Um Curso de Programação a Distância com Metodologias Ativas e Análise de Aprendizagem por Métricas de Software

Márcia Gonçalves de Oliveira – CEFOR/IFES – marcia.oliveira@ifes.edu.br
Adler Neves – IFES – adlerosn@gmail.com
Mônica Ferreira Silva Lopes – IFES – ferreirasilvalopes@gmail.com
Helen França Medeiros – IFES – helenfranca93@gmail.com
Mariella Berger Andrade – CEFOR/ IFES – mariella.andrade@ifes.edu.br
Leonardo Leal Reblin – UFES– leoreblin@gmail.com

Resumo. *Este trabalho apresenta um curso a distância de Programação C com metodologias ativas e com perfis de alunos representados por métricas de software para análise de dificuldades, habilidades e até competências em programação. O objetivo do curso é formar programadores a partir de metodologias centradas na aprendizagem do aluno e que favoreçam o desenvolvimento das habilidades de programação. Propomos, portanto, para esse curso uma prática assistida apoiada por tecnologia em que o professor possa acompanhar um conjunto de indicadores de aprendizagem e realizar intervenções. Para isso, criamos um sistema que gera automaticamente, para cada programa construído, uma representação de perfil do aluno composta por 348 métricas que quantificam esforço e qualidade de programação. Nos primeiros experimentos, as métricas possibilitaram o reconhecimento de classes de perfis de alunos, dificuldades de aprendizagem, soluções suspeitas de plágio e boas práticas de programação.*

Palavras-chave: *programação, metodologias ativas, métricas de software, perfil do aluno.*

A Distance Programming Course with Active Methodologies and Learning Analysis by Software Metrics

Abstract. *This work presents a distance course of C Programming with active methodologies and with student profiles represented by software metrics to analyze difficulties, abilities and even programming skills. The objective of this course is training programmers from methodologies focused on student learning and that favor the development of programming skills. We propose, therefore, for this course an assisted practice supported by technology in which the teacher can follow a set of learning indicators and carry out interventions. For this, we created a system that automatically generates, for each built program, a representation of the student profile by 348 metrics that quantify effort and quality of programming. In the first experiments, the metrics enabled the recognition of classes of student profiles, learning difficulties, suspected solutions of plagiarism and good practices of programming.*

Keywords*: programming, active methodologies, software metrics, student profile*

1. Introdução

Os processos de ensino, aprendizagem e avaliação de programação têm sido temáticas de ampla discussão no domínio da Educação e da Informática, principalmente por causa das dificuldades do domínio da programação. Essas dificuldades têm assim despertado o

interesse de muitos pesquisadores para o desenvolvimento de metodologias (Abuid, 2010) e tecnologias (Oliveira et al., 2015) de apoio à aprendizagem de programação.

Durante o processo de ensino e de aprendizagem de programação, muitos alunos apresentam dificuldades em raciocinar e assimilar as abstrações necessárias na criação e compreensão dos algoritmos (Dijkstra, 1982). Neste contexto, a dificuldade apresentada com o raciocínio lógico necessário à programação acarreta falta de motivação para muitos alunos, podendo ser fator determinante para reprovações e evasão (Rodrigues, 2002).

Com o objetivo de favorecer a aprendizagem de programação com metodologias de ensino e tecnologias de avaliação, este trabalho apresenta um Curso de Programação C a Distância. Nesse curso, são destaques: uma metodologia de estruturação de atividades associadas às habilidades de programação (compreensão, análise, revisão, reflexão, sequenciação lógica, abstração e construção); um fórum tira-dúvidas onde todos assumem a função de tutor para ajudar uns aos outros e recebem bônus pela qualidade da resposta dada para ajudar um colega; e as tecnologias de mapeamento de perfis para a avaliação de dificuldades de aprendizagem, de boas práticas de programação e de plágios.

A principal contribuição deste trabalho para a Educação Profissional de Informática é a apresentação de um curso de programação a distância metodologicamente centrado na aprendizagem do aluno e tecnologicamente suprido para assistir a prática da programação em uma perspectiva clínica e multidimensional.

Para apresentar o curso proposto, este trabalho está organizado conforme a ordem a seguir. Na Seção 2, apresentamos os trabalhos relacionados a nossa proposta de curso a distância com metodologias ativas e análise de aprendizagem. Na Seção 3, descrevemos a nossa proposta metodológica de curso de programação a distância e os instrumentos utilizados na análise de aprendizagem. Na Seção 4, apresentamos e discutimos alguns resultados de aplicação das metodologias ativas e da análise de aprendizagem. Na Seção 5, concluímos com as considerações finais e trabalhos futuros.

