Avaliação de Ferramentas para Ensino de Programação para Crianças e Adolescentes

Renata F. Pontes[1], Jaindson V. Santana[1], Mirko B. Perkusich[1], Anderson F. Barbosa[1], Vitor H. Gomes[1], Moisés Simões[1], Carlos A. S. Camelo[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Campus Monteiro (IFPB) – Monteiro, PB – Brasil

{renata.pontes, jaindson.santana}@ifpb.edu.br, {mperkusich, anderson.barbosa}@gmail.com, vitor_gomes_18@live.com, {moises316wwe, carlos.avelino2.0}@gmail.com

Abstract. *There are many tools available to teach computer programming and Computational Thinking (CT), however, a challenge to select them for an instructional unit. The goal of this paper is to compare tools Scratch and Stencyl from the viewpoint of the efficacy of teaching CT. For this purpose, we executed a case study we groups of students from the eighth and ninth grade. For each group, we designed and implemented similar instructional units, in which each group worked with only one of the evaluated tools. To measure the development of CT during the study, we used a questionnaire based on the Bebras Challenge. As a result, we verified a similarity in the evolution of both groups of students, and, therefore, of the efficacy of the tools. Scratch had a small advantage in terms of the time to reach the learning objectives.*

Resumo. *Diversas ferramentas estão disponíveis para o ensino de programação de computadores e do Pensamento Computacional (PC), porém, um desafio é definir quais serão utilizadas para ensinar crianças ou adolescentes. O objetivo deste artigo é comparar as ferramentas Scratch e Stencyl quanto a sua eficácia no ensino do PC. Para tal, foi realizado um estudo de caso com turmas compostas por alunos do 8º e 9º anos do ensino fundamental. Para cada turma, unidades instrucionais similares foram planejadas e executadas, cada, utilizou uma das ferramentas sob análise. As habilidades do PC, durante o estudo, foram mensuradas com base no Bebras Challenge. Como resultado, percebeu-se uma similaridade na evolução das turmas e, portanto, da eficácia das ferramentas, mesmo se percebendo uma leve vantagem da ferramenta Scratch, em termos de uma curva de aprendizado e de tempo para o alcance dos objetivos esperados.*

1. Introdução

Com todo o avanço tecnológico, a programação tem se tornado uma habilidade tão fundamental quanto aprender a ler e escrever. O interesse da comunidade científica e dos governos de diversos países pelo ensino-aprendizagem da programação e do pensamento computacional, a partir da educação básica, aparecem em estudos como Mannila et al. (2014).

Segundo Resnick (2012), existe uma visão equivocada sobre a capacidade dos nativos digitais em relação ao uso de tecnologias, pois ele relata, na sua reflexão, que os

jovens dedicam grande parte do seu tempo consumindo tecnologias (seja através das redes sociais ou dos jogos) porém, isso não os tornam fluentes em tecnologia. Dessa forma, eles acabam desenvolvendo a capacidade de serem bons consumidores de informações, produzidas e filtradas através da tecnologia, mas não aprendem a produzir conhecimento novo, não sendo, portanto, indivíduos capazes de entender o potencial de criação existente através dos recursos tecnológicos disponíveis.

Em Grover e Pea (2013), como resultado de uma revisão na literatura, foi concluído que o uso de Pensamento Computacional (PC) em escolas, apesar de ter sido estudado nos anos 80 e 90 com as Linguagens de Programação LOGO, devem ser reavaliadas com as novas linguagens disponíveis.

O ensino de programação aos jovens possibilita o desenvolvimento de diversas capacidades, dentre elas, o aprimoramento do raciocínio lógico dos estudantes. De acordo com um grupo de defesa do ensino de ciência da computação no ensino fundamental (Code.org *Advocacy Coalition*), "A Ciência da Computação desenvolve as habilidades de Pensamento Computacional e crítico dos alunos e mostra como criar, não simplesmente usar, novas tecnologias. Este conhecimento fundamental é necessário para preparar os alunos para o século 21, independentemente do seu campo de estudo ou ocupação" [Computing in the Core, 2018]. Programar possibilita trabalhar a habilidade de desenvolver uma solução para um problema, necessitando também de outras habilidades, criando, assim, com interdisciplinaridade. Algumas sugestões para se trabalhar essas habilidades em um cenário multidisciplinar podem ser encontradas em Sendova (2006) e Wursthorn (2005).

