ScratchAnalytics: Um Framework para Coleta e Análise de Interações em Projetos Scratch

Jorge Luis Nachtigall Vaz Junior, Ana Marilza Pernas, Tiago Thompsen Primo

[1]Programa de Pós-Graduação em Computação
Universidade Federal de Pelotas (UFPel)

{jlnvjunior,tiago.primo,}@inf.ufpel.edu.br

Abstract. *Technological advancement and its impact on our lives gradually highlights the need to use technological resources in the educational environment. One of these resources is Scratch, a block programming tool, widely used in educational projects which aim to provide students with the teaching of programming from a creative and constructionist perspective. As a result, evaluations and validations become necessary, in order to provide more information on the impact generated by these projects. This work presents a proposal to facilitate the execution of these analyzes, through the implementation of a framework which automates the process of collecting and analyzing interactions made by students in Scratch.*

Resumo. *O avanço tecnológico e seu impacto em nossas vidas gradativamente deixa em evidência a necessidade da utilização de recursos tecnológicos no ambiente educacional. Um destes recursos é o Scratch, uma ferramenta de programação em blocos, amplamente utilizada em projetos educacionais que visam proporcionar a alunos o ensino de programação sob uma perspectiva criativa e construcionista. Como resultado, avaliações e validações tornam-se necessárias, visando fornecer maiores informações do impacto gerado por estes projetos. O presente trabalho apresenta uma proposta para facilitar a execução destas análises, através da implementação de um framework o qual automatiza o processo de coleta de interações feitas por alunos no Scratch.*

1. Introdução

A revolução causada pela maior acessibilidade e avanço dos mais diferentes tipos de tecnologias em nossa sociedade gerou diversas mudanças nas mais diferentes áreas. Uma destas, é a área da educação. Cada vez mais novos projetos educacionais amparados em conceitos da computação criativa [Zhang and Yang 2013], pensamento computacional [Wing 2006] e ciência da computação ganham popularidade, levando a alunos de todas as idades alternativas criativas para o processo de aprendizado. 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 ferramenta capaz de desenvolver estas competências sobre uma perspectiva criativa é o Scratch.

O Scratch é uma VPL (*Visual Programming Language*) [Burnett 1999] desenvolvida pelo MIT Media Lab em 2007. É uma ferramenta gratuita, de código aberto e fácil

DOI: 10.5753/cbie.sbie.2020.862

acesso, rodando no próprio navegador ou através da sua aplicação offline. Além disso, o Scratch estimula a colaboração, as trocas de experiências e possibilita aos seus usuários a capacidade de lidar com sugestões e críticas relativas ao seus projetos de forma construtiva.

Como demonstrado por [Resnick et al. 2009], o Scratch foi desenvolvido tendo como público alvo crianças entre 8 a 16 anos, porém a sua comunidade conta com usuários de todas as idades. Como se trata de uma VPL, o código escrito é substituído por blocos de programação. Possuindo uma interface simples, nomenclatura dos blocos intuitiva e por manter os usuários longe de mensagens de erro, o Scratch mostra-se uma ótima porta de entrada para crianças e adultos que queiram arriscar os primeiros passos com uma linguagem de programação, sob uma perspectiva criativa, construtiva e colaborativa. Tais facilidades permitem que rapidamente usuários possam criar seus projetos e compartilhá-los no site.

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 na ferramenta. Trabalhos como o de [Gomes et al. 2014], que utilizou o Scratch como meio de incentivar o interesse de meninas do ensino médio em áreas da computação, e o trabalho desenvolvido por [Santana et al. 2017], que levou o ensino da computação de uma maneira criativa para 209 alunos do 5º ao 9º ano da rede pública de Itajaí, em Santa Catarina, são exemplos de projetos desenvolvidos que utilizam Scratch como ferramenta de ensino, porém utilizam para a análise de seus resultados, processos manuais ou ferramentas que limitam o compreendimento do processo de criação dos alunos.

