Ferramenta para edição de uma linguagem visual para desenvolvimento de habilidades do Pensamento Computacional

MATHEUS IANZER HERTZOG[1]; SIMONE ANDRÉ DA COSTA CAVALHEIRO[2]; LUCIANA FOSS[3]

[1]*Universidade Federal de Pelotas – mihertzog@inf.ufpel.edu.br*
[2]*Universidade Federal de Pelotas – simone.costa@inf.ufpel.edu.br*
[3]*Universidade Federal de Pelotas – lfoss@inf.ufpel.edu.br*

1. INTRODUÇÃO

Pensamento Computacional, como Jeannette Wing, a criadora do termo, coloca, se trata de uma abordagem para resolução de problemas, concepção de sistemas e entendimento do comportamento humano que se baseia em conceitos fundamentais da computação (WING, 2006). Portanto, se compreendermos a ciência da computação como a automação de abstrações (ULLMAN, 1992), o processo de abstração torna-se o mais importante, e de mais alto nível, processo de pensamento. Dessa forma, a abstração é usada para definir padrões, generalizando e parametrizando, e assim, capturando as propriedades essenciais de um conjunto de objetos. Por exemplo, um algoritmo é uma abstração de um processo que recebe entradas, executa passos, e gera uma saída de acordo com o objetivo desejado. Nesse sentido, abstração nos fornece a capacidade de construir sistemas cada vez maiores e poderosos para lida com a complexidade, sendo os bits (0's e 1's) o caso base.

Sendo uma habilidade fundamental e poderosa, engana-se aqueles que acreditam que é somente valiosa para os cientistas da computação, em realidade, todos podem se beneficiar dela. Tanto a leitura, a escrita, o tratamento de problemas matemáticas, quanto as habilidades analíticas de crianças podem se valer do Pensamento Computacional como demonstrado por CHRISTIAN; BRACKMANN (2017). Essa noção de que a abordagem computacional pudesse mudar a maneira como percebemos problemas em diferentes áreas não é nova, em 1962, preconizou-se que a automatização de processos mudaria a maneira com que profissionais de diferentes áreas enxergariam seus respectivos trabalhos (PERLIS, 1962). E na década de 1980, a ideia de que o pensamento procedural e o computador poderiam afetar a maneira que crianças pensam e aprendem foi popularizada (PAPERT, 1980); Seymour Papert advoga que o educando poderia construir o seu próprio conhecimento por intermédio de uma ferramenta, como o computador, naquilo que denominou construcionismo, embasando-se no construtivismo de Jean Piaget. Portanto, longe da ideia de "pensar como um computador" o Pensamento Computacional é uma distinta capacidade criativa, crítica e estratégica, de usar os fundamentos e os recursos da computação acompanhado de nossa inteligência com a finalidade de resolver problemas de diferentes tipos e de uma maneira individual ou colaborativa.

O Pensamento Computacional se fundamenta em quatro diferentes pilares que orientam o processo de solução de um determinado problema qualquer: a **decomposição**, como proposto pela segunda regra do método descartiano (DESCARTES, 2009), trata de quebrar um problema complexo em partes menores e mais simples de resolver; o **reconhecimento de padrões**, este se dá pela identificação de similaridades em diferentes disposições de um mesmo problema de maneira que uma solução satisfaça a todas; a **abstração**, que trata-se da distinção, pela análise, dos elementos relevantes dos que podem ser ignorados; e os **algo-**

ritmos, que abrangem todos os pilares anteriores, constituem o processo de criação das regras para a resolução de problemas. (BRACKMANN, 2017)

Ainda assim, tudo isso levanta um novo problema: como inserir o Pensamento Computacional no ensino básico de forma efetiva e quais técnicas utilizar? Tentando responder a essa pergunta, um projeto de pesquisa, intitulado "**Proposta Metodológica para a Introdução do Raciocínio Computacional no Ensino Fundamental**", foi proposto. Dentro do escopo deste projeto, foi proposta uma linguagem visual que segue a abordagem funcional, a qual deve ser usada para a introdução do conceito de algoritmos. Esta linguagem permite a abstração de procedimentos, focando nos dados de entrada e nos resultados obtidos. Além disso, a incorporação dos conceitos de decomposição e generalização (reconhecimento de padrões) é favorecida devido à natureza funcional da linguagem. Para que o uso da linguagem possa de fato ser efetiva, torna-se necessária uma ferramenta que permita a edição e simulação dos algoritmos construídos. Sendo assim, este trabalho visa apresentar o projeto e uma implementação de uma ferramenta que atende essa demanda.