2. Trabalhos relacionados

Entre as metodologias já desenvolvidas com as finalidades de desenvolver habilidades e despertar a motivação na prática da programação, destacamos a Metodologia *Khan Academy* (Medeiros e Moura, 2016) e as metodologias de aprendizagem ativa baseada em problemas (Dos Santos e Costa, 2006; Looi e Seyal, 2014).

A metodologia *Khan Academy* consiste basicamente em uma disponibilização de atividades práticas e vídeos em que, em um ambiente de EaD, alunos interagem como tutores dos colegas de forma que aprendam uns com os outros.

A aprendizagem ativa baseada em problemas é uma abordagem derivada do construtivismo e caracteriza-se pelo processo de autoaprendizagem em que o aprendiz constrói ativamente o seu conhecimento (Looi e Seyal, 2014). O estudo de Looi e Seyal (2014) tem investigado se essa metodologia melhora a aprendizagem de programação e reforça o desenvolvimento das habilidades de análise e de resolução de problemas.

Além das metodologias de ensino, em um ambiente virtual de aprendizagem, é importante mapear perfis de estudantes para a personalização do ensino conforme habilidades, dificuldades e competências dos estudantes. Um exemplo de objeto de aprendizagem desenvolvido para mapeamento de competências é o *CompMap2*, que tem a função de ser o recurso digital com um conteúdo desenvolvido especificamente para o mapeamento de competências com foco no aluno da EaD (Behar e Silva, 2012).

Um exemplo de metodologia de ensino de programação integrada à utilização da tecnologia como instrumento de apoio ao ensino é o ambiente virtual denominado *O HALYEN*, que é um Sistema Tutor Inteligente com o objetivo de atuar como ferramenta tecnológica de auxílio ao ensino, através da escolha dinâmica da estratégia pedagógica, segundo o perfil e outras características dos alunos (Gonzalez e Tamariz, 2014).

A ideia de mapeamento de perfis também pode ser estendida à avaliação de aprendizagem. Um exemplo desse mapeamento é quando há a necessidade de se utilizar e visualizar um grande número de variáveis para diagnóstico da aprendizagem de programação. Uma tecnologia que destacamos, que é bem próxima à tecnologia utilizada neste trabalho é o *PCodigo* (Oliveira *et al*., 2015).

As principais funcionalidades do *PCodigo* para apoiar o trabalho docente são as seguintes: executar programas em massa, representar perfis de estudantes em vetores cujas dimensões são componentes de habilidades quantificadas na frequência de ocorrência de *tokens* e indicadores de execução da Linguagem C (Oliveira *et al*., 2015) e analisar programas para identificação de soluções divergentes e indícios de plágios.

Este trabalho, conforme Looi e Seyal (2014), busca experimentar metodologias que melhorem a aprendizagem de programação e reforcem o desenvolvimento das habilidades de análise e de resolução de problemas. Além disso, como na Metodologia *Khan Academy,* visamos que, em ações de tutoria mútua, os alunos também aprendam uns com os outros. Já, conforme a proposta de Oliveira *et al*. (2015), buscamos mapear perfis de estudantes quantificando informações extraídas da análise de códigos-fontes.

3. Um Curso a Distância de Programação C Essencial e Avançada

O *Curso de Programação C Essencial e Avançada* a distância, criado em 2016, foi ofertado duas vezes pelo Centro de Referência em Formação e Educação a Distância do Instituto Federal do Espírito Santo (Ifes). O objetivo desse curso é formar programadores a partir do desenvolvimento das habilidades de programação. Para alcançar esse objetivo, o curso foi planejado com metodologias de aprendizagem ativa orientada a problemas e experimentado no uso tecnologias de análise de aprendizagem para auxiliar o trabalho de avaliação de professores de programação.

3.1. Metodologias ativas orientadas ao desenvolvimento de habilidades

De acordo com Antunes (2001), o desenvolvimento das habilidades de um domínio do conhecimento pode ser realizado a partir dos próprios conteúdos ministrados em um curso. No caso da programação de computadores, isso pode ser realizado nos processos de programar, uma vez que a programação se aprende com a prática, e a prática melhora a taxa de processamento de informações de um aprendiz (Anderson, 2000).

