OntoScratch: ontologias para a avaliação do ensino de Pensamento Computacional através do Scratch

Nícolas O. de Araujo[1], **Tiago T. Primo**[1], **Ana M. Pernas**[1]

[1]Instituto de Informática – Universidade Federal de Pelotas (UFPel)
Caixa Postal 15.064 – 91.501-970 – Pelotas – RS – Brazil

Abstract. *This paper presents an ontology-based knowledge representation to characterize Scratch projects and the Computational Thinking assessments performed on it, enabling the performance of analyzes and inferences that assist in the understanding o the student's knowledge, how it evolves and other learning-related aspects. Results show the efficiency and potential of this knowledge representation, proving able to faithfully reproduce, through SPARQL queries, the Computational Thinking skills' evaluation of the tool Dr. Scratch.*

Resumo. *Este artigo apresenta uma representação de conhecimento baseada em ontologias para representar um projeto Scratch e as avaliações de habilidades de pensamento computacional realizadas sobre ele ao longo do tempo, como forma de prover suporte para a realização de análises e inferências que auxiliem na compreensão do conhecimento adquirido pelo aluno, sua evolução e outros aspectos relacionados a seu aprendizado. Os resultados demonstram a eficiência e potencial desta representação, sendo capaz de reproduzir fielmente, através de consultas SPARQL, a avaliação de habilidades do pensamento computacional da ferramenta Dr. Scratch.*

1. Introdução

A computação está presente de forma quase intrínseca na vida humana, não sendo possível mais conceber uma sociedade sem o uso do computador [BLINKSTEIN 2019]. Assim, diversas iniciativas visando introduzir conceitos relacionados a computação na educação básica tem iniciado em países da Europa, Estados Unidos e, mais recentemente, no Brasil.

Dentre a lista de habilidades e conhecimentos introduzidos nestas iniciativas, o Pensamento Computacional (PC) talvez seja o mais importante e o menos compreendido [Blikstein 2008]. Ele baseia-se em conceitos de Computação, envolvendo solução de problemas, compreensão do comportamento humano e capacidade de projetar sistemas, representando uma forma de pensamento analítico. Esta habilidade busca ser incentivada pela educação *maker*, uma metodologia que combina o Construcionismo e o Construtivismo Social [Dousay 2017]. Assim, ela combina duas epistemologias distintas: o aprendizado através de interações em um grupo e o aprendizado através do ato de criação.

Nestas iniciativas, o uso de ferramentas de programação visual, como o Scratch, vem sendo a principal abordagem, destacando-se como uma ferramenta prática e de baixo custo. Por sua vez, o Scratch é um ambiente de programação visual desenvolvido pelo MIT Media Lab, que permite a criação de histórias interativas e jogos. Seu objetivo é introduzir conceitos relacionados a programação de forma visual e interativa, aliado ao suporte de uma comunidade que possibilite a colaboração e cooperação. Criado em 2007, o

DOI: 10.5753/cbie.sbie.2020.1823

Scratch vem apresentando crescente popularidade, consolidando sua ferramenta e comunidade, apresentando mais de meio milhão de usuários ativos mensalmente [scr 2020a].

Diversos projetos utilizam-se da ferramenta Scratch para explorar conceitos e práticas computacionais em salas de aula, buscando estimular o desenvolvimento do PC [De França and do Amaral 2013]. Neste cenário, é necessária a geração de evidências para demonstrar o impacto da realização destas atividades. Assim, surge o desafio de armazenar os dados de atividades, além de como utilizá-los para avaliar o progresso de um aluno, ou seja, como visualizar sua evolução. Visando atacar este problema, diversas metodologias de avaliação foram criadas, porém, em sua grande maioria, aplicadas de forma manual e pouco prática [Brennan and Resnick 2012, Wilson et al. 2012, Seiter and Foreman 2013].

