Grau de Dificuldade de Problemas de Programação Introdutória: Uma Revisão Sistemática da Literatura

Rodrigo Elias Francisco[1], **Cleon X. Pereira Júnior**[2], **Ana Paula Ambrósio**[3]

[1] Instituto Federal Goiano (IFGoiano)
Morrinhos – GO – Brasil

[2]Instituto Federal Goiano (IFGoiano)
Campos Belos – GO – Brasil

[3]Instituto de Informática – Universidade Federal de Goiás (UFG)
Goiânia – GO – Brasil

{frodrigo.francisco,cleon.juniorg}@ifgoiano.edu.br, apaula@inf.ufg.br

Abstract. *Computer Science 1 (CS1) presents alarming failure rates, being considered one of the disciplines that retain the most in the initial phase of computer education. Implementation of problem solutions is key to introductory programming teaching, and often means solving a large number of problems. This impacts the teacher's workload as he has to create problem lists and correct them. Due to several constraints, lists are generic, not taking into account student's specific needs. Aiming the automatic generation of personalized exercise lists, a classified problem database must be implemented. This article presents a Systematic Review of Literature focusing on strategies to define the level of difficulty of introductory programming problems. The review aimed to answer the following question: What is the most appropriate strategy to measure the difficulty of CS1 problems? At the end, it was possible to verify that researchers, when working on this theme, emphasized the concepts involved, aspects related to problem solving and error analysis.*

Resumo. *Ciência da Computação 1 (CS1) apresenta números alarmantes com relação à taxa de reprovação, sendo considerada uma das disciplinas que mais reprovam na fase inicial de formação em Computação. Resolver problemas com implementações é fundamental para a aprendizagem de programação introdutória, e quase sempre significa resolver um grande número de problemas. Isso aumenta a carga de trabalho do professor, pois ele precisa criar listas de problemas e corrigi-los. Devido a várias restrições, as listas são genéricas, não levando em conta as necessidades específicas do aluno. Visando a geração automática de listas de exercícios personalizadas, um banco de dados de problemas classificados precisa ser criado. Este artigo apresenta uma Revisão Sistemática de Literatura focada em estratégias para definir o grau de dificuldade de problemas de programação introdutória. A revisão teve como objetivo responder à seguinte pergunta: Qual é a estratégia mais adequada para medir a dificuldade dos problemas da CS1? Ao final, foi possível verificar que os pesquisadores, ao trabalharem sobre esse tema, enfatizaram os conceitos envolvidos, aspectos relacionados à solução de problemas e análise de erros.*

1. Introdução

A disciplina introdutória de programação, referenciada na literatura como CS1 (*Computer Science 1*), é fundamental para a formação de profissionais de diversas carreiras, como Computação e Engenharia. [Watson and Li 2014], a partir de uma Revisão Sistemática da Literatura, encontraram uma taxa de 32,3% de alunos que não são aprovados na disciplina de CS1. Em um cenário nacional, [Bosse and Gerosa 2015] mostraram que na USP (Universidade de São Paulo), de 2010 a 2014, essa disciplina teve um percentual médio de reprovações e trancamentos de 29,31%, sendo que este número se manteve praticamente constante e que é uma das que mais reprovam no primeiro semestre dos cursos de Computação, o que evidencia os desafios no ensino de CS1. No intuito de melhorar a situação, pesquisadores investigam questões que se relacionam com o ensino de programação. O uso de diversas linguagens e metodologias têm sido propostos, mas ainda é um problema em aberto.

No entanto, é comum entre os docentes, nas suas práticas de ensino, enfatizarem os aspectos do conhecimento de programação (sintaxe e semântica) em detrimento das estratégias de resolução de problemas, talvez porque as estratégias de resolução de problemas são consideradas bastante difíceis na percepção dos alunos [Malik and Coldwell-Neilson 2016]. Para facilitar este trabalho, ferramentas vêm sendo desenvolvidas para promover a habilidade de resolução de problemas e o desenvolvimento da lógica de programação [Noschang et al. 2014].