Desta maneira, o trabalho aqui descrito, tem como proposta a implementação de um *framework* para coleta e análise de interações realizadas por usuários em projetos de computação criativa desenvolvidos na ferramenta Scratch, permitindo uma melhor compreensão sob cada etapa do processo de criação dos alunos. Espera-se que o *framework* contribua para o desenvolvimento da área de análise baseada em interações, em projetos de computação criativa no Scratch, uma vez que a mesma até o presente momento foi pouco explorada.

O texto deste trabalho está organizado em 5 seções. Na seção 1 é apresentada uma breve introdução e contextualização do trabalho. A seção 2 apresenta os trabalhos relacionados. A seção 3 apresenta todas as etapas de desenvolvimento do trabalho. A seção 4 descreve o experimento e apresenta os resultados. Por fim, a seção 5 descreve as conclusões do trabalho e trabalhos futuros esperados para complementarem o *framework*.

2. Trabalhos Relacionados

O Hairball é um analisador estático para projetos Scratch. Foi desenvolvido em python e é constituído através da implementação de plugins que definem quais conceitos do código serão analisados. A análise rotula cada uma das instâncias de um projeto Scratch, baseando-se nos conceitos analisados do plugin como: correto, semanticamente incorreto, incorreto ou incompleto [Boe et al. 2013].

A ferramenta foi utilizada em um acampamento de verão aplicado pelos autores do projeto e sua análise de resultados foi realizada em cima de 58 projetos Scratch, coletados

ao longo de 2 semanas de acampamento. Os conceitos analisados neste experimento buscaram verificar a implementação correta do estado inicial de um programa Scratch, a sincronização entre os blocos de fala e o blocos de som, o uso apropriado de blocos de transmissão e recebimento de mensagens, e a utilização de animações.

Os testes mostraram-se satisfatórios para o autor, uma vez que o Hairball apresentou melhores taxas de acertos do que a análise manual dos projetos para os conceitos mencionados anteriormente, contendo uma taxa de falso positivo de 0,5% e uma taxa de falso negativo 13,5% [Boe et al. 2013].

O Dr. Scratch é uma ferramenta web de código aberto e gratuita que pode ser utilizada por professores e alunos para analisar projetos desenvolvidos no Scratch [Moreno-León et al. 2015]. A análise do Dr. Scratch é constituída por dois pontos: procura por boas práticas de programação no projeto e pontuação em categorias baseadas em competências do pensamento computacional.Cada projeto analisado recebe uma pontuação entre 0 e 21 resultantes da combinação de pontuações individuais para estas categorias. As categorias analisadas e os critérios de pontuação estão descritos na Figura 1, retirada do artigo de [Moreno-León et al. 2015]. Foram realizados *workshops* para testar a ferramenta com alunos entre 8 e 14 anos de oito escolas diferentes e os resultados mostraram que os alunos foram capazes de melhorar suas pontuações em 1,45 pontos após a primeira análise de seus projetos no Dr. scratch.

O trabalho de [Aivaloglou and Hermans 2016] descreve uma análise feita sob um *dataset* de mais 250 mil projetos retirados do repositório público do Scratch. A análise inicial realizada consistiu na verificação dos tipos de blocos usados, tamanho dos códigos dos projetos coletados, o uso de abstrações e conceitos de programação aplicados aos projetos e, por fim, más práticas de programação. Os resultados demonstraram que os códigos desenvolvidos no Scratch normalmente são pequenos e não utilizam em sua maioria conceitos mais complexos de programação como estruturas de repetição e condicionais.

Os trabalhos relacionados citados, junto dos demais trabalhos apresentados na Seção 1, evidenciam a ampla utilização do Scratch como ferramenta educacional e reforçam a importância da utilização de ferramentas para realização de análises sobre os projetos desenvolvidos.

