Informática Aplicada à Educação

Everson Mario Novak
(Organizador)

Editora Atena
Ano 2018

Everson Mario Novak
(Organizador)

Informática Aplicada à Educação

Atena Editora
2018

Editora Chefe: Profª Drª Antonella Carvalho de Oliveira

Edição de Arte e Capa: Geraldo Alves e Natalia Sandrini
Revisão: Os autores

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

I43 Informática aplicada à educação [recurso eletrônico] / Organizador Everson Mario Novak. – Ponta Grossa (PR): Atena Editora, 2018.
10.596 kbytes

Formato: PDF
Requisitos de sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
Inclui bibliografia
ISBN 978-85-85107-14-7
DOI 10.22533/at.ed.147181308

1. Educação. 2. Informática. 3. Tecnologia educacional. I. Novak, Everson Mario.

CDD 371.334

Elaborado por Maurício Amormino Júnior – CRB6/2422

www.atenaeditora.com.br
E-mail: contato@atenaeditora.com.br

APRESENTAÇÃO

Este livro foi divido em 3 eixos, fruto de pesquisa científica de ótima qualidade acadêmica sobretudo por equipes multidisciplinares e de diversas instituições. Os trabalhos realizados são para auxiliar na Educação a distância e presencial, utilizando recursos computacionais para o planejamento e desenvolvimento de aplicativos para apoiar o aprendizado de matemática e de atividades cotidianas para crianças autistas, desenvolvimento de jogos educacionais e ainda para avaliar os dados armazenados em LMS (Learning Management Software) da plataforma Moodle.

No primeiro eixo temos o desenvolvimento de softwares e aplicativos voltados para a EAD, iniciamos por uma aplicação m-learning Genius para o auxiliar no ensino de matemática na educação infantil, explorando formas geométricas, números e a adição e subtração através de figuras e sons. O ENEN foi tema de um aplicativo focado em preparar os alunos na disciplina de matemática. O relacionamento social, comunicação e alterações de comportamento do autista são o tema de estudo para o desenvolvimento de um aplicativo para auxiliar os autistas no aprendizado e no relacionamento social.

A Cloud Computing apoia a aprendizagem em ambientes U-learning para verificar os estilos de aprendizagem e aplicabilidade em ambientes educacionais. As métricas de software são utilizadas para fazer uma análise da aprendizagem em cursos de programação a distância. Uma base de conhecimento gerada das questões e códigos inseridos nas plataformas digitais de ensino, foi feita a classificação de códigos da linguagem C em medidas similares para fazer os agrupamentos para formação de uma base de questões com códigos e soluções associadas para correções de questões de forma automatizada.

O segundo eixo entra em jogos digitais e gamificação, auxiliam na aprendizagem de pessoas com deficiência visual, tenta garantir no processo pedagógico uma inclusão digital e social destas pessoas. O processo de aprendizado utilizou-se dos jogos construcionistas para propor quatro jogos educativos, simplificando a complexidade na sua criação. Problemas motivacionais dos alunos são tratados na gamificação para verificar o que ocorre em processos de aprendizagem em ambientes educacionais.

No terceiro e último eixo é abordada a aprendizagem de máquina (machin-learning), aplicada a educação e aprendizado. O conceito de Estilos de Aprendizagem (EA) da psicologia cognitiva e da pedagogia, são propostos em sistemas educacionais adaptativos, com algumas aplicações da Aprendizagem por Reforço, foi proposto uso de algoritmos relacionados a aprendizagem de máquina para obter os estilos de Aprendizagem. Aplicabilidade de modelos de Regressão Múltipla no contexto da EAD foi abordado para validar as variáveis de comportamento de autorregulação da aprendizagem na plataforma LMS – Moodle.

Ao escrever este prefácio contextualizei o alinhamento das análises e teorias desenvolvidas nos artigos contidos neste livro. Sugiro que o leitor faça este caminho para uma compreensão ampla destes trabalhos, agradeço a oportunidade de fazer parte de grupo e felicito a todos os integrantes.

Everson Mario Novak
Mestrando em Informática - PUCPR

SUMÁRIO

EIXO 1: SOFTWARES E APLICATIVOS VOLTADOS PARA A EAD

CAPÍTULO 1 1

GENIUS MATH: UMA APLICAÇÃO MOBILE PARA AUXILIAR A APRENDIZAGEM DA MATEMÁTICA NA PRÉ-ESCOLA

Stefane Vieira Menezes
Jiani Cardoso da Roza

CAPÍTULO 2 13

APLICATIVO MÓVEL PARA PREPARAÇÃO DE ESTUDANTES PARA O ENEM NO CONTEXTO DA DISCIPLINA DE MATEMÁTICA

Hannderson Faria Arantes
Rodrigo Duarte Seabra

CAPÍTULO 3 27

COTIDIANO: UM SOFTWARE PARA AUXILIAR CRIANÇAS AUTISTAS EM SUAS ATIVIDADES DIÁRIAS

Afranio Furtado de Oliveira Neto
Hugo Leonardo Pereira Rufino
Diovane de Godoi Beira
Rodolfo Bocado Palis
Paula Teixeira Nakamoto

CAPÍTULO 4 41

APRENDIZAGEM SIMULADA NA NUVEM

Rafaela R. Jardim
Roseclea Duarte Medina
Giliane Bernardi
Fabricio Herpich
Andressa Facalde
Eduardo Lemos

CAPÍTULO 5 55

ANÁLISE DA APRENDIZAGEM DE PROGRAMAÇÃO POR MAPEAMENTO DE PERFIS EM MÉTRICAS DE SOFTWARE

Márcia Gonçalves de Oliveira
Ádler Oliveira Silva Neves
Helen França Medeiros
Mônica Ferreira Silva Lopes
Leonardo Leal Reblin
Elias Silva de Oliveira