Algumas pesquisas realizadas foram capazes de encontrar relações cognitivas com a capacidade de aprendizagem de programação. [Martins et al. 2010] destacam o relacionamento da disciplina de CS1 com processos cognitivos como a criatividade e racionalidade, que necessitam de meta-habilidades como abstração e inferência, e se relacionam com habilidades de leitura, interpretação e contextualização. [Kramer 2007] também destaca a necessidade da habilidade de abstração por parte dos alunos.

Para desnvolver estas habilidades, o aluno precisa resolver muitos problemas. No entanto, preparar e corrigir extensas listas é muito dispendioso para os docentes, o que motiva investir em estratégias para automatizar o processo. Isto implica em ter bancos de problemas para uso nestes sistemas. Para permitir selecionar os problemas adequados para cada situação, é necessário que se tenha previamente classificado os problemas em categorias relevantes, como por exemplo: conteúdo, dificuldade, etc. O estudo apresentado colabora com uma estratégia maior, que visa propor cursos de CS1 personalizados, trabalhando com Juízes *Online* e Ambientes Virtuais de Aprendizagem (AVA), para serem usados nos ensinos presencial e a distância (EaD).

Esse cenário motiva a pesquisar sobre a dificuldade de problemas de CS1, o que contribui para estratégias didáticas mais efetivas, e.g. com avaliações contendo notas mais justas. Neste aspecto, será apresentada uma Revisão Sistemática da Literatura (RSL) sobre a dificuldade de problemas de CS1.

Este artigo está organizado como se segue. A Seção 2 apresenta a metodologia aplicada na RSL, dividida em planejamento, execução e resultados. A Seção 3 apresenta os resultados da RSL após os critérios de inclusão e exclusão. Por fim, na Seção 4 são apresentadas as conclusões obtidas através da RSL.

2. Metodologia

Esta RSL segue as instruções elaboradas por [Kitchenham 2004], cuja condução se subdivide em três etapas: Planejamento, Execução e Resultados (Figura 1).

Figura 1. Fases de uma RSL - adaptado de [Vieira et al. 2016]

2.1. Planejamento

Esta RSL tem como objetivo para responder à questão **Qual é a estratégia mais adequada para medir a dificuldade de problemas de CS1?**. Para isso, outras duas questões de pesquisa foram definidas:

- **QP1:** Quais estratégias já foram usadas para medir a dificuldade de problemas de CS1?
- **QP2:** Como validar a estratégia para medir a dificuldade de problemas de CS1?

Para responder às perguntas levantadas, foi realizada uma busca nas bases de artigos científicos, considerando os trabalhos de 2010 a 2016 (Com a data limite de 14/12/2016). Foram selecionadas as bases de pesquisa: *IEEEXplorer Digital Library*, *ACM Digital Library*, *Science Direct* e *Scopus*. Estas bases foram escolhidas pois disponibilizam mecanismo de consulta via *web*, estão relacionadas a temas de Computação e Informática e permitem filtro por ano de publicação. Foram considerados trabalhos publicados em português e inglês. A Tabela 1 descreve as *strings* usadas nas buscas.

Tabela 1. Strings de busca para a seleção de artigos

STRINGS DE BUSCA	
Idioma	String
Português	(”dificuldade”) AND (”problema”OR ”problemas”) AND (”programação introdutória”OR ”introdução a programação”OR ”ciência da computação 1”OR ”CS1”)
Inglês	(”difficulty”) AND (”problem”OR ”problems”) AND (”introductory programming”OR ”introduction to programming”OR ”computer science 1” OR ”CS1”)

A tabela 2 apresenta os critérios de inclusão e exclusão, que foram construídos considerando as questões de pesquisa.

2.2. Execução

Como mostra a Figura 1, a execução é composta de três etapas: identificação dos trabalhos, seleção e extração. A identificação dos trabalhos visa executar o plano da RSL e obter a lista de artigos. A seleção ocupa-se da leitura dos artigos realizada de uma maneira mais geral (título, palavras-chave e resumo). A extração exige que seja feita a leitura completa dos artigos, buscando realizar uma síntese com ênfase nas questões de pesquisa levantadas no planejamento.