Levando em consideração todos os aspectos individuais levantados dentre os trabalhos correlacionados, podemos notar que todas ferramentas realizam suas análises apenas sob o estágio final de desenvolvimento dos projetos. É justamente onde este trabalho insere sua principal contribuição no panorama atual. Como principal diferencial, o *framework* aqui descrito possibilita a compreensão de todo processo de criação desenvolvido por um aluno. Desta forma, por exemplo, podemos identificar e correlacionar eventos externos ao Scratch, como estímulos realizados pelo professor (dicas, aconselhamentos, etc) em sala de aula, com os dados coletados no exato momento em que estes eventos ocorrem, buscando entender melhor como estes eventos impactam nas interações dos alunos.

3. Framework

Nesta seção iremos falar sobre as etapas de estudo e desenvolvimento do *framework* para coleta de interações dos alunos junto a ferramenta Scratch.

CT Concept	Competence Level			
	Null (0)	Basic (1 point)	Developing (2 points)	Proficiency (3 points)
Abstraction and problem decomposition	-	More than one script and more than one sprite	Definition of blocks	Use of clones
Parallelism	-	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, create clone, two scripts when %s is > %s, two scripts on when backdrop change to
Logical thinking	-	If	If else	Logic operations
Synchronization	-	Wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
Flow control	-	Sequence of blocks	Repeat, forever	Repeat until
User Interactivity	-	Green flag	Key pressed, sprite clicked, ask and wait, mouse blocks	When %s is >%s, video, audio
Data representation	-	Modifiers of sprites properties	Operations on variables	Operations on lists

Figura 1. Critérios de pontuação utilizados no Dr. Scratch.

3.1. Anatomia dos Blocos Scratch

Visando obter maior compreensão do funcionamento interno da ferramenta Scratch e dos dados por ela gerados, foi realizado um estudo sobre a estrutura dos blocos nela presentes.

Após a inserção de um bloco em um projeto, o mesmo é incorporado a uma estrutura na forma de um JSON para posteriormente ser interpretado pela máquina virtual do Scratch, o componente scratch-vm, o qual será discutido na subseção 3.2. A parte que define a representação de um bloco nesta estrutura JSON está demonstrada na Figura 2.

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 através de uma tabela a qual possui os dados completos dos mesmos[1].

3.2. Componentes da Ferramenta Scratch e seus Eventos

Para a implementação da coleta de dados no *framework*, foi realizada uma análise da estrutura interna do Scratch. O Scratch 3.0 foi desenvolvido em HTML5 e *JavaScript*, e é constituído através da junção de diversos componentes os quais formam a ferramenta. Compreender estes componentes é de extrema importância, pois cada um desempenha uma função diferente no funcionamento da ferramenta.

O componente scratch-gui, implementa a interface gráfica da ferramenta para criar e executar projetos. O componente scratch-blocks é o responsável pela implementação e funcionamento dos blocos no Scratch 3.0. Através deste componente são definidos os visuais e funcionamento de cada um dos blocos, assim como os eventos de interação (os quais serão descritos mais à frente) à serem são emitidos. O scratch-vm é o componente

[1]https://bityli.com/3uccY

```
{
  "_blocks": {
    "Q]PK~yJ@BTV8Y~FfISeo": {
      "id": "Q]PK~yJ@BTV8Y~FfISeo",
      "opcode": "event_whenkeypressed",
      "inputs": {
      },
      "fields": {
        "KEY_OPTION": {
          "name": "KEY_OPTION",
          "value": "space"
        }
      },
      "next": null,
      "topLevel": true,
      "parent": null,
      "shadow": false,
      "x": -69.333333333333,
      "y": 174
    }
  },
  "_scripts": [
    "Q]PK~yJ@BTV8Y~FfISeo"
  ]
}
```


Figura 2. Anatomia de um bloco Scratch: Estrutura JSON.

utilizado para executar os códigos desenvolvidos no Scratch. Através dele também é feita a representação do código, através da estrutura ilustrada na Figura 2 e discutida na subseção 3.1.

As alterações no código ocorrem em um trabalho conjunto entre a scratch-vm e o scratch-blocks. Sempre que um evento é emitido pelo scratch-blocks, o mesmo é capturado pela scratch-vm que realiza as alterações necessárias baseadas no evento recebido.