Neste sentido, a aplicação Dr. Scratch [Moreno-León et al. 2015] foi proposta como forma de auxiliar a avaliação do projeto resultante de uma atividade. Esta ferramenta realiza uma análise automatizada, pontuando o projeto com base na decomposição da nota em 7 conceitos relacionados, como "Pensamento Lógico" e "Abstração". Desta forma, a ferramenta permite, através do *upload* do projeto, realizar sua avaliação de forma rapida, mostrando-se uma valiosa ferramenta de apoio [Oluk and Korkmaz 2016]. Entretanto, esta avaliação refere-se a apenas um ponto no tempo, ou seja, ao resultado final do projeto. Ela não permite a avaliação do progresso e da evolução do aluno em cada um de seus conceitos. Além disso, surge como obstáculo a ausência de uma representação de conhecimento que permita armazenar estes dados de forma a permitir que sejam trabalhados e gerem evidências sobre o trabalho realizado.

Esta representação de conhecimento possibilitaria a realização de análises, raciocínios e inferências sobre os dados coletados, permitindo a detecção de padrões e a tomada de ação, como a adaptação de atividades com base nas dificuldades dos alunos [Campos et al. 2019]. Esta demanda fica evidente através de trabalhos como [Troiano et al. 2019] e [Seiter and Foreman 2013], onde são realizadas análises sobre o conhecimento construído no ensino do PC, porém as mesmas tem seu escopo limitado por utilizarem o arquivo padrão de projeto, sendo limitadas devido a ausência de um modelo de dados mais abrangente, com capacidade de representação temporal. Além disso, existem iniciativas que visam a coleta destes dados, como [Junior et al. 2019], porém sem a formalização destes dados coletados em um formato padronizado e de fácil inferência para trabalhos futuros. Assim, se faz necessária a organização dos dados coletados, criando sua devida representação de conhecimento de modo a possibilitar a execução de análises temporais e a utilização de modelos de inferência.

Desta forma, o principal objetivo deste trabalho é prover uma representação de conhecimento capaz de caracterizar os dados de um projeto Scratch e de avaliações de habilidades do PC demonstradas nele, permitindo o acompanhamento do aluno e dando suporte para a utilização de mecanismos inteligentes, como ferramentas de inferência.

Este artigo está estruturado da seguinte maneira: a Seção 2 apresenta o escopo e a representação desenvolvida; a Seção 3 traz a análise da avaliação realizada e dos resultados obtidos; por fim, a Seção 4 apresenta as considerações finais e trabalhos futuros.

2. OntoScratch

Esta modelagem foi concebida visando a representação dos projetos desenvolvidos e a avaliação de habilidades do PC, de modo a permitir a realização do acompanhamento do progresso de alunos, provendo suporte a realização de inferências e a criação de ferramentas de suporte para o professor com base em trabalhos, como por exemplo o de [Junior et al. 2019].

Assim, foi dado enfoque à representação de um projeto em sua totalidade, minimizando perdas semânticas e de representatividade, de modo a não limitar o escopo de análises e inferências futuras. Além disso, a respeito da representação da avaliação sobre cada projeto, foi dedicada especial atenção para a possibilidade de acompanhamento da evolução do aluno ao longo da construção de um projeto, de modo a permitir o acompanhamento do caminho de aprendizado para cada conceito específico avaliado. Este fator permite que sejam realizadas análises sobre quais conceitos são mais facilmente desenvolvidos por cada aluno e em quais estes apresentam maior dificuldade.

Devido a importância do suporte a inferências e raciocínios por este modelo de dados, optou-se pela representação do mesmo através de um conjunto de ontologias. A divisão desta representação em ontologias distintas tem como principal motivação facilitar o reuso do vocabulário e sua futura expansão.