Tabela 2. Critérios de Inclusão e Exclusão

Critérios	ID	Descrição
Exclusão	E1	Texto completo não disponível para acesso na Web
	E2	Escrito em outro idioma
	E3	Trabalho não aborda dificuldade de problemas de CS1
	E4	Não é um capítulo de livro com resumo ou artigo de periódico ou de conferência
	E5	Publicação duplicada
Inclusão	I1	Aborda estratégias usadas para medir dificuldade de problemas de CS1
	I2	Aborda validação de estratégias usadas para medir dificuldade de problemas de CS1

- A identificação dos trabalhos trouxe um total de 39 trabalhos.
- Na seleção, 24 trabalhos foram excluídos por não estarem ligados ao tema (E3) e 7 foram excluídos por estarem duplicados (E5), restando 8 artigos.
- Na extração, somente um trabalho foi removido, por não estar ligado ao tema (E3).

A Figura 2 mostra a distribuição dos 7 trabalhos, resultantes da extração, para as questões de pesquisa. É preciso considerar a legenda: I1 - Aborda estratégias usadas para medir dificuldade de problemas de CS1, e I2 - Aborda validação de estratégias usadas para medir dificuldade de problemas de CS1.

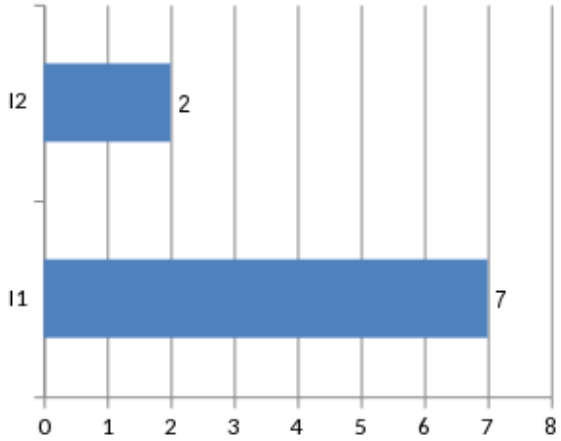

Figura 2. Trabalhos incluídos na extração conforme critérios

3. Resultados

A leitura dos trabalhos considerando os critérios da RSL permitiu realizar uma síntese dos dados. Isso possibilitou responder a cada uma das duas questões de pesquisa e formar uma visão geral sobre o assunto.

QP1: Quais estratégias já foram usadas para medir a dificuldade de problemas de CS1?

A Tabela 3 mostra os grupos de características encontrados nos trabalhos estudados.

A compreensão da dificuldade de problemas a partir dos erros ocorridos foi praticada nos trabalhos [Bryce et al. 2010, Cherenkova et al. 2014, Lakanen et al. 2015]. A partir de atendimentos de tutores a alunos visando assistência em atividades,

Tabela 3. Características encontradas nos trabalhos sobre dificuldade de problemas de CS1

CONJUNTO	CARACTERÍSTICA	REF.	Nº TRAB.
Erros	Erros dos alunos	[Bryce et al. 2010, Lakanen et al. 2015]	3
	Erros nos programas	[Cherenkova et al. 2014]	
	Cálculo de percentuais de acertos	[Cherenkova et al. 2014]	
Conceitos de programação	Sintaxe da linguagem	[Malik and Coldwell-Neilson 2016]	4
	Sequência	[Simon 2011]	
	Estruturas de seleção	[Cherenkova et al. 2014, Bryce et al. 2010]	
	Estruturas de repetição	[Bryce et al. 2010, Cherenkova et al. 2014] [Malik and Coldwell-Neilson 2016]	
	Matrizes unidimensionais	[Malik and Coldwell-Neilson 2016]	
	Matrizes bidimensionais	[Bryce et al. 2010]	
	Strings	[Bryce et al. 2010]	
	Funções	[Malik and Coldwell-Neilson 2016]	
	Entrada e saída em arquivos	[Bryce et al. 2010]	
Estratégias de pesquisa	Dificuldade analisada por professor	[Llana et al. 2012, Denny et al. 2015]	7
	Análise da Solução escrita pelo autor do problema	[Denny et al. 2015]	
	Resolução do problema para estimar a dificuldade	[Llana et al. 2012]	
	Estudo da dificuldade para o contexto da disciplina	[Bryce et al. 2010, Simon 2011, Cherenkova et al. 2014] [Malik and Coldwell-Neilson 2016]	
	Estudo da dificuldade para o contexto do problema	[Llana et al. 2012, Denny et al. 2015, Lakanen et al. 2015]	
Resolução de Problemas	Raciocínio lógico	[Bryce et al. 2010]	6
	Síntese a partir da leitura do código-fonte	[Simon 2011]	
	Lidar com o problema sem o computador	[Lakanen et al. 2015]	
	Compreensão das estratégias de resolução de problemas	[Malik and Coldwell-Neilson 2016]	
	Compreensão textual	[Denny et al. 2015, Cherenkova et al. 2014, Bryce et al. 2010] [Lakanen et al. 2015]	

