Mensurando o desenvolvimento do Pensamento Computacional por meio de Mapas Auto-Organizáveis: um estudo preliminar em uma Oficina de Jogos Digitais

Thiago Barcelos[1], Alexandra Souza[1,2], Leandro Silva[2], Roberto Muñoz[3,4], Rodolfo Villarroel[4]

[1] Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – IFSP
Av. Salgado Filho, 3501, Guarulhos, SP, Brasil – CEP 07115-000

[2] Universidade Presbiteriana Mackenzie
Rua da Consolação, 930, São Paulo, SP, Brasil – CEP 01302-907

[3] Universidad de Valparaíso – Escuela de Ingeniería Civil Informática
General Cruz, 222, Valparaíso, Chile – CP 2362905

[4] Pontifícia Universidad Católica de Valparaíso – PUCV
Brasil, 2241, Valparaíso, Chile – CP 2362807

{tsbarcelos, alexandra.souza}@ifsp.edu.br;
leandroaugusto.silva@mackenzie.br; roberto.munoz@uv.cl;
rodolfo.villarroel@pucv.cl

Abstract. *The automated analysis of programming code developed during Computational Thinking development activities has demonstrated a potential to identify employed strategies for problem solving. However, the evolution of code produced by students and its relationships with Computational Thinking skill acquisition has not been explored in full detail. In this study we used self-organizing maps to analyze games produced by students in a Game Building Workshop offered in 2013 and 2017. As a result of the training process, clusters in the produced map were consistent with employed strategies that were previously observed in classroom activities.*

Resumo. *A análise automatizada de código desenvolvido em atividades de desenvolvimento do Pensamento Computacional vem mostrando potencial em identificar estratégias utilizadas para a resolução dos problemas propostos. No entanto, ainda não tem sido explorada em detalhes a evolução do código e sua relação com a aquisição de competências do Pensamento Computacional. Neste estudo foram empregados mapas auto-organizáveis na análise de jogos produzidos por alunos em uma Oficina de Produção de Jogos Digitais nos anos de 2013 e 2017. O resultado do processo de treinamento produziu agrupamentos no mapa que apresentaram coerência com estratégias utilizadas pelos alunos e previamente observadas em sala de aula.*

1. Introdução

Da mesma forma que em outros contextos do processo de ensino-aprendizagem, avaliar adequadamente o desenvolvimento de competências e habilidades relacionadas ao

DOI: 10.5753/cbie.wcbie.2017.932

Pensamento Computacional se constitui como um desafio. A avaliação da aprendizagem em contextos experimentais tem sido frequentemente realizada com a utilização de técnicas de pesquisa relacionadas ao paradigma qualitativo, tais como a observação etnográfica e as entrevistas, bem como testes e questionários específicos (ARAÚJO; ANDRADE; GUERRERO, 2016). Por outro lado, outras iniciativas para assegurar o desenvolvimento de um conjunto mínimo de competências e habilidades incluem a definição de currículos de referência (CSTA, 2011) e rubricas educacionais para contextos mais específicos (MORENO-LEÓN; ROBLES, 2015).

Rubricas educacionais podem estar associadas a ambientes de desenvolvimento utilizados para fomentar o Pensamento Computacional. O Scratch concentra algumas dessas iniciativas devido a sua ampla utilização, chegando inclusive a ser citado recentemente dentre as 20 linguagens de programação mais populares no índice TIOBE (TIOBE, 2017). Sistemas para análise do código produzido com o Scratch visam identificar desde estatísticas básicas de utilização das estruturas de programação (WOLZ; HALLBERG; TAYLOR, 2011) até análises estruturais mais elaboradas (BOE *et al.*, 2013; MORENO-LEÓN; ROBLES, 2015).

No entanto, ainda há poucos trabalhos de pesquisa que explorem a utilização de métricas obtidas a partir de código Scratch para obter inferências de nível mais alto sobre o desenvolvimento de competências ao longo do tempo de forma semi-automatizada. A utilização de técnicas de Analítica de Aprendizagem (SILVA; PERES; BOSCARIOLI, 2016) poderia, por exemplo, permitir que um professor visualizasse o avanço de seus alunos de forma individualizada e ao mesmo tempo comparada com uma base histórica de resultados, permitindo um suporte adequado a um aluno que eventualmente apresentasse dificuldades em uma determinada etapa do desenvolvimento do Pensamento Computacional.