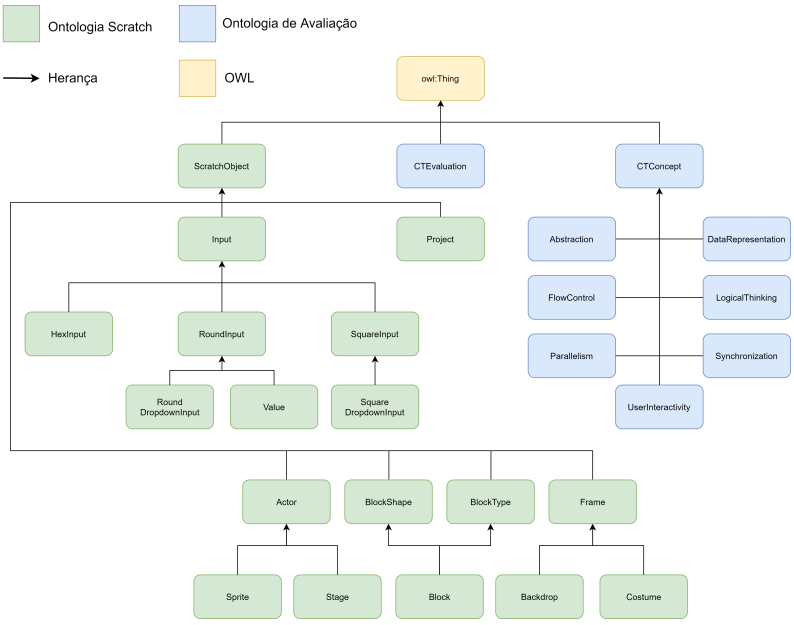

Figura 1. Hierarquia de classes simplificada do OntoScratch

Devido a extensão da representação de conhecimento desenvolvida – esta possui 190 classes, 45 propriedades de objeto e 9 propriedades de dados – é inviável a apresentação de sua hierarquia de classes completa em uma única figura. Assim, é possível visualizar uma versão resumida da representação desenvolvida na Figura 1, simplificada para as classes mais específicas, enquanto uma versão completa está disponível no repositório [1].

[1] https://github.com/Naraujo13/OntoScratch

Apresentada em verde na figura, a primeira ontologia é responsável por representar o universo de um projeto Scratch, com todos seus objetos e relações, sendo abordada em detalhes na Subseção 2.1. Por sua vez, a segunda ontologia, é responsável pela representação da avaliação de PC sobre um projeto, suas regras e conceitos avaliados, sendo detalhada na Subseção 2.2, e é apresentada em azul na hierarquia.

2.1. Ontologia do Universo Scratch

Apesar de sua simplicidade de uso, a representação de um projeto Scratch é extensa e complexa. A ferramenta apresenta 119 blocos únicos em sua terceira versão [scr 2020b], sem considerar as diversas extensões disponíveis, que adicionam novos blocos. Estes podem ser combinados em diversos *scripts*, com diferentes sequências e parametrizações, aumentando ainda mais esta gama de possibilidades. A documentação do Scratch está disponível em uma wiki[2] e esta foi utilizada como dicionário de dado para a criação desta ontologia. Esta opção se deu com o intuito de evitar conflitos entre classificações, buscando tornar mais simples a transição entre vocabulário da ferramenta e da ontologia.

2.1.1. Hierarquia Base

Assim, criou-se uma classe base, chamada de *ScratchObject*, para representar todo e qualquer objeto pertencente ao universo virtual do Scratch. Subclasse de *Thing*, ela serve como base para todas as outras classes a serem desenvolvidas nesta ontologia.

Esta classe é especializada em outras 5 classes principais: (1) Projeto, representando o projeto em si, com seu nome, url, versão do Scratch e autor; (2) Ator, representando aqueles que atuam no projeto, ou seja, que possuem os blocos para alterar seu comportamento; (3) *Frames*, representando as diversas formas visuais que um personagem pode possuir, como diferentes fantasias ou diferentes visuais de palco para fases de um jogo; (4) *Input*, representando as entradas de dados dos blocos, podendo estas ser em diferentes formatos – entrada livre, escolha de opção em lista e etc - e com diferentes tipos de dados – booleanos, numéricos ou textuais; e (5) Bloco, sendo este a representação da estrutura visual de programação do Scratch, que permite seu encaixe em outros blocos como uma peça de quebra-cabeça, permitindo a criação de sequências de ações e comportamentos.