Na Europa, o European Schoolnet (2014) publicou um relatório sobre a situação em vinte países europeus no ano de 2014 em relação ao ensino de programação para crianças e adolescentes. Em treze desses países, a programação já estava presente ou em iminência de ser introduzida na educação fundamental. Em sete desses países, como por exemplo, a Estônia, a Inglaterra e a Grécia, a programação já estava incluída como parte obrigatória do currículo. No entanto, são poucas as crianças e adolescentes aqui no Brasil que têm a oportunidade de aprender a programar, pois nossas escolas, em sua grande maioria, não incluem programação em seus currículos.

Para ensinar os conceitos de programação para crianças e adolescentes pode-se usar diversas ferramentas já estudadas na literatura, por exemplo: Scratch (RESNICK et al., 2009), Alice (PAUSCH, 1995), Kodu (STOLEE, 2010), Greenfoot (KOLLING, 2010), Lego Mindstorms (Lego Mindstorms, 2019) e Stencyl (Stencyl, 2019). Diversos artigos relatam experiências utilizando essas ferramentas, de forma individualizada, para verificar o desempenho do aprendizado dos alunos nas escolas (CHAUDHARY et al., 2016, HERMANS et al., 2017, MEYER et a., 2016). Por outro lado, há uma escassez de estudos empíricos comparativos entre ferramentas com relação sua eficácia no ensino do pensamento computacional.

Neste contexto, neste artigo, apresenta-se o resultado de um estudo empírico no qual duas ferramentas, Scratch e Stencyl, foram comparadas com relação eficácia no ensino do pensamento computacional. Para tal, foi realizado um estudo de caso com duas turmas compostas, majoritariamente, por alunos do 8º e 9º anos do ensino fundamental de uma escola pública. Para cada turma, unidades instrucionais similares foram planejadas e executadas, tendo cada turma usando uma das ferramentas sob análise. Para mensurar o desenvolvimento do Pensamento Computacional durante o

estudo, foram utilizados questionários baseados no *Bebras Computing Challenge* (2019). Como resultado, percebeu-se uma similaridade na evolução das turmas e, portanto, das ferramentas, mesmo percebendo uma leve vantagem da ferramenta Scratch em termos de uma curva de aprendizado e de tempo para o alcance dos objetivos esperados.

O restante deste artigo é estruturado como se segue: na Seção 2, é apresentada a metodologia do estudo empírico realizado; na Seção 3, os resultados do estudo empírico são apresentados; e, por fim, na Seção 4, as considerações finais.

2. Metodologia

Realizar avaliações comparativas de plataformas de ensino de programação para crianças não é trivial e atualmente há uma escassez de modelos de avaliação especificamente voltados para este intuito. Nesta seção, uma abordagem para guiar a realização desse estudo empírico comparativo de plataformas de ensino de programação para crianças e adolescentes é apresentada.

Na Etapa 1: definição dos objetivos de avaliação. Tais objetivos podem estar relacionados à motivação, experiência do aluno ou aprendizagem [PETRI et al. 2017]. Para o trabalho foi escolhido avaliar a aprendizagem do aluno, no que diz respeito ao desenvolvimento das habilidades do Pensamento Computacional. Para a condução do estudo, um grupo de alunos do 8º e 9º anos de uma mesma escola, foram escolhidos e organizados em e duas turmas, nivelando, assim, o nível de conhecimento deles.

Na Etapa 2: definição das unidades de análise e plataformas. Nesta foram definidas as quais ferramentas seriam utilizadas, e como seria realizado o acompanhamento da evolução do aprendizado dos alunos com o uso da ferramenta. As ferramentas escolhidas foram: Scratch (Scratch, 2019), por ser a que mais apareceu em estudos e artigos, no levantamento bibliográfico feito, e o Stencyl, baseado na comparação do mesmo com outras ferramentas que tinham características/objetivos similares ao Scratch, podendo assim, ser possível uma melhor análise dos resultados. Com relação a forma de acompanhamento da aprendizagem, optou-se pela divisão das etapas de ensino em: (i) realização de nivelamento em termos de raciocínio computacional; (ii) uma atividade avaliativa contendo a metade do assunto a ser ensinado sobre conceitos de programação; e (iii) realização de atividade avaliativa contendo a segunda parte do assunto definido. Com essas etapas seria possível aplicar uma avaliação para mensurar as habilidades do Pensamento Computacional de cada aluno, tais como: abstração, decomposição, avaliação, pensamento algorítmico. A escolha das questões presentes nos instrumentos avaliativos baseou-se no *Bebras Computing Challenge*[1] que, além de já ser utilizado pelo mundo para a avaliação de crianças e adolescentes quanto a seu desempenho em PC, já apresenta a classificação das questões de acordo com a idade e habilidade a ser avaliada.

