificuldades ao longo da disciplina de programação. Nesse sentido, os estudantes também foram questionados a opinar sobre quais seriam as mudanças necessárias para evitar que esse fato continue ocorrendo. Os resultados estão alinhados às sugestões apresentadas pelos estudantes. Dentre as mudanças mais indicadas, destacam-se: mais atividades de monitoria para auxiliar alunos/professores na disciplina (21%), mudança no método de ensino (14%) e uso de ferramentas/ambientes de aprendizagem (7%).

5. Conclusões

Neste trabalho foi apresentado um survey que teve como objetivo investigar as principais dificuldades e problemas enfrentados pelos estudantes na aprendizagem de disciplinas relacionadas à programação de computadores.

Os resultados obtidos evidenciam que a maioria dos estudantes (247 de um total de 284) possui certa dificuldade na aprendizagem de programação devido à sua inerente complexidade. As dificuldades variam desde o entendimento da lógica de programação até a definição e uso de estruturas de dados mais complexas. Tais resultados são semelhantes aos resultados de outros estudos realizados fora do contexto brasileiro [Gomes 2007, Gomes et al. 2008].

Diante do exposto, muitos estudantes acabam ficando desmotivados, constituindo um fator preponderante pela desistência do curso, além de contribuir com o aumento do índice de evasão em cursos da área de Computação (que já é elevado). De maneira análoga, estudos na literatura [Santos 2006, Santiago 2016, Kantorski et al. 2016, Pascoal et al. 2016] mostram que os altos índices de reprovação e desistências em cursos de Computação têm relação com as dificuldades dos estudantes no aprendizado das disciplinas de programação.

Dentre as sugestões dos participantes do survey para a melhoria do ensino e aprendizagem de programação, destaca-se a necessidade de mudanças na metodologia de ensino adotada em sala de aula. De acordo com [Fotaris et al. 2016] a abordagem

tradicional de ensino de programação ministrada em sala de aula e os materiais didáticos comumente disponibilizados aos estudantes não favorecem o seu engajamento na disciplina, nem fomentam a sua motivação e interesse em permanecer estudando programação. Sob essa perspectiva, abordagens lúdicas, técnicas de gamificação e robótica educacional emergem como meios de tornar a aprendizagem de programação mais atrativa e fomentar a motivação e o engajamento de estudantes.

Em complemento ao estudo conduzido neste trabalho, propõe-se investigar os índices de evasão e reprovação nos cursos de Computação. Pretende-se também propor uma ferramenta lúdica baseada em gamificação para apoiar o ensino e aprendizagem de conceitos introdutórios de programação. Ainda, a ideia é utilizar a robótica educacional em conjunto com a plataforma arduino como estratégia de ensino e aprendizagem mais lúdica, atrativa e menos abstrata, facilitando a introdução de conceitos de programação.

Referências

Campos, R. L. (2009). ERMC: Uma proposta de metodologia para melhoria do ensino-aprendizado de lógica de programação. In *XI Congreso Chileno de Educación Superior en Computación (CCESC)*, pages 1–5. Jornadas Chilenas de Computación.

Fotaris, P., Mastoras, T., Leinfellner, R., and Rosually, Y. (2016). Climbing up the leaderboard: an empirical study of applying gamification techniques to a Computer Programming Class. *The Electronic Journal of e-Learning*, 14(2):94–110.

Gil, A. C. (2002). *Como elaborar projetos de pesquisa*. São Paulo: Atlas, 4 edition.

Gomes, A., Areias, C. M., Henriques, J., and Mendes, A. (2008). Aprendizagem de programação de computadores: Dificuldades e ferramentas de suporte. *Revista Portuguesa de Pedagogia*, 42(2):161–179.

Gomes, A.; Mendes, A. J. (2007). Learning to program - difficulties and solutions. In *International Conference on Engineering Education (ICEE)*, pages 1–5.

Kantorski, G. Z., Flores, E. G., Hoffmann, I. L., Schmitt, J. A., and Barbosa, F. P. (2016). Predição da evasão em cursos de graduação em instituições públicas. In *Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 906–015.

Pascoal, T. A., Brito, D. M., Andrade, L. P., and Rêgo, T. G. (2016). Evasão de estudantes universitários: diagnóstico a partir de dados acadêmicos e socioeconômicos. In *Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 926–355.

Ramos, V., Freitas, M., Galimbert, M., Mariani, A. C., and Wazlawick, R. (2015). Comparação da realidade mundial do ensino de programação para iniciantes com a realidade nacional: Revisão sistemática da literatura em eventos brasileiros. In *Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 318–327.

Rapkiewicz, C. E., Falkemback, G., Seixas, L., Rosa, N. D., Cunha, V. V., and Klemann, M. (2006). Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. *Revista Novas Tecnologias na Educação (RENOTE)*, 4(2):1–11.

Santiago, A. D. V.; Kronbauer, A. H. (2016). Um Modelo lúdico para o ensino de conceitos de programação de computadores. In *Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 420–429.

Santos, R. P.; Costa, H. A. X. (2006). Análise de metodologias e ambientes de ensino para algoritmos, estrutura de dados e programação aos iniciantes em Computação e Informática. *INFOCOMP Journal of Computer Science*, 5(1):41–50.

Silva, T. S. C., Melo, J. C. B., and Tedesco, P. C. A. R. (2015). Um modelo para promover o engajamento estudantil no aprendizado de programação utilizando gamification. In *Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 71–80.

Souza, D. M., Batista, M. H., and Barbosa, E. F. (2016). Problemas e dificuldades no ensino e na aprendizagem de programação: Um mapeamento sistemático. *Revista Brasileira de informática na Educação (RBIE)*, 24(1):39–52.

Viegas, T. R., Okuyama, F. Y., Paravisi, M., and Bertagnolli, S. D. (2015). Uso das TICs no processo de ensino-aprendizagem de programação. pages 780–785.

Wazlawick, R. S. (2014). *Metodologia de pesquisa para Ciência da Computação*. Rio de Janeiro: Elsevier, 2 edition.