2.1.2. Bloco

Dentre estas estruturas, os blocos destacam-se como a principal estrutura de programação no Scratch. É possível distingui-los em duas hierarquias diferentes: em relação ao seu formato e em relação a sua categoria. A primeira é responsável por definir as relações de um bloco com outros, definindo se o mesmo pode ter sucessores, antecessores ou blocos internos. Já em relação a suas categorias, estas são definidas pela funcionalidade esperada.

O formato dos blocos Scratch determina o encaixe que os mesmos podem ter com outros blocos. Ou seja, a forma do bloco determina as relações que este bloco pode ter com outros, como blocos sucessores e antecessores ou até mesmo a parametrização. Ao

[2]https://en.scratch-wiki.info/

todo, existem seis formatos distintos de blocos: *Hat Blocks*, *Stack Blocks*, *Boolean Blocks*, *Reporter Blocks*, *C Blocks* e *Cap Blocks*. Deste modo, o formato do bloco foi representado como a classe *BlockShape*, uma subclasse de *ScratchObject*. Ademais, foram criadas subclasses para cada um dos formatos específicos, com cada subclasse representando blocos que possuem aquele formato e, consequentemente, as restrições de relacionamento com outros blocos pertinentes ao mesmo.

A categoria de um bloco determina sua funcionalidade, ou seja, o comportamento esperado. Assim, blocos com efeitos similares são agrupados na mesma categoria e isto é visível pela sua tonalidade. No total, existem 10 (dez) categorias distintas de blocos: *Motion Blocks*, *Looks Blocks*, *Sound Blocks*, *Event Blocks*, *Control Blocks*, *Sensing Blocks*, *Operators Blocks*, *Variables Blocks*, *List Blocks* e *My Blocks*. Assim, a classe *BlockType* foi criada como uma subclasse de *ScratchObject*, representando uma categoria de bloco. Também foram criadas subclasses de *BlockType* para cada uma das categorias específicas, cada uma representando um conjunto de blocos que compartilha determinadas características como explicitado acima.

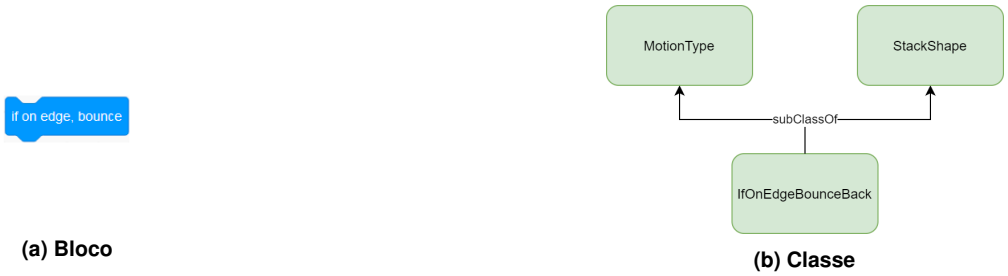

Figura 2. Exemplo de classe para um bloco

Assim, a classe que representa um bloco Scratch sempre será uma subclasse de um formato e de uma categoria, de modo a definir sua função e seu relacionamento com outros blocos. Um exemplo disto é a definição do bloco *IfOnEdgeBounceBack* (Figura 2), utilizado para fazer *Sprites* rebaterem ao chegarem nas bordas da tela, sendo esta uma subclasse de uma categoria de bloco, neste caso *MotionType*, e de um formato de bloco, neste caso *StackShape*. Esta classe representa este bloco, porém, no Scratch, o mesmo bloco pode ser utilizado como diversas instâncias diferentes. Assim, cada instância de um bloco é representada como um indivíduo do mesmo, diferenciada pelo seu identificador único e com diferentes relações de objeto representando suas diferentes entradas de dados e diferentes conexões. Este formato de representação se estende para cada um dos 119 (cento e dezenove) blocos existentes no Scratch.

