International Journal of Internet, Broadcasting and Communication Vol.15 No.4 134-141 (2023)
http://dx.doi.org/10.7236/IJIBC.2023.15.4.134

IJIBC 23-4-15

Robot Software Framework using Robot Operation System(ROS2) based on Behavior Tree

Sangho Lee[1], Hyejin Chang[2], Seulgi Jeon[3], Janghwan Kim[4],e R. Young Chul Kim[5*]

[1]M.S student, Software Engineering Laboratory, Hongik University,
[1,2,3]Director, Dept. of Applied Technology Research, Rastech,
[4]PhD. Candidate, Dept. of Computer Engineering, Mokpo National University
[5]Professor, Dept. Software and Communications Engineering, Hongik University,*
E-mail: { [1]project, [2]hjchang, [3]jseulgi}@rastech.co.kr,
[4]janghwan.kim@g.hongik.ac.kr, [5]bob@ hongik.ac.kr*

Abstract

As robotic technology expands into various fields, robots need to execute some complicated tasks in diverse environments. However, the previous robotic software solutions were limited to independent systems. We can not adapt to diverse functionalities and environments. This makes it hard to provide rapid and effective services and leads to costs and losses in the development process. To overcome these problems, we propose a robot software framework with behavior trees based on ROS2. This framework simplifies complex robot behaviors through behavior trees and makes it easy to modify, extend, and reuse robot behaviors. Furthermore, ROS2 standardizes connections between software modules, enhances the robot's flexibility, and enables independent development and testing of software. Our framework aims to provide a foundation for high-quality robot service provision by supporting the modularity, reusability, independent development, and testing required by intelligent robots that need to provide services in various environments.

Keywords: *Robot Software Framework, ROS2, Behavior Tree*

1. Introduction

Modern robot technology is expanding from traditional manufacturing to various fields such as home, medical care, education, and agriculture. As the tasks and environments performed by robots are diversified, the functions that robots must perform are also becoming more complex. These tasks are operated by complex interworking and designing various software modules such as sensor fusion algorithm, dynamic object detection algorithm, and robot control algorithm. Therefore, robots should be able to dynamically combine software modules for different environments, adjust their behavior as needed, or learn new behavior.

However, most robot software solutions are currently organized in an independent system depending on the

Manuscript Received: September. 14, 2023 / Revised: September. 19, 2023 / Accepted: September. 23, 2023
Corresponding Author: bob@hongik.ac.kr
Tel: +82-44-860-2477, Fax: +82-44-865-0460
Professor, Department of Software and Communications Engineering, Hongik University, Korea

robot, making it difficult to replace or connect different software modules, making it difficult for one robot to use a different system of software modules or for multiple robots to share and use one software module. This reduces the robot's adaptability to various functions and environments, preventing rapid and effective service provision. In addition, in the existing traditional robot software system, software modularity and reusability depend greatly on the developer's ability. This requires advanced and skilled developers to modify, expand and replace software modules, resulting in a complex and expensive development process.

To solve this problem, we propose a robot software framework using a ROS2 (Robot operation system)-based Behavior Tree. The behavior tree is a tool that allows the robot's complex behavior to be decomposed into several simple behaviors and structured, making it easy to modify, expand, and reuse the robot's behavior. In addition, ROS2 standardizes connections between modules to increase robot flexibility and enable independent development and testing of software. The proposed framework seeks to provide a foundation for providing high-quality robot services by supporting the modularization, reuse, independent development, and testing required for intelligent robots that need to provide services in various environments.

