PCodigo II: O Sistema de Diagnóstico da Aprendizagem de Programação por Métricas de Software

Adler Neves[1], Márcia Gonçalves de Oliveira[1], Helen França Medeiros[1], Mônica Ferreira S. Lopes[1], Leonardo Leal Reblin[2], Elias Silva de Oliveira[2]

[1]Instituto Federal do Espírito Santo (Ifes) - Vitória - ES – Brazil
[2] Universidade Federal do Espírito Santo (Ufes) – Vitória, ES – Brazil

marcia.oliveira@ifes.edu.br, adlerosn@gmail.com,
elias_oliveira@acm.org

Abstract. *Evaluating programming exercises with the purpose of understanding learning difficulties and comparing solutions is a challenge for programming teachers. Looking at this challenge, we have developed PCodigo II, a system of assisted programming practice that maps programming solutions into learning profiles represented by 348 software metrics. The main functions of PCodigo II are mass execution of exercises, diagnosis of learning difficulties and analysis of plagiarism. PCodigo II is therefore an important tool to assist the evaluation work of teachers by providing them with reports that favors a fine and multidimensional analysis of programming learning.*

Keywords: *programming, learning analysis, PCodigo II*

Resumo. *A avaliação de exercícios de programação com as finalidades de compreender as dificuldades de aprendizagem e de comparar soluções representa um desafio para professores de programação. Contemplando esse desafio, desenvolvemos o PCodigo II, um sistema de apoio à prática assistida de programação que mapeia soluções de programação em perfis de aprendizagem representados por 348 métricas de software. As principais funções do PCodigo II são execução em massa de exercícios, diagnóstico de dificuldades de aprendizagem e análise de plágios. O PCodigo II apresenta-se, portanto, como uma importante ferramenta para auxiliar o trabalho de avaliação de professores ao fornecer-lhes relatórios que favorecem uma análise fina e multidimensional da aprendizagem de programação.*

Palavras-chave*: programação, análise de aprendizagem, PCodigo II*

1. Cenário de uso

O processo de ensino e de aprendizagem da programação de computadores é considerado complexo porque a programação é um conhecimento que para ser aprendido envolve a operacionalização de várias habilidades cognitivas e demanda extensa prática de

DOI: 10.5753/cbie.wcbie.2017.339

exercícios (Pea & Kurland, 1984). Dessa forma, as dificuldades de aprendizagem de programação têm sido associadas aos altos índices de reprovação e de evasão em cursos de informática e têm sido uma temática muito discutida na literatura acadêmica de Educação em Informática (Giraffa & Mora, 2013; Souza, Batista, & Barbosa, 2016).

Embora nas últimas décadas tenham sido desenvolvidas muitas metodologias e tecnologias para assistir o processo de ensino e de aprendizagem de programação, elas ainda são direcionadas, na maioria das vezes, para facilitar ou tornar atrativa a aprendizagem de programação. Há ainda poucas estratégias tecnológicas para assistir a prática da programação identificando as dificuldades de aprendizagem, monitorando as habilidades e intervindo no processo de aprendizagem. Além disso, considerando que a programação exige extensa prática de exercícios, o trabalho de avaliação dos professores de programação contemplando as funções diagnóstica, formativa e somativa da avaliação torna-se uma tarefa sobremaneira onerosa.

Com o objetivo de auxiliar professores de programação no acompanhamento da aprendizagem de seus alunos na prática da programação, contemplando os desafios de monitorar várias habilidades da programação em vários exercícios de vários alunos, desenvolvemos o *PCodigo II*, o sistema de diagnóstico da aprendizagem de programação por métricas de software. O *PCodigo II* foi desenvolvido para ser utilizado por professores de programação e reúne as seguintes funcionalidades:

i. Execução em massa de soluções de exercícios de programação a serem recebidos de uma interface *web do Moodle*. No atual protótipo do *PCodigo II*, ainda não há a integração com essa interface para receber automaticamente as submissões do *Moodle*. Dessa forma, a transferência das submissões de cursos de programação para o *PCodigo II* ainda está sendo realizada de forma manual até ser implementada a integração com a interface *web do Moodle*.

ii. Representação vetorial de perfis de alunos por 348 métricas de software que quantificam esforço e qualidade de programação a partir da análise dos códigos-fontes escritos por estudantes.

iii. Formação de grupos de perfis similares

iv. Análise de plágios

v. Emissão de relatórios em gráficos de mapas de calor (em integração)

Atualmente, o *PCodigo* realiza o mapeamento de perfis em métricas de softwares a partir de programas escritos em Linguagem C, mas já está sendo considerada uma adaptação para outras linguagens de programação.

