UM FRAMEWORK DE ANÁLISE DE DADOS PARA PROJETOS SCRATCH

JORGE NACHTIGALL[1]; TIAGO PRIMO[2]; ANA MARILZA PERNAS[3]

[1]*Universidade Federal de Pelotas – jlnvjunior@inf.ufpel.edu.br*
[2]*Universidade Federal de Pelotas – tiago.primo@inf.ufpel.edu.br*
[3]*Universidade Federal de Pelotas – marilza@inf.ufpel.edu.br*

1. INTRODUÇÃO

O desenvolvimento tecnológico trouxe impacto em todas áreas de nossa vida. Uma dessas áreas é a da educação. Cada vez mais novos projetos educacionais amparados em conceitos da computação criativa (L. Zhang; H. Yang, 2013), pensamento computacional (WING, 2006) e ciência da computação ganham popularidade e levam a alunos de todas as idades novas formas criativas e modernas de aprender. A grande parte destes projetos busca alternativas acessíveis e de fácil compreensão para aplicar conceitos de algoritmos e resolução de problemas, assim como, o desenvolvimento de aspectos socio-emocionais (ex: colaboração, empatia ou capacidade de lidar com a auto-crítica). Uma dessas alternativas é o Scratch.

Como demonstrado por Mitchel Resnick em sua publicação (RESNICK et al., 2009), o Scratch foi desenvolvido tendo como público alvo crianças entre 8 a 16 anos, porém conta com uma grande audiência adulta também. Como se trata de uma VPL, o código escrito é substituído por blocos de programação. Possuindo uma interface simples, o Scratch demonstra-se uma ótima porta de entrada para crianças e adultos os quais queiram arriscar os primeiros passos com uma linguagem de programação.

Devido a todos esses fatores citados, o Scratch apresenta uma crescente popularidade em projetos educacionais. Em conjunto, surge a necessidade de uma plataforma que forneça algum tipo de validação relativa aos conteúdos desenvolvidos nos mesmos. Em contrapartida, a análise automatizada de tais dados não é uma tarefa trivial. A aplicação da interdisciplinaridade e de intervenção humana nestes dados faz-se necessária em muitos casos de forma a garantir a sua confiabilidade, compreensão e estabelecimento de inferências mais complexas. Os trabalhos de Bryce Boe com o Hairball (BOE et al., 2013), Moreno León com Dr. Scratch (Moreno-León, 2015) utilizam o Scratch em diversos domínios como ferramenta de ensino. Entretanto a avaliação de seus trabalhos ainda é dependente de processos manuais e pouco exploratórios.

Este artigo descreve a implementação de um framework que coleta dados provenientes da interação dos alunos participantes de clubes de computação criativa, com o uso da ferramenta Scratch em uma versão modificada, realizando o acompanhamento de todos os passos de desenvolivmento de seus projetos.

2. METODOLOGIA

O desenvolvimento da aplicação foi dividido em 3 etapas de estudo e implementação, e cada uma delas será descrita a seguir. São elas: estudo da anatomia dos blocos Scratch, coleta de dados: estudo sobre a scratch-vm e seus eventos e o desenvolvimento de uma API para manipulação dos dados coletados.

Na etapa de estudo da anatomia dos blocos Scratch, analizamos o funcionamento e estrutura de todos blocos presentes nas variadas categorias da

ferramenta. Após a inserção de um bloco em um projeto o mesmo assume a forma de um *JSON* (*JavaScript Object Notation*) para posteriormente ser interpretado pela máquina virtual do Scratch. Esta estrutura contém todas as informações sobre um bloco específico, como por exemplo: sua identificação única no projeto, o nome do bloco, os parâmetros de entrada e campos de preenchimento específicos, o bloco ao qual ele está conectado como filho bem como o bloco seguinte conectado a ele.

Embora a estrutura interna de um bloco Scratch seja genérica, nem todos os blocos preenchem todos os campos contidos nela, apresentando algumas variações nesta estrutura. Por isso foi realizado um levantamento individual desses blocos descrevendo todos os campos utilizados e seus significados. O acesso a tabela que contem estas informações pode ser feito através do seguinte link: https://bit.ly/2vhgH02.