A escolha de se utilizar do paradigma funcional advém deste possuir um alto grau de abstração e, portanto, maior foco nas partes de **abstração** e **decomposição** de procedimentos. Uma aplicação real no Ensino básico seria, por exemplo, o exercício de estruturar, em Linguagem Visual, operações elementares como divisão, multiplicação, até mesmo equações.

2. METODOLOGIA

Para atingir o objetivo deste trabalho, primeiro houve a necessidade de entender a linguagem a ser considerada. A Figura 1 mostra um exemplo de procedimento funcional que poderia ser definido nessa nova linguagem. A "**Acao YYY**" representa uma função que recebe três argumentos **v1**, **v2**, **v3**, de tipos, respectivamente, Numérico, String, e Booleano. As ações "**acao1**", "**acao2**", e "**acao3**" são funções também definidas pelo usuário de maneira que a execução se dará na ordem definida pelo fluxograma; primeiramente, "**acao1**" e "**acao2**" são executadas em paralelo com os seus respectivos argumentos, que são os valores recebidos através dos parâmetros da "**Acao YYY**", então "**acao3**" será executada com os resultados das computações de "**acao1**" e "**acao2**", gerando o resultado final do tipo String.

A ferramenta aqui proposta é constituída por dois componentes: um que provê uma interface para edição de código, e outro que realiza a compilação/interpretação de código. O código gerado em um ambiente que suporte à linguagem (editor) é traduzido pelo compilador/interpretador para que seja possível simulá-lo. Para a criação de código, cada função deve ser construída definindo-se as suas entradas, as subações que a compõe, e o seu resultado, onde as subações são descritas em um alto nível de abstração (informando-se apenas as suas entradas e saída). E, caso haja necessidade de detalhar alguma subação, esta pode ser selecionada e descrita da mesma forma que a ação principal.

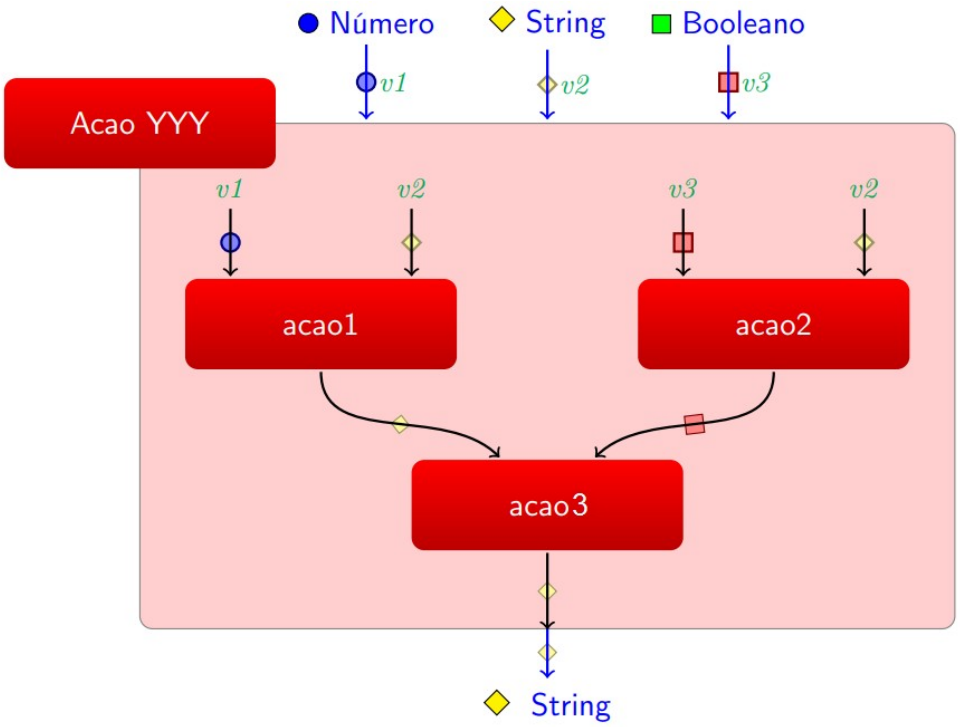

Figura 1. Sintaxe para procedimentos funcionais.

Os limites impostos pelas finalidades da ferramenta, limites estes como a utilização da ferramente em escolas públicas e, portanto, em máquinas de baixo desempenho, demandaram uma ferramenta leve e portável, dentre as estudadas (JavaFX/Swing, WxPython, Qt), o framework Qt se mostrou a opção mais viável. Após a definição do projeto da ferramenta, iniciou-se sua implementação. Até o presente momento, realizou-se a implementação da interface gráfica do editor de código fazendo uso do framework Qt em C++.