Dessa forma, o objetivo deste artigo é apresentar uma validação preliminar da viabilidade do uso de técnicas de aprendizado de máquina, em específico os Mapas Auto-Organizáveis, para identificar o desenvolvimento ao longo do tempo de competências e habilidades do Pensamento Computacional em alunos participantes de uma Oficina de Desenvolvimento de Jogos Digitais. A sequência deste artigo é organizada da seguinte forma: na seção 2 é apresentada uma revisão da literatura sobre a utilização de métricas obtidas a partir de código Scratch para avaliar a aprendizagem e o funcionamento dos mapas auto-organizáveis; na seção 3 são descritos os métodos e técnicas utilizados na pesquisa, bem como a estrutura e instâncias de oferecimento da Oficina de Jogos Digitais cujos dados foram utilizados; os resultados e discussão são apresentados na seção 4 e na seção 5 são apresentadas as conclusões preliminares da pesquisa e os possíveis trabalhos futuros.

2. Referencial teórico

2.1. Análise automatizada de código Scratch

Desde que Brennan e Resnick (2012) propuseram que a aquisição de competências do Pensamento Computacional poderia ser evidenciada pela complexidade dos artefatos computacionais produzidos por alunos, tais como o código de programas, algumas estratégias para a análise de códigos produzidos no ambiente Scratch vêm sendo

desenvolvidas e avaliadas. Wolz *et al*. (2011) propuseram o Scrape, um sistema para análise e apresentação visual do tipo e frequência de utilização das estruturas de programação em um programa Scratch. Uma abordagem mais sofisticada foi descrita por Boe *et al*. (2013) com o Hairball, um sistema baseado em *plug-ins* com o objetivo de extrair diversas características por meio da análise do código Scratch, tais como o emprego de práticas não recomendadas (*bad smells*).

Moreno-León e Robles (2015) propuseram uma rubrica que associa o nível de complexidade e frequência das estruturas de programação utilizadas a sete categorias conceituais relacionadas ao Pensamento Computacional: abstração, paralelismo, lógica, sincronização, controle de fluxo, interação com o usuário e representação de dados. A cada categoria é atribuída uma pontuação 1, 2 ou 3 em função da complexidade das estruturas e estratégias de programação utilizadas. A rubrica foi implementada em um sistema *web* denominado Dr. Scratch que se baseia em uma extensão do funcionamento do Hairball. A partir da análise automatizada de código tem sido possível obter resultados relacionados às boas e más práticas de programação utilizando Scratch (AIVALOGLOU; HERMANS, 2016; TECHAPALOKUL, 2017). No entanto, ainda não tem sido explorada em detalhes a evolução do código produzido por alunos e sua relação com a aquisição de competências do Pensamento Computacional.

2.2 Mapas Auto-Organizáveis

Mapas Auto-Organizáveis ou simplesmente SOM (do inglês, *Self-Organizing Maps*) se constitui como um tipo de rede neural artificial interconectada e não supervisionada que permite um mapeamento auto-organizável de um espaço de dados multidimensional em um plano bidimensional, resolvendo problemas envolvendo tarefas relacionadas ao agrupamento de dados, visualização e abstração (KOHONEN, 1988). O SOM também pode ser utilizado para um estudo mais amplo da correlação entre as múltiplas variáveis existentes em uma base de dados, sem restrição da quantidade a analisar, graças a sua propriedade de mapear dados com elevado número de dimensões em dimensões reduzidas, convertendo relações estatísticas não lineares complexas em relações geométricas, mas preservando a relação topológica original visto que a localização física dos dados no mapa mostra a similaridade entre os mesmos no espaço multidimensional original (KOHONEN, 2013).