CAPÍTULO 6 68

CLASSIFICAÇÃO DE CÓDIGOS C USANDO MEDIDAS DE SIMILARIDADE PARA APOIO AO ENSINO DE PROGRAMAÇÃO

José Carlos Campana Filho
Elias Silva de Oliveira
Márcia Gonçalves de Oliveira

EIXO 2: JOGOS DIGITAIS E GAMIFICAÇÃO

CAPÍTULO 7 79
BEM EXPRESSÕES: JOGO DIGITAL VOLTADO PARA O ENSINO INCLUSIVO DA MATEMÁTICA

André Luis Bitencourt Fernandes
Claudia Pinto Pereira
Kayo Costa de Santana
Ana Jaize de Oliveira Silva Santos
Bruno Gonzaga de Mattos Vogel

CAPÍTULO 8 95
JINDIE: UMA LINHA DE PRODUTO DE SOFTWARE PARA JOGOS EDUCATIVOS COM FOCO NO CONSTRUCIONISMO

Carlos Alberto Correia Lessa Filho
Arturo Hernandez Dominguez

CAPÍTULO 9 107
METODOLOGIAS GAMIFICADAS PARA A EDUCAÇÃO: UMA REVISÃO SISTEMÁTICA DETECÇÃO AUTOMÁTICA DE ESTILOS DE APRENDIZAGEM: UMA ANÁLISE COMPARATIVA DE CLASSIFICADORES APLICADOS EM UM CENÁRIO REAL DE APRENDIZADO

André Luiz de Souza Brito
Charles Andryê Galvão Madeira

EIXO 3: APRENDIZAGEM DE MÁQUINA APLICADA A EDUCAÇÃO

CAPÍTULO 10 120
DETECÇÃO AUTOMÁTICA DE ESTILOS DE APRENDIZAGEM: UMA ANÁLISE COMPARATIVA DE CLASSIFICADORES APLICADOS EM UM CENÁRIO REAL DE APRENDIZADO

Lucas Daniel Ferreira
José Fernando Rodrigues Jr

CAPÍTULO 11 140
DETECÇÃO AUTOMÁTICA E DINÂMICA DE ESTILOS DE APRENDIZAGEM EM SISTEMAS ADAPTATIVOS E INTELIGENTES PARA A EDUCAÇÃO UTILIZANDO DYNAMIC SCRIPTING

Júlio César da Costa Silva
Cristiano Grijó Pitangui
Alessandro Vivas Andrade
Luciana Pereira de Assis
Cristiano Maciel da Silva

CAPÍTULO 12 156
UM PROCESSO DE VALIDAÇÃO DE VARIÁVEIS COMPORTAMENTAIS DE AUTORREGULAÇÃO DA APRENDIZAGEM EM PLATAFORMAS DE LMS

Rodrigo Lins Rodrigues
João Carlos Sedraz Silva
Jorge Luis Cavalcanti Ramos
Fernando da Fonseca de Souza
Alex Sandro Gomes

SOBRE O ORGANIZADOR 166

EIXO 1 – SOFTWARES E APLICATIVOS VOLTADOS PARA A EAD

APRESENTAÇÃO

No primeiro eixo temos o desenvolvimento de softwares e aplicativos voltado para EAD, iniciamos por uma aplicação m-learning Genius para o auxiliar no ensino de matemática na educação infantil, explorando formas geométricas, números e a adição e subtração através de figuras e sons. Com atividades lúdicas viabilizando práticas contemporâneas ao cotidiano infantil.

Agora abordando outro tema pertinente o ENEN, um aplicativo focado em preparar os alunos para o Exame Nacional do Ensino Médio na disciplina de matemática.

As dificuldades apresentadas em relacionamento social, comunicação e alterações de comportamento por um autista são o tema de estudo para o desenvolvimento de um aplicativo para auxiliar os autistas no aprendizado e no relacionamento social.

A Cloud Computing está apoiando a aprendizagem em ambientes U-learning, criando um laboratório virtual U-Lab Cloud para verificar os estilos de aprendizagem para adotar a tecnologia em ambientes educacionais.

O software PCodigo II, utiliza métricas de software para fazer a análise da aprendizagem em cursos de programação a distância, para que sejam observadas dificuldades de aprendizagem, boas práticas de programação e perfis de aprendizagem de forma rápida, detalhada e holística.

Neste outro tema é gerado uma base de conhecimento de forma organizada das questões e códigos gerados nas plataformas digitais de ensino a distância. Abordando uma classificação de códigos da linguagem C baseada em medidas similares para fazer os agrupamentos para formação de uma base de questões com códigos e soluções associadas para correções de questões de forma automatizada.

Everson Mario Novak
Mestrando em Informática - PUCPR

CAPÍTULO 5

ANÁLISE DA APRENDIZAGEM DE PROGRAMAÇÃO POR MAPEAMENTO DE PERFIS EM MÉTRICAS DE SOFTWARE

Márcia Gonçalves de Oliveira
Instituto Federal do Espírito Santo (Ifes)
Vitória - ES

Ádler Oliveira Silva Neves
Instituto Federal do Espírito Santo (Ifes)
Vitória – ES

Helen França Medeiros
Instituto Federal do Espírito Santo (Ifes)
Vitória – ES

Mônica Ferreira Silva Lopes
Instituto Federal do Espírito Santo (Ifes)
Vitória – ES

Leonardo Leal Reblin
Universidade Federal do Espírito Santo (Ufes)
Vitória - ES

Elias Silva de Oliveira
Universidade Federal do Espírito Santo (Ufes)
Vitória – ES

RESUMO Este trabalho apresenta o *PCodigo* II, um sistema de mapeamento automático de perfis de estudantes em métricas de software para análise da aprendizagem de programação Além do mapeamento de perfis em 348 métricas de software, o *PCodigo II* possui as funcionalidades de execução em massa, de agrupamento de perfis similares, de visualização da informação e de análise de plágios. As primeiras aplicações do *PCodigo II* em exercícios reais de cursos de programação a distância mostram que professores, atentando para o que as métricas informam, podem reconhecer as dificuldades de aprendizagem, boas práticas de programação e classes de perfis de aprendizagem de toda uma turma de forma rápida, detalhada e holística.
PALAVRAS-CHAVE análise de aprendizagem, métricas de software, programação.