O *PCodigo* II está sendo aplicado experimentalmente nas turmas do Curso a Distância de Programação C Essencial e Avançada do Centro de Referência em Formação e Educação a Distância (Cefor) do Instituto Federal do Espírito Santo (Ifes).

2. Desenvolvimento

O *PCodigo II* foi desenvolvido nas linguagens *Shell script*, PHP e R para rodar em ambientes Linux. Outros softwares e códigos livres que foram integrados ao *PCodigo II*

foram os seguintes: códigos das métricas de software (Berry & Meekings, 1985; Curtis, Sheppard, Milliman, Borst, & Love, 1979), o *script* de execução em massa do *PCodigo* original de Oliveira *et al. (2015)* e os algoritmos de *clustering* do *software Cluto 2.1.2*[1]. Também está sendo integrado ao *PCodigo II*, *scripts* em Linguagem R para geração de mapas de calor.

As etapas de desenvolvimento do *PCodigo* foram as seguintes:

i. **Construção do Núcleo Executor**: O *Núcleo Executor* recebe como entrada o diretório *submissoes* contendo vários subdiretórios com os arquivos das soluções em Linguagem C submetidas por alunos. Em seguida, copia-o para o diretório de processamento *questoesseparadas.* Um programa *shell script* foi desenvolvido para entrar em cada diretório de *questoesseparadas* e executar cada submissão com restrição de tempo e de tamanho do *arquivosaida* gerado.

ii. **Pré-Processamento:** o pré-processamento do *PCodigo II* consiste em calcular, para cada solução submetida para uma atividade de programação, os valores das métricas de software, gerar os vetores de perfis cujas dimensões são os valores dessas métricas e formar uma matriz para reunir esses vetores. As métricas foram calculadas a partir de *implementações de funções de* métricas de código-fonte (Berry & Meekings, 1985; Curtis et al., 1979). A matriz, por sua vez, foi normalizada a valores entre 0 e 1 por meio de *scripts* escritos na Linguagem de Programação *Python*. Esses valores entre 0 e 1 significam que todos os valores de uma métrica foram divididos pelo maior valor obtido dessa métrica em uma solução de programação. Dessa forma, o valor 1 significa que a métrica assume o seu valor máximo e, se for zero, assume o seu valor mínimo.

iii. **Núcleo de análise**: para implementação do *Núcleo de análise*, utilizamos os algoritmos e as visualizações de *clustering* do software *Cluto 2.1.2* (para realização dos processos de *clustering*) e um *script* em Linguagem R para geração da matriz de similaridade entre os vetores da matriz de submissões. O relatório de plágio é gerado por um *script* PHP que recebe como entrada a matriz de similaridade e a matriz com os vetores das submissões e retorna como saída os pares de submissões com índices de similaridades superiores a 90%. As saídas do *PCodigo II* são direcionadas para os subdiretórios *processado/metricas e processado/plagio*.

3. Apresentação do Software

O *PCodigo II* é uma extensão do *PCodigo*, que é um sistema de execução em massa e de análise de exercícios de programação submetidos por meio de ambientes virtuais ou outras interfaces web. O *PCodigo* original foi criado em 2015 com o objetivo de ser um instrumento de avaliação de apoio à prática assistida de programação. As principais funcionalidades do *PCodigo* são o mapeamento de perfis de estudantes em componentes de habilidades, análise de soluções divergentes e a análise de plágio.

[1] Software *Cluto* e Documentação: http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

O *PCodigo II* é um sistema que recebe de uma interface *web* soluções de atividades de programação submetidas por alunos, executa-as e emite relatórios de avaliação para professores. A arquitetura do *PCodigo II*, ilustrada na Figura 1, é formada por uma interface web (*Moodle*, por exemplo), um *Núcleo Executor,* um módulo de *Pré-processamento* e um *Núcleo de Análise*, assim como a versão original do *PCodigo* (Oliveira et al, 2015). No entanto, a representação de perfis de estudantes de programação no *PCodigo II* é diferente da representação de perfis do *PCodigo* original.