O treinamento da rede SOM consiste em submeter iterativamente o conjunto de dados de entrada (um objeto do conjunto de treinamento é escolhido aleatoriamente) a um modelo de aprendizagem competitiva, na qual os neurônios disputam entre si a representatividade dos objetos do conjunto de treinamento, num processo que envolve a adaptação dos pesos da rede a partir dos estímulos fornecidos pelos dados de entrada (cálculo da distância de cada vetor de peso de todos os neurônios do mapa, onde a menor indica o neurônio mais próximo, ou com maior intensidade, e a maior os neurônios adjacentes, ou com menor intensidade). No final do treinamento, cada neurônio representa um subconjunto de objetos usado no treinamento, que são semelhantes entre si, mas que também possuem características similares com os neurônios mais próximos e distintas em relação aos neurônios mais distantes. Na representação bidimensional, o treinamento dos dados multidimensionais de entrada compõe uma matriz pré-definida, de x colunas e y linhas.

No contexto desta pesquisa a utilização do SOM se justifica pela variada quantidade de métricas que podem ser obtidas a partir de código Scratch produzido por alunos e a necessidade de exploração visual da correlação com outras variáveis cujo efeito no desenvolvimento do Pensamento Computacional ainda não é bem esclarecida.

3. Métodos e técnicas

A Oficina de Produção de Jogos Digitais, descrita anteriormente em (MUÑOZ *et al.*, 2015; BARCELOS *et al.*, 2014) foi criada com o objetivo de promover o desenvolvimento de competências do Pensamento Computacional relacionadas à automação de processos e construção de algoritmos por meio da construção de jogos digitais no ambiente Scratch. Sua estrutura, inspirada em princípios do construcionismo e da Aprendizagem Baseada em Problemas (ABP), propõe desafios de complexidade crescente a partir dos quais os participantes são convidados a explorar conceitos relacionados ao desenvolvimento de jogos (animação de *sprites*, colisão, controles por teclado e mouse) e a fundamentos de programação (variáveis, estruturas condicionais, laços e mensagens). Os desafios estão, na maior parte do tempo, relacionados à construção de jogos digitais reais.

No escopo deste estudo foram selecionados dois oferecimentos da Oficina a alunos com perfis semelhantes. Em 2013, a Oficina foi oferecida em 12 semanas para alunos do curso técnico concomitante ao ensino médio do Instituto Federal de São Paulo, matriculados na disciplina de Lógica de Programação. Naquela ocasião, 40 alunos de 15 a 17 anos participaram das atividades. No primeiro semestre de 2017 a Oficina foi novamente oferecida na disciplina de Lógica de Programação, mas agora para os alunos do curso técnico integrado ao ensino médio. Novamente o oferecimento aconteceu para 40 participantes matriculados no primeiro ano do ensino médio, com idades entre 14 e 15 anos. O tempo disponível para aplicação das atividades foi ligeiramente maior nesta ocasião, chegando a 18 semanas.

A estrutura da Oficina nos anos de 2013 e 2017 é apresentada na Tabela 1. A estrutura prevê que nas primeiras semanas as atividades são relacionadas aos fundamentos de programação e da utilização do ambiente do Scratch e que a partir do segundo encontro os estudantes já começam a implementar jogos com funcionalidades completas. Deve-se ressaltar que no oferecimento de 2017 foram acrescentados dois novos jogos – um Simulador de Batalha Aérea, e uma versão digital do conhecido Jogo da Velha. Por razões administrativas, três semanas foram alocadas para procedimentos avaliativos e exercícios de teoria de programação. Na coluna referente ao oferecimento em 2013 tais atividades foram omitidas por terem ocorrido após as 12 semanas de atividades da Oficina em si.

Outra alteração refere-se ao projeto final proposto para os alunos: enquanto que em 2013 todos os alunos desenvolveram uma versão do jogo Pacman, em 2017 foi feita uma proposta mais aberta. Como houve tempo hábil para introduzir a construção de desenhos via programação no Scratch, de forma muito semelhante aos gráficos da linguagem Logo, foi solicitado que os participantes desenhassem uma grade regular de 10 x 10 quadrados, e que o jogo criado por cada equipe acontecesse nesse espaço da tela.

Tabela 1. Programação da oficina em 2013 e 2017.