A relação entre blocos pode ocorrer: (1) como uma relação de sequência, com um bloco sendo encaixado como sucessor ou antecessor de outro; e, (2) como uma relação de aninhamento, com um bloco sendo encaixado como interno a outro, como no caso de estruturas de repetição ou condicionais, constituindo uma relação de pai e filho. Estas relações foram mapeadas como hierarquia de propriedades de objeto para a ontologia.

2.2. Ontologia de Avaliação do Pensamento Computacional

Para realização do acompanhamento da evolução de um aluno em relação ao seu conhecimento de PC, é necessária a representação das avaliações realizadas sobre seus projetos

ao longo do tempo. Desta forma, será possível verificar sua evolução ao longo de uma ou mais atividades.

Devido a grande divergência entre os conceitos e habilidades relevantes para mensuração do PC, além da grande variedade de metodologias para execução desta avaliação, foi dada atenção especial à modelagem da ontologia de avaliação, para permitir uma fácil expansão, além de fácil inferência. Com estes fatores em mente, optou-se por uma adaptação da abordagem utilizada em [Araújo et al. 2019], realizando a separação da representação dos conceitos de PC da execução da avaliação em si. Este formato permite uma fácil adição de novos termos, como novos conceitos ou habilidades a serem mensurados, permitindo a expansão e reuso do vocabulário.

Para isto, optou-se pela representação dos conceitos de PC como uma classe específica ao invés de apenas propriedades de dados da avaliação. Este formato permite que sejam atreladas anotações que busquem auxiliar na compreensão do conceito representado. Isto facilita para o usuário o entendimento de quais fatores são considerados em cada conceito, permitindo compreender melhor sua representação. Além disso, essa representação permite a fácil expansão do vocabulário para a inclusão de novos conceitos.

Já para o acompanhamento da avaliação de um aluno, optou-se pela representação da mesma como uma classe específica na ontologia, chamada de *CTEvaluation*, onde cada instância representa a execução de uma avaliação de habilidades do PC demonstradas em um projeto Scratch. Para permitir a representação temporal destas avaliações, foi definida uma propriedade de dados para esta classe chamada *evaluatedAt*, representando o momento de avaliação.

3. Avaliação e Resultados

Como forma de avaliar a representação de conhecimento, propôs-se o seguinte cenário: "Um professor deseja avaliar e comparar as habilidades de PC demonstradas por dois de seus alunos ao desenvolverem um projeto em Scratch utilizando a metodologia da ferramenta Dr. Scratch". Através deste cenário busca-se avaliar a capacidade de representatividade da OntoScratch, além de avaliar também seu suporte para a realização de inferências. Assim, a avaliação consiste dos seguintes passos:

- Representar ambos os projetos na ontologia Scratch.
- Avaliar os projetos através de mecanismos de inferência sobre a ontologia, representando as avaliações executadas na ontologia de Avaliação de PC.
- Avaliar os projetos diretamente na ferramenta Dr. Scratch, utilizando a versão beta disponibilizada em sua plataforma web[3].
- Comparar a avaliação realizada na OntoScratch com a avaliação realizada diretamente na ferramenta do Dr. Scratch.