3. RESULTADOS E DISCUSSÃO

Os resultados aqui obtidos tratam de fornecer um ambiente para geração de código (gráfico) na linguagem supracitada, e como descrito anteriormente, cada função precisa ser construída. A Figura 2 representa a interface já implementada em execução, onde "**foo**", é a função que está sendo definida, a qual possui três argumentos dos tipos: String, Imagem, Booleano, e a saída de tipo Numérico. Nela estão presentes três subações, "**foo2**", "**foo3**", e "**fun**", com suas devidas entradas e saídas. A ferramenta apresenta um menu na esquerda com botões que permitem selecionar o tipo de subação que será inserido na função, e um botão de texto que permite inserir comentários na tela. O menu superior trata da formatação do texto que se insere na tela. Um clique direito em uma subação mostrará um menu de subação que apresenta as funcionalidades de mudar o nome da subação, inserir novas entradas, ou deletá-la. Um clique direito em "**foo**" permite inserir novas entradas na mesma.

Entre funcionalidades desejadas estão, um duplo clique em uma subação que abrirá uma nova aba dentro da mesma interface que permitirá a construção da mesma, uma maneira interativa (com o uso do mouse) de conectar as saídas das subações, e nomear as entradas e saídas de uma maneira a manter a ordem visual.

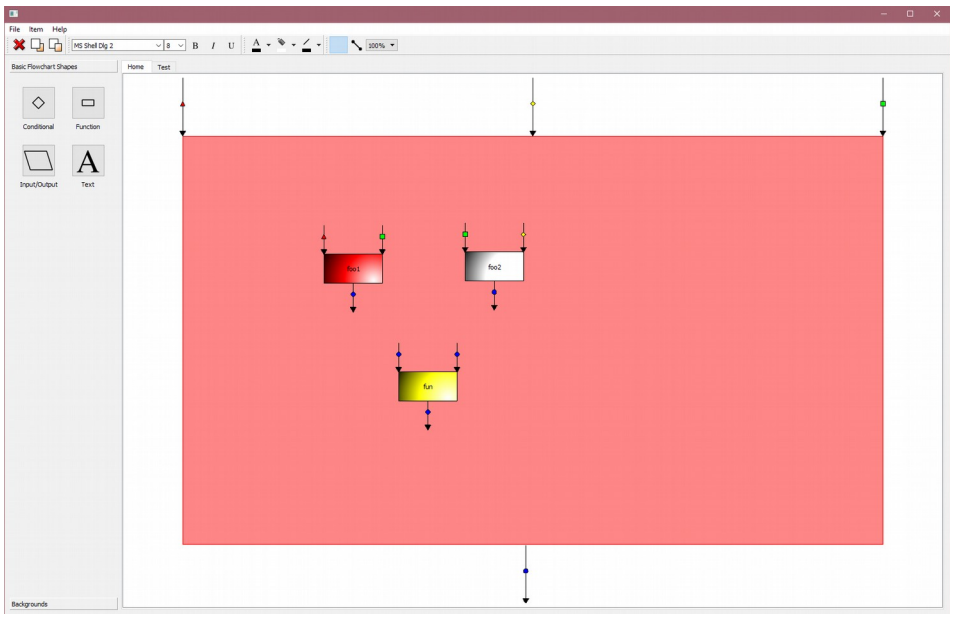

Figura 2. Interface da ferramenta de edição.

4. CONCLUSÕES

Neste artigo são apresentados o projeto e parte da implementação de uma ferramenta que dará suporte a construção de algoritmos usando uma linguagem visual. Além disso, o Pensamento Computacional é apresentado e sua relevância, principalmente no contexto do ensino, é usada como motivação para o projeto da ferramenta aqui descrita. Como trabalhos futuros são esperados: a finalização da interface gráfica, a geração da árvore sintática que represente cada código definido no ambiente aqui descrito, bem como a implementação do componente de compilação/interpretação.

5. REFERÊNCIAS BIBLIOGRÁFICAS

AHO, Al; ULLMAN, Jeff. **Foundations of Computer Science**. NY, USA. Computer Science Press, Inc. 1992.

WING, J. M. Computational thinking. **Communications of the ACM.** New York, v.49, n.3, p 33–35. 2006.

DESCARTES, René. Discurso do método. Brasil: Escala, 2009.

PERLIS, Alan. **Computers and the World of the Future**, Massachussetts: The MIT Press, 1964.

PAPERT, Seymour. **Mindstorms: Children, Computers, and Powerful Ideas**. Basic Books, Inc., New York, USA, 1980.

BRACKMANN, C.P,; **Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica**. 2017. Dissertação (Doutorado em Ciência da Computação), Universidade Federal do Rio Grande do Sul.

CAVALHEIRO, Simone; FOSS, Luciana; RIBEIRO, Leila. **Entendendo o Pensamento Computacional.** 2017. Disponível em https://arxiv.org/abs/1707.00338. Acesso em agosto de 2018.