VERSÃO 2013

Semana	Atividade/Conteúdo
1	Conceito de Algoritmo. Familiarização com o Ambiente do Scratch. Conceitos de sprites e colisão entre sprites
2	Variáveis e Laços de Repetição. Criar o jogo Pescaria.
3	Laços de Repetição e Estruturas Condicionais. Criar o jogo Adivinhe o Número
4	Criar o jogo Pedra-Papel-Tesoura
5-6	Criar o jogo Simulação de Guerra
7-8	Criar o jogo Breakout
9-11	Criar o jogo Pacman (projeto final)
12	Apresentação dos Projetos Finais

VERSÃO 2017

Semana	Atividade/Conteúdo
1	Conceito de Algoritmo. Familiarização com o Ambiente do Scratch. Conceitos de sprites e colisão entre sprites
2	Variáveis e Laços de Repetição. Criar o jogo Pescaria.
3	Laços de Repetição e Estruturas Condicionais. Criar o jogo Adivinhe o Número
4	Criar o jogo Simulação de Batalha Aérea
5	Exercícios de teoria de programação
6	Criar o jogo Pedra-Papel-Tesoura
7	Criar o Jogo da Velha
8-9	Criar o jogo Simulação de Guerra
10	Semana de avaliação intermediária
11	Finalizar o jogo Simulação de Guerra
12-13	Criar o jogo Breakout
14-15	Gráficos e procedimentos em Scratch
16-18	Criar o projeto final da disciplina
19	Semana de avaliação

Em ambos os oferecimentos da Oficina todos os arquivos Scratch produzidos pelos alunos ao longo do tempo foram coletados. Os arquivos foram processados utilizando o Hairball e executando o plugin *mastery*[1], que implementa a coleta das métricas definidas por Moreno-León e Robles (2015). Dessa forma, cada programa processado gerou um registro composto pela identificação do jogo desenvolvido, a identificação do aluno, o oferecimento da Oficina no qual o jogo foi desenvolvido (2013 ou 2017) e as métricas para cada uma das sete categorias da rubrica. Foram processados 220 arquivos desenvolvidos pelos alunos em 2013 e 333 arquivos produzidos pelos alunos em 2017, em um total de 553 registros. Cabe ressaltar em que ambas as situações alguns programas foram desenvolvidos em equipes de dois ou três alunos pela própria dinâmica colaborativa da atividade, particularmente o projeto final.

A análise de agrupamento e da correlação existente entre os dados foi realizada por intermédio da linguagem R e o seu pacote Kohonen (*Supervised and Unsupervised Self-Organising Maps*) com a função *som* parametrizada conforme apresentado na Tabela 2.

Tabela 2. Parametrização da função som utilizada para análise.

Parâmetro	Configuração
Dados	Base de 553 registros
Grid	Dimensão 8x8 com *lattice* Regular
Vizinhança	Hexagonal
Épocas	2000
Taxa de Aprendizado	Inicial 0,05 com redução linear até 0,01

[1] https://github.com/jemole/hairball/blob/master/hairball/plugins/mastery.py

O treinamento da rede SOM foi realizado com a pontuação de cada programa nos níveis de competência das sete categorias definidas na rubrica de Moreno-León e Robles (2015). O propósito da análise foi verificar a influência do nível de competência mensurado em cada categoria na composição de cada nodo e a consequente distribuição topológica dos elementos da base de dados sobre o mapa. Os demais dados de cada registro (jogo, aluno e ano de oferecimento da oficina) não fizeram parte do agrupamento e foram analisados para a identificação de tendências.

4. Resultados

A visualização dos pesos do vetor de características em cada nodo do mapa como um gráfico de elementos circulares permite a análise de tendências de agrupamento e similaridade dos dados. Na Figura 1 é apresentado o mapa com essa visualização e a visão equivalente na qual o número do jogo implementado em cada programa é exibido dentro de cada nodo.