[Bryce et al. 2010] categorizaram os principais erros dos estudantes com dificuldades na disciplina de CS1 ao escrever programas. Já [Lakanen et al. 2015] investigaram os erros dos alunos na resolução do problema *Soloway's Rainfall*, que pede a criação de um programa que leia números inteiros até quando a entrada for igual a 99999 e escreva a sua média. A partir de uma análise minuciosa, concluíram que os principais erros foram relacionados às tarefas (divisão por zero, contagem de entradas válidas, controle das repetições, cálculo da média envolvendo a sequência do algoritmo e entendimento do texto).

A pesquisa de [Cherenkova et al. 2014] sugere a tendência de certos tipos de erros nos programas que não são considerados pelas avaliações do sistema *CodeLab* (Sistema *web* de problemas de programação introdutória para turmas de programação introdutória), e.g. um problema que enfatiza estruturas de seleção possui o erro *Reversed comparison operator* executado por 28.52% dos estudantes. Por outro lado, [Cherenkova et al. 2014] calcularam os percentuais de acertos nas implementações visando identificar os problemas difíceis no sistema *CodeLab*.

Houve o interesse em associar a dificuldade dos problemas com os conceitos de CS1 nos trabalhos [Cherenkova et al. 2014, Bryce et al. 2010, Simon 2011, Denny et al. 2015]. [Bryce et al. 2010] concluem que os conceitos (estruturas de repetição, instruções *switch*, matrizes, entrada/saída em arquivos, *strings*) se relacionam com as dificuldades dos alunos de CS1 e que estas dificuldades envolvem o entendimento da lógica e da sintaxe, e [Cherenkova et al. 2014] identificaram, a partir de uma base de 266.852 registros, que os conceitos de CS1 que os alunos têm mais dificuldade são estruturas de seleção e estruturas de repetição. [Malik and Coldwell-Neilson 2016] aponta como dificuldade compreender as estruturas de programação, e relaciona os temas: estruturas de repetição, funções e vetores.

No entanto, [Simon 2011] fez pesquisas que confirmaram a dificuldade dos alunos

em entender a sequência dos algoritmos. Existem alunos que parecem acreditar que grupos de declarações são executados simultaneamente ao invés de sequencialmente e, por isso, não entendem atribuições de variáveis nos programas. Esse conceito é pré-requisito para que os alunos consigam entender os demais, e se relaciona com a dificuldade de aprender a sintaxe da linguagem de programação [Malik and Coldwell-Neilson 2016].

Houve o interesse dos trabalhos [Bryce et al. 2010, Cherenkova et al. 2014, Simon 2011, Malik and Coldwell-Neilson 2016] em buscar a dificuldade dos alunos na disciplina de CS1 como um todo. No entanto, a preocupação em levantar a dificuldade específica de cada problema foi verificada nos trabalhos [Denny et al. 2015, Lakanen et al. 2015, Llana et al. 2012]. [Lakanen et al. 2015] investigaram a resolução do problema *Soloway's Rainfall*.