Para a implementação da coleta de dados no framework, foi realizada uma análise da estrutura interna do Scratch. Em sua versão 3.0. O Scratch é construído através da junção de diversos componentes os quais constituem a ferramenta. O scratch-gui, desenvolvido em React, engloba toda a interface gráfica da ferramenta para criar e rodar projetos. Ele é a porta de entrada para os demais componentes.

A cada interação realizada na interface do Scratch outros dois componentes são acionados: o scratch-blocks e o scratch-vm. O scratch-blocks provém especificações de design e a base de código para a criação de interfaces interativas. Através dele temos acesso ao *workspace* do Scratch, podendo assim interagir com os blocos e construir nossos códigos. Este componente trabalha juntamente com o scratch-vm (*Scratch Virtual Machine*), pois é através dele que todo o código criado pelo scratch-blocks é construído, representado e executado.

O componente scratch-vm é acionado através de eventos emitidos pela interface do scratch-blocks, assim sempre que alguma interação com o *workspace* da ferramenta é realizado um evento é gerado para a scratch-vm construir o estado atual do código. A construção do estado do código é normalizada para uma estrutura JSON contendo todas as informações presentes no projeto, como os blocos presentes no código, informações sobre imagens utilizadas, variáveis criadas e versão do scratch-vm na qual o estado foi gerado.

Conhecendo os eventos emitidos pelo componente scratch-blocks para a scratch-vm podemos capturar com precisão cada interação realizada pelo usuário da ferramenta. A Fig. 1 apresenta a listagem de alguns eventos emitidos pelo componente scratch-blocks e seus respectivos significados.

Nome do evento	Descrição do evento
`PROJECT_START`	Nome do evento quando o projeto é iniciado.
`PROJECT_STOP_ALL`	Nome do evento quando o projeto é parado pelo usuário.
`TARGETS_UPDATE`	Nome do evento de atualização do workspace de blocos.
`BLOCK_DRAG_UPDATE`	Nome do evento de atualização da ação de pegar um bloco.
`EXTENSION_ADDED`	Nome do evento que reporta que uma nova extensão foi adicionada.
`PERIPHERAL_LIST_UPDATE`	Nome do evento de atualização da lista de periféricos disponíveis.
`PERIPHERAL_CONNECTED`	Nome do evento que reporta quando um novo periférico é conectado.

FIG. 1: EVENTOS DO COMPONENTE SCRATCH-VM.

Após o estudo e aprofundamento de seus componentes componentes, lançamos uma versão própria do Scratch, contendo alterações em sua estrutura para a realização da coleta dos sinais de interação dos alunos com seus projetos. Essa versão modificada nos permite a identificação cada usuário e capturar todas as modificações realizadas no *workspace*.

As ações definidas que ativam a captura dos dados foram selecionadas de forma à compreender o caminho realizado pelo aluno na construção do seu programa e suas tentativas de executar o código. Sempre que o aluno adiciona, remove ou troca um bloco de lugar, bem como tenta executar seu programa, uma nova entrada no banco de dados do *framework* é gerada contendo a identificação deste aluno, o horário em que a ação ocorreu, o estado atual do código através do arquivo JSON gerado pela scratch-vm e qual evento desencadeou esta nova entrada no banco de dados. As entradas no banco de dados relativas a modificações nos blocos são geradas através do evento "DRAGS_UPDATE" e entradas relativas a tentativas de execução do código por parte do aluno são geradas através do evento "PROJECT_START".

Com base nos eventos escolhidos para ativarem a captura de dados, temos informações suficientes para diferenciarmos os estados entre si e identificarmos todas as mudanças ocorridas entre as coletas, bem como temos também a listagem de todos os estados nos quais o aluno executou seu código. Tais ações nos capacitam a construção do caminho percorrido pelo aluno durante a criação do seu projeto e possibilita a realização de inferências futuras relativas as tentativas de execução do mesmo.

Para possibilitar a manipulação dos dados coletados conforme descritos na subseção B, foi realizado o desenvolvimento de uma *API* na linguagem de programação Python. Através dessa *API* o usuário pode ter acesso aos dados coletados pela ferramenta de Scratch modificada descrita aqui.