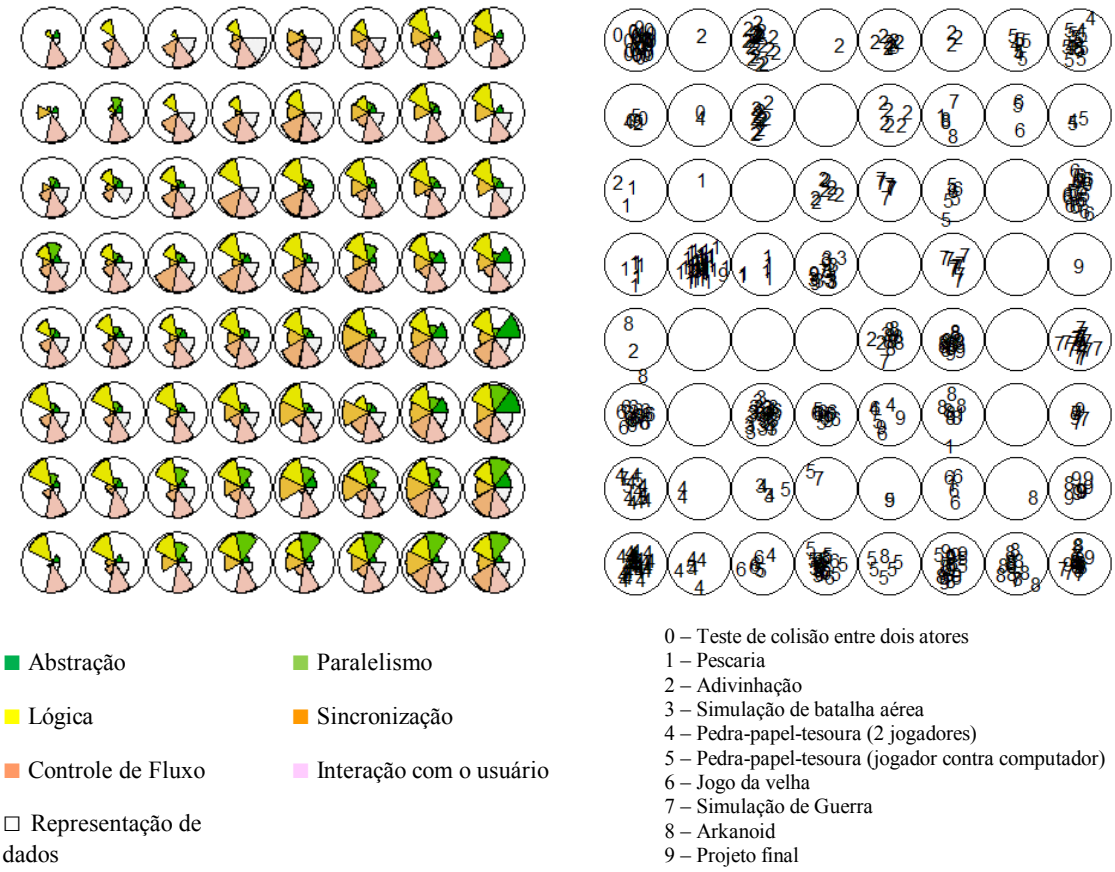

Figura 1. Mapa de características produzido pelo treinamento da rede SOM e agrupamento por tipo de jogo.

A topologia de 8x8 nodos foi selecionada experimentalmente visando à minimização da distância entre os nodos. Considerando a característica da distância entre nodos, é possível identificar de forma mais marcante o agrupamento de alguns jogos. Na Figura 2 a seguir o mapa de características é apresentado sobreposto por uma indicação preliminar de agrupamentos. O critério para definir um agrupamento foi a predominância

de instâncias de um jogo em um ou mais nodos adjacentes no mapa. Esse critério busca produzir uma melhor visualização da coesão dos agrupamentos gerados. Em azul são indicados os jogos cujo agrupamento resultou em um único *cluster*, em vermelho aqueles que foram agrupados em dois *clusters* e em verde aqueles que foram agrupados em três *clusters*. O código numérico que rotula cada cluster é o mesmo atribuído a cada jogo e utilizado na Figura 1.