ABSTRACT This work presents *PCodigo II*, a system of an automatic mapping of student profiles in software metrics to analyze programming learning. In addition to profile mapping in 348 software metrics, *PCodigo II* has mass execution, similar profile grouping, information visualization, and plagiarism analysis capabilities. The first applications of *PCodigo II* in real programming exercises demonstrate the effectiveness of this system for the diagnostic evaluation of programming learning. The first applications of *PCodigo II* in real programming exercises show that teachers, taking into account what the metrics say, can recognize the learning difficulties, good programming practices and classes of learning profiles of a whole class in a fast, detailed and holistic way.
KEYWORDS learning analysis, software measures, programming.

1 I INTRODUÇÃO

A avaliação da aprendizagem de programação com as finalidades de diagnosticar, regular e qualificar um processo de aprendizagem tem sido um verdadeiro desafio, uma vez que a análise de aprendizagem de programação é, de modo geral, baseada em indicadores subjetivos. Dessa forma, a carência de indicadores padronizados inviabiliza uma avaliação para reconhecimento de dificuldades de aprendizagem, habilidades e até competências em programação. Por conseguinte, a automatização desse processo também é dificultada.

Embora já existam soluções de avaliação automática de programação (PIETERSE, 2013), uma importante questão a ser discutida é a seguinte: em vez de aplicar testes padronizados, acatar a avaliação subjetiva de professores ou buscar na verbalização de estudantes possíveis indicadores de aprendizagem, não seria mais viável uma avaliação a partir da análise dos códigos de programação sob diferentes métricas para inferir dificuldades, habilidades e competências em programação?

Uma alternativa para essa avaliação é o uso de métricas de software para análise de códigos-fontes visando quantificar esforço e qualidade de programação (CURTIS *et al.*, 1979, BERRY; MEEKINGS, 1985). Essas métricas podem ser utilizadas, conforme Pettit *et al.* (2015), como instrumentos de análise dos processos de programação a partir de códigos-fontes para auxiliar professores na avaliação de seus alunos.

Com o objetivo de analisar a aprendizagem de programação a partir de códigos-fontes para reconhecimento de dificuldades de aprendizagem, habilidades e até competências em programação, foi desenvolvido o *PCodigo II*, uma evolução do sistema *PCodigo* desenvolvido por Oliveira (2015) na sua forma de mapear perfis que consiste em representar soluções de programação desenvolvidas por estudantes em 348 métricas de software.

A principal contribuição deste trabalho para a Informática na Educação é propor um instrumento de apoio à avaliação que possibilite professores realizarem uma análise multidimensional da aprendizagem de seus alunos na prática da programação. Dessa forma, através de um amplo leque de métricas, professores poderão identificar classes de perfis de alunos, analisar indicadores de possíveis causas de dificuldades de aprendizagem e comparar soluções de programação em detalhes.

Para apresentar os fundamentos e funcionalidades do *PCodigo II*, este trabalho está organizado conforme a ordem a seguir. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve a arquitetura do *PCodigo II* e as principais métricas de software utilizadas na representação de perfis. Na Seção 4, destacam-se a aplicação do *PCodigo II* em cursos a distância de programação e os principais resultados. A Seção 5 conclui este trabalho destacando os principais achados, os trabalhos futuros e as considerações finais.

2 | TRABALHOS RELACIONADOS

Os principais trabalhos relacionados à proposta deste trabalho são os trabalhos de Oliveira *et al*. (2013), de Munson (2017) e de Oliveira (2015). O trabalho de Oliveira *et al*. (2013) mapeia informações de código C em componentes de habilidades. Além dessas componentes de habilidades, as métricas de avaliação de classificação automática *multi-label* foram utilizadas por Oliveira *et al*. (2013) como instrumentos de avaliação diagnóstica fornecendo importantes indicadores de dificuldades de aprendizagem a serem acompanhados por professores.

Um estudo mais recente visa encontrar métricas para determinar, no início de um curso, quais alunos podem estar em risco (MUNSON, 2017). Nessa proposta, os *logs* de atividades de programação gerados por um ambiente de programação foram utilizados para gerar um conjunto de dados de variáveis, como o tempo e a quantidade de erros.

O *PCodigo* de Oliveira (2015), por sua vez, é um sistema de apoio à prática assistida de programação que recebe soluções de programação submetidas por estudantes via interface web, executa-as e emite relatórios de avaliação para professores. As principais funcionalidades do *PCodigo* para apoiar o trabalho docente e favorecer a aprendizagem de programação são as seguintes (OLIVEIRA, 2015): executar programas em massa, representar perfis em componentes de habilidades e analisar códigos-fontes.

O *PCodigo II* herda as funcionalidades do *PCodigo* original de Oliveira (2015), evoluindo-o na representação de perfis e, conforme Oliveira *et al*. (2013), dá novos significados a diferentes métricas para avaliação diagnóstica da aprendizagem de programação visando identificar, assim como Munson (2017), alunos com dificuldades de aprendizagem.

3 | O PCODIGO II

O *PCodigo II* foi desenvolvido com a finalidade de ser um sistema de análise multidimensional da aprendizagem de programação a partir da análise de códigos-fontes escritos por estudantes. Para isso, cada programa escrito caracterizando um perfil foi representado por um vetor de 348 dimensões, onde cada dimensão representa uma métrica de software.

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 realizado através de variáveis de avaliação que indicassem, por exemplo: esforço de programação, complexidade de código, estilo de programação, número de variáveis, número de linhas, número de comentários, eficiência, complexidade e número de

funções.