Na Etapa 3: planejamento das unidades instrucionais. Foi planejado uma unidade instrucional específica para cada ferramenta, respeitando suas características, mas com as ementas em conformidade, para minimizar o efeito da didática e do instrutor na performance dos estudantes, dado que o objetivo era comparar as

[1] http://www.bebraschallenge.org/

plataformas. Planos de aula foram definidos (e.g., sequenciamento de assuntos e tempo reservado) e padronizados, para minimizar efeitos subjetivos e para que cada etapa fosse finalizada de forma concomitante. Além disso, as unidades instrucionais foram planejadas de modo que os alunos não se sentissem parte de um experimento, mas dentro de um ambiente de aprendizagem, como coloca McGowan (2011).

Etapa 4: Operação. Nesta etapa, as aulas foram ministradas e as avaliações foram administradas ao final de cada etapa do experimento, de modo que os dados pudessem ser coletados e analisados. No início da execução das aulas, foi realizada uma avaliação diagnóstica para que se tivesse uma linha de base para análise dos dados no fim do estudo. Logo após a avaliação diagnóstica, como descrito anteriormente, as etapas foram seguidas de forma similar e em paralelo nas duas turmas.

Etapa 5: Análise e interpretação. Com o fim da coleta das informações acerca do desempenho dos alunos ao longo do aprendizado de programação foi possível iniciar a análise. Com a análise, as comparações entre as ferramentas puderam ser feitas e as conclusões puderam ser definidas.

Finalmente, na Etapa 6: toda a documentação sobre o estudo foi gerada. Relatórios foram criados e o relato dessas conclusões para fins de divulgação do experimento está sendo descrito neste artigo.

3. Resultados e Discussão

O objetivo do estudo de caso aplicando a abordagem proposta foi comparar duas ferramentas com relação aos seus efeitos no desenvolvimento no ensino de programação em crianças do 8º e 9º anos. Inicialmente, foi realizada a seleção das plataformas a serem comparadas. A primeira ferramenta (i.e., ferramenta base) selecionada foi o Scratch por ser uma plataforma bastante conhecida e consolidada no meio acadêmico, como dito anteriormente. Para selecionar a segunda ferramenta, utilizou-se como base o levantamento realizado em Gomes (2017) e o conhecimento dos autores. O resultado da avaliação comparativa das ferramentas em função do nível de abstração e objetivo é apresentado na Tabela 1.

Tabela 1. Comparativo entre ferramentas similares ao *Scratch*

Ferramenta	Nível de abstração	Objetivo
Scratch	Alto	Animações ou jogos
Stencyl	Alto	Jogos
AppInventor	Alto	Aplicativo móvel
Yenka	Alto	Simulador 3D
Kodu	Alto	Jogos
Kodable	Alto	Aplicativo ou jogos
Sparki	Alto	Robótica

Dentre as ferramentas analisados, verificou-se que o Stencyl, apesar de ser uma ferramenta profissional e específica para desenvolvimento de jogos, é similar ao Scratch por ter a opção de utilizar programação em blocos, ter uma interface amigável, também ser gratuito e ser no domínio de desenvolvimento de jogos ou animações, fazendo com que o estímulo em relação ao objetivo do curso, para os alunos, fosse o mesmo.

Para a comparar as duas plataformas foi selecionada uma turma de 50 alunos do 8º e 9º anos de uma escola pública municipal. Essa turma foi dividida em duas unidades

de análise, uma para cada plataforma, com uma unidade instrucional específica para cada possuindo os mesmos objetivos de aprendizagem, mas contextualizadas com a plataforma em questão. No início do estudo, em maio de 2018, foi realizado um treinamento de nivelamento com o Code.org com todos os alunos, durante duas semanas, para que todos pudessem partir de um mesmo estágio inicial de conhecimento acerca de programação.