2. Related Works

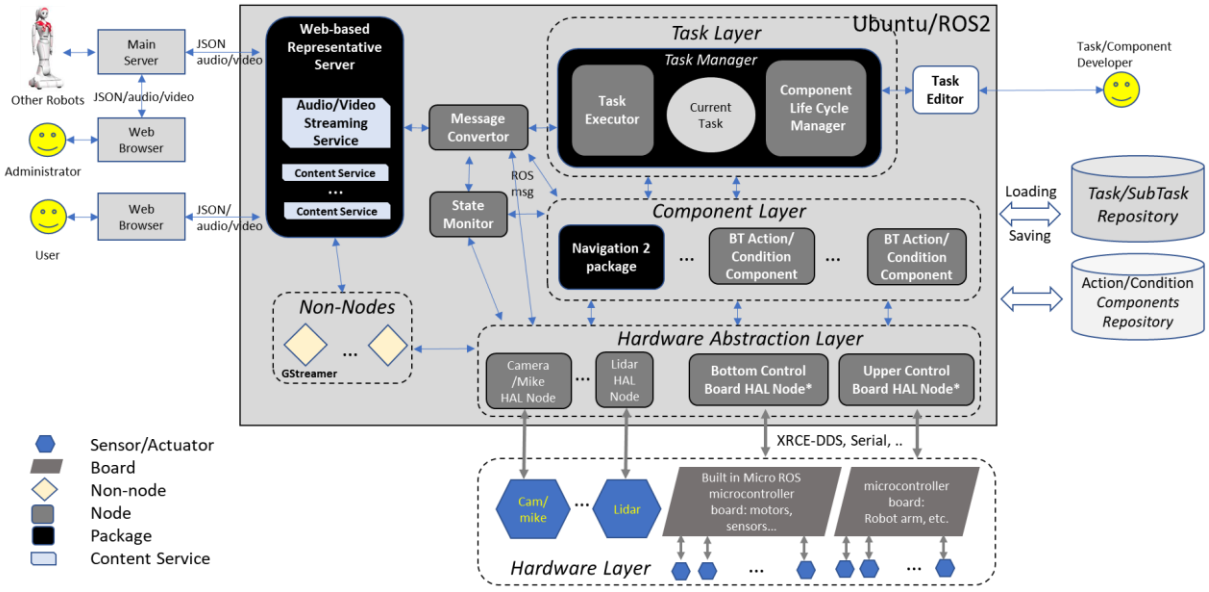

Figure 1. Software Framework using a ROS2-based Behavior Tree

A behavior tree is a theory originating in the field of game development that classifies complex behaviors into smaller sub-behaviors. Behavior trees greatly improve the flexibility and reuse of software and are very useful for designing and implementing tasks for complex systems, which have been demonstrated by several studies.

Behavior tree-related research has been actively studied in addition to the structure of the basic behavior tree, such as how to optimize the behavior tree through genetic algorithms. With the recent acceleration of artificial intelligence research, studies on the convergence of reinforcement learning and behavior trees are being conducted [1-5]. Kartasev [1] proposes a behavior tree with deep Q-Network (DQN) and Proximal Policy Optimization (PPO). Big problems were classified hierarchically through the action tree, hierarchically

classified sub-problems were solved through reinforcement learning, and their performance was verified. As the robotics field develops, the behavior tree is recognized as a very useful tool for robot behavior establishment and management, and various studies are underway [6-12]. Michele and Ogren [6] mention a comprehensive introduction to the behavior tree and how to apply it in the robotics and artificial intelligence fields. Alejandro, et al [7] mention an integrated behavior tree framework for robot control, which can integrate multiple robot control architectures and easily add new control architectures [7]. Jeong [8] proposes an efficient framework using the behavior tree, focusing on the robot's autonomous driving system [8]. The behavior tree-based autonomous driving framework has shown that it is possible to control multiple robots by communicating through a Data Distribution Service (DDS) between multiple robots to exchange data.

3. Behavior tree based on Robot Software Framework Implementation

This paper proposes a method to ensure the common use, reusability, portability, and scalability of robotic software. That is, we propose a software system that can reuse the components of the robot software module like Lego blocks and assemble them to define the robot's task. In addition, in this text, we build a whole-electric robot software development platform that ranges from robot software development to simulation, testing, and distribution.

3.1 Development and Operational Environment

It was developed in the ROS2 environment using Ubuntu 22.04 as the basic operating system. This is to utilize the flexibility and scalability provided by ROS2, and various libraries and tools linked to ROS2. The programming languages used are C++ 14 or later and Python 3.8 or later, both of which are well compatible with ROS2. Taking advantage of each language, C++ was used for low-level robot control that required high computational speed, while Python was used to define high-level robot behavior.