A prática de professores medirem as dificuldades dos problemas foi verificada nos trabalhos [Llana et al. 2012, Denny et al. 2015]. [Denny et al. 2015] mediram as dificuldades de problemas inventados pelos alunos a partir da análise da descrição do exercício, da solução padrão escrita pelo autor e dos conceitos que haviam sido ensinados no curso até o momento em que a atividade foi inventada. Já [Llana et al. 2012], no projeto *FLOP*, usaram uma estratégia que solicita que professores resolvam os problemas para contribuir com a medição de sua dificuldade.

Segundo os trabalhos [Bryce et al. 2010, Simon 2011, Lakanen et al. 2015, Cherenkova et al. 2014, Denny et al. 2015, Malik and Coldwell-Neilson 2016], as questões relacionadas à resolução de problemas impactam na dificuldade. A capacidade de compreensão e o entendimento da lógica representam os maiores problemas relacionados às dificuldades dos alunos identificadas na pesquisa de [Bryce et al. 2010]. [Lakanen et al. 2015] identificaram a dificuldade no entendimento do texto a partir da resolução do problema *Soloway's Rainfall*. Entretanto, [Malik and Coldwell-Neilson 2016] trazem como dificuldade compreender as estratégias de resolução de problemas. Essas considerações são essenciais para a evolução da pesquisa.

Outros trabalhos realizados mostram a inclinação das pesquisas para essa linha que envolve a dificuldade de compreensão. [Cherenkova et al. 2014] se empenharam para levantar a dificuldade textual do problema por meio de análises de comprimento do texto, vocabulário e referências culturais, porém os resultados não chegaram a nenhuma conclusão. [Denny et al. 2015], para calcular a dificuldade de problemas de CS1, realizaram análises da descrição do exercício a partir da sua leitura.

Por outro lado, porém ainda envolvendo a resolução de problemas, [Simon 2011] menciona sobre a dificuldade dos alunos em ler um código-fonte e explicá-lo de maneira sintetizada e entende que essa dificuldade, em muitos casos, está relacionada à não compreensão da sequência do algoritmo.

QP2: Como validar a estratégia para medir a dificuldade de problemas de CS1?

A Tabela 4 resume as características encontradas nos trabalhos referentes à validação.

Existem ameaças que impedem de generalizar os resultados da pesquisa de

Tabela 4. Características encontradas nas validações dos trabalhos sobre dificuldade de problemas de CS1.

CARACTERÍSTICA	REF.	Nº TRAB.
Impossibilidade de generalizar informações	[Bryce et al. 2010]	1
Considerar conceitos ensinados no curso até o momento da invenção da atividade	[Denny et al. 2015]	1
Relacionar proporção de submissões corretas para cada problema com o nível de dificuldade	[Denny et al. 2015]	1

[Bryce et al. 2010] a todos os estudantes. Essas ameaças apareceram ao observar que: (a) existem erros adicionais que os alunos não conhecem, (b) a coleta de dados ocorreu apenas de alunos que visitaram o laboratório de tutoria, e (c) a eficácia do professor pode distorcer esses resultados. É preciso considerar essas informações na validação das estratégias.

[Denny et al. 2015] fizeram uma validação empírica para a classificação humana da dificuldade e a apontaram como satisfatória. A proporção de submissões corretas para cada problema foi usada, visando considerar que os exercícios mais fáceis possuem uma proporção maior de submissões bem sucedidas. Porém, a possibilidade de plágio pode ameaçar essa validação. Os autores encontraram uma relação entre a dificuldade do exercício inventado pelo aluno e seu desempenho no exame. Essa relação mostrou que os estudantes que criaram exercícios mais difíceis tiveram melhor desempenho no exame.

Ao medirem as dificuldades de problemas inventados pelos alunos, [Denny et al. 2015] consideraram os conceitos que haviam sido ensinados no curso até o momento em que a atividade foi inventada. Isso traz novas perspectivas para a área, trazendo questionamentos sobre a validade de diversas pesquisas que não observaram este detalhe.

QP: Qual é a estratégia mais adequada para medir a dificuldade de problemas de CS1?