No *PCodigo* original, o mapeamento de perfis consistia em representar um programa em Linguagem C desenvolvido por um aluno em um vetor de *n* dimensões representadas pelos valores de componentes de habilidades, que quantificavam informações de códigos-fontes. As componentes de habilidades eram quantificadas pelo número de palavras reservadas, símbolos, operadores, comentários e *strings* e por indicadores de execução (variáveis lógicas com valores 0 e 1 que informavam se um programa compilava e executava) (Oliveira, Nogueira, & Oliveira, 2015)

No *PCodigo II* a representação de perfis é realizada por métricas de software que quantificam praticamente tudo que é escrito no código-fonte de um programa em Linguagem C, os indicadores de funcionamento e informações de qualidade, estilo e esforço de programação.

A ideia de se utilizar métricas de software na representação de perfis de estudantes de programação surgiu da necessidade de quantificar o trabalho de programação através de variáveis que indicassem, por exemplo: esforço, complexidade de código, número de variáveis, número de linhas de código, eficiência e número de funções. Para isso, utilizamos métricas de análise de códigos em Linguagem C (Berry & Meekings, 1985), as métricas de *Halstead* e *McCabe* (Curtis et al., 1979) e os indicadores de compilação e execução de programa (Oliveira et al., 2015). Ao todo, o *PCodigo II* implementa 348 métricas para análise de códigos e caracterização de perfis de estudantes.

De acordo com a Figura 1, o processamento do *PCodigo II* se inicia após os alunos realizarem a *Submissão* de soluções de programação na *Visão do aluno*.

A *Submissão*, conforme a Figura 1, deve conter um programa em Linguagem C formado por um ou mais arquivos, o *arquivoentrada* contendo as entradas desse programa e um arquivo *makefile* com as instruções de execução do programa submetido. Uma vantagem do modelo de submissão com *makefile* é a flexibilização do formato de submissão, que permite o aluno gerenciar a execução do seu programa desde o recebimento das entradas até a geração da saída (Oliveira et al., 2015)

Quando a *Submissão* de um aluno chega ao *Núcleo Executor* do *PCodigo II,* é gerada uma *Saída* com os arquivos do projeto em Linguagem C, o programa *Executável* e o *Arquivosaida* contendo os resultados de execução (Oliveira et al., 2015).

No módulo de *Pré-Processamento* da Figura 1, os arquivos de cada *Submissão* são recebidos pelo programa de *Cálculo de Métricas* que obtém, a partir das informações de código-fonte, os valores das 348 métricas de software caracterizando qualidade e esforço de programação. Essas métricas quantificam informações como, por exemplo: se um código compila e executa, variabilidade de palavras reservadas, esforço, dificuldade de programação, uso de funções, linhas de código e média de complexidade ciclomática

de funções. Em seguida, a partir dos valores das métricas, é gerado um vetor multidimensional representando um *Perfil* de aluno. Esse perfil junto com outros P*erfis de alunos* formam uma *Matriz A*, cujas linhas representam os *Perfis de alunos* e as colunas, as métricas de software. Em resumo, a *Matriz A* reúne as representações vetoriais de todas as submissões de alunos para um exercício de programação que poderão ser analisadas e comparadas no *Núcleo de Análise*.