Para Anderson (2000), monitorando cuidadosamente as componentes de uma habilidade, é possível levar estudantes rapidamente ao domínio de habilidades mais complexas. Dessa forma, um curso planejado com metodologias para ativar a aprendizagem na prática da programação e com tecnologias que auxiliem professores no trabalho de avaliação da aprendizagem de programação poderá vencer os desafios de ensinar e de aprender programação com possibilidades de êxitos coletivos de aprendizagem.

Seguindo essa ideia, a proposta de curso a distância de programação deste trabalho tem como principais objetivos contribuir para o desenvolvimento de habilidades de programação, assistir a prática da programação não só pelos programas desenvolvidos,

mas principalmente pelos seus processos visando a formação de bons programadores. Para isso, através do uso de metodologias de aprendizagem ativa baseadas em problemas, os módulos da nossa proposta de curso de programação a distância são organizados conforme os componentes (Silbermann, 1996) e atividades da Tabela 1:

Tabela 1 – Atividades orientadas ao desenvolvimento de habilidades

Componentes	Tipos de Atividades
Conteúdos para *ler, assistir e ouvir*	• Disponibilização de *slides* com os conteúdos sintetizados e ilustrados; • Apresentação de exercícios de programação resolvidos e comentados; • **Indicação de tutoriais e de vídeos do Youtube**.
Atividades para *discutir*, *perguntar* e *ensinar*	• Webconferência tira-dúvidas com tutores e professores, • Fórum do tipo u*m ao outro ajudou* com participação de alunos, professores e tutores com bônus para uma boa explicação para sanar a dúvida de um colega • Tarefa de produção de vídeo explicando uma solução de programação **• Fórum do tipo Leilão em que se propõem diferentes soluções para um problema com bônus de participação e bônus extra para quem oferecer a melhor solução.** **• Fórum do tipo q*ual é o problema* em que se descobre por que um programa não está funcionando corretamente.**
Atividades para *compreender* problemas	• Tarefas de sequenciação lógica, • Tarefas para identificar entrada, processamento e saída de um programa • Tarefas para reconhecer estados iniciais e finais de variáveis locais e globais após processamento em funções. **• Tarefa para analisar e comentar instruções de um programa**
Atividades para *criar* programas	Tarefa de construção de programas com envio do programa, do arquivo contendo entradas e de *makefile* contendo as instruções de execução do programa.
Atividades para *revisar e depurar* programas	Tarefa do jogo dos sete erros em que é apresentado um programa com erros de sintaxe e de lógica e o estudante deve identificar pelo menos sete erros. Para incentivar a análise minuciosa do código, cada erro descoberto além dos sete, recebe um bônus.

3.2 Instrumentos de análise de aprendizagem

Para gerar as representações de perfis de estudantes de programação, integramos as tecnologias de análise de código de Curtis *et al*. (1979) e Berris e Meekings (1985) às funcionalidades do *PCodigo de* Oliveira *et al. (2015).* As representações de perfis consistem em vetores com 348 dimensões, onde cada dimensão é o valor de uma métrica de software que quantifica esforço ou qualidade de programação.

Entre algumas das métricas utilizadas, destacamos as seguintes: esforço e dificuldade de Halstead (Curtis et al. 1979), variabilidade de palavras reservadas (Berris e Meekings, 1985), quantidade de funções[1], número de linhas de código[2] e média da complexidade ciclomática de todas as funções[3].

[1] https://github.com/terryyin/lizard/tree/master
[2] https://github.com/dborowiec/commentedCodeDetector/tree/master (Com alterações)
[3] https://github.com/terryyin/lizard/tree/master

Para realizar os processos de *clustering*, utilizamos o algoritmo *Bissecting K-Means* e a visualização gráfica *3D Mountain Visualization* (Visualização Montanha), ambos do software *gCluto* (Karypis *et al*., 2003). Os principais parâmetros de *clustering* configurados para formar esses agrupamentos foram a medida de similaridade *cosseno* e o número de *clusters* igual a dez.

4. Experimentos e resultados

A primeira turma do *Curso de Programação C Essencial e Avançada* em que foram experimentadas as metodologias ativas e as tecnologias de análise de aprendizagem deste trabalho era formada por cerca de 25 alunos. Essa turma era bastante heterogênea e havia predominantemente alunos do ensino médio e da pós-graduação.