A *API* fornece ao usuário diversos métodos para o resgate dos dados coletados. A partir do instanciamento da classe principal, o usuário define qual turma deseja obter os dados fornecendo a chave de identificação da mesma, obtendo assim todos os estados e suas informações relativas aos alunos presentes na turma selecionada.

Essas informações compreendem no número de blocos presentes em um estado, quais blocos estão presentes, a listagem de todas as informações de cada um destes blocos, a listagem de atores presentes no estado e o conteúdo veiculado a cada um deles, como as fantasias e áudios. Desta forma temos acesso a todos os dados coletados presentes dentro de um projeto Scratch, seja de forma macro, através de consultas em cima de uma turma afetando todos os alunos presentes nela, ou de forma micro, analisando individualmente cada um dos alunos.

3. RESULTADOS E DISCUSSÃO

Com a implementação da *API* temos o fluxo do framework completo conforme podemos ver na Fig. 2. Disponibilizamos todas as ferramentas necessárias para a realização da coleta e manipulação dos dados, possibilitando a avaliação de estados específicos do desenvolvimento dos projetos dos alunos.

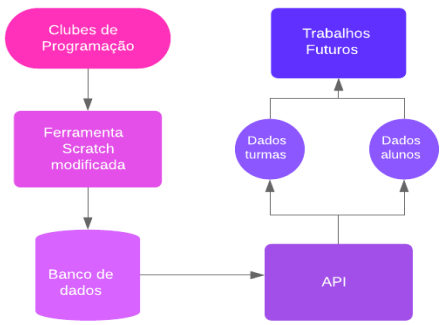

FIG. 2: FLUXOGRAMA DO FRAMEWORK

Este framework já encontra-se em ação coletando dados de oficinas realizadas junto aos clubes de computação criativa na cidade de Pelotas e espera-se que até o fim do ano seja realizada uma análise completa de cada uma destas oficinas, permitindo mensurar os avanços obtidos nestas análises ao utilizarmos o framework, porém já é evidente o ganho em conhecimento de cada uma das etapas de desenvolvimento dos projetos feitos pelos alunos, permitindo a realização de mapeamentos entre os estados coletados pelo framework e interações realizadas dentro da sala de aula, sejam elas entre os próprios alunos ou interações do professor com a turma.

4. CONCLUSÕES

Neste artigo apresentamos um framework o qual provê mecanismos de suporte para os professores realizarem a avaliação de seus alunos em projetos desenvolvidos no Scratch. Possibilitamos o desenvolvimento de uma ferramenta capaz de mensurar o impacto e analisar o uso desta como ambiente de ensino, bem como a realizar inferências até mesmo sobre o comportamento de alunos e suas características. Demonstramos uma versão modificada do Scratch capaz de coletar todas as interações dos alunos com seus projetos, bem como coletar todos os estados do desenvolvimento destas tarefas. Agregado a isso, temos uma *API* que disponibiliza métodos os quais permitem a manipulação destes dados coletados, possibilitando ao professor visualizá-los e utilizá-los da forma que for conveniente ao seu trabalho. Todos estes processos abrem novas possibilidades na área de análise de projetos Scratch, permitindo uma avaliação pontual e maior compreensão das tomadas de decisões dos alunos.

Os trabalhos futuros consistem na expansão da interdisciplinaridade com outras áreas de conhecimento para o desenvolvimento de modelos de inteligência artificial capazes de realizarem análises mais complexas relativas aos dados coletados e análises preditivas quanto ao desenvolvimento das atividades.

5. REFERÊNCIAS BIBLIOGRÁFICAS

L. Zhang and H. Yang, "Definition, research scope and challenges of creative computing" 2013 19th International Conference on Automation and Computing, London, 2013, pp. 1-6.

WING, Jeannette M. Computational thinking. Communications of the ACM, v. 49, n. 3, p. 33-35, 2006.

RESNICK, Mitchel et al. Scratch: Programming for all. Commun. Acm, v. 52, n. 11, p. 60-67, 2009.

Moreno-León, Jesús, Gregorio Robles, and Marcos Román-González. "Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking." RED. Revista de Educación a Distancia 46 (2015): 1-23.

BOE, Bryce et al. Hairball: Lint-inspired static analysis of scratch projects. In: Proceeding of the 44th ACM technical symposium on Computer science education. ACM, 2013. p. 215-220.