Buscando responder a essa pergunta, sugere-se a necessidade de pesquisas com planejamento que considere as questões relacionadas à validação e que explorem as características relacionadas à dificuldade de problemas trazidas por esta RSL.

O estudo de [Bryce et al. 2010] apresenta ameaças que impactam diretamente na validação da pesquisa, e.g, a existência de erros adicionais que os alunos não conhecem, dados coletados apenas de alunos que visitaram o laboratório de tutoria e a eficácia do professor. Diferente da abordagem dos trabalhos [Llana et al. 2012, Denny et al. 2015], que usam a classificação da dificuldade pelo professor, é sugerido que a validação trabalhe com dados que representem as opiniões dos alunos.

[Denny et al. 2015] abordou o uso da informação de quais conceitos foram apresentados para os alunos até o momento da classificação da dificuldade, trazendo considerações importantes para a validação das pesquisas. Entende-se que a capacidade do aluno é variável no tempo e existem alunos com diferentes capacidades em uma sala de aula. Logo, é importante trabalhar com conjuntos de alunos quanto às suas habilidades e/ou capacidades cognitivas para ter informações mais reais sobre a dificuldade.

Quanto às características que relacionam-se com a dificuldade, é possível perceber uma inclinação em trabalhar com (1) conceitos envolvidos

[Bryce et al. 2010, Cherenkova et al. 2014, Simon 2011, Denny et al. 2015], (2) aspectos referentes à resolução de problemas [Bryce et al. 2010, Simon 2011, Lakanen et al. 2015, Cherenkova et al. 2014, Denny et al. 2015], envolvendo, e.g., a compreensão textual e a capacidade de síntese a partir da leitura do código-fonte, e (3) análise de erros [Bryce et al. 2010, Lakanen et al. 2015, Cherenkova et al. 2014].

Propõe-se, como resultado da QP, visando a evolução da pesquisa, a criação de instrumentos para avaliar a dificuldade de problemas de CS1 que considerem as reflexões trazidas por esta RSL. É importante lembrar que os instrumentos precisam ter reprodutibilidade, pois esta característica permitirá a comparação de diferentes contextos socioeconômico-culturais de maneira mais objetiva, e.g. usando estatística. Observa-se as dificuldades em reproduzir as pesquisas.

4. Conclusão

Tornar o ensino mais justo é desafio e obrigação da educação como um todo. No entanto, ter avaliações mais justas depende de atividades com pesos de dificuldade corretos e que o professor se dedique no processo ensino-aprendizagem. Sabe-se que programas de computadores são ricos em detalhes e que nem sempre o professor consegue dar *feedback* necessário ao aluno devido à sua alta carga de trabalho. Este cenário atual, por um lado, pode criar o perfil de um aluno mais autônomo e, por outro, traz a necessidade de investigar a prática docente na área de programação.

Ao estudar a possível estratégia para medir a dificuldade de problemas de CS1, depara-se com a questão da validação. A validação depende de dados que representem a dificuldade na percepção dos alunos. A subjetividade destes dados torna a tarefa bastante crítica, além da necessidade de considerar a capacidade do aluno.

Para que o aluno resolva corretamente um problema de CS1, ele precisa conhecer a linguagem de programação e a solução do problema. Isso relaciona-se com a explicação de [Cukierman et al. 2007], que aborda a multidisciplinaridade no trabalho da área de *software*, considerando que construir *software* não é só programar. Esse cenário traz o desafio de compreender a dificuldade de entender o problema de CS1 mesmo antes de iniciar a programação, o que se relaciona com as estratégias de resolução de problemas [Malik and Coldwell-Neilson 2016].

Diante deste cenário de pesquisa, diversas estratégias já foram realizadas. Por exemplo, [Oliveira et al. 2016] propuseram a ferramenta *NextStep*, cujo foco foi sequenciar de maneira inteligente e adaptativa os enunciados de CS1 usando grafos genéticos. A intenção foi, a partir de informações referentes ao modelo do aluno, sugerir a sequência de problemas mais adequada. Verificou-se que os autores acabaram trabalhando com a dificuldade de problemas de maneira indireta, ao solicitar que docentes realizassem uma sequência prévia, o que afirma a relevância desta RSL.