4.1. Primeiras avaliações da aplicação de metodologias ativas

As principais vantagens de aplicação das metodologias ativas no curso de Programação C foram as seguintes: a preparação dos alunos para construir programas a partir de atividades envolvendo as principais habilidades de programação como a compreensão, a análise, a observação de detalhes, a sequenciação e a abstração; a tutoria mútua nos fóruns tira-dúvidas envolvendo alunos de ensino médio, de graduação e de pós-graduação; e a organização do material e das atividades do curso para ambientes *online*.

De acordo com as avaliações do curso, para 91.3% dos alunos que concluíram o curso, foi indispensável a aplicação de atividades de diferentes níveis de dificuldades. Para 65.2% dos alunos, também foi indispensável a aplicação de exercícios articulando a teoria com a prática e, para 78.3% dos alunos o planejamento dessa estrutura de curso de programação para ambientes *online* foi adequada.

Como pontos positivos do Curso de Programação C, os alunos destacaram a fácil introdução mesmo para os alunos inexperientes em programação, as atividades, as discussões colaborativas, os exercícios individuais práticos e teóricos, os jogos dos sete erros e os exercícios de compreensão. Em relação aos pontos negativos, houve algumas críticas em relação aos prazos das disciplinas, do cronograma do curso, em relação ao material instrucional, mas praticamente não houve críticas em relação às metodologias utilizadas nas atividades do curso.

4.2. Experimentos e resultados de avaliação diagnóstica por métricas de software

Os experimentos iniciais tiveram como objetivo realizar uma avaliação diagnóstica para compreender como os alunos programam a partir da análise de seus códigos-fontes e inferir dificuldades, boas práticas e até plágios de programação.

Para realizar essa análise, escolhemos um exercício de desenvolvimento de programa aplicado no meio do curso. Esse exercício consistiu em identificar por número de pontos o campeão e o vice-campeão de um campeonato de futebol. Esse exercício foi o mesmo utilizado nos testes de Oliveira *et al*. (2015) em outras turmas de programação. O critério de escolha desse exercício foi o uso de estruturas de repetição, de estruturas condicionais e, principalmente, de expressões lógicas. Uma boa solução desse exercício utiliza no máximo de três a cinco comparações, enquanto uma solução ruim utiliza muitas comparações, mesmo que o programa funcione corretamente.

Os alunos submeteram a solução para esse exercício através do *Moodle* junto com um arquivo de *makefile* e um arquivo contendo as entradas do programa.

Em seguida, organizamos as submissões em diretórios no formato de entrada do *PCodigo* de Oliveira et al. (2015). Para gerar vetores de representação das submissões, utilizamos as variáveis (ou componentes de habilidades) do *PCodigo* e as métricas de Curtis *et al*. (1979) e Berris e Meekings (1985). Após formar uma matriz *M* de 25 linhas representando alunos e 348 dimensões representando os valores das métricas, dela retiramos as colunas cujas métricas estavam zeradas para todos os alunos. Ao final, foi gerada uma nova matriz *M'* com 104 dimensões e 25 linhas.

Para termos uma melhor visualização das dificuldades, das boas práticas e dos indícios de plágios de programação, utilizamos um *script* em Linguagem R para gerar mapas de calor da matriz *M'*. Um exemplo de mapa de calor gerado é apresentado na Figura 1.