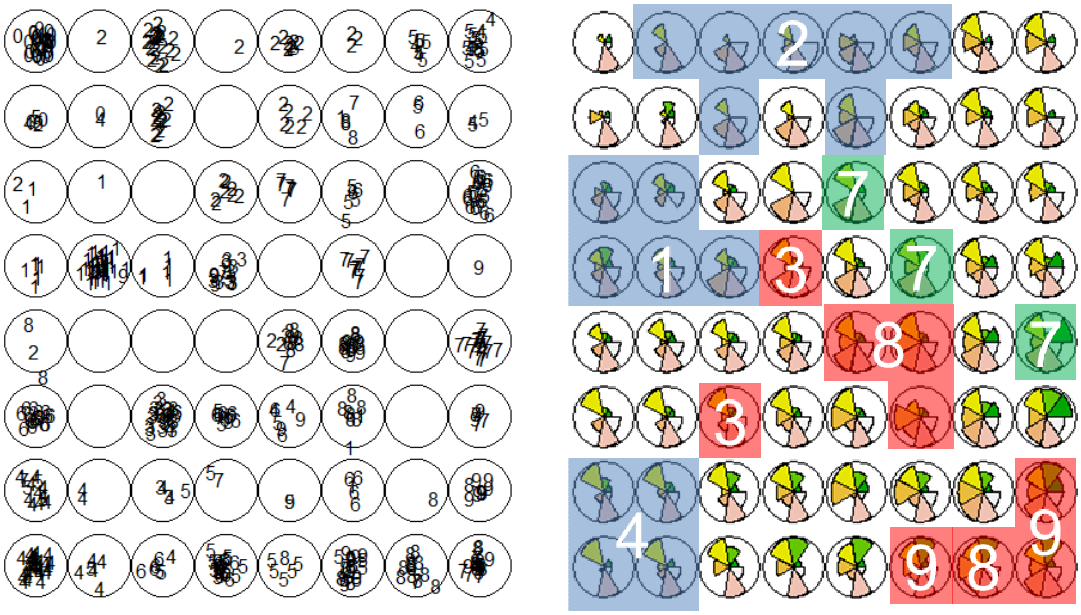

Figura 2. Mapa de características sobreposto pelos agrupamentos identificados.

Os jogos 1 (Pescaria), 2 (Advinhação) e 4 (Pedra-Papel-Tesoura Manual) possuem características que podem explicar o agrupamento em um único *cluster*: as estruturas de programação utilizadas sofrem pouca variação, especialmente no caso dos dois primeiros jogos, que são as atividades introdutórias da Oficina. No caso do jogo 4 (Pedra-Papel-Tesoura manual), também constata-se que tanto os mecanismos de controle propostos, que são as teclas do teclado, quanto o funcionamento lógico do jogo, baseado na verificação do estado da jogada feita por cada um dos jogadores humanos, não permite muitas variações na implementação. O jogo com a numeração 0 se trata de um teste inicial de colisão entre dois atores controlados pelo teclado, desenvolvido pelos alunos no primeiro encontro, que teve os menores valores nas categorias definidas pela rubrica devido à simplicidade da funcionalidade proposta.

Já ao analisar os jogos que não apresentaram coesão de agrupamento, ou seja, não foram agrupados em um único *cluster* é possível identificar potenciais diferenças que se confirmam na observação *in loco* das duas edições da Oficina, realizada por um dos autores do artigo. As instâncias produzidas do jogo 3 (Simulador de Batalha Aérea) foram agrupadas em dois nodos, cujos pesos se diferenciam predominantemente na categoria de "Controle de Fluxo" (laranja escuro). Na definição da rubrica, é possível verificar que o nível mais alto de competências nessa categoria relaciona-se ao uso da estrutura de repetição "repita até que", pela exigência da definição de uma condição booleana de parada. Como a proposta desse jogo demanda a programação de objetos que representam uma nave e um submarino inimigos que se movimentam na tela com

uma animação complexa (BOE *et al.*, 2013), verificou-se que cada um dos nodos agrupou implementações do jogo com uma maior ou menor sofisticação no uso de tais estruturas de controle.

Os projetos finais (jogo 9), apesar de estarem localizados em nodos próximos no mapa, apresentam duas diferenciações importantes: a presença de um peso maior para o "Controle de fluxo" no *cluster* mais à direita. Em análise conjunta com o ano de oferecimento da Oficina, foi possível verificar que todos os jogos do *cluster* à direita foram oriundos da oficina de 2017. Considerando a similaridade dos pesos das demais características do Pensamento Computacional nos dois *clusters*, é possível inferir que a estratégia de utilizar um projeto com temática mais "aberta", conforme mencionado anteriormente, pode ter possibilitado aos alunos exercitar de forma mais efetiva algumas competências em relação à edição de 2013.