Para a execução deste cenário, foram utilizadas consultas SPARQL[4] processadas pelo *framework* Snap-SPARQL[Horridge and Musen 2015] diretamente na ferramenta *Protégé*[5] e utilizando o *reasoner* Pellet [Sirin et al. 2007] em sua versão incremental. As consultas para realização da inferência seguem o padrão de regras apresentado pela

[3]http://www.drscratch.org/
[4]https://www.w3.org/TR/sparql11-overview/
[5]https://protege.stanford.edu/

ferramenta Dr. Scratch. Assim, são avaliados sete conceitos relacionados ao PC, atribuindo uma pontuação de 0 a 3 para cada, onde o somatório das sete notas representa a pontuação final do projeto. Todas as consultas podem ser conferidas em maiores detalhes no repositório das ontologias [6]. As consultas utilizaram-se da detecção de padrões para identificar o nível de cada habilidade e criar a relação correspondente de dados, com todas elas seguindo a estrutura apresentada na Listagem 1.

Listagem 1. Estrutura de consultas de inferência

```
CONSTRUCT { ?evaluated_concept ctc:ConceptScore ?score }
WHERE {
  OPTIONAL{
    # Padrao de grafo para pontuacao de nivel 1
  }
  # Rule 2
  OPTIONAL{
    # Padrao de grafo para pontuacao de nivel 2
  }
  # Rule 3
  OPTIONAL {
    # Padrao de grafo para pontuacao de nivel 3
  }
  BIND(
    if(BOUND(?level_3_block), 3,
      if(BOUND(?level_2_block), 2,
        if(BOUND(?level_1_block), 1, 0
  ))) AS ?score)
}
```

Para construção deste cenário de teste, optou-se pela escolha de dois projetos dentre os exemplos disponibilizados na documentação oficial do Scratch chamados *Pong Starter*[7] e *Maze Starter*[8]. Ambos foram escolhidos por apresentar uma boa variação de blocos e diferentes padrões de programação, porém não serem grandes a ponto de tornar difícil sua visualização e compreensão.

No primeiro projeto, *Pong Starter*, o jogador interage movendo uma barreira com o objetivo de rebater a bola, sem deixá-la tocar na parte inferior da tela. Assim, dois atores possuem blocos de modificação de comportamento: a barreira, realizando a movimentação do jogador através do mouse, e a bola, realizando a sua movimentação e a detecção das colisões.

A seguir, executou-se a avaliação na ferramenta Dr. Scratch diretamente através de sua interface web. O resultado foi uma avaliação do projeto com uma pontuação de 9 pontos, dentre os 21 possíveis, classificando a proficiência demonstrada no projeto como "Em desenvolvimento". Para a execução da avaliação na representação ontológica, foram executadas as consultas SPARQL desenvolvidas obtendo-se a avaliação apresentada na Figura 3. A avaliação realizada através da ontologia apresentou as mesmas pontuações individuais para todas as habilidades em comparação com a realizada diretamente no Dr. Scratch e, consequentemente, também apresentou a mesma pontuação total.

Já no segundo projeto de testes, *Maze Starter*, o jogador deve mexer a bola de forma a desviar dos obstáculos e chegar no objetivo. Desta forma, dois atores possuem

[6]https://github.com/Naraujo13/OntoScratch
[7]https://scratch.mit.edu/projects/10128515/
[8]https://scratch.mit.edu/projects/10128431/

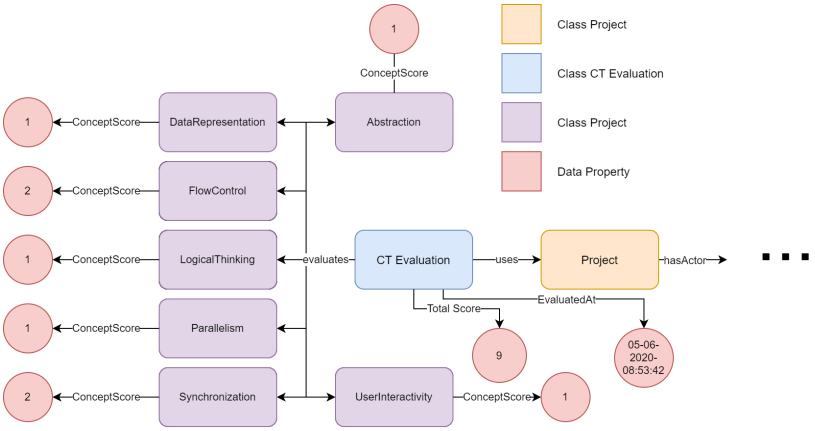

Figura 3. Avaliação do Projeto 1 na OntoScratch

scripts, sendo eles o jogador, encarregado da movimentação e colisão com obstáculos, e o objetivo, encarregado de verificar a condição de vitória.