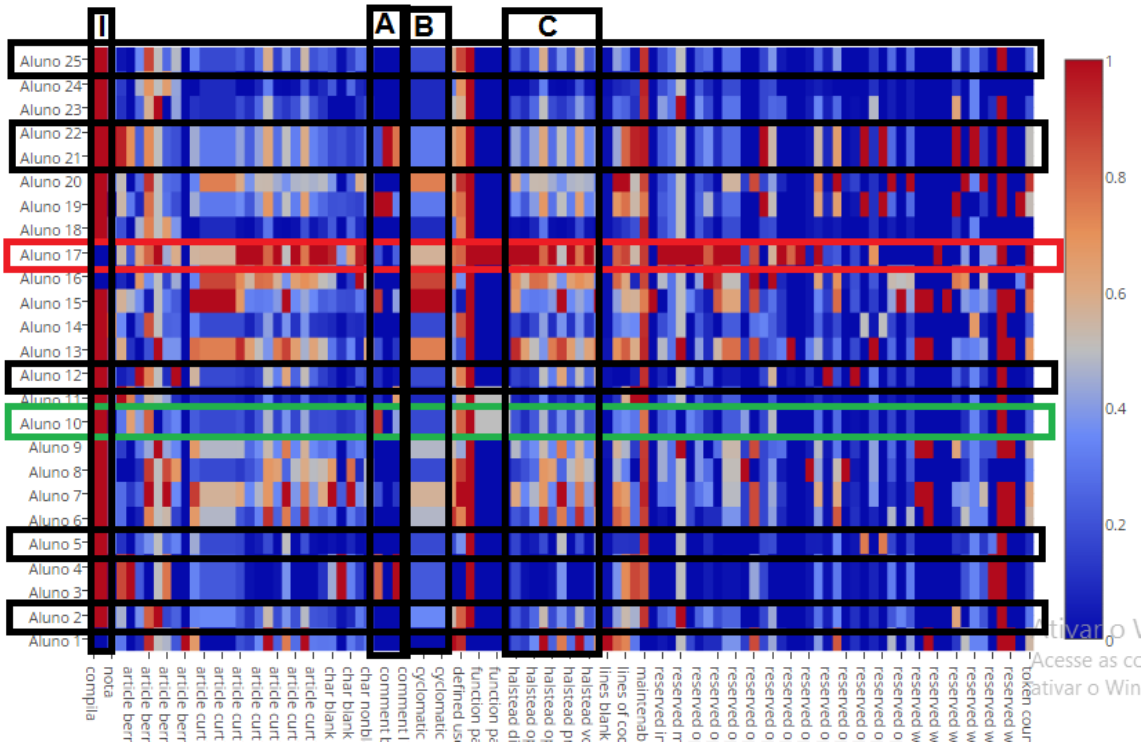

Figura 1 – Mapa de calor Alunos x Métricas

No gráfico da Figura 1, as linhas representam os alunos e as colunas, as métricas de avaliação. Para essa visualização, os valores das métricas foram normalizados a valores entre 0 e 1. Quanto maior o valor da métrica, mais vermelha é a sua célula. Da mesma forma, quanto mais azul, menor o valor da métrica. Em geral, os resultados são bons quando os valores das métricas são baixos, exceto para a *coluna I* que indica se um programa compila (1=Compila=Vermelho; 0= Não compila = Azul).

Na Figura 1, a primeira observação a ser feita é em relação ao *Aluno 17* (marcado de vermelho no gráfico) e ao Aluno 10 (marcado de verde). O programa do *Aluno 17* não compila (Coluna *I* do gráfico com cor azul) e muitas métricas apresentam altas taxas, principalmente as métricas do *Grupo C* no gráfico, que são as métricas de *Halstead*, que caracterizam esforço e dificuldade na escrita do código (Curtis et al., 1979). Isso nos leva a concluir que, embora o programa do aluno tenha sido considerado correto pela avaliação humana, esse aluno não conseguiu fazer o código compilar e fez um grande esforço

escrevendo muito "para dar certo" e não para "fazer certo". Dessa forma, fica evidente que esse aluno apresenta dificuldades de aprendizagem, embora seja esforçado.

Por outro lado, o *Aluno 10* e o *Aluno 5* do gráfico da Figura 1 desenvolveram boas soluções de programação porque elas compilaram corretamente (conforme cor vermelha na coluna *I*) e apresentam taxas mais baixas nas diferentes métricas, em especial nas métricas de *Halstead* do *Grupo C*. Há, porém, uma diferença entre as soluções na coluna *A*, que indica número de comentários. A solução do *Aluno 5* é considerada a melhor por ter menos textos, porém, a solução do *Aluno 10* indica boas práticas de programação ao inserir muitos comentários no código, conforme cor vermelha nas colunas do *Grupo A*.

Analisando os demais alunos na Figura 1, observamos que os alunos 25 e 12 também desenvolveram soluções corretas por compilarem e apresentarem, em geral, baixas taxas nas demais métricas. Já o *Aluno 2*, embora o código compile e apresente baixas taxas nas demais métricas, é um código considerado incorreto pelo professor por uso incorreto de operadores e expressões lógicas.

Uma análise importante a ser realizada também é a complexidade de código representada pelas colunas do *Grupo B* na Figura 1. As soluções dos alunos 15 e 16 aparecem como as mais complexas (cor vermelha no *Grupo B*). Ao verificar os códigos desses alunos, observamos uma alta quantidade de comparações e, no caso da solução do *Aluno 16*, as métricas de *Halstead* (*Grupo C*) têm valores um pouco altos. Essas métricas informam, portanto, que esses alunos têm dificuldades em utilizar operadores e expressões lógicas, o que é grave, já que as expressões lógicas é que controlam as estruturas de controle condicional e de repetição de um programa.