No caso do jogo 7 (Simulação de Guerra) é possível identificar uma característica também vinculada ao ano de oferecimento da disciplina, conjuntamente com o acréscimo de novas funcionalidades no Scratch. Neste caso dois nodos agruparam exclusivamente jogos desenvolvidos em 2013 e um nodo os jogos desenvolvidos em 2017. Neste último a categoria com peso mais diferenciado foi a "Abstração" (verde escuro), cujos valores mais altos na rubrica se relacionam ao uso de "clones", um mecanismo da linguagem introduzido em sua versão 2.0 que permite a cópia de um objeto com todo o seu comportamento, mantendo valores próprios de atributos para cada cópia, de forma muito semelhante ao mecanismo de instanciação presente em linguagens orientadas a objeto. A versão 2.0 do ambiente foi utilizada apenas em 2017, quando os alunos foram estimulados a explorar o mecanismo de clonagem para simplificar a implementação de um tiro de canhão proposto na mecânica do jogo. Esse aspecto muito provavelmente explica o agrupamento identificado no mapa.

No caso do jogo 8 (Arkanoid) a diferença entre os dois *clusters* identificados se concentra em um destacado peso para a categoria "Paralelismo" (verde claro) no *cluster* situado na porção inferior do mapa. Os valores mais altos dessa característica na rubrica envolvem o uso de mais de um *script* em um mesmo ator que trate o recebimento de mensagens, que é o mecanismo mais sofisticado possível para detectar e tratar um evento muito frequente no jogo: a colisão da bola com um dos blocos a serem destruídos. Como neste caso os *clusters* não agruparam exclusivamente jogos desenvolvidos em 2013 ou 2017, o peso das características forneceu um indício para direcionar uma análise mais específica do código dos jogos em questão.

A principal limitação identificada na análise do agrupamento no mapa refere-se aos jogos 5 (Pedra-Papel-Tesoura, com jogador contra computador) e 6 (Jogo da Velha). As instâncias desses jogos se distribuíram de maneira mais esparsa por sete nodos do mapa sem um padrão mais claro de proximidade. Ambos os jogos têm uma característica técnica semelhante: a possibilidade do uso de mensagens para indicação e tratamento de eventos relevantes da mecânica do jogo (a jogada do jogador humano, no Pedra-Papel-Tesoura, e a verificação de vitória ou derrota logo após uma jogada no Jogo da Velha). Uma análise mais aprimorada desses jogos será necessária para identificar se este é um caso da adoção de diferentes estratégias de implementação por parte dos alunos ou uma limitação da rubrica em identificar as características relevantes desses jogos, permitindo o agrupamento.

Quanto à definição da rubrica e sua relação com o processo de agrupamento do algoritmo para treinamento da rede, verificou-se uma uniformidade nos pesos atribuídos à característica "Interação com o usuário". A rubrica considera que o nível mais alto de competência nessa característica relaciona-se ao uso dos novos recursos de reconhecimento de vídeo e áudio presentes no Scratch 2.0. Como a Oficina não previa a utilização desses recursos, é esperado que a avaliação dos jogos dos participantes seja sempre abaixo do máximo nesse critério, provavelmente o tornando pouco determinante no processo de treinamento. Devido ao fato desses novos recursos não envolverem nenhuma complexidade para sua modelagem ou uso, pois conceitualmente tratam-se apenas de variáveis pré-definidas que indicam a maior ou menor presença de movimentação no vídeo ou volume no áudio capturado no microfone, seria possível questionar se seu uso realmente denotaria um nível mais alto de competência no Pensamento Computacional.

5. Conclusões e trabalhos futuros

A análise automatizada do código produzido por alunos é uma técnica promissora para identificar e monitorar as estratégias de resolução de problemas empregadas e sua relação com o Pensamento Computacional. No entanto, técnicas mais sofisticadas de análise podem ser necessárias para avaliar o avanço da sofisticação das estratégias utilizadas por um aluno ao longo do tempo e mesmo a comparação dos resultados obtidos por ele no desenvolvimento de um programa em relação a soluções de referência previamente coletadas.

Este artigo apresentou uma validação preliminar do uso de mapas auto-organizáveis como técnica de análise de programas desenvolvidos em Scratch a partir da rubrica proposta por Moreno-León e Robles (2015) e implementada no software Dr. Scratch. Os resultados apresentaram, na maioria dos casos, um agrupamento dos registros relacionados a jogos do mesmo tipo no mapa. Em granularidade mais fina, os agrupamentos ainda apresentaram coerência com características dos jogos vinculadas a estratégias utilizadas pelos alunos e previamente observadas em sala de aula. Os resultados obtidos até o momento permitem concluir a viabilidade e o potencial do uso de mapas auto-organizáveis como estratégia de análise.