3.2 Behavior Tree-based Robotic Software Framework

We mention our robot software framework consisting of the Behavior Tree, Presentive Server, Task Layer, Component Layer, and Hardware Abstraction Layer, and the overall structure is shown in Figure 1.

3.2.1 Behavior Tree

The behavior tree is a key element of the robot development framework that defines and manages the tasks of robots. Each node of the behavior tree is represented by an XML file, and the behavior tree consists of a terminal node representing condition check or action performance and a control node representing repetition, sequential performance, etc. Creating XML files based on previously developed nodes and sub-action trees can define tasks for robots, and Figure 2 shows an example of a ROS2-based behavior tree structure.

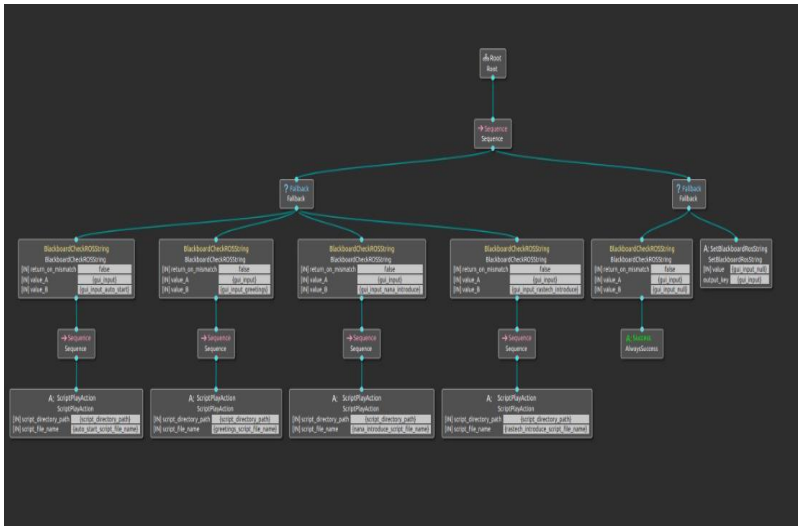

Figure 2. A ROS2-based behavior tree structure

3.2.2 Robot Representative Server

The robot representative server is the representative server of the robot and is responsible for all services outside the robot and outside the robot. It is implemented as a web server that supports WebSocket, and can be accessed by robots using ID/password and SSL/TLS from the outside. The conceptual diagram of the representative server is shown in Figure 3.

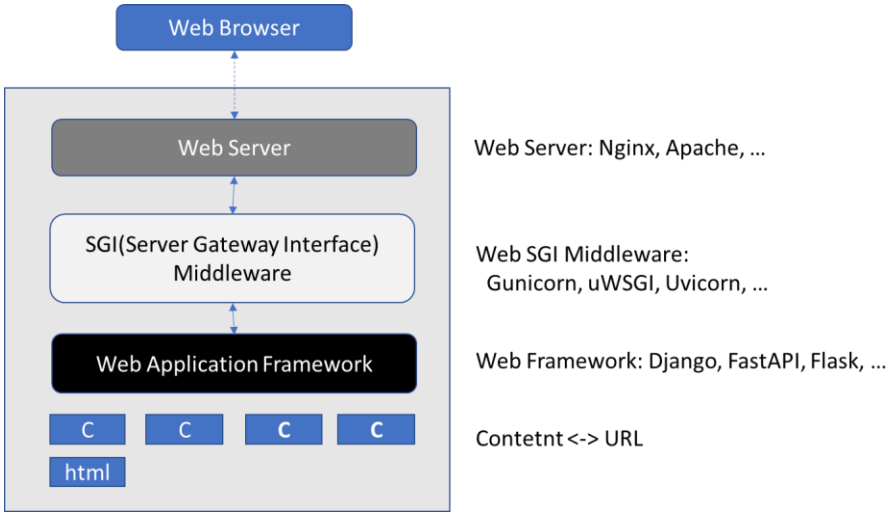

Figure 3. Representative Server Conceptual Diagram

3.2.3 Task Layer

The Task Layer is a software module that controls the robot at the top level. In the Task Layer, the Task Executor performs an action tree edited by the Task Editor using the Component Life Cycle Manager according to commands received by the Task Manager through the Message Handler. For example, functions such as emergency suspension that are not expressed as tasks are handled by calling components directly without going through Task Manager. Figure 4 shows the conceptual diagram of the Task Editor and defines the task in XML form.