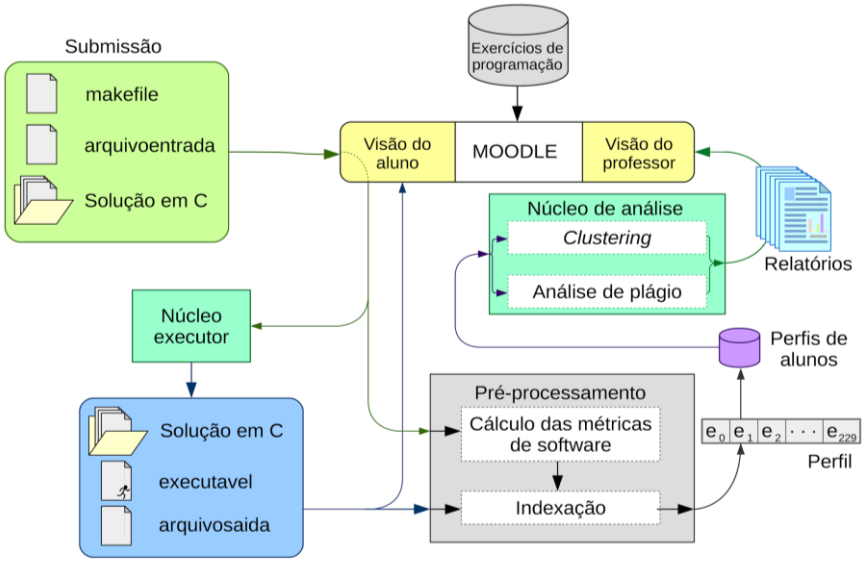


Figura 1. Arquitetura do *PCodigo II*

No módulo de *Análise de Plágio* do *Núcleo de Análise* da Figura 1, os vetores da *Matriz A* são comparados dois a dois e é gerada uma *Matriz As,* formada pelos índices de similaridade entre cada par de vetores calculados pela medida de similaridade *cosseno* (Baeza-Yates & Ribeiro-Neto, 2013). Os índices de similaridade variam de 0 (dissimilaridade) a 1 (similaridade total). Assumimos que duas submissões apresentam altas possibilidades de plágio quando o índice de similaridade ultrapassa o valor de 0.9, isto é, de 90% de similaridade.

Para executar o *PCodigo II*, basta seguir as orientações do roteiro de execução anexo aos arquivos deste artigo. A Figura 2 apresenta o arquivo de *makefile* que inicia a execução de todas as instruções do *PCodigo II* ao ser chamado dentro do diretório onde está inserido através da seguinte linha de comando em terminal Linux:

$ make

Os primeiros resultados de aplicação do *PCodigo II* em cursos a distância de Programação C Essencial e Avançada do *Cefor/Ifes* mostraram que é possível diagnosticar um processo de aprendizagem de programação em uma perspectiva multidimensional evidenciando dificuldades de aprendizagem, boas práticas de programação e até plágios de forma rápida, detalhada e holística (Oliveira et al., 2017).

```
makefile
default:
    make cleanpipeline
    make -C avaliador
    make delprocessado
    make copyoutput
    make cleanpipeline

cleanpipeline:
    # Limpando saída
    cd avaliador; ./remover.sh
    -rm -r avaliador/questoesmetrificas
    # Limpando entrada
    -rm -r avaliador/questoesseparadas
    -mkdir avaliador/questoesseparadas

copyoutput:
    # Copiando saída
    cp -r avaliador/indexados processado/metricas
    cp -r avaliador/analises/matrizes processado/plagio

delprocessado:
    # Limpando área de saída
    -rm -r processado
    -mkdir processado
```

Figura 2. *Makefile* de execução do *PCodigo II*

O gráfico de mapa de calor da Figura 3 e o relatório de plágio da Figura 4 apresentam os primeiros resultados gerados pelo *PCodigo II*.