Considerando os potenciais e limitações do presente estudo, propõe-se como trabalhos futuros as seguintes linhas: (i) incorporar outras métricas de software no processo de treinamento do mapa, como aquelas relacionadas a *bad smells*; (ii) incorporar variáveis relacionadas ao nível prévio de competência dos alunos no processo de treinamento do mapa; (iii) testar o potencial do mapa previamente treinado em estimar o nível de competência evidenciado por um aluno em novos programas desenvolvidos por ele.

Referências

AIVALOGLOU, E.; HERMANS, F. How Kids Code and How We Know: An Exploratory Study on the Scratch Repository. ICER '16, 2016. **Proceedings of the 2016 ACM Conference on International Computing Education Research**. New York, NY, USA: ACM, 2016. p. 53–61.

ARAÚJO, A. L. S. O.; ANDRADE, W. L.; GUERRERO. Um Mapeamento Sistemático sobre a Avaliação do Pensamento Computacional no Brasil. In: 2º WORKSHOP DE ENSINO EM PENSAMENTO COMPUTACIONAL, ALGORITMOS E PROGRAMAÇÃO, 2016, Uberlândia. **V Congresso Brasileiro de Informática na Educação**. Uberlândia: SBC, 2016.

BARCELOS, T. *et al.* Informal HCI: Fixing Playability Issues As A Strategy To Improve The Skills Of Novice Programmers. **IEEE Latin America Transactions**, v. 12, n. 1, p. 29–35, jan. 2014.

BOE, B. *et al.* Hairball: lint-inspired static analysis of scratch projects. 2013, Denver, Colorado, USA. **Proceeding of the 44th ACM technical symposium on Computer science education**. Denver, Colorado, USA: ACM, 2013. p. 215–220.

BRENNAN, K.; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. In: **Proceedings of AERA 2012**. Vancouver: American Educational Research Association, 2012.

KOHONEN, T. Essentials of the self-organizing map. **Neural Networks**, v. 37, p. 52–65, jan. 2013.

KOHONEN, T. Self-organized formation of topologically correct feature maps. In: ANDERSON, J. A.; ROSENFELD, E. (Org.).**Neurocomputing: foundations of research**. [S.l.]: MIT Press, 1988. p. 509–521.

MORENO-LEÓN, J.; ROBLES, G. Analyze your Scratch projects with Dr. Scratch and assess your Computational Thinking skills. In: 7TH INTERNATIONAL SCRATCH CONFERENCE, 2015, Amsterdam. **Proceedings of Scratcm2015AMS**. Amsterdam: [s.n.], 2015.

MUÑOZ, R. *et al.* Diseño e Implementación de un Taller de Programación de Juegos Digitales con Scratch como Apoyo a Fundamentos de Programación. In: IV CBIE E X LACLO. **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. Maceió: Sociedade Brasileira de Computação, 2015.

SILVA, L. A.; PERES, S. M.; BOSCARIOLI, C. **Introdução à Mineração de Dados com Aplicações em R**. Rio de Janeiro: Elsevier, 2016.

TECHAPALOKUL, P. Sniffing Through Millions of Blocks for Bad Smells. SIGCSE '17. **Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2017. p. 781–782.

THE CSTA STANDARDS TASK FORCE. **CSTA K-12 Computer Science Standards**. . New York: ACM Computer Science Teachers Association, 2011. Disponível em: <http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf>. Acesso em: 3 fev. 2012.

TIOBE. **TIOBE Index - The Scratch Programming Language**. . [S.l: s.n.]. Disponível em: <https://www.tiobe.com/tiobe-index/scratch/>. Acesso em: 14 jul. 2017. , 2017

WOLZ, U.; HALLBERG, C.; TAYLOR, B. Scrape: A tool for visualizing the code of Scratch programs. In: 42ND SIGCSE, 2011, Dallas. **Proceedings of the 42nd ACM Technical Symposium on Computer Science Education**. Dallas: ACM, 2011.