Para a avaliação de cada etapa foi escolhido como instrumento de medição, o *Bebras Challenge*, por ser uma iniciativa internacional de promoção e avaliação do PC (Pensamento Computacional), já consolidada e validada no meio acadêmico. Este instrumento de avaliação é composto de questões classificadas por idade, nível de dificuldade e habilidades (e.g., abstração, generalização, pensamento algorítmico, dentre outros), podendo ter mais de uma habilidade sendo avaliada em uma mesma questão. Optou-se por aplicar a avaliação quatro vezes durante o projeto: uma no início, para se saber a nível inicial dos alunos; uma após o nivelamento das turmas, para se checar o grau de conhecimento adquirido, e assim dar-se continuidade ao estudo; outra no meio do projeto, após dois meses de aulas com as ferramentas; e uma ao final dos quatro meses de aulas com as ferramentas, para se ter ideia de como eles evoluíram ao longo do projeto, sendo possível comparar as duas turmas e ver qual teve o melhor desempenho.

No início das aulas os alunos tiveram dificuldades, já que nunca tinham estudado lógica de programação. Por esse motivo foi aplicado o nivelamento com o Code.org introduzindo conceitos básicos de: variáveis, eventos, sistemas de coordenadas, o fluxo de controle de um programa, estruturas condicionais (IF, ELSE IF, ELSE), operadores booleanos e lógicos, e estruturas de repetição (FOR, WHILE). Com isso eles passaram para a próxima etapa de aulas, que tiveram como foco a criação de jogos 2D utilizando as plataformas Scratch ou Stencyl.

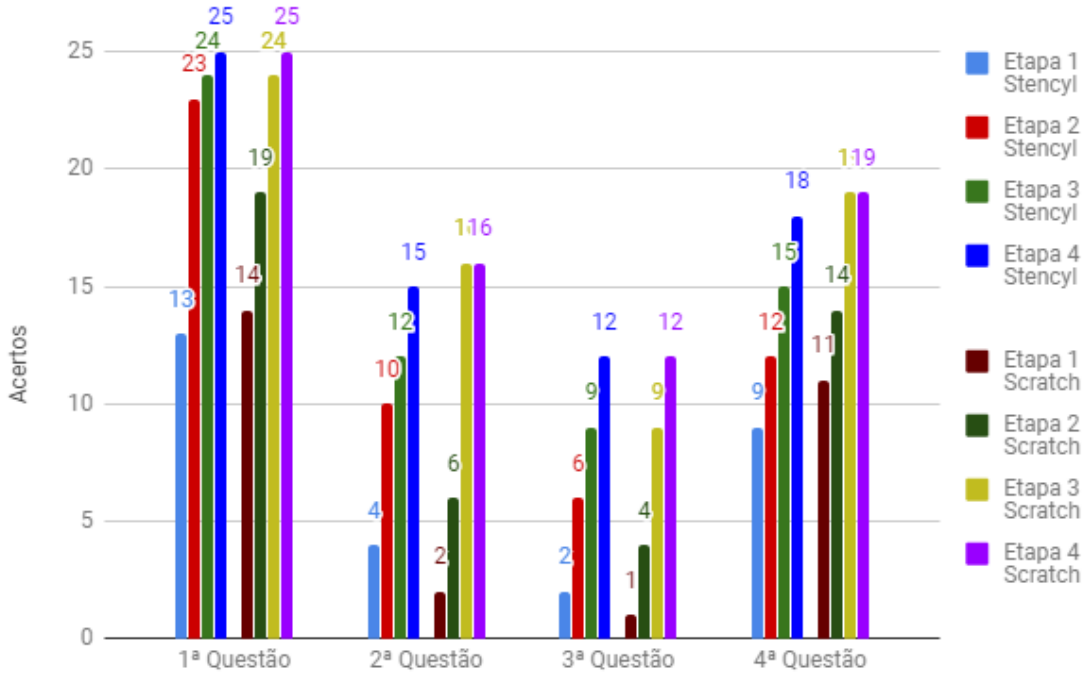

Figura 1 – Número de acertos pelas turmas nas quatro etapas do projeto.

A Figura 1 ilustra o número de acertos das turmas nas provas ao longo das etapas. Com base no gráfico é possível observar uma evolução no número de acertos das questões de forma crescente e, relativamente homogênea, mostrando a curva de aprendizagem esperada com o curso. Observa-se também que o número de acertos de cada turma, na terceira (e penúltima) etapa, já foi grande, mostrando que não era necessário saber todas as etapas de programação do curso para desenvolver a lógica de programação esperada nas avaliações, bastava já ter contato com essa forma de raciocínio através das ferramentas utilizadas. Essa informação já foi importante, de uma forma geral, para mostrar que o ensino de programação, de fato, faz uma diferença, não só para o ato de programar em si, mas para se aprender a resolver problemas de forma computacional, mesmo sem se tornar um programador ao final do processo. Percebe-se também que as plataformas tiveram pouca diferença entre os resultados obtidos, mostrando que as duas atingiram bem o objetivo de desenvolver o pensamento computacional para crianças nessa faixa etária. No entanto, é interessante destacar que a ferramenta Scratch atinge valores iguais ou mais altos já na penúltima etapa, mostrando que, talvez, o desenvolvimento do pensamento computacional seja um pouco mais rápido ao se utilizar essa ferramenta. Além disso, considerando que o resultado da avaliação inicial (pré-nivelamento) também foi menor nesta ferramenta, a curva de aprendizado foi maior para que esses alunos atingissem o mesmo objetivo. Essa pode ser uma informação relevante para escolha de uma ferramenta para cursos de curta duração com esse objetivo.