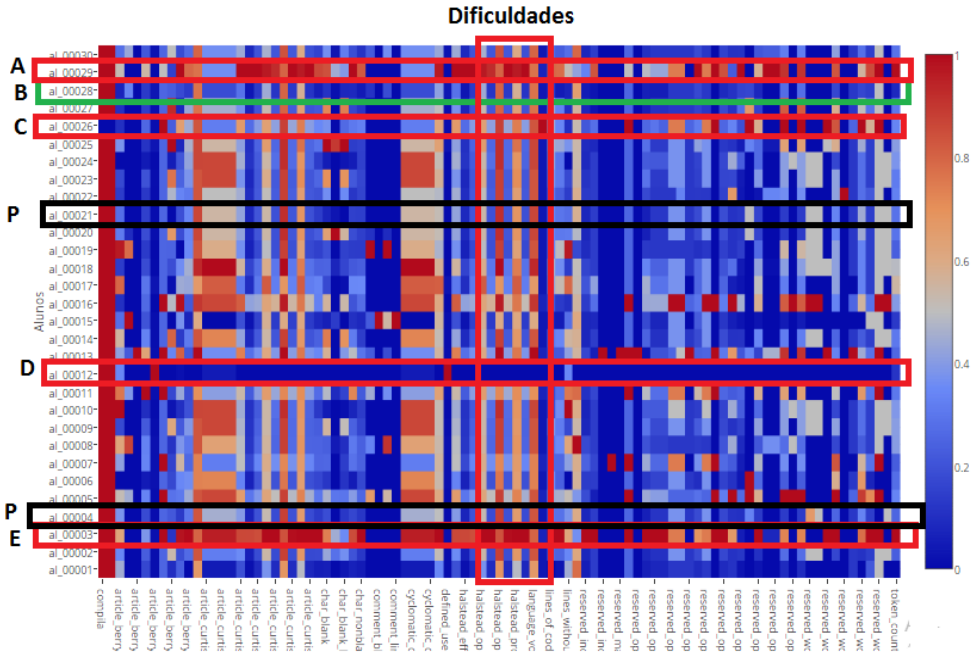
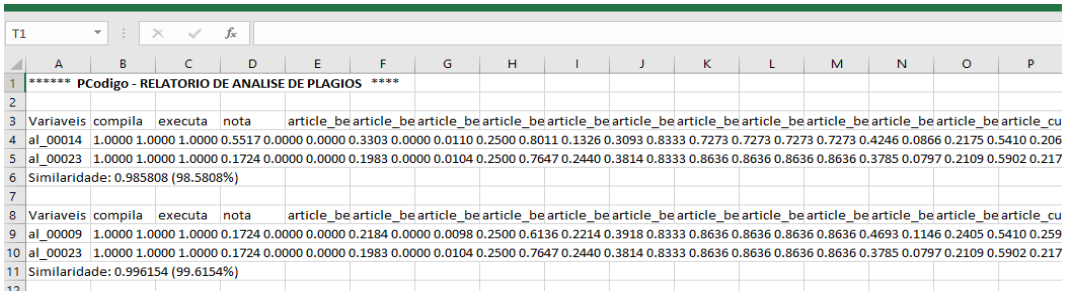


Figura 3. Mapa de calor das soluções de exercícios mapeadas em métricas de software



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	****** PCodigo - RELATORIO DE ANALISE DE PLAGIOS ****															
2																
3	Variaveis	compila	executa	nota	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_cu
4	al_00014	1.0000	1.0000	1.0000	0.5517	0.0000	0.0000	0.3303	0.0000	0.0110	0.2500	0.8011	0.1326	0.3093	0.8333	0.7273	0.7273	0.7273	0.7273	0.4246	0.0866	0.2175	0.5410	0.206
5	al_00023	1.0000	1.0000	1.0000	0.1724	0.0000	0.0000	0.1983	0.0000	0.0104	0.2500	0.7647	0.2440	0.3814	0.8333	0.8636	0.8636	0.8636	0.8636	0.3785	0.0797	0.2109	0.5902	0.217
6	Similaridade: 0.985808 (98.5808%)															
7																
8	Variaveis	compila	executa	nota	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_be	article_cu
9	al_00009	1.0000	1.0000	1.0000	0.1724	0.0000	0.0000	0.2184	0.0000	0.0098	0.2500	0.6136	0.2214	0.3918	0.8333	0.8636	0.8636	0.8636	0.8636	0.4693	0.1146	0.2405	0.5410	0.259
10	al_00023	1.0000	1.0000	1.0000	0.1724	0.0000	0.0000	0.1983	0.0000	0.0104	0.2500	0.7647	0.2440	0.3814	0.8333	0.8636	0.8636	0.8636	0.8636	0.3785	0.0797	0.2109	0.5902	0.217
11	Similaridade: 0.996154 (99.6154%)															

Figura 4. Relatório de análise de plágio

As soluções *A, C, D e E* evidenciam dificuldades de aprendizagem pelos altos valores de suas métricas (em cor vermelha), em especial nas métricas de *Halstead* (nas colunas rotuladas de *Dificuldades*), que indicam esforço e dificuldade de codificação. A *Solução C* não compila, conforme a cor azul indicando valor 0 (não compila) na primeira coluna do gráfico de mapa de calor. A *Solução D* também sinaliza dificuldades não pelo excesso de instruções, mas pela escassez delas (predomínio da cor azul). Já as soluções *P* são reconhecidas como plágios no relatório de análise de plágio. A *Solução B* foi considerada a melhor solução pelas baixas taxas nos valores das métricas, mas não uma solução 100% correta, segundo o professor da turma.

