# A robotics software framework to change execution based on Behavior Trees at runtime

MAHIRO IGUCHI[†1] HARUMI WATANABE[†1]

***Abstract***: This article contributes to easily dealing with robots' behavior on the model at runtime. Modern robot systems have been rapidly more complex and supported more varied services. Many of them must change their behavior at runtime. To solve this problem, we propose a framework of Behavior Tree (BT) to change execution at runtime. The difficulty of this framework is the delay time at changing BT. In this article, we focus on the delay time problem.

***Keywords***: ROS2, Behavior Trees, Robotics software framework

## 1. Introduction

Recently, robot systems have been more complex and supported larger and varied services. Many of them must change their behavior at runtime because of their learning or maintenance. To support building such systems, we focus on changing models at runtime, called runtime change behavior, and choose Behavior Trees (BT) as the modeling language. The BT helps us understand such complex robot systems, and the runtime change behavior on the BT contributes to easily dealing with behavior on the model at runtime.

However, ordinary BTs do not contain functions to change the built behavior. Also, they don't support the function to change the robot's behavior at runtime. In this article, we introduce a framework of BT with the runtime change behavior. Our framework uses BehaviorTree.CPP [2] and robotics software platform ROS2[3]. In the execution process, we can change the behaviors without stopping the ROS2 program. The framework supports building such systems that change behavior at runtime. The difficulty of this framework is the delay time at changing BT. Therefore, in this article, we engage in the problem of the delay time problem.

The remainder of this article is as follows. Chapter 2 introduces Behavior Trees. Chapter 3 introduces the framework. Chapter 4 applies to our framework for the ROS2 program. Finally, chapter 5 concludes this article.

## 2. Behavior Trees

This chapter explains BT. The BT is a way of constructing the behavior of autonomous robots and non-player characters as tree models[1].

### 2.1 Structure of BT

Fig. 1 shows an example of BT. The root on the top node sends a signal called a Tick to the child's nodes. If there are branches, send a Tick from left to right. The nodes on the bottom are called execution nodes. When each execution node receives a Tick, it executes the program on its node and sends back a Tick to the top node. Nodes between the top and bottom are called control flow nodes. At a receiving Tick, each control flow node distinguishes whether it sends the Tick to the next child node. Otherwise, it returns the Tick to the parent node. The callbacks of Ticks consist of RUNNING, SUCCESS, or FAILURE. One of the control flow nodes is the Sequence node. When the Sequence node receive SUCCESS on a Tick, it sends a Tick to the next child node.

### 2.2 Behavior of BT

The behavior of Fig. 1 is as follows. First, the left node receives Ticks and checks anyone there. If anyone is there, the Sequence nodes send the next child's nodes, resulting in execution actions with smiles and waving hands. If no one is there, the Sequence nodes do not send the next child's nodes and return Ticks to the Root node.

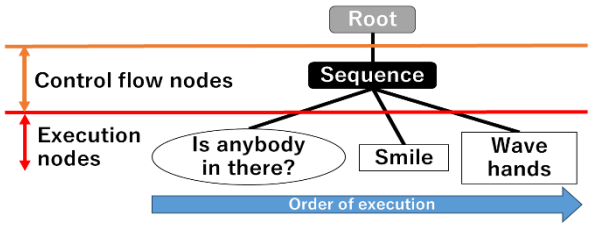


Fig. 1 Example of BT

## 3. Change of behavior on BT

In this chapter, we present a robotics software framework to change software on BT at runtime. In our framework, users use BehaviorTree.CPP[2] and robotics software platform ROS2[3]. Fig. 2 shows a process to change behavior based on BT at runtime. In the BehaviorTree.CPP, tree models of BT are written in XML file format. The sequence of the execution is decided based on the XML file. The robot can change behavior at runtime by modifying the XML file and reloading it.

The following describes the normal process. It means that the process doesn't happen with the changing behavior at runtime. In the behavior of the blue line in Fig. 2, first, the execution thread of BT acquires "update_mutex." Then, the thread releases the mutex after the program finishes the execution.

The following describes the runtime change behavior shown in the red line of Fig. 2. First, when users change the XML file, a method "Behavior Trees rebuild call" is called, then tried to acquire "update_mutex." Second, it acquires the mutex when the

†1 Tokai University the Graduate School of Information and Telecommunication Engineering

"Behavior Trees execution" thread releases the mutex at the end of the program. When the" Behavior Trees execution" thread doesn't acquire the mutex, it rebuilds the program based on the XML file. After that, the" Behavior Trees rebuild call" method releases the mutex. The "Behavior Trees execution" thread returns to the execution normal process.

### 3.1 Issue

This section clarifies the issue to the delay time of changing BT. The above process needs to wait until the end of the program to change execution. It means that the delay time happens to wait for the changing execution. The article calls this problem the delay time of the change in the BT problem. This article solves the problem by changing the contents of the methods corresponding to nodes without reconstructing BT.