O *PCodigo II* estende o *PCodigo* original de Oliveira (2015) na representação de perfis de estudantes pelas métricas de análise de códigos-fontes em Linguagem C (BERRY; MEEKINGS, 1985), pelas métricas de *Halstead* e *McCabe* (CURTIS *et al.*, 1979) e pelos indicadores de execução *compila* e *executa* de Oliveira (2015). Ao todo, o *PCodigo II* implementa 348 métricas para a caracterização de perfis de estudantes. A Tabela 4 apresenta as principais métricas do *PCodigo II*.

A Figura 1 apresenta a arquitetura do *PCodigo II* com suas funcionalidades, processos, entradas e saídas.

De acordo com a Figura 1, o processo de mapeamento de perfis em métricas de software começa com a Submissão de uma Solução em C junto com um arquivo makefile e um arquivoentrada. A Solução em C contém um ou mais arquivos de um projeto contendo programas em Linguagem C, arquivos de bibliotecas e outros arquivos necessários para rodar um programa. O arquivo makefile contém as instruções de execução da Solução em C e o arquivoentrada contém as entradas utilizadas para testar a Solução em C.

Cada Submissão é armazenada em uma base de Exercícios de Programação via interface web do MOODLE na Visão Aluno, conforme a Figura 1. Todas as submissões para cada tarefa são executadas pelo Núcleo Executor do PCodigo II e os arquivos executavel e arquivosaida com os resultados de execução de uma Solução em C são gerados em cada diretório de Submissão.

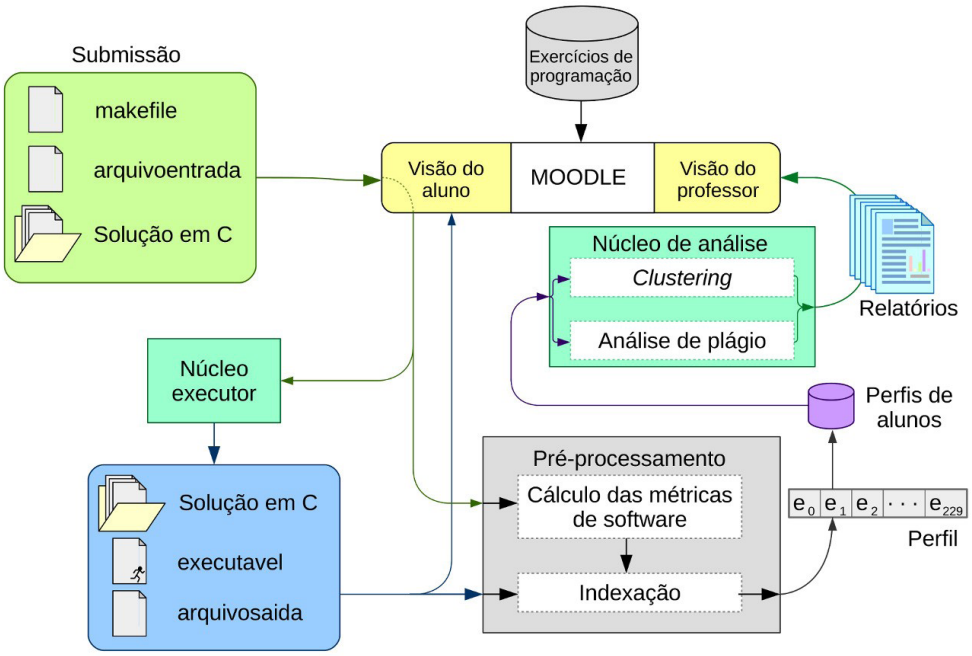

Figura 1. Arquitetura do PCodigo II (NEVES *et. al*, 2017)

O *Pré-processamento* da Figura 1 consiste em duas etapas: *Cálculo das Métricas* e *Indexação*. O *Cálculo das Métricas* consiste em análisar os códigos de programação

em Linguagem C e a partir deles calcular os valores das métricas de Curtis *et al*. (1979) e de Berry e Meekings (1985) e dos indicadores de execução de Oliveira (2015). A *Indexação* consiste em criar uma representação vetorial chamada *Perfil* em que cada dimensão contém o valor $e_i (i = 1 \ldots 348)$ de uma métrica de software calculada. A Figura 2 apresenta algumas métricas utilizadas na representação de perfis de estudantes de programação.

Em seguida, conforme a Figura 1, o Perfil gerado na Indexação é armazenado junto com outros perfis da mesma tarefa na Matriz M (OLIVEIRA, 2015). Essa matriz é normalizada a valores entre 0 e 1 e submetida a algoritmos de Clustering para agrupamento de perfis similares e ao algoritmo de Análise de Plágio para verificar similaridades entre perfis que caracterizem plágios.

Finalizando os processos, na Figura 1, os Relatórios de Clustering, de Visualização da Informação e de Análise de Plágio, que são descritos nas subseções a seguir, são enviados para um professor via interface web Visão do Professor do Moodle.

Através dos *Relatórios* do *PCodigo II*, os professores podem análisar a aprendizagem de seus alunos comparando perfis e reconhecendo, por meio das métricas, dificuldades de aprendizagem, boas práticas de programação e plágios.

3.1 Clustering e visualização da informação

Para agrupar os perfis por similaridade, foi utilizado o algoritmo de *clustering Bissecting K-means* do software *Cluto* (KARYPIS, 2002). As entradas foram o número de *clusters*, a matriz *M'* formada pelos vetores de perfis, os rótulos das linhas de *M'* com os identificadores das soluções de cada aluno e os rótulos das colunas com os identificadores das métricas de software.