Os eventos capturados pela scratch-vm nos proporcionam a possibilidade de compreendermos as interações que são realizadas no Scratch por um usuário. Através deles temos informações relativas às mudanças no código ou interações feitas em lugares específicos da interface gráfica da ferramenta. Existem diversos eventos que são emitidos pelos componentes do Scratch, porém no escopo deste trabalho, que é o de compreensão do caminho de desenvolvimento de um projeto feito por um aluno, vamos focar no estudo apenas dos seguintes eventos:

- "BLOCK_DRAG_UPDATE": este evento é disparado pelo componente scratch-blocks toda vez que ocorre alguma interação do usuário com algum bloco presente no *workspace*. Estas interações podem envolver a adição de um novo bloco, a remoção ou simplesmente o remanejo do bloco no código.
- "TARGETS_UPDATE": este evento é disparado sempre que o usuário interage com a área de pré-visualização do projeto no Scratch. Esta é a área a qual apresenta o resultado do seu projeto baseado na lógica desenvolvida no código e nos blocos utilizados. Através dela é possível interagir diretamente com os atores presentes na cena e testar o funcionamento do seu projeto.
- "PROJECT_START": este evento é disparado sempre que o usuário tenta executar seu projeto utilizado o botão "bandeira verde" na interface gráfica do Scratch.

Através destes eventos podemos identificar as interações realizadas pelos alunos, qual ação foi tomada, de que forma isso impactou no estado atual do código e assim, podemos capturar com precisão cada interação realizada pelo usuário da ferramenta.

Após o estudo e aprofundamento de seus componentes, lançamos uma versão própria do Scratch, contendo alterações em sua estrutura para a realização da coleta de interações dos alunos com seus projetos baseado nos eventos já listados. 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.

Desta forma temos informações suficientes para diferenciarmos os estados entre si e identificarmos todas as mudanças ocorridas entre as interações, bem como temos também a listagem de todos os estados nos quais o aluno executou seu código. Estas informações nos possibilitam visualizar o caminho percorrido pelo aluno durante a criação do seu projeto e viabilizam análises sob o processo completo de desenvolvimento dos projetos.

Por fim temos a implementação de uma API em Python para resgate das informações relativas aos projetos presentes no banco de dados, concluindo assim o fluxo de funcionamento do *framework* completamente conforme podemos ver na figura 3.

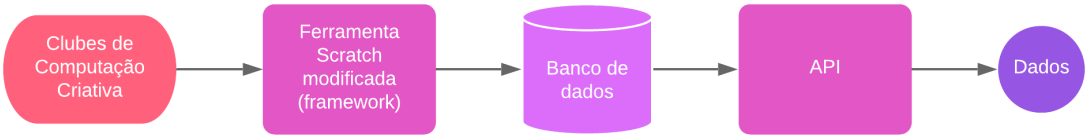

Figura 3. Fluxograma do framework.

3.3. Oficina

A oficina desenvolvida para a utilização na etapa de experimentação do *framework*, a qual terá seus resultados descritos na seção 4, consistiu em uma atividade lúdica com o objetivo dos alunos desenvolverem uma máquina do tempo, amparada pelos conceitos de aprendizagem criativa [Resnick 2014] e computação criativa [Brennan et al. 2011].

Ao todo, foram elaboradas 6 etapas de desenvolvimento diferentes dentro oficina, onde as 4 primeiras possuem foco na programação dos elementos do projeto com os blocos Scratch, a 5ª etapa voltada para teste das funcionalidades implementadas, e a 6ª etapa aberta para os alunos criarem novos elementos para o projeto, sejam eles visuais ou voltados para a programação de blocos.

Para cada etapa da oficina, um material de apoio foi disponibilizado aos alunos, utilizando uma dinâmica semelhante aos Scratch Cards[2] desenvolvidos pelo MIT Media Lab, porém com o diferencial de que os blocos demonstrados nestes cartões embora resolvam a tarefa proposta para a respectiva etapa, ao mesmo tempo encontram-se fora de ordem, cabendo ao aluno descobrir qual a melhor forma de conectá-los em seu projeto. O modelo de um destes cartões pode ser visto na Figura 4.