Essa pesquisa desencadeou novos questionamentos e interesses. A Computação pode ser compreendida a partir de diferentes camadas de abstração e dependendo de como se quer aplicá-la são alterados os detalhes a serem enfatizados, pois escrever um algoritmo paralelo para Inteligência Artificial é diferente de construir um Sistema de Informação Empresarial ou de um *Software* Embarcado para Robótica. No entanto, o conteúdo de CS1 é pré-requisito necessário para o aluno avançar nestas profissões. Isso mostra a im-

portância de investigar questões sobre a formação na área de Computação e suas relações com a prática profissional em diferentes perspectivas.

4.1. Trabalhos Futuros

Pretende-se: (a) verificar como são calculadas as dificuldades de problemas de outras áreas (e.g. Física e Matemática), pois pode contribuir com parte do problema; (b) criar um repositório de problemas de programação com diversos atributos (e.g. disciplina, características textuais e do código-fonte, interação problema-aluno), o que possibilitará realizar classificações conforme necessidade; (c) identificar o método de resolução de problemas usado pelo aluno e analisar sua relação com a dificuldade do problema percebido; (d) projetar um Juiz *Online*, usando padrões de Engenharia de *Software* que enfatizem comunicabilidade e manutenibilidade, que contribuam com este tipo de pesquisa ao acoplar o repositório (item b) e possibilite exportar dados anônimos para cientistas de dados.

Referências

Bosse, Y. and Gerosa, M. A. (2015). Reprovações e trancamentos nas disciplinas de introdução à programação da universidade de são paulo: Um estudo preliminar. In *XXIII WEI–Workshop sobre Educação em Informática. Recife, Julho*.

Bryce, R. C., Cooley, A., Hansen, A., and Hayrapetyan, N. (2010). A one year empirical study of student programming bugs. In *Frontiers in Education Conference (FIE), 2010 IEEE*, pages F1G–1. IEEE.

Cherenkova, Y., Zingaro, D., and Petersen, A. (2014). Identifying challenging cs1 concepts in a large problem dataset. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 695–700. ACM.

Cukierman, H. L., Teixeira, C., and Prikladnicki, R. (2007). Um olhar sociotécnico sobre a engenharia de software. *Revista de Informática Teórica e Aplicada*, 14(2):199–219.

Denny, P., Cukierman, D., and Bhaskar, J. (2015). Measuring the effect of inventing practice exercises on learning in an introductory programming course. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, pages 13–22. ACM.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

Kramer, J. (2007). Is abstraction the key to computing? *Communications of the ACM*, 50(4):36–42.

Lakanen, A.-J., Lappalainen, V., and Isomöttönen, V. (2015). Revisiting rainfall to explore exam questions and performance on cs1. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, pages 40–49. ACM.

Llana, L., Martin-Martin, E., and Pareja-Flores, C. (2012). Flop, a free laboratory of programming. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, pages 93–99. ACM.

Malik, S. I. and Coldwell-Neilson, J. (2016). A model for teaching an introductory programming course using adri. *Education and Information Technologies*, pages 1–32.

Martins, S. W., Mendes, A. J., and Figueiredo, A. D. (2010). Comunidades de investigação em programação: Uma estratégia de apoio ao aprendizado inicial de programação. *IEEE-RITA*, 5(1):39–46.

Noschang, L. F., Fillipi Pelz, E. A., and Raabe, A. (2014). Portugol studio: Uma ide para iniciantes em programaçao. *Anais do CSBC/WEI*, pages 535–545.

Oliveira, C. M., Pimentel, A., and Maschio, E. (2016). Nextstep: Um protótipo para o sequenciamento inteligente e adaptativo de enunciados em programação de computadores. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 5, page 1238.

Simon (2011). Assignment and sequence: Why some students can't recognise a simple swap. *Koli Calling '11*.

Vieira, M. A. et al. (2016). Modelagem de espaços inteligentes pessoais e espaços inteligentes fixos no contexto de cenários de computação ubíqua.

Watson, C. and Li, F. W. (2014). Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 39–44. ACM.