Métrica	Origem	Significado
compila	PCodigo	programa escrito corretamente
executa	PCodigo	O código executa?
article_berry_reserved_words	ref:RBerry85	o número de diferentes palavras reservadas e
article_curtis_halstead_difficulty	ref:BCurtis79	Dificuldade de Halstead
article_curtis_halstead_effort	ref:BCurtis79	Esforço de Halstead
article_curtis_halstead_volume	ref:BCurtis79	Volume de Halstead
function_count	git:lzd	Quantidade de funções utilizadas
halstead_difficulty	git:ccd	Dificuldade de Halstead
halstead_effort	git:ccd	Esforço de Halstead
token_count	git:ccd@	Número de símbolos utilizados em todo o programa
lines_of_code_total	git:ccd@	Contagem de linhas do código fonte.
reserved_word_for	git:ccd@	Símbolo reservado pela gramática da linguagem
reserved_word_if	git:ccd@	Símbolo reservado pela gramática da linguagem
cyclomatic_complexity_avg_per_func	git:lzd	Média da complexidade ciclomática de todas as funções.

Abreviações

Prefixos

ref: referencia a bibliografia em BibTeX, abaixo
git: referencia a saída de um código vindo do GitHub

Corpo

ccd: https://github.com/dborowiec/commentedCodeDetector/tree/master
lzd: https://github.com/terryyin/lizard/tree/master

Sufixo

@: indica que o código recebeu edições para desempenhar tal funcionalidade

Figura 2. Exemplos de métricas de software (NEVES et. al, 2017)

As saídas dos algoritmos de *clustering* foram um arquivo contendo a identificação dos *clusters* a que cada solução da matriz *M'* pertence, um gráfico de *clustering com* os vetores de soluções distribuídos entre os *clusters* e um relatório com as informações de cada *cluster* e dos processos de *clustering*.

Para a visualização de informação, foram utilizados algoritmos escritos em Linguagem R para geração de mapas de calor (KOLDE, 2015). Os algoritmos de *clustering* do *Cluto 2.1.2* (KARYPIS, 2002) geraram uma visualização de perfis reunidos em *clusters*. A Figura 4 é um exemplo de gráfico gerado pelo *Cluto*.

Os *clusters* foram obtidos usando a medida de similaridade coeficiente de correlação. Dessa forma, os valores das métricas no gráfico gerado correspondem aos valores do vetor original subtraídos do vetor-média.

3.2 Análise de plágios

No módulo de *Análise de Plágio* da Figura 1, os vetores da Matriz *M'* são comparados dois a dois e égerada uma Matriz M_A, formada pelos índices de similaridade entre cada par de vetores (OLIVEIRA, 2015) calculados pela medida *cosseno* (BAEZA-YATES *et al.*, 1999). Os índices de similaridade variam de 0 (dissimilaridade) a 1 (similaridade total).

Para a análise do professor, o módulo *Análise de Plágio* retorna os pares de

vetores com índices de similaridade acima de 0.9, isto é, com semelhanças acima de 90% (OLIVEIRA, 2015), conforme exemplo da Figura 5.

A vantagem do *PCodigo II* em relação ao *PCodigo* original na *Análise de Plágio* é não precisar realizar processos de normalização de códigos para retirada de *tokens* que geram ambiguidades, de *strings* e de comentários, uma vez que, comparando soluções a partir de 348 variáveis de avaliação, as similaridades acima de 90% evidenciam fortemente práticas de plágios.

4 | EXPERIMENTOS E RESULTADOS

Para coletar códigos de programação e para testar as funções de representação de perfis em métricas de software, de *clustering*, de visualização da informação e de análise de plágios do *PCodigo* II, foi ofertado um curso a distância de programação C para 80 estudantes do ensino médio, de graduação e de pós-graduação.

Durante o curso, foi aplicado um exercício de programação para avaliação diagnóstica, o mesmo utilizado por Oliveira (2015). Esse exercício é adequado para a avaliação diagnóstica porque utiliza expressões lógicas, estruturas de controle condicional e estruturas de controle de repetição. Para o professor que aplicou a atividade, o critério de avaliação foi o uso das expressões lógicas. Dessa forma, seria uma evidência de dificuldades o uso excessivo de comparações e um número alto de linhas de código. Uma boa solução utilizaria no máximo três comparações e poucas linhas de código.

O experimento foi realizado com trinta alunos que submeteram uma solução em código C para o exercício proposto por meio do ambiente virtual *Moodle*. Essas soluções submetidas foram executadas e mapeadas em vetores de 84 dimensões correspondentes às métricas de software não nulas para todos os alunos. Em seguida, foi gerada uma visualização desses vetores, que é apresentada na Figura 3.