[2]https://resources.scratch.mit.edu/www/cards/pt-br/scratch-cards-all.pdf

Figura 4. Cartão dica apresentado na etapa 1 da oficina.

4. Estudo de Caso e Resultados

A etapa de experimentação do *framework* consistiu na aplicação da oficina descrita na subseção 3.3, realizada em uma escola municipal da cidade de Pelotas, no estado do Rio Grande do Sul, em uma turma dos clubes de computação criativa, através de um projeto em parceria da Universidade Federal de Pelotas e a prefeitura da cidade. A turma na qual o experimento foi realizado consistiu num total de 7 alunos, entre 8 e 12 anos de idade. Todos os alunos presentes na turma já haviam em ocasiões anteriores realizado projetos utilizando Scratch, não sendo esta a primeira experiência delas utilizando a ferramenta e também eram alunas frequentes do clube de computação criativa. A oficina teve duração de aproximadamente 1 hora. Todos os dados coletados são totalmente anônimos.

Para a análise dos resultados, quebramos o evento "BLOCK_DRAG_UPDATE", descrito na seção 3.2, em três novos eventos, que representam os três possíveis tipos de interação que o mesmo representa. Os novos eventos são: "ADD", "REMOVE" e "DRAG". O evento "ADD" descreve uma interação de adição de um bloco ao *workspace*. O evento "REMOVE" descreve uma interação de remoção de um bloco presente no *workspace*. Por fim, o evento "DRAG" descreve uma interação na qual o usuário reposicionou um bloco já presente no *workspace* e não realizou a remoção do mesmo.

A tabela 1 demonstra o número total de interações realizadas pelos alunos, classificadas pelos seus tipos. Ao longo de toda a oficina, foram coletadas 1290 interações. Podemos ver que o evento com maior número de interações realizadas é o "TARGETS_UPDATE", com 664 interações realizadas, o que indica que os alunos realizaram mais interações com a área de pré-visualização e interação, do que de fato interações com os blocos. O evento com o menor número de interações é o "REMOVE", com apenas 37 interações. Um fato curioso com relação a esta métrica, é que o aluno 3 apresentou um total de 0 interações do tipo "REMOVE". Estes baixos valores para exclusões de blocos podem indicar diversas situações referentes aos alunos e a própria tarefa a desenvolvida, por exemplo: o material de apoio recebido pode ter suprido todas e quaisquer dificuldades dos alunos ou até mesmo a tarefa pode ser fácil demais para a turma. Estes tipos de conclusões requerem uma análise mais profunda entre diversos outros aspectos, mas podemos ter uma ideia do potencial e a riqueza de informações que podem ser obtidas através da coleta de interações.

Tabela 1. Número total de interações de cada tipo realizadas pelos alunos.

Aluno	Eventos				
	ADD	DRAG	REMOVE	TARGETS_UPDATE	PROJECT_START
1	26	8	3	139	2
2	28	15	3	142	5
3	25	63	0	91	10
4	36	41	6	60	11
5	36	78	12	138	7
6	38	76	12	20	1
7	26	35	1	74	22
$\sum$	215	316	37	664	58

Tabela 2. Resultados estatísticos relativos às interações realizadas pelos alunos ao longo das tarefas.