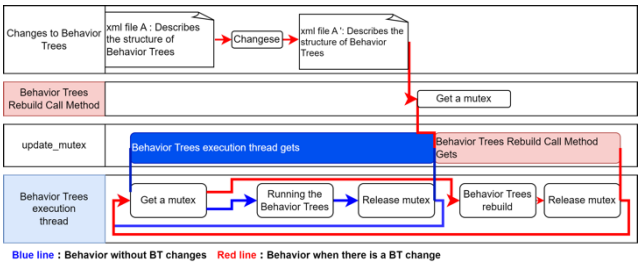


Fig. 2 Process of changing BT

### 3.2 Runtime change behavior considering the delay time

We propose a method to change behavior before a system built using the BT has finished processing using ROS2's pluginlib[4]. To deal with the runtime change behavior, we must change the behavior before the node completes its execution. For the issue, we use a pluginlib of ROS2[4]. ROS2's pluginlib can dynamically plug or unplug other package classes. Fig. 3 shows an example of pluginlib to change the main method. The figure contains a main method and three packages. The main method has two "printAction" methods: one is marked in red dashed lines, and another is marked in blue dashed lines. The main method executes the "printAction" method. Thus, the behavior of "printAction" method before plugging the package is different from "printAction" method after plugging. Each package contains the behavior of a node. Thus, if the package is switched to another package, the node's behavior is changed. We call such "printAction" methods to hot methods. Fig. 4 shows a process of changing node behavior by using pluginlib.

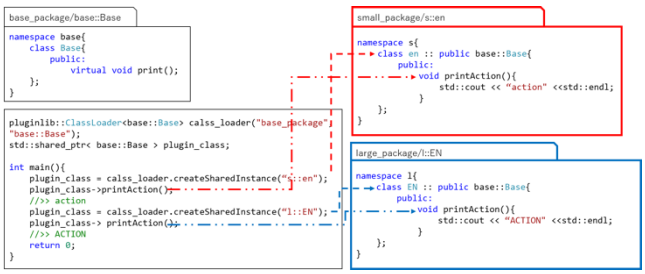


Fig. 3 Example of change the main class by using pluginlib

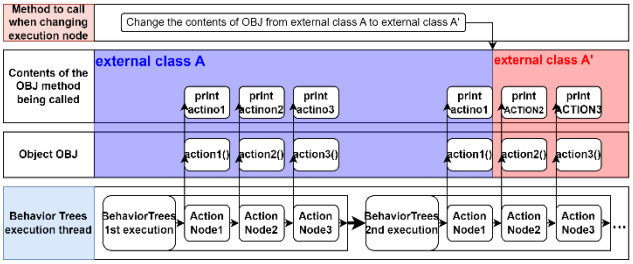


Fig. 4 Process of changing hot methods on BT

## 4. Application

This chapter applies our framework to ROS2 program. Fig. 4 shows a result of the application. To confirm the runtime change behavior, we prepared two types of nodes: one node shows sentences with small letters; Another shows sentences with uppercase letters. The outline of the behavior is shown in the left of Fig. 5. The result is show in the right of Fig. 5. As shown in this figure, we confirmed the behavior change from small letters to uppercase letters without waiting to finish the program.

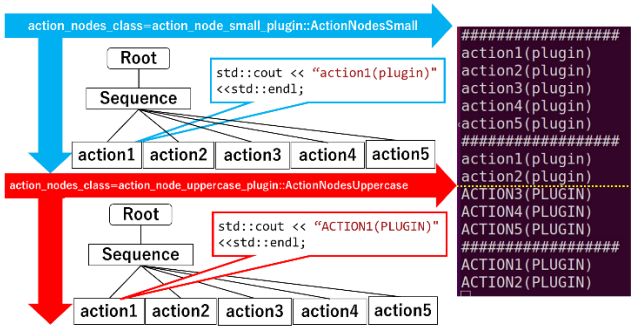


Fig. 5 The result that runs to the proposed framework

## 5. Conclusion

In this article, we proposed a framework of BT with the runtime change behavior. This framework solved the problem of the delay time at changing behavior. To confirm the behavior, we applied the framework to the small example. That was changed the sentences from small letters to uppercase letters.

In future work, we will apply the framework to more practical systems and evaluate the performance of the runtime change behavior.

## Reference

[1] Colledanchise, Michele, and Petter Ögren. "Behavior Trees in Robotics and AI: An Introduction." arXiv preprint arXiv:1709.00084 (2017).

[2] " Behavior Trees Library in C++. Batteries included.". https://github.com/BehaviorTree/BehaviorTree.CPP, (accessed 2023-07-14).

[3] Steve Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, William Woodall. "Robot Operating System 2: Design, architecture, and uses in the wild." Science Robotics 7.66 (2022): eabm6074.

[4] "Creating and using plugins (C++) − ROS 2 Documentation: Foxy documentation" . https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Pluginlib.html, (accessed 2023-07-14).