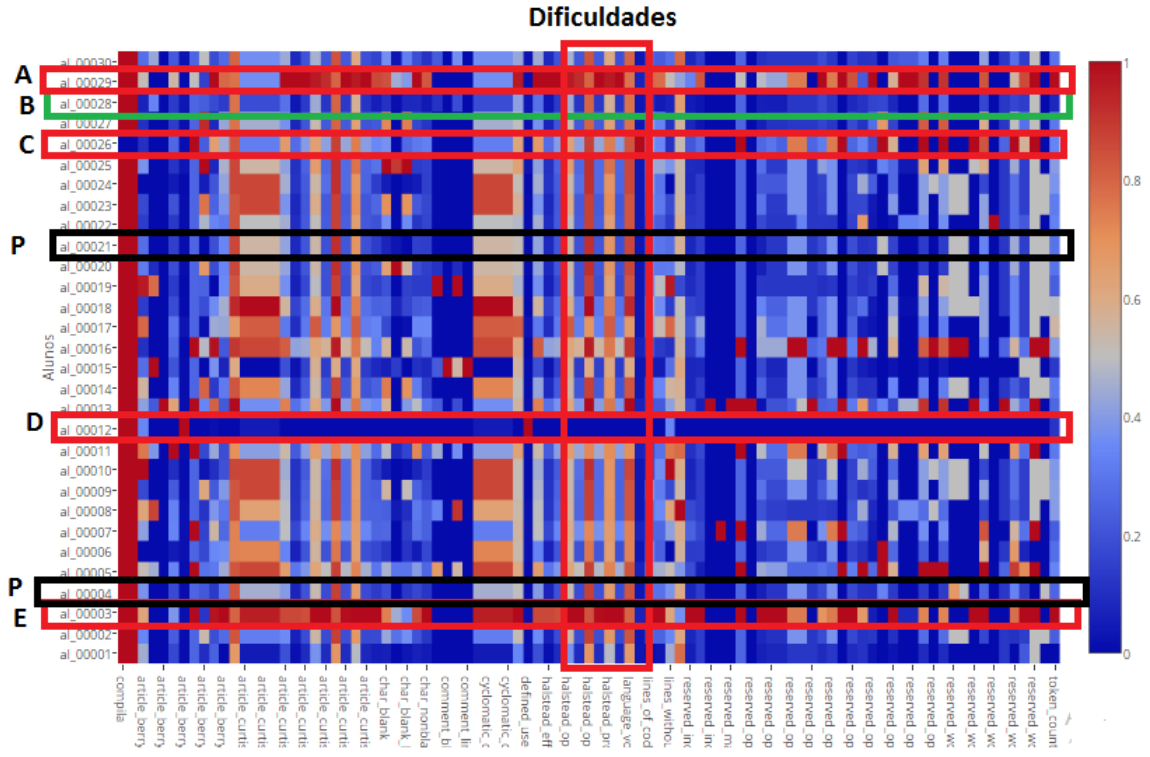

Figura 3. Visualização das métricas em mapa de calor (NEVES et. al, 2017)

Na Figura 3, as linhas são as soluções dos alunos e as colunas, os valores das métricas de software normalizados, isto é, divididos, em cada coluna, pelo maior valor de métrica. Quanto mais vermelha uma célula, maior o valor de uma métrica e, quanto mais azul, menor é o valor da métrica. A primeira coluna informa se uma solução compila (cor vermelha) ou não (cor azul). Nas demais métricas, as taxas mais altas aparecem em vermelho e as mais baixas, em azul.

No mapa de calor da Figura 3, analisamos sete soluções identificadas com as letras *A, B, C, D, E* e *P*. De acordo com as métricas, foram identificadas soluções que evidenciam mais dificuldades (*A, C, D* e *E*), plágios (*P*) e a solução menos incorreta (B). As dificuldades das soluções A, B, C, *D* e *E* no gráfico da Figura 3 foram reconhecidas principalmente pela presença da cor vermelha em várias métricas, principalmente nas colunas das métricas de *Halstead* (métricas de esforço e dificuldade), marcadas no gráfico com o rótulo *Dificuldades*. Ao verificar a Solução A, por exemplo, identificamos excesso de linhas de código (112 linhas), de instruções de entrada e de saída, de estruturas condicionais, de variáveis e de *tokens*, conforme cor vermelha na última coluna do gráfico, que informa a quantidade de *tokens*.

De acordo com a Figura 3, a Solução C, além do excesso de *tokens*, conforme a cor vermelha nas métricas mais à direita, não compila, conforme a cor azul da primeira coluna da Solução C. A Solução E, por sua vez, além de ter os mesmos excessos da Solução A, excede no número de comparações e de palavras reservadas, conforme observamos na cor vermelha nas métricas mais à direita das métricas de *Dificuldades*. Nesse caso, o aluno utilizou vários recursos da Linguagem C, mas não desenvolveu uma solução correta.

Ao contrário das soluções *A, E* e *C*, a Solução *D* do gráfico da Figura 3 evidencia dificuldades de aprendizagem pelas taxas muito baixas ou nulas nos valores de várias métricas. Nesse caso, o predomínio da cor azul, em especial nas métricas de *Dificuldades*, indica dificuldades pela ausência de instruções de programação. Ao verificar essa solução, observou-se que o aluno só escreveu instruções básicas de entrada e saída, não utilizando expressões lógicas nem estruturas de controle condicional e de repetição. O aluno não conseguiu, portanto, operar logicamente, o que é grave, uma vez que as expressões lógicas são conteúdos fundamentais na aprendizagem de programação.

A Figura 4 apresenta a visualização gerada pelos algoritmos de *clustering*. As linhas são rotuladas pelos identificadores dos alunos, *underline* e a nota do aluno com valores de 0 a 1 (0=0% e 1=100%). Quanto mais vermelho estiver o valor de uma métrica, maior é esse valor em relação ao valor médio da métrica e, quanto mais verde, menor é o valor dessa métrica em relação ao seu valor médio. A cor preta indica valor zero, isto é, a métrica assume o valor médio.

De acordo com o gráfico da Figura 4, as soluções plagiadas *P* aparecem no mesmo *cluster*. No entanto, a alta similaridade entre os dois códigos passou despercebida na avaliação do professor, uma vez que uma solução obteve nota 0.6 (60%) e a outra 1.0 (100%). Além disso, conforme o predomínio da cor vermelha nas métricas de complexidade e de dificuldade do gráfico, as duas soluções indicam dificuldades de aprendizagem por excesso de código. Através desse tipo de visualização, podemos ver a importância do uso das métricas para auxiliar o trabalho de avaliação de professores de programação.

Uma outra importante observação no gráfico da Figura 4 aparece nas soluções da parte mais baixa do gráfico. De forma isolada, aparecem as soluções *A, C, D* e *E*, isto é, as soluções que evidenciaram mais dificuldades de aprendizagem. As soluções *A* e *E*, segundo critérios do professor, obtiveram nota máxima e as soluções *C* e *D* obtiveram notas 0.0 e 0.4, respectivamente. As soluções *A* e *E*, embora indicassem muito esforço, o professor possivelmente considerou que a solução “deu certo” embora não “estivesse certa”. A *Solução C* possivelmente recebeu nota zero por não ter compilado e a *Solução D* perdeu 0.6 porque estava incompleta sem as expressões lógicas e sem as estruturas condicionais e de repetição.