Apesar de ambos os projetos possuírem o mesmo número de atores, suas estruturas de programação são amplamente distintas. O primeiro projeto possui apenas três fluxos de execução em um mesmo ator, com estes fluxos sendo mais longos e complexos. Enquanto isso, no projeto em questão, a abordagem utilizada privilegia um maior número de fluxos paralelos, com seis deles apenas no ator do jogador principal, sendo estes, em geral, mais curtos e simples.

A seguir, foi executada a avaliação do projeto na ferramenta Dr.Scratch. Foi atribuída uma pontuação total para o projeto de 8 pontos dentre os 21 possíveis, categorizando-o também como "Em desenvolvimento". A seguir, o conjunto de consultas SPARQL foi executado sobre a representação do projeto na ontologia. Esta execução resultou na representação apresentada na Figura 4. Foram instanciados nodos com as mesmas pontuações da avaliação realizada diretamente na ferramenta para cada habilidade e, consequentemente, também para pontuação total.

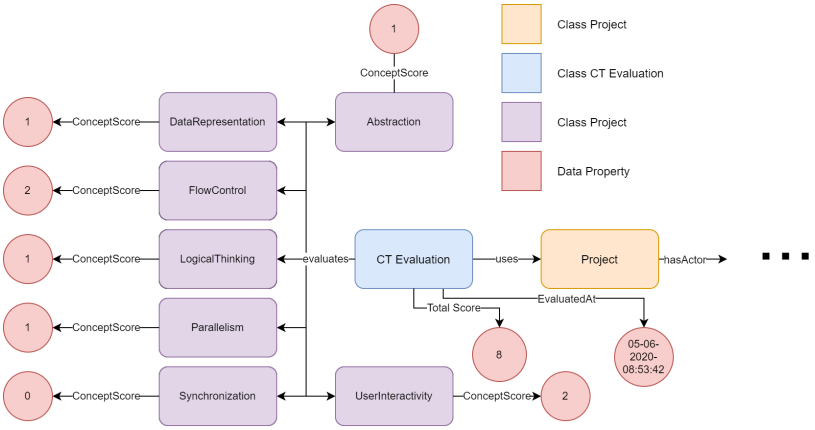

Figura 4. Avaliação do Projeto 2 na OntoScratch

Através deste teste foi possível observar que a inferência das habilidades de PC realizada através da representação de conhecimento apresentou resultados equivalentes a avaliação realizada na ferramenta especializada Dr. Scratch. Consequentemente, a

representação de conhecimento OntoScratch mostrou-se eficaz para a representação e visualização dos projetos e de suas avaliações, provando-se efetiva para o suporte a realização de inferências.

4. Considerações Finais

A expansão de iniciativas para introduzir conceitos relacionados a computação no ensino à crianças vêm ocorrendo em diversos países e, nestas iniciativas, destaca-se o uso de ferramentas de programação visual, como o Scratch. Neste cenário, surge o desafio de armazenar os dados coletados nestas atividades, para que estes sejam trabalhados e gerem evidências do resultado deste trabalho, porém a ausência de uma representação de conhecimento capaz de representar estes dados de forma a prover suporte para a análise do progresso do aluno ou da realização de outras inferências surge como obstáculo.