4. Ameaças à validade

A ameaça à validade de construção refere-se ao quanto às medidas operacionais representam os atributos investigados. Neste estudo, a análise foi baseada em questões do Bebras Challenge, que é uma ferramenta consolidada. Por outro lado, a seleção das questões foi realizada pelos pesquisadores, e isso é um risco a ser considerado.

A ameaça à validade interna refere-se às relações causais. Neste estudo, assume-se que as ferramentas são o único fator relevante no resultado das turmas. Por parte dos pesquisadores, foi realizado um esforço para projetar os cursos de forma o mais similar o possível para minimizar o efeito de diferenças nas unidades instrucionais. Outro fator relevante é o instrutor, que, devido ao seu estilo, pode ter tido mais empatia por parte de uma turma do que da outra.

A ameaça à validade externa refere-se ao poder de generalização dos resultados. Generalizações estatísticas não são possíveis com apenas um estudo de caso da dimensão do realizado. Por outro lado, os resultados podem ser relevantes para escolas que pretendem utilizar uma das duas ferramentas.

4. Conclusão

Esse trabalho teve o objetivo de fazer uma avaliação comparativa entre duas ferramentas já existentes para ensino de programação para crianças e adolescentes. De acordo com levantamento prévio, poucos estudos com esse objetivo foram encontrados, sendo mais encontrados estudos que avaliavam ferramentas de forma isolada em estudos de casos, com objetivos de melhora no desempenho de alunos em outras disciplinas. [Fessakis et al, 2013]. As comparações encontradas geralmente eram do ponto de vista das características das ferramentas, e não estudos de caso com a

avaliação do uso de duas ou mais ferramentas de forma comparativa com o mesmo objetivo.

Foram utilizadas como unidade de análise, duas turmas de crianças entre o 8º e o 9º ano contendo 25 alunos cada, de uma escola pública municipal. Essas turmas foram escolhidas para serem os alvos do estudo em questão por terem crianças com faixa etária muito próxima, e por virem da mesma escola, mantendo, assim, um nível de conhecimento inicial próximo entre as crianças acompanhadas. Após a escolha das ferramentas a serem comparadas, e a forma de avaliar a evolução das turmas, um nivelamento com raciocínio lógico e pensamento computacional foi realizado para que as crianças pudessem sair de um ponto de partida similar. Planos de ensino foram definidos para que o ensino de programação utilizando as duas ferramentas tivesse a evolução mais próxima possível em termos de conteúdo.

Após a finalização do ensino nas turmas, foi realizada a coleta dos dados referentes às avaliações feitas ao longo do curso, tendo sido elas realizadas: ao iniciar os estudos, após nivelamento com code.org, na metade do ensino de programação em si e ao final do curso, algumas conclusões puderam ser observadas. A primeira delas é que houve um crescimento gradual da quantidade de questões acertadas por parte dos alunos, mostrando que, de fato, esse estudo estava desenvolvendo o pensamento computacional e o raciocínio lógico dos alunos. Outra observação importante é que, na metade do curso, os alunos já atingiram médias muito próximas às que iriam atingir ao final do curso, mostrando que a lógica de programação e suas habilidades (e.g., abstração, generalização, pensamento algorítmico) são desenvolvidos mesmo que o aluno ainda não tenha visto todos os conceitos de programação. O contato com o raciocínio computacional introduzido com a ideia de resolução de problemas através das ferramentas utilizadas já surte bastante efeito. E por último, percebeu-se que os alunos que utilizaram o Scratch tiveram uma avaliação inicial pior que a outra turma e, mesmo assim, no meio do curso já atingiram médias muito mais próximas das médias finais, indício que o Scratch, pelo menos nessas crianças, gerou uma curva de aprendizado maior, tendo obtido um desempenho mais rápido no alcance dos objetivos esperados.