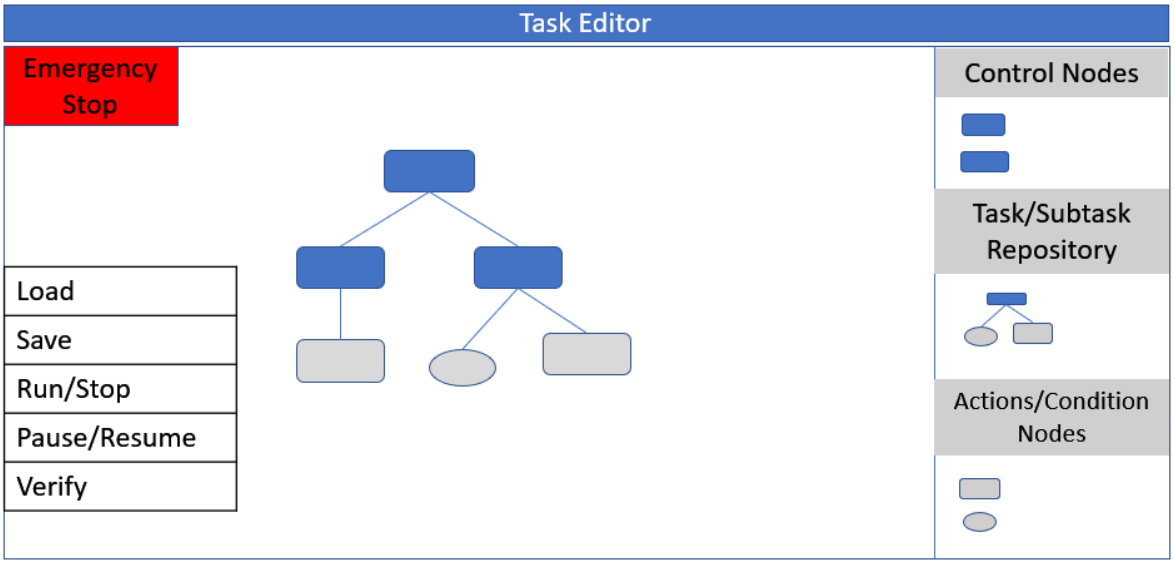

Figure 4. Task Editor Conceptual Diagram

3.2.4 Component Layer

The component layer is a layer in which leaf nodes such as conditions and actions that make up the action tree are developed, stored, and performed. The component is implemented as a C++ or Python component that supports the dynamic loading of ROS2 and has high reuse.

3.2.5 Hardware Abstraction Layer

Hardware Abstraction Layer (HAL) is a layer that allows hardware elements such as ultrasonic sensors, drive motors, Lidar sensors, cameras, and microphones to be treated like ROS2 nodes. This layer provides HAL nodes for devices directly connected to the PC and microcontroller boards connected to multiple sensors.

4. A Case Study for Humanoid Robot

Figure 5. Humanoid Robot

The robot software framework using the proposed ROS2-based behavior tree is applied to a humanoid-type guidance service robot developed by Las Tech Co., Ltd. to evaluate the framework performance. Figure 5 shows the shape of a humanoid robot and plays a role in providing commentary and guidance services to visitors in public relations and exhibition halls. The main task of a robot is defined as follows.