A representação de conhecimento baseada em ontologias OntoScratch, permite armazenar de maneira estruturada o percurso de alunos durante a elaboração de projetos Scratch visando analisar seu processo de aprendizagem de habilidades relacionadas ao PC. A avaliação conduzida e apresentada na Seção 3 demonstrou sua viabilidade para representação destes conhecimentos, além de sua eficiência para realização de inferências, sendo capaz de reproduzir fielmente, através de consultas SPARQL, a avaliação de PC da ferramenta Dr. Scratch.

Como uma nova representação de conhecimento para o acompanhamento do ensino de PC através de projetos Scratch, o presente trabalho apresenta diversas possibilidades de trabalhos futuros. Inicialmente, visualiza-se a conexão desta representação com coletores de dados que capturam todas as ações realizadas pelo usuário no projeto, como o apresentado em [Junior et al. 2019]. Através da sequência temporal de ações, ser possível reconstruir o estado do projeto em qualquer momento de tempo desejado e representá-lo com a OntoScratch.

Assim, através desta conexão, seria possível representar o projeto periodicamente, recriando o seu estado a cada intervalo de tempo desejado. Este fator permitiria a realização de inferências temporais mais facilmente, como, por exemplo, verificar a evolução de cada uma das habilidades de PC para auxiliar o educador na tomada de decisão e planejamento dos conteúdos. Um exemplo de ferramenta possibilitada é a utilização de mecanismos de recomendação para auxiliar o educador a identificar as habilidades a serem reforçadas em uma turma com base no desempenho e resultado dos alunos.

Referências

(2020a). Scratch. `https://scratch.mit.edu/statistics/`. Acessado em 11/07/2020.

(2020b). Scratch wiki. `https://en.scratch-wiki.info/`. Acessado em 07/07/2020.

Araújo, C., Lima, L. V., and Henriques, P. R. (2019). An ontology based approach to teach computational thinking. In *2019 International Symposium on Computers in Education (SIIE)*, pages 1–6. IEEE.

Blikstein, P. (2008). O pensamento computacional e a reinvenção do computador na educação. *Education & Courses*.

BLINKSTEIN, P. (2019). O pensamento computacional e a reinvenção do computador na educação. 2008.

Brennan, K. and Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, volume 1, page 25.

Campos, F., Soster, T., and Blikstein, P. (2019). Sorry, i was in teacher mode today: Pivotal tensions and contradictory discourses in real-world implementations of school makerspaces. In *Proceedings of FabLearn 2019*, pages 96–103.

De França, R. S. and do Amaral, H. J. C. (2013). Proposta metodológica de ensino e avaliaçao para o desenvolvimento do pensamento computacional com o uso do scratch. In *Anais do Workshop de Informática na Escola*, volume 1, page 179.

Dousay, T. A. (2017). Defining and differentiating the makerspace. *Educational Technology*, pages 69–74.

Horridge, M. and Musen, M. (2015). Snap-sparql: A java framework for working with sparql and owl. In *International Experiences and Directions Workshop on OWL*, pages 154–165. Springer.

Junior, J. L. N. V., Primo, T. T., Pernas, A. M., and Júnior, D. A. M. (2019). A framework for collecting and analyzing interactions in scratch projects. In *XIV Latin American Conference on Learning Technologies (LACLO)*, pages 50–54. IEEE.

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.

Oluk, A. and Korkmaz, Ö. (2016). Comparing students' scratch skills with their computational thinking skills in terms of different variables. *Online Submission*, 8(11):1–7.

Seiter, L. and Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the ninth annual international ACM conference on International computing education research*, pages 59–66.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53.

Troiano, G. M., Snodgrass, S., Argımak, E., Robles, G., Smith, G., Cassidy, M., Tucker-Raymond, E., Puttick, G., and Harteveld, C. (2019). Is my game ok dr. scratch? exploring programming and computational thinking development via metrics in student-designed serious games for stem. In *Proceedings of the 18th ACM International Conference on Interaction Design and Children*, pages 208–219.

Wilson, A., Hainey, T., and Connolly, T. (2012). Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. In *European Conference on Games Based Learning*, page 549. Academic Conferences InternationalLimited.