CSSR for

Culturally Sensitive Social Robotics for Africa

6.1 Use Case Implementation

Due date: **31/03/2025**
Submission Date: **18/04/2025**
Revision Date: **01/06/2025**

Start date of project: **01/07/2023** Duration: **36 months**

Lead organisation for this deliverable: **Carnegie Mellon University Africa (for Wits)**

Responsible Person: **D. Vernon** Revision: **1.2**

Project funded by the African Engineering and Technology Network (Afretec) Inclusive Digital Transformation Research Grant Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including Afretec Administration)	
RE	Restricted to a group specified by the consortium (including Afretec Administration)	
CO	Confidential, only for members of the consortium (including Afretec Administration)	

Executive Summary

Deliverable D6.1 documents the outcome of Task 6.1, i.e., the implementation of the two use cases defined in Work Package WP2, using the outcomes of WP1 - WP5, i.e., the cultural knowledge, the scenario specification, and the integrated robot's sensory and interaction capabilities. Specifically, the use cases are captured using the robot mission specification methodology documented in Deliverable D5.4.2 Robot Mission Language, i.e., using behavior trees [1, 2]. The resultant behavior tree provides the input to the `behaviorController` ROS node documented in Deliverable D5.4.3. Running the `cssr_system` ROS package against the behavior tree robot mission specification provides a demonstration of the complete working system for the corresponding use case.

In the CSSR4Africa work plan, this deliverable is assigned to the University of the Witswatersrand. However, the material in this report was developed and written by Carnegie Mellon University Africa. This was necessary because, due to extensive delays in the delivery of the Pepper robot to the University of the Witswatersrand, little or no progress had been made on three key inputs for Task 6.1 and Deliverable D6.1, viz. Deliverables D5.4.1 Cultural Knowledge Ontology & Culture Knowledge Base, D5.4.2 Robot Mission Language, and D5.4.3 Robot Mission Interpreter, all of which were assigned to the University of the Witswatersrand. Consequently, Carnegie Mellon University Africa took joint responsibility for these four deliverables (among others, specificially D5.5.2.1, D5.5.4, D6.2). Since this involved a significant amout of additional, unplanned effort, only one use case, the laboratory tour, has been implemented to date, leaving the second use case, the receptionist, to be implemented later.

Contents

1 Introduction

This deliverable provides a detailed walkthrough of the behaviour tree implementation the Lab Tour robot mission specification, as defined by the D2.1 Use Case Scenario. Section 2.1 introduces the generated XML output, which is the representation of robot mission specification behavior tree, by describing the organization of the XML file. Section 2.3 discusses the incorporation and structuring of subtrees, which modularize complex behaviors within the mission specification. For each subtree, corresponding XML definition and graphical illustration are provided. Additionally, Section 2.4 explains mission nodes, which represent specific tasks or actions executed by the robot.

2 Behaviour Tree Implementation of Use Case Scenario 1: Laboratory Tour

This section details the implementation of a robot mission specification based on the operational guidelines provided in the "Lab Tour" scenario (see Deliverable D2.1 User Scenario Specification). While the mission is grounded in the scenario described in D2.1, there are minor deviations from the original step-by-step narrative to accommodate implementation constraints and practical considerations. The mission is structured as a behavior tree, which serves as the control architecture for coordinating the robot's actions. To design and visualize this behavior tree, we employ the Groot2 IDE, using its intuitive drag and drop mechanisms as outline above. The mission interpreter, implemented using the BehaviorTree.CPP library (described in Deliverable D5.4.3 Robot Mission Interpreter), executes the behavior tree within a ROS-based framework, forming an integral part of the CSSR4Africa software system. For an overview of the system architecture, please refer to D3.1 System Architecture.

2.1 XML Output

As explained above, the mission is designed using the Groot2 IDE. The Groot2 IDE allows for the design of behavior trees using a graphical interface. The mission specification represented as a behavior tree is then exported as an XML file, which is used by the mission interpreter to execute the mission. The XML representation of the behavior tree designed for the "Lab Tour" scenario is explained in the following sections, along with the graphical illustrations for each segment of the behavior tree.

The robot mission specification file is has three sections, the file root, the subtree specifications, and the mission nodes. These are described in the following sections.

2.2 File Root

The root element of the XML file is defined as:

```
<?xml version="1.0" encoding="UTF-8"?>
<root BTCPP_format="4" main_tree_to_execute="TourGuide">
<!-- .... -->
</root>
```

The attribute `BTCPP_format="4"` specifies that this behavior tree is built for version 4 of the BehaviorTree.CPP library. This is important because the robot mission interpreter must be implemented with the correct version, in this case, version 4, for the behavior tree specification to be executed correctly. It is worth noting that the Groot2 IDE supports both version 3 and version 4 of the library, configurable within the settings, so this attribute helps ensure compatibility between the designed behavior tree and the mission interpreter. The attribute `main_tree_to_execute="TourGuide"` designates `TourGuide` as the primary behavior tree to be executed, indicating which tree within the file serves as the entry point for the mission.

2.3 Subtrees

To facilitate an efficient mission design process and enhance readability and clarity, the overall mission was structured into several distinct subtrees, each representing a logical segment of the overall task defined in the use case scenario. This enabled easier development and debugging and allows for future modifications. By organizing the behavior tree into modular, clearly defined subtrees, the complexity of mission specifications is significantly reduced. Then, by connecting these modular subtrees in a well-defined logical sequence, the final behavior tree is constructed, which serves as the complete mission specification for the use case scenario as defined in Deliverable D2.1 Use Case Scenario.

2.3.1 TourGuide Subtree

The `TourGuide` subtree is the main behavior tree that orchestrates the robot's actions during the tour. It begins with the `StartOfTree` action node and is composed of five sequential subtrees, each representing a distinct phase of the tour interaction. The segments are executed sequentially, with the robot transitioning from one segment to the next based on the outcome of the previous segment. The behavior tree diagram of the `TourGuide` subtree is shown in Figure 1. The XML representation of the TourGuide subtree is shown in Listing 1.

