Ferramentas de Visualização de Programas na Compreensão de Funções de Alta-Ordem

Marcos Rogério Martins
martinsmrogerio@gmail.com
Universidade Estadual de Feira de Santana, Feira de Santana, Brasil

Rodrigo Duran
rodrigo.duran@ifms.edu.br
Instituto Federal de Mato Grosso do Sul, Nova Andradina, Brasil

PALAVRAS-CHAVE

Visualização de Programas, Funções de Alta-Ordem, Compreensão de Programas

Uma questão central em Educação em Computação (EC) é a investigação de como se dá o aprendizado de programação, uma vez que, historicamente, a maioria dos cursos de graduação em Ciência da Computação foram desenhados para formar desenvolvedores de software [3]. Além das necessidades do desenvolvimento de software para estudantes dos cursos de Computação, e de tecnologia de forma geral, existe uma crescente demanda da programação em outras áreas do conhecimento. Muitas das atividades realizadas por não-desenvolvedores é o que convencionou-se chamar de Centralidade de Dados [4], um novo paradigma de computação coloca um foco muito maior em aplicações que visem atender demandas organizacionais de armazenamento, recuperação, movimentação e processamento de dados. Esse manuseio massivo de dados requer profissionais com conhecimento para implementar algoritmos que automatizem processos de análise de dados, auxiliando na tomada de decisão e com experiência em outras áreas do conhecimento para interpretar tais dados e suas análises.

Assim, é fundamental prover ferramentas computacionais eficientes e eficazes que possam realizar tarefas de forma simples. As Funções de Alta-Ordem (FAO) são descritas como funções que operam em outras funções, seja tomando-as como argumentos ou retornando-as como resultado, além de permitirem composição de funções, ou seja, possuir pequenas funções que compõem funcionalidades que realizam tarefas mais extensas e complexas. Dentre as inúmeras FAO, podemos destacar três funções amplamente utilizadas por desenvolvedores e não-desenvolvedores: *Map, Filter e Reduce*, utilizadas para transformar, filtrar e agregar dados, respectivamente. Estes métodos são úteis para reduzir a complexidade e, muitas vezes, tornar o código mais legível [1].

Estudos recentes mostram que estudantes ainda tem dificuldades na compreensão de funções como o reduce [5] e na composição de funções [6]. Tais dificuldades sugerem que o uso da visualização e simulação de programas pode melhorar a compreensão das funcionalidades e aplicações que utilizam-se de FAO pois exploram abordagens e métodos de representação de código e processamento gráfico estático e dinâmico de elementos deste código. A apresentação destes elementos de código está relacionado principalmente à análise e desenvolvimento de programas, visando melhorar a compreensão do programa inerentemente invisível e intangível e apresentar concretamente a semântica de uma determinada linguagem de programação de forma simplificada e com objetivos educacionais, ou seja, a sua *Notional Machine* [2, 7]. O principal desafio é encontrar traduções eficazes dos vários aspectos do código para representações gráficas usando metáforas visuais. [8].

O objetivo desse trabalho é propor potenciais designs para sistemas de visualização e simulação de FAO adaptando o sistema de visualização e simulação JSVEE [1], o qual já possui suporte à linguagem Python, para incorporar a visualização de FAO. Inicialmente será feito um estudo acerca dos guias de design existentes para sistemas de visualização e sistemas multimídia para explorar quais características deverão ser incorporadas ao sistema para representar dinamicamente a simulação e visualização de FAO. Após o desenvolvimento dos protótipos, será conduzida uma avaliação empírica a qual será empregada em um cenário de pesquisa quase-experimental utilizando um design de pesquisa com um grupo controle e grupos experimento (um grupo para cada protótipo de design). Todos os grupos serão expostos ao mesmo material de ensino preparado para essa pesquisa e todos serão avaliados pela compreensão do mesmo grupo de questões. A seleção dos indivíduos será feita por conveniência entre os estudantes do curso de Computação da Universidade Estadual de Feira de Santana que cursaram a disciplina introdutória de programação em Python.

REFERÊNCIAS

[1] Rodrigo Duran, Juha Sorva, and Sofia Leite. 2018. Towards an Analysis of Program Complexity From a Cognitive Perspective. In *Proceedings of the 2018 ACM Conference on International Computing Education Research* (Espoo, Finland) *(ICER '18)*. Association for Computing Machinery, New York, NY, USA, 21–30.

[2] Rodrigo Duran, Juha Sorva, and Otto Seppälä. 2021. Rules of Program Behavior. *ACM Trans. Comput. Educ.* 21, 4, Article 33, 37 pages.

[3] Mark Guzdial. 2019. *Computing for Other Disciplines.* Cambridge University Press, 584–605.

[4] Shriram Krishnamurthi and Kathi Fisler. 2020. Data-centricity: a challenge and opportunity for computing education. *Commun. ACM* 63, 8, 24–26.

[5] Shriram Krishnamurthi and Kathi Fisler. 2021. Developing Behavioral Concepts of Higher-Order Functions. In *Proceedings of the 17th ACM Conference on International Computing Education Research* (Virtual Event, USA) *(ICER 2021)*. Association for Computing Machinery, New York, NY, USA, 306–318.

[6] Elijah Rivera, Shriram Krishnamurthi, and Robert Goldstone. 2022. Plan Composition Using Higher-Order Functions. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1* (Lugano and Virtual Event, Switzerland) *(ICER '22)*. Association for Computing Machinery, New York, NY, USA, 84–104.

[7] Juha Sorva. 2013. Notional Machines and Introductory Programming Education. *ACM Trans. Comput. Educ.* 13, 2, Article 8, 31 pages.

[1] https://github.com/Aalto-LeTech/jsvee

EduComp'23, Abril 24-29, 2023, Recife, Pernambuco, Brasil (On-line)

[8] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A Review of Generic Program Visualization Systems for Introductory Programming Education. *ACM Trans. Comput. Educ.* 13, 4, Article 15, 64 pages.