Além de fornecer indicadores de dificuldades de boas práticas de programação as métricas podem informar ações intencionais de prática do plágio ou ações não intencionais como a submissão de soluções duplicadas. No caso dos alunos 21 e 22 da Figura 1, é evidente que houve um caso de plágio ou de envio de soluções duplicadas, pois as métricas apresentam-se idênticas. Considerando as caraterísticas das soluções, informações de comentários com o nome do aluno e informações de data e hora dos arquivos, o professor considerou as soluções como duplicadas e considerou a hipótese dos alunos 21 e 22 serem uma mesma pessoa que realizou duas submissões.

Após os processos de *clustering*, foram gerados alguns arquivos de saída, dentre eles um relatório que resumimos na Figura 2.

ANÁLISE DE CLUSTERS

Cluster	Sim	Rotulo	Somalinha	Observação	Cluster	Sim	Rotulo	Somalinha	Observação
0	***	Aluno_1	30.0455	Confusão de instruções	7	91.9%	Aluno_16	42.044	Excesso de Instruções
1	100%	**Aluno_4**	**22.0351**	**Solucao Duplicada**	7		Aluno_20	40.3458	Excesso de Instruções
1		**Aluno_3**	**22.0351**	**Solucao Duplicada**	8	89.7%	Aluno_6	34.7018	Uso do Switch
2	***	**Aluno_17**	**61.7443**	**Correta com muito Esforço**	8		Aluno_15	45.6963	Uso do Switch
3	99.99%	Aluno_18	**13.7678**	**Fortes indícios de plágio**	8		Aluno_7	43.8759	Uso do Switch
3		Aluno_24	**13.7678**	**Fortes indícios de plágio**	8		Aluno_13	46.7111	Uso do Switch
4	***	Aluno_8	30.3597	Solução atípica	8		Aluno_9	36.5193	Uso do Switch
5	96.8%	Aluno_21	**37.3936**	**Solução duplicada**	9	88.3%	Aluno_25	23.4273	Correta
5		Aluno_22	**37.3936**	**Solução duplicada**	9		Aluno_14	21.5007	Incorreta
5		Aluno_19	36.8049	Confusão de instruções	9		Aluno_2	26.8392	Incorreta
6	93%	Aluno_10	26.0579	**Solução boa e comentada**	**9**		**Aluno_5**	**16.6205**	**Melhor Solução**
6		Aluno_11	21.6784	Incompleta	9		Aluno_12	24.6777	Correta
					9		Aluno_23	20.684	Incorreta

Figura 2 – Análise de c*lusters*

A Figura 2 apresenta os dez *clusters* formados pelo algoritmo *Bissecting K-means*

contendo as soluções de programação mapeadas em 348 métricas de software. Na Figura 2, o *Cluster* é o identificador de *cluster* numerado de 0 a 9, *Sim* é o índice de similaridade interna do *cluster*, *Rotulo* é o identificador da solução de um aluno e *Somalinha* é a soma das métricas da solução de um aluno.

De acordo com a Figura 2 e com a Figura 1, identificamos como a melhor solução a do *Aluno 5* no *Cluster 9* porque possui menor valor de *Somalinha*, confirmando os baixos valores das métricas do *Aluno 5* na Figura 1. Já a solução do *Aluno 17*, isolada como uma solução atípica no *Cluster 2*, apresenta-se com o maior valor de *Somalinha*, o que sugere muito esforço e dificuldade, conforme analisamos na Figura 1.

O *Cluster 3* da Figura 2 reúne duas soluções com fortes indícios de plágio por terem similaridade de 100%, o que não é tão perceptível no mapa de calor da Figura 1. Nesse caso, não ocorreu submissão duplicada, uma vez que houve ligeiras modificações no código no qual se evidenciou o plágio e as informações de arquivo como data e hora diferenciam-se nas duas soluções.

No gráfico do tipo *Mountain Visualization* da Figura 3, a forma de cada relevo é uma curva gaussiana, em que a altura é proporcional ao índice de similaridade interna do *cluster*, o volume é proporcional à quantidade de elementos do *cluster* e a cor de cada pico vai de acordo com o desvio interno do *cluster:* vermelho indica alto desvio interno (alta heterogeneidade), enquanto azul indica baixo desvio (baixa heterogeneidade).