Referências

AKPINAR Y., ASLAN, Ü. (2015). "Supporting Children's Learning of Probability Through Video Game Programming". In: Journal of Educational Computing Research, Vol. 53, Edição 2.

Bebras Challenge - Bebras® Computing Challenge. Disponível em: http://www.bebraschallenge.org/. Acesso em 04 de julho de 2019.

CHAUDHARY, Vidushi et al. An experience report on teaching programming and computational thinking to elementary level children using lego robotics education kit. In: 2016 IEEE Eighth International Conference on Technology for Education (T4E). IEEE, 2016. p. 38-41.

Computing in the Core - What is Computer Science and What do People Do Once They Know It? – Disponível em: http://www.computinginthecore.org/. Acesso em 27 de fevereiro de 2018.

European Schoolnet. Computing our future: Computer programming and coding - Priorities, school curricula and initiatives across Europe, October 2014.

Fessakis, G., Gouli, E., Mavroudi, E. Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. Computers & Education, Vol 63, Páginas 87-97, Elsevier, Abril 2013.

GOMES, V. H., PONTES, R. F., CAMELO, C. A. S., CAVALCANTI, G. A. S., PERKUSICH, M. B. Ensino de programação para crianças e adolescentes: um estudo exploratório. In Anais dos Workshops do VI Congresso Brasileiro de Informática na Educação (WCBIE 2017).

GROVER, S., PEA, R. Computational thinking in K-12. A review of the state of the field. Educational Researcher, v. 42, n. 1, p.38-43, 2013.

HERMANS, Felienne; AIVALOGLOU, Efthimia. Teaching software engineering principles to k-12 students: a mooc on scratch. In: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET). IEEE, 2017. p. 13-22.

KOLLING, M. The Greenfoot Programming Environment. ACM Trans. Comput. Educ., v. 10, n. 3, 14, 2010.

Lego Mindstorms. Disponível em: <https://www.lego.com/en-us/mindstorms>. Acesso em 04 de julho de 2019.

MANNILA, L., DAGIENE,V., DEMO, B., GRGURINA, N., MIROLO, C., ROLANDSSON, L. Computational Thinking in K-9 Education. In: Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference - ITiCSE-WGR 2014.

McGowan, H. M. (2011). Planning a comparative experiment in educational settings. Journal of Statistics Education, 19(2).

MEYER, Dany; BATZNER, Ansgar. Engaging computer science non-majors by teaching K-12 pupils programming: first experiences with a large-scale voluntary program. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research. ACM, 2016. p. 174-175.

PAUSCH, R., BURNETTE, T., CAPEHEART, A.C., CONWAY, M., COSGROVE, D., DELINE, R., DURBIN, J., GOSSWEILER, R., KOGA, S., WHITE, J. IEEE Computer Graphics and Applications, 1995.

PETRI, Giani; WANGENHEIM, Christiane Gresse von; BORGATTO, Adriano Ferreti. Evolução de um Modelo de Avaliação de Jogos para o Ensino de Computação. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI-SBC), 25. , 2017, São Paulo. Anais do XXV Workshop sobre Educação em Computação. Porto Alegre: Sociedade Brasileira de Computação, julho 2017 . ISSN 2595-6175

RESNICK M., MALONEY, J., MONROY-HERNÁNDEZ, A., EASTMOND, E., BRENNAN, K., MILLNER, A., ROSENBAUM, E., SILVER, J., SILVERMAN, B., KAFAI, Y., Scratch: Programming for all. Communications of the ACM, v. 52, n. 11, p. 60-67, 2009.

Scratch - Imagine, Program, Share. Disponível em: <https://scratch.mit.edu/>. Acesso em 04 de julho de 2019.

SENDOVA, E. Handling the diversity of learners' interests by putting informatics content in various contexts. In Proc. of the 2nd ISSEP, volume 4226 of LNCS, páginas 71- 82, 2006.

Stencyl - Create Amazing Games Without Code. Disponível em: <http://www.stencyl.com/>. Acesso em 04 de julho de 2019.

STOLEE, K. Kodu Language and Grammar Specification. Microsoft Resarch whitepaper, 2010.Tainton, B. (1990). The unit of analysis 'problem'in educational research. Journal of Educational Research, 6(1):4–19.

WURSTHORN, B. Fundamental concepts of CS in a Logo-environment. In Proc. of EuroLogo, 2005.