		Eventos				
		ADD	DRAG	REMOVE	TARGETS_UPDATE	PROJECT_START
Etapa 1	Média	5	10.4285	1.8571	2	0.4285
	Mediana	4	10	1	2	0
	Desvio Padrão	2.3904	7.4230	2.0995	2.2038	0.7284
Etapa 2	Média	5	25.1428	1.2857	4.8571	4
	Mediana	4	21	1	5	1
	Desvio Padrão	2.3904	16.8304	1.3850	2.7479	4.4721
Etapa 3	Média	5.5714	5.7142	1.1428	11.4285	3.5714
	Mediana	4	6	0	11	4
	Desvio Padrão	2.9692	2.9623	1.4568	4.1007	2.3211
Etapa 4	Média	8.4285	1.5	0.5714	17	0.1428
	Mediana	8	2	0	15	0
	Desvio Padrão	0.7284	1.1180	0.7284	13.4589	0.3499
Etapa 5	Média	4.1428	2.1428	0.2857	21.8571	0.1428
	Mediana	4	2	0	16	0
	Desvio Padrão	0.3499	2.5314	0.4517	19.2014	0.3499
Etapa 6	Média	0.5714	0.2857	0.1428	37.7142	0
	Mediana	0	0	0	37	0
	Desvio Padrão	0.9035	0.6998	0.3499	26.2445	0

A Tabela 2 apresenta os resultados estatísticos baseado nos tipos de interações realizadas pelos alunos em cada uma das 6 etapas da oficina. Podemos ver que, em média, as 4 primeiras etapas da oficina apresentam um valor consistente para interações relacionadas à adição e manipulação dos blocos no workspace, ou seja, eventos do tipo "ADD" e "DRAG", os quais contrastam com os baixos valores iniciais para interações mais relacionadas à interface do Scratch, como o evento "TARGETS_UPDATE" o qual vai crescendo ao decorrer da oficina. Isso se dá pelo fato de que, conforme descrito na subseção 3.3, as etapas iniciais focam basicamente em tarefas voltadas para a implementação de recursos utilizando os blocos Scratch. Desta forma, com o passar do tempo de aplicação da oficina, podemos ver o evidente crescimento das interações "TARGETS_UPDATE", indicando que os projetos estão avançados o suficiente para serem testados e prontos para a interação com o usuário.

A Figura 5 demonstra o número total de blocos utilizados pelos alunos ao longo

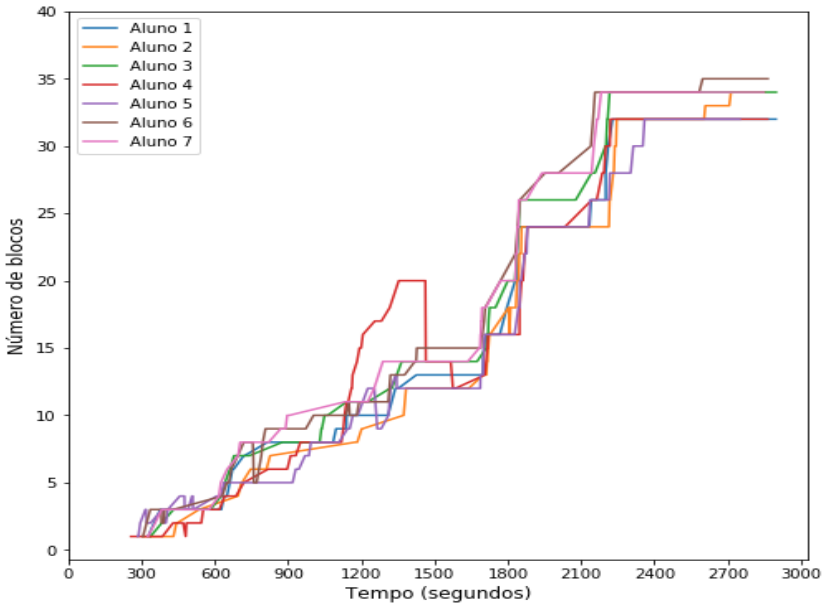

Figura 5. Quantidade de blocos utilizadas pelos alunos ao longo da oficina.

de toda a aplicação da oficina. Podemos notar uma grande homogeneidade nesta métrica entre os alunos da turma. O aluno 4 durante o segundo quarto da oficina, apresentou uma leve alteração, adicionando mais blocos que os demais alunos durante um espaço de tempo considerável, porém acabou excluindo-os e voltando para a o fluxo geral da turma. Este é outro ponto muito interessante que pode ser explorado com coletas de dados neste nível de capacidade: investigar que acontecimentos levam os alunos a manterem-se ou desviarem-se do comportamento geral da turma e, utilizando o caso citado como exemplo, o que os leva a voltarem ao fluxo após apresentarem alterações como a descrita.