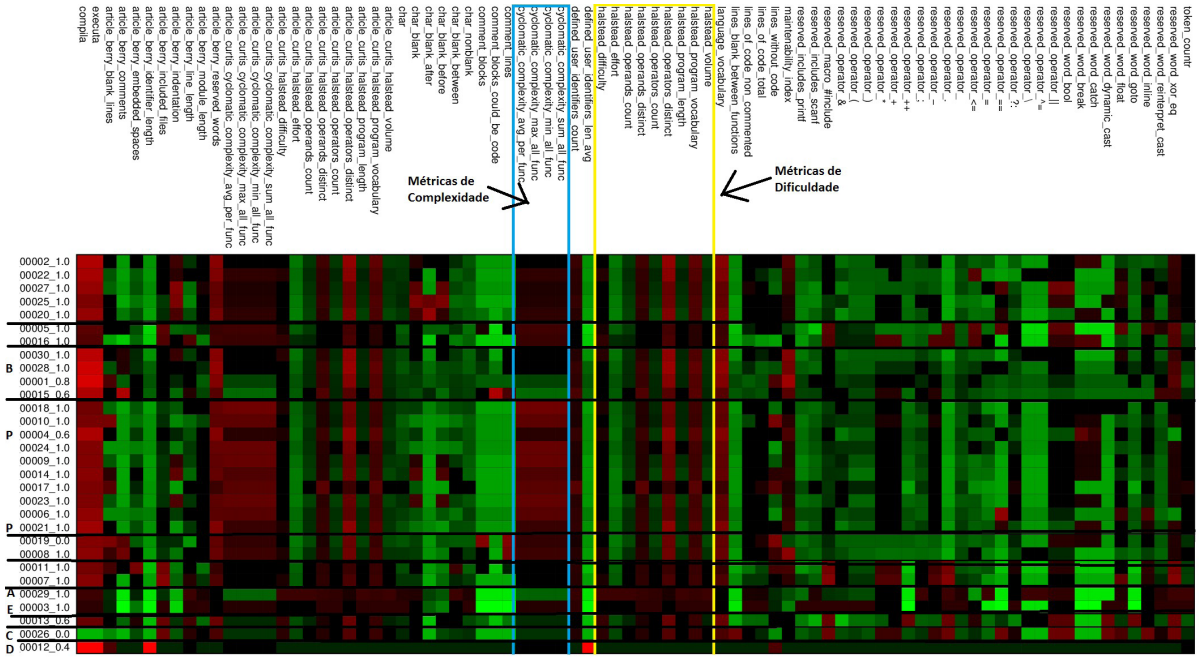

Figura 4. Gráfico de clustering

A *Solução B* e as demais soluções presentes no mesmo *cluster* na Figura 4 aparecem com predomínio de baixos valores de complexidade e a *Solução B* e a solução abaixo dela aparecem com predomínio da cor preta nas métricas de complexidade, confirmando, por estarem com os valores médios dessas métricas, que são as melhores soluções da turma, embora não estejam 100% corretas conforme análise de um professor de programação.

A Figura 5 apresenta o relatório de similaridade das soluções submetidas reconhecidas como suspeitas de plágio, isto é, com similaridade acima de 90%. Nas linhas 19 e 20 do relatório, aparecem as soluções *P*, chamadas, respectivamente, de *Solução 4* e *Solução 21*. Essas soluções têm similaridade de 97.78%, indicando fortes indícios de plágio. Analisando os vetores das linhas 19 e 20, observamos que os valores normalizados das métricas eram muito próximos. A Tabela 1 apresenta os códigos dessas soluções e confirma a alta similaridade entre elas.

De acordo com a Tabela 1, os códigos suspeitos de plágio assemelham-se pelas instruções e pelo erro comum de utilizar a estrutura de seleção da Linguagem C *switch*. Conforme Oliveira (2015), quando os alunos têm dificuldades em programar e resolvem ”colar”, eles costumam mudar apenas os nomes dos identificadores como, por exemplo, os nomes de variáveis. No entanto, eles não alteram os códigos de programação por não entendê-los. Observando o início das duas soluções da Tabela 1, conclui-se que o plagiador fez a alteração apenas nos nomes das variáveis.

Solução 4

```
#include <stdio.h>i
nt main()

{


intt, gn, vf, gp, vc, e, pt1,p
        t2,pt3,camp,vice;

for(t=1;t<=3;t++)

{
        printf(@texto,t);

        printf(@texto);

        scanf("%d",&gp);

    ...

        switch(t){ca
        se 1:

       pt1 = (5*gp–gn + 3*vf + 2*vc + e);pri
            ntf(@texto,t,pt1);

            break;

       ...

        case3:

      pt3 = (5*gp–gn + 3*vf + 2*vc + e);pri
            ntf(@texto,t,pt3);

          break;

          default:br
          eak;

                }

       }


        if(pt1>pt2)

                camp=1;

        else

                camp=2;



        if(pt3>camp)

                camp=3;

        ...

        if(camp==3)

                if(pt2>pt1)

                        vice=2;

                else

                        vice=1;

        printf(@texto,camp,vice)

        }
```

Solução 21

```
#include<stdio

.h>int main()

{
int time, GP, GN, VF, VC, E, P1, P2,

      P3,primeiro,seg
            undo;



for(time=1;time<=3;time++)

{
printf(@texto,time);

printf(@text
o);scanf("%d"
,&GP);

...

switch(time){

case1:P1=(5*GP–GN+3*VF+2*VC+E);

printf(@texto,time,P1);

break;

...

case3:P3=(5*GP–GN+3*VF+2*VC+E);

printf(@texto,time,P3);

break;

default:break;

}}
if(P1>=P2)

primeiro
=1;else

primeiro=2;



if(P3>prime
iro)primeir
o=3;



...

if(primeiro
==P3)if(P2>P1)

segundo
=2;else

segundo=1;


printf(@texto, primeiro, segun

do);return0;

}
```

Tabela 1. Programas suspeitos de plágios

Os resultados apresentados demonstram que a análise de aprendizagem por métricas de software é uma possibilidade muito favorável ao trabalho de avaliação de professores de programação, pois auxiliam-nos a compreenderem as dificuldades de aprendizagem de seus alunos principalmente quando estas se evidenciam em uma turma inteira. Além disso, as métricas podem ajudar professores a descobrirem o que caracteriza as boas soluções.