Podemos observar que o *Cluster 9* da Figura 3 é o mais diferenciado. Olhando para as informações desse *cluster* na Figura 1, observa-se que ele reúne soluções corretas e incorretas, o que o caracteriza como um *cluster* representativo da diversidade de soluções. O *Cluster* 8, por sua vez, apresenta-se um pouco mais homogêneo, mas com alguma diferenciação. Nesse cluster, as soluções se aproximam pelo uso do comando de seleção *Switch* da Linguagem C e diferenciam-se na forma de organização das instruções.

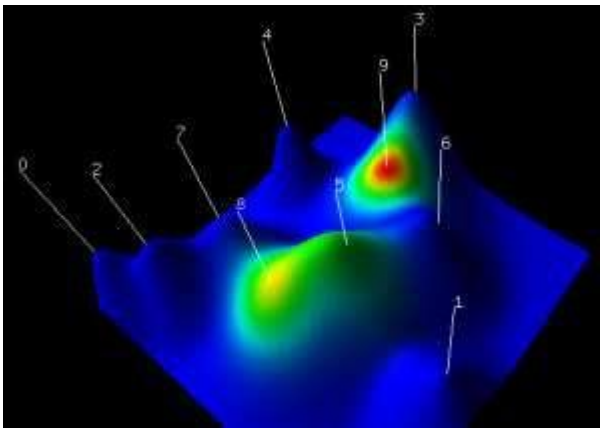

Figura 3 – Análise de *clusters* em gráfico de montanhas

Os demais *clusters* da Figura 3 apresentam-se mais uniformes. Os *clusters* mais homogêneos, conforme análises, podem ser interpretados de acordo com as seguintes hipóteses: quando há muitos exemplos, é possível que reúnam as soluções de gabarito, pois as boas soluções tendem a se aproximar (Oliveira *et al*., 2016); quando há poucos exemplos, podem estar reunindo soluções plagiadas, duplicadas, com poucas informações (como no *Cluster 3*), soluções de gabarito ou soluções únicas (como no *Cluster 4*).

Concluindo, conforme as análises realizadas utilizando as métricas de software, os algoritmos de *clustering* e as ferramentas de visualização de informação, mostramos que temos suma proposta de curso de programação a distância com possibilidades de um acompanhamento minucioso da aprendizagem dos alunos na prática da programação. Dessa forma, esses instrumentos poderão ser utilizados por professores para compreender as dificuldades de seus alunos e ajudá-los personalizando o ensino e reorientando as ações pedagógicas para que de fato se alcancem êxitos de aprendizagem na programação.

5. Considerações Finais

Este trabalho apresentou a proposta de um curso de programação a distância orientado ao desenvolvimento de habilidades de programação baseado em aprendizagem ativa orientada a problemas. Para diagnóstico do processo de aprendizagem, utilizamos instrumentos de análise de aprendizagem baseados em métricas de software e tecnologias de análise de aprendizagem com as finalidades de reconhecer dificuldades de aprendizagem, classes de perfis, possíveis plágios e boas práticas de programação.

Os resultados obtidos nas avaliações do curso apontam para êxitos na metodologia utilizada para ensino e aprendizagem de programação a distância mesmo em uma turma bastante heterogênea com diferentes níveis de aprendizagem. Os diferenciais dessa metodologia foram a realização da prática da programação a partir das habilidades envolvidas no processo de programação, a tutoria afetiva dos tutores e a tutoria colaborativa dos alunos através dos fóruns tira-dúvidas.

Os relatórios de avaliação diagnóstica, que foram gerados pelas tecnologias de análise de aprendizagem com as funcionalidades de mapeamento de perfis em métricas de software, *clustering* e visualização de informação, revelaram dificuldades de aprendizagem, classes de perfis, suspeitas de plágios e características de boas práticas de programação. Tais informações podem ser utilizadas por professores para personalização do ensino.

Como trabalhos futuros a partir deste, propomos evoluir as metodologias aplicando atividades com abordagem de aprendizagem baseada em projetos e agregar aos instrumentos de análise de aprendizagem a funcionalidade de previsão de desempenhos a partir de um histórico de perfis de aprendizagem gerados pelas métricas de software.

