

Behavior Trees and State Machines in Robotics Applications (Summary)

Razan Ghzouli ¹, Thorsten Berger ^{2,1}, Einar Broch Johnsen ³, Andrzej Wasowski ⁴, and Swaib Dragule ¹

Abstract: We summarize our article published in IEEE Transactions on Software Engineering [Gh23].

Keywords: behavior trees, state machines, robotics, behavior specification, empirical study



Robotics systems are complex and software-intensive cyber-physical systems performing increasingly complex tasks in our everyday life. The software controlling robots is typically realized with specific software architectures, allowing to structure the software that controls the different parts of a robot, ranging from controlling low-level behavior (typically called skills) to controlling the higher-level behaviors (typically called missions). This allows modularizing decisions of different levels of complexity, such as simple reactive control or the more deliberative planning. Given the complexity of decisions, behavior is often specified in models expressed in domain-specific languages (DSLs) [WB23], many of which have been developed for modeling the behavior of robotic systems [Dr21a, Dr21b].

State machines are arguably the most common notation to specify behavior in software systems, specifically in robotic systems. Many state-machine DSLs are available for practitioners: either as external DSLs [Dr21a] to be used with dedicated tooling (modeling environment), or as internal DSLs, such as the libraries SMACH and FlexBE [Gh23], to be used directly within the source code of the system. However, while being the lingua franca of behavior specification, state machines have shortcomings, requiring reasoning about system behavior in terms of states and state transitions, as well as modularity being an issue.

Recently, behavior trees have become popular as an alternative. Stemming from the games domain, behavior trees offer an extensible tree-based representation of robotic missions. They allow reasoning about behavior in terms of actions and conditions, as well as they promise better modularity and foster reuse [CÖ18]. However, while various implementations are available, little is known about their relationship to traditional languages, such as state machines, as well as their scope and their use in practice.

We present a study of the core language concepts offered by behavior trees, their implementation in real DSLs (libraries), and their use in real-world robotics applications (mined from GitHub) implemented with the Robotics Operating System (ROS). We investigate five

1 University of Zambia, Zambia, and Chalmers | University of Gothenburg, Sweden, mukelabai.mukelabai@unza.zm,  <https://orcid.org/0000-0002-3868-4319>

1 Chalmers | University of Gothenburg, Gothenburg, Sweden, razan@chalmers.se,  <https://orcid.org/0000-0002-5428-8113>; wasowski@itu.dk,  <https://orcid.org/0000-0002-9916-400X>

2 Ruhr University Bochum, Bochum, Germany,

DSLs—three behavior-trees DSLs and two state machine DSLs—in depth, comparing the concepts of behavior trees and state machines (an earlier version of our article also includes a comparison to UML activity diagrams [Gh20]).

All the studied DSLs show similarities in language design, offering constructs for frequent control-flow patterns, while the range of support differs between behavior-tree and state machine DSLs. Openness is a common property—the DSLs do not enforce a fixed model, expect the robotics projects to extend the DSLs by-need. For their execution, they all follow the models-at-runtime paradigm. We observed similar usage patterns at model structure and at code reuse in the behavior-tree and state-machine models within the mined open-source projects. Interestingly, the external DSLs with their modeling environments foster the use of built-in language constructs, such as Decorators in behavior trees and Concurrency containers in state machines, while the others leaked these constructs to the code instead of the model. Our analysis of the DSLs' use in robotics projects shows that the use of behavior trees is rising rapidly. Their use was similar from a structural perspective. Developers kept the models fairly simple; shallow and moderately sized models were observed in the majority of the sampled projects both for behavior trees and state machines. Finally, we observed three model-reuse patterns.

Data Availability

An online appendix with our can be found online [On23].

Bibliography

- [CÖ18] Colledanchise, Michele; Ögren, Petter: Behavior Trees in Robotics and AI: An Introduction. CRC Press, 2018.
- [Dr21a] Dragule, Swaib; Berger, Thorsten; Menghi, Claudio; Pelliccione, Patrizio: A Survey on the Design Space of End-User Oriented Languages for Specifying Robotic Missions. *International Journal of Software and Systems Modeling*, 20(4):1123–1158, 2021.
- [Dr21b] Dragule, Swaib; Garcia, Sergio; Berger, Thorsten; Pelliccione, Patrizio: Languages for Specifying Missions of Robotic Applications. In (Cavalcanti, Ana; ad Rob Hierons, Brijesh Dongol; Timmis, Jon; Woodcock, Jim, eds): *Software Engineering for Robotics*. Springer, 2021.
- [Gh20] Ghzouli, Razan; Berger, Thorsten; Johnsen, Einar Broch; Dragule, Swaib; Wasowski, Andrzej: Behavior Trees in Action: A Study of Robotics Applications. In: *SLE*. 2020.
- [Gh23] Ghzouli, Razan; Berger, Thorsten; Johnsen, Einar Broch; Wasowski, Andrzej; Dragule, Swaib: Behavior Trees and State Machines in Robotics Applications. *IEEE Transactions on Software Engineering*, 49(9):4243–4267, 2023.
- [On23] Online Appendix. <https://bitbucket.org/easelab/behaviortrees>, 2023.
- [WB23] Wasowski, Andrzej; Berger, Thorsten: *Domain-specific Languages: Effective Modeling, Automation, and Reuse*. Springer, 2023.