O mapa de calor da Figura 3 foi importante para o professor concluir que a turma inteira teve dificuldades nesse exercício, conforme evidenciam as métricas de *Dificuldades*. Uma observação nesse mapa de calor é que algumas métricas têm valores similares. Dessa forma, um melhoramento que pode ser realizado na composição do gráfico é a seleção das métricas mais relevantes para auxiliar a avaliação de professores.

Destacamos a seguir os principais arquivos de documentação do *PCodigo II*:

- Vídeo de apresentação do *PCodigo II*: https://www.youtube.com/watch?v=ZkXKl8bPFzA
- Vídeo que apresenta uma explicação e uma demonstração mais detalhada do funcionamento do *PCodigo II* em https://youtu.be/xVupJTDfn4I
- O roteiro de instalação do software *PCodigo II* está no *link* https://drive.google.com/file/d/0B0kktp6kkP9CbFd4c0xwLVhpalE/view
- Para fazer o *download* do *PCodigo II*, acesse o *link* https://drive.google.com/open?id=0B0kktp6kkP9CMkotLWlMc1RtWjA

Obs.: No diretório *submissoes*, há alguns exemplos de bases de exercícios para testes.

4. Considerações finais

A principal contribuição do *PCodigo II* para o ensino e para a aprendizagem de programação é oferecer um amplo leque de variáveis de avaliação para uma análise fina e multidimensional da aprendizagem de programação a partir das informações de códigos-fontes. Através dos relatórios do *PCodigo*, professores de programação, antes

mesmo de corrigirem os exercícios, poderão ter uma visualização geral de seus alunos pela lente das métricas de software.

Os próximos passos para avançar no desenvolvimento do *PCodigo II* são a integração dos *scripts* de visualização da informação dos resultados das métricas de software, a integração da ferramenta ao ambiente virtual de aprendizagem *Moodl*e e desenvolver um mecanismo para selecionar apenas as principais métricas que caracterizem exercícios de programação visando facilitar a emissão de um parecer sobre a aprendizagem dos alunos de uma forma mais resumida.

Concluindo, o *PCodigo II* mostrou-se uma ferramenta muito útil para avaliação diagnóstica de programação que possibilita professores entenderem como seus alunos programam e por que eles têm dificuldades e, a partir dessa compreensão, reorientar o ensino de forma a promover êxitos coletivos de aprendizagem.

5. Agradecimentos

À Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES) e à Pró-reitoria de Pesquisa e Pós-Graduação do Instituto Federal do Espírito Santo (PRPPG/Ifes).

Referências

Baeza-Yates, R. A., & Ribeiro-Neto, B. (2013). *Recuperação de Informação: Conceitos e Tecnologia das Máquinas de Busca* (2a. Edição). Bookman Editora.

Berry, R. E., & Meekings, B. A. E. (1985). A style analysis of C programs. *Commun. ACM*, *28*, 80–88.

Curtis, B., Sheppard, S. B., Milliman, P., Borst, M. A., & Love, T. (1979). Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics. *IEEE Trans. Softw. Eng.*, *5*(2), 96–104.

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

Oliveira, M., Neves, C. A., Lopes, M. F. S., Medeiros, F. H., Andrade, M. B., & Reblin, L. L. (2017). Um Curso de Programação a Distância com Metodologias Ativas e Análise de Aprendizagem por Métricas de Software. *RENOTE - Revista Novas Tecnologias Na Educação*, *15*(1).

Oliveira, M., Nogueira, M. de A., & Oliveira, E. (2015). Sistema de apoio à prática assistida de programação por execução em massa e análise de programas. In *CSBC 2015-Workshop de Educação em Informática (WEI)*. Recife, PE: SBC.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*(2), 137–168.

Souza, D. M., Batista, M. H. da S., & Barbosa, E. F. B. (2016). Problemas e Dificuldades no Ensino e na Aprendizagem de Programação: Um Mapeamento Sistemático. *Revista Brasileira de Informática Na Educação*, *24*(1).