Com as metodologias propostas e as tecnologias de análise de aprendizagem utilizadas, apresentamos uma solução favorável à aprendizagem de programação a distância visando auxiliar o ensino de programação e contribuir para a formação de bons programadores.

6. Agradecimentos

Agradecemos à Fundação de Apoio à Pesquisa e Inovação do Espírito Santo (FAPES) pelo apoio dado ao projeto *Tecnologias de Avaliação Semi-automática da Aprendizagem de Programação* (*do EDITAL 006/2014 – Universal Projeto Individual de Pesquisa*) do qual resultou esta publicação.

7. Referências

ABUID, B. A. ADRI–Self assessment model for teaching and learning. In: **2nd international conference on global trends and challenges in higher education and quality assurance**. 2010. p. 12-13.

ANDERSON, J. R. **Cognitive psychology and its implications**. New York and Basingstoke: Worth Publishers, p. 181-183, 2000.

ANTUNES, C. **Trabalhando habilidades: construindo ideias**. São Paulo, SP: Scipione, 2001.

BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. **Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca**. Bookman Editora, 2013.

BEHAR, P. A; DA SILVA, K.K. Mapeamento de competências: um foco no aluno da educação a distância. **RENOTE**, v. 10, n. 3, 2012.

DIJKSTRA, Edsger W. On the Teaching of Programming, ie on the Teaching of Thinking. In: **Language hierarchies and interfaces**. Springer, Berlin, Heidelberg, 1976. p. 1-10.

BERRY, R. E.; MEEKINGS, B. AE. A style analysis of C programs. **Communications of the ACM**, v. 28, n. 1, p. 80-88, 1985.

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.

DOS SANTOS, Rodrigo Pereira; COSTA, Heitor Augustus Xavier. Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática. **INFOCOMP Journal of Computer Science**, v. 5, n. 1, p. 41-50, 2006.

GONZALEZ, S.; TAMARIZ, A. Integração de uma metodologia de ensino presencial de programação com um sistema tutor inteligente. **Revista Brasileira de Informática na Educação**, v. 22, n. 02, p. 16, 2014.

KARYPIS, George et al. CLUTO: a software package for clustering high-Dimensional data sets. **University of Minnesota, Dept. of Computer Science**, 2003.

LAHTINEN, ESSI; ALA-MUTKA, KIRSTI; JÄRVINEN, HANNU-MATTI. A study of the difficulties of novice programmers. In: **Acm Sigcse Bulletin**. ACM, 2005. p. 14-18.

LOOI, H.C; SEYAL, A. H. Problem-based learning: An analysis of its application to the teaching of programming. **International Proceedings of Economics Development and Research**, v. 70, p. 68, 2014.

MALIK, S.I.; COLDWELL-NEILSON, J.. A model for teaching an introductory programming course using ADRI. **Education and Information Technologies**, p. 1-32, 2016.

MEDEIROS FILHO, Dante Alves; MOURA, Ernani Guilherme Groff. A Metodologia de Ensino da Khan Academy para a Área Tecnológica. **Universidade Estadual de Maringá-Departamento de Informática. Disponível em:< http://www. espweb. uem. br/site/files/tcc/2011/Ernan Guilherme Groff Moura-A metodologia de ensino da Khan Academy para a area tecnologica. pdf>. Acesso em**, v. 19, 2016.

OLIVEIRA, M.; NOGUEIRA, M. A.; OLIVEIRA, E. (2015). Sistema de Apoio à Prática Assistida de Programação por Execução em Massa e Análise de Programas. In **XXIII Workshop sobre Educação em Computação (WEI) - CSBC 2015, Recife, PE**. SBC.

OLIVEIRA, Marcia et al. Reconhecimento Automático de Representações de Rúbricas em Agrupamentos de Soluções de Exercícios de Programação. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. 2016. p. 1106.

RODRIGUES, M. C. Como Ensinar Programação? **Informática–Boletim Informativo Ano I**, n. 01, 2002.

SILBERMAN, Mel. **Active Learning: 101 Strategies To Teach Any Subject**. Prentice-Hall, PO Box 11071, Des Moines, IA 50336-1071, 1996.

SOUZA, Draylson M.; DA SILVA BATISTA, Marisa Helena; BARBOSA, Ellen Francine. Problemas e Dificuldades no Ensino e na Aprendizagem de Programação: Um Mapeamento Sistemático. **Revista Brasileira de Informática na Educação**, v. 24, n. 1, 2016.