Com a aplicação desta oficina pode-se concluir que o *framework* cumpriu o seu objetivo, provendo primeiramente a possibilidade da coleta de dados baseado em interações dentro da ferramenta Scratch modificada, apresentando ótimo desempenho e estabilidade. Também foi possível evidenciar o potencial que o tipo de coleta disponibilizada pelo *framework* provê, possibilitando análises baseadas não somente em estados únicos dos projetos, e sim de todo o processo de desenvolvimento realizado pelos alunos.

5. Conclusões e Trabalhos Futuros

O trabalho aqui descrito teve como objetivo a implementação de um *framework* para a coleta de interações de alunos na elaboração de seus projetos no Scratch. Para alcançar este objetivo, foram realizadas modificações sobre a ferramenta, resultando em um Scratch personalizado capaz de obter todos os sinais de interações dos alunos com os blocos e com a área de pré-visualização de seus projetos. As principais contribuições deste trabalho a serem destacadas são o desenvolvimento de um mecanismo, capaz de coletar de forma automatizada, informações relativas as interações de alunos em seus projetos Scratch. Também destaca-se o estudo realizado sobre a estrutura de funcionamento dos blocos e dos componentes do Scratch, possibilitando uma maior compreensão da ferramenta. Estas contribuições mostram-se importantes para o desenvolvimento da área de análise de projetos Scratch, apresentando uma nova maneira de obter informações sob o

processo de criação de usuários da ferramenta.

Os trabalhos futuros pretendidos para este projeto envolvem a implementação de métodos de inteligência artificial para a realização de inferências mais complexas relativas aos dados obtidos nas coletas. Também pretende-se disponibilizar o *framework*, adicionando um painel de visualização, que mostre informações personalizadas relativas às análises realizadas sob as interações dos alunos como suporte aos professores que utilizem Scratch como ferramenta de ensino em projetos de computação criativa.

Referências

Aivaloglou, E. and Hermans, F. (2016). How kids code and how we know: An exploratory study on the scratch repository. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, pages 53–61. ACM.

Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., and Franklin, D. (2013). Hairball: Lint-inspired static analysis of scratch projects. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 215–220. ACM.

Brennan, K., Chung, M., and Hawson, J. (2011). Creative computing: A design-based introduction to computational thinking. *Retrieved May*, 9:2012.

Burnett, M. (1999). Encyclopedia of electrical and electronics engineering, chapter what is visual programming.

Gomes, W. F., Louzada, C. S., Nunes, M. A. S. N., Salgueiro, E. M., and Andrade, B. T. (2014). Incentivando meninas do ensino médio à área de ciência da computação usando o scratch como ferramenta. In *Anais do Workshop de Informática na Escola*, volume 20, page 223.

Moreno-León, J., Robles, G., and Román-González, M. (2015). Dr. scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia*, (46):1–23.

Resnick, M. (2014). Give p'sa chance: Projects, peers, passion, play. In *Constructionism and creativity: Proceedings of the Third International Constructionism Conference. Austrian Computer Society, Vienna*, pages 13–20.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J. S., Silverman, B., et al. (2009). Scratch: Programming for all. *Commun. Acm*, 52(11):60–67.

Santana, A. L. M., de Jesus, E. A., Raabe, A., Santana, L., Cucco, L., and Ramos, G. (2017). Tem ideia na rede: Inserindo o pensamento computacional na rede municipal de ensino. In *Anais do Workshop de Informática na Escola*, volume 23, page 1032.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.

Zhang, L. and Yang, H. (2013). Definition, research scope and challenges of creative computing. In *2013 19th International Conference on Automation and Computing*, pages 1–6. IEEE.