- Drive autonomously at the specified waypoint, move, and stop
- Moving the charging station to charge the battery
- 14 Free Body and Arm Motion Control
- Voice output synchronized with motion
- Speech recognition for interacting with people

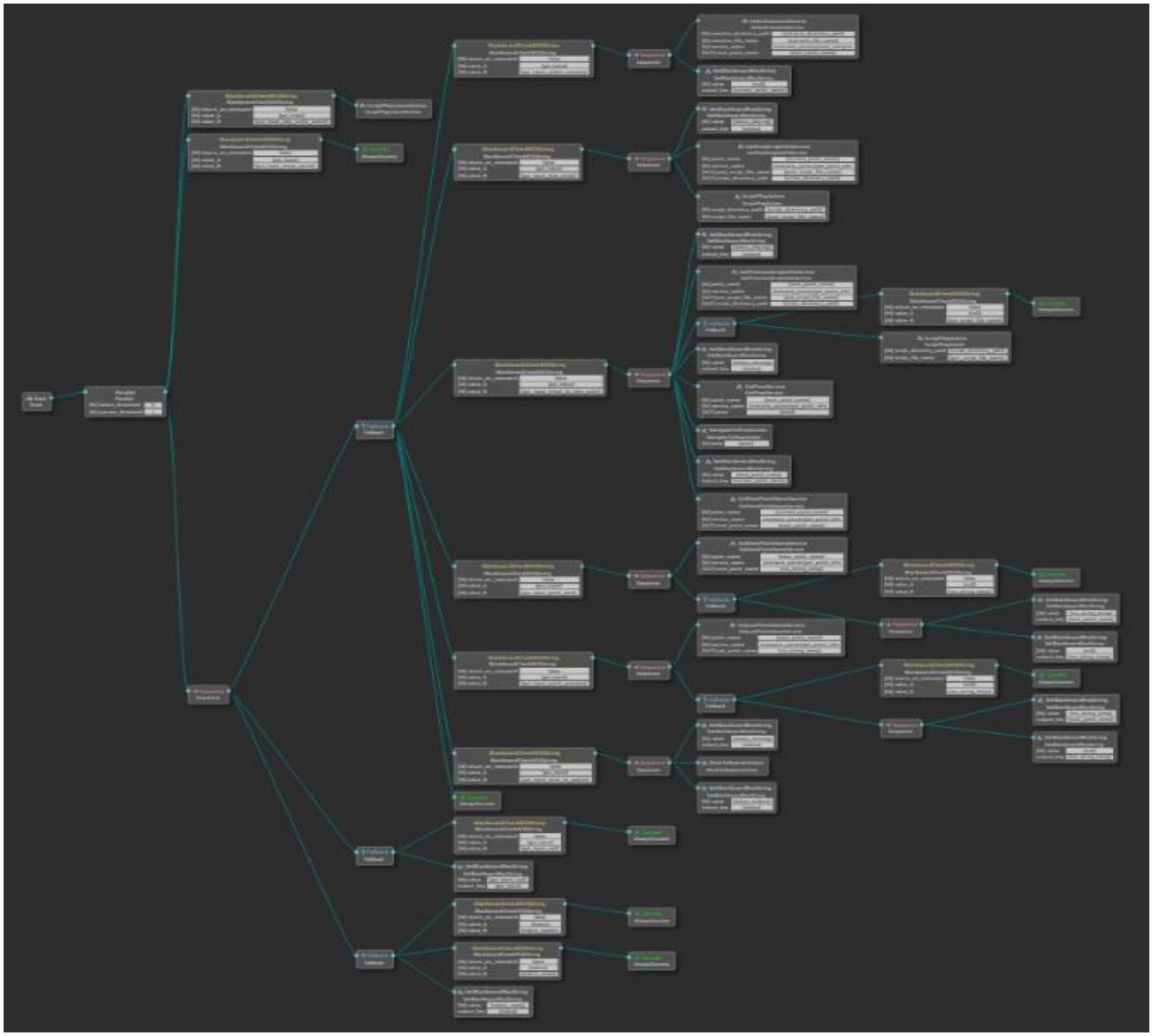

Figure 6. Behavior Tree Structure of Humanoid Service Robot

Sensors and various hardware elements essential for implementing the function of the robot are abstracted through the HAL node to configure an interface to acquire sensor data in real time. Component layer is a unit in which robots can be independently tested and developed, and it can operate effectively in complex

environments by developing and dynamically loading leaf nodes such as conditions and actions that make up the action tree.