B	C	D	E	F	G	H
******		**PCodigo**	**-**	**RELATÓRIO**	**DE**	**ANÁLISE**
Variaveis	compila	executa	nota	article_berry_blank_lines	article_berry_comments	article_berry_constant_definitions
al_00014	1.0000	1.0000	1.0000	0.5517	0.0000	0.0000
al_00023	1.0000	1.0000	1.0000	0.1724	0.0000	0.0000
Similaridade:	0.985808	(98.5808%)				
Variaveis	compila	executa	nota	article_berry_blank_lines	article_berry_comments	article_berry_constant_definitions
al_00009	1.0000	1.0000	1.0000	0.1724	0.0000	0.0000
al_00023	1.0000	1.0000	1.0000	0.1724	0.0000	0.0000
Similaridade:	0.996154	(99.6154%)				
Variaveis	compila	executa	nota	article_berry_blank_lines	article_berry_comments	article_berry_constant_definitions
al_00009	1.0000	1.0000	1.0000	0.1724	0.0000	0.0000
al_00014	1.0000	1.0000	1.0000	0.5517	0.0000	0.0000
Similaridade:	0.98235	(98.235%)				

Figura 5. Análise de plágios

O *PCodigo II* apresenta-se, portanto, como uma ferramenta muito útil para avaliação diagnóstica da aprendizagem de programação e que muito pode contribuir para 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 I CONSIDERAÇÕES FINAIS

Este trabalho, que também foi publicado no Simpósio Brasileiro de Informática na Educação 2017 (NEVES, 2017), apresentou o sistema *PCodigo II* como um instrumento de análise da aprendizagem de programação a partir de códigos-fontes por mapeamento de perfis em 348 métricas de software. Os resultados da primeira experiência apontam o *PCodigo II* como uma ferramenta adequada para análise multidimensional da aprendizagem de programação.

Como trabalhos futuros a partir deste, sugerimos selecionar automaticamente as métricas mais relevantes para a avaliação de diferentes exercícios de programação e criar significados de grupos de métricas.

Em resumo, o *PCodigo II* apresenta-se como uma ferramenta de avaliação diagnóstica eficaz que auxilia professores a melhor compreenderem os processos de aprendizagem de seus alunos. Dessa forma, a contribuição dessa ferramenta para favorecer a aprendizagem de programação é ter um mecanismo para mostrar como os alunos programam, apontar dificuldades individuais e comuns em um turma e também reconhecer boas práticas de programação que caracterizam programadores hábeis e competentes.

REFERÊNCIAS

BAEZA-YATES, Ricardo et al. **Modern information retrieval**. New York: ACM press, 1999. Berry, R. and Meekings, B. A. (1985). A style analysis of C programs. Communications of the ACM, 28(1):80–88.

CURTIS, Bill et al. **Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics**. IEEE Transactions on software engineering, n. 2, p. 96-104, 1979.

KARYPIS, George. **CLUTO-a clustering toolkit**. Minnesota Univ Minneapolis Dept of Computer Science, 2002.

KOLDE, Raivo. **Pheatmap: pretty heatmaps**. R package version, v. 61, 2012.

MUNSON, Jonathan P. **Metrics for timely assessment of novice programmers**. Journal of Computing Sciences in Colleges, v. 32, n. 3, p. 136-148, 2017.

NEVES, Adler et al. **Mapeamento Automático de Perfis de Estudantes em Métricas de Software para Análise de Aprendizagem de Programação**. In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE**)**. 2017. p. 1337.

OLIVEIRA, Márcia Gonçalves; CIARELLI, Patrick Marques; OLIVEIRA, Elias. **Recommendation of programming activities by multi-label classification for a formative assessment of students**. Expert Systems with Applications, v. 40, n. 16, p. 6641-6651, 2013.

OLIVEIRA, M.; NOGUEIRA, M.; OLIVEIRA, Elias. **Sistema de apoio à prática assistida de programação por execução em massa e análise de programas**. In: XIV Workshop de Educação em Computação (WEI)-SBC 2015. 2015.

PETTIT, Raymond et al. **An empirical study of iterative improvement in programming assignments**. In: Proceedings of the 46th ACM Technical Symposium on Computer Science Education. ACM, 2015. p. 410-415.

PIETERSE, Vreda. **Automated assessment of programming assignments**. In: Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research. Open Universiteit, Heerlen, 2013. p. 45-56.

SOBRE O ORGANIZADOR

Everson Mario Novak Possui graduação em Tecnologia em Sistemas para Internet, Especialização em Desenvolvimento Web e MBA em Gestão de TI pela Faculdade Educacional de Ponta Grossa (Faculdade UNIÃO). Atualmente está cursando Mestrado em Informática na PUCPR - Pontifícia Universidade Católica do Paraná é professor do curso de Sistemas de Informação na Faculdades Integradas de Itararé – FAFIT. Ainda como Professor pela PUCPR na TECPUC na unidade de Ponta Grossa. É Analista de Sistemas, programador e tem experiência na área de Ciência da Computação, com ênfase em Arquitetura de Sistemas de Computação, Agentes de Software e Inteligência artificial.

Agência Brasileira do ISBN
ISBN 978-85-85107-14-7
9 788585 107147