For example, the Navigation 2 component node developed for indoor and outdoor autonomous driving was reused to implement a function to autonomously drive to a target point, and it was expanded to avoid static/dynamic obstacles and to move around prohibited areas. As such, the main task of the robot specified above was developed as an independent node for the component layer, and the integrated system according to the state of the robot was organized as a behavior tree as shown in Figure 6 through the task editor.

5. Conclusions

In this paper, we introduce a sophisticated software framework that integrates behavior trees within the context of a ROS2 environment. This framework has been meticulously crafted to support the implementation of discrete functional components, each designed to address the diverse and specific needs of robotic systems. The unique aspect of this approach lies in the dynamic loading of these developed components, which are strategically used to define and execute robotic tasks utilizing the behavior tree methodology. The practicality and effectiveness of this framework have been validated through a series of empirical experiments, notably in the application to humanoid guidance service robots.

Our proposed framework demonstrates how robots work effectively in complex environments in the real world based on the reuse, scalability, and efficiency of behavior trees. It is achieved through the framework's emphasis on the reuse, scalability, and efficiency inherent in behavior trees. We anticipate that our approach will significantly streamline the complexity and enhance the flexibility inherent in the development of robotic systems. This is expected to address and alleviate the financial and logistical challenges often encountered in robot development, while simultaneously elevating the overall quality of the software produced. Our future research aims to further improve the proposed framework to improve reliability and robustness in diverse and unpredictable environments.

References

[1] Kartasev, Mart, “Integrating Reinforcement Learning into Behavior Trees by Hierarchical Composition”, MS Thesis, KTH R. Inst. Technol., Stockholm, 2019.

[2] YanChang, Fu, Long, Qin, Quanjun, Yin, “A Reinforcement Learning Behavior Tree Framework for game AI”, International Conference on Economics, Social Science, Arts, Education and Management Engineering August 2016. DOI: 10.2991/essaeme-16.2016.120

[3] Hao, Hu, et al. “Self-Adaptive Traffic Control Model with Behavior Trees and Reinforcement Learning for AGV in Industry 4.0”, IEEE Transactions on Industrial Informatics December 2021. DOI: 10.1109/TII.2021.3059676

[4] Lei, Li, et al. “Mixed Deep Reinforcement Learning-behavior Tree for Intelligent Agents Design”, ICAART February 2021. DOI: 10.5220/0010316901130124

[5] Kartasev Mart, Justin Saler, Petter Orgen “Improving the Performance of Backward Chained Behavior Trees that use Reinforcement Learning”, Arxiv Dec 2021. DOI: 10.48550/arXiv.2112.13744

[6] Colledanchise Michele, Petter Ogren “Behavior Trees in Robotics and AI: An Introduction” CRC Press 2018.

[7] Marzinotto Alejandro, et al. "Towards a unified behavior trees framework for robot control", IEEE ICRA MAY 2014. DOI: 10.1109/ICRA.2014.6907656

[8] Jeong Seungwoo, et al. "Behavior Tree-Based Task Planning for Multiple Mobile Robots using a Data Distribution Service", IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) 2022.

DOI: 10.1109/AIM52237.2022.9863364

[9] Colledanchise Michele, Lorenzo Natale. "On the implementation of behavior trees in robotics", IEEE Robotics and Automation Letters July 2021. DOI: 10.1109/LRA.2021.3087442

[10] Ghzouli Razan, et al. "Behavior trees in action: a study of robotics applications", Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering November 2020.

DOI: 10.1145/3426425.3426942

[11] Colledanchise Michele. “Behavior trees in robotics”, Diss. KTH Royal Institute of Technology, 2017.

[12] Paxton Chris, et al. "CoSTAR: Instructing collaborative robots with behavior trees and vision", IEEE ICRA May 2017. DOI: 10.1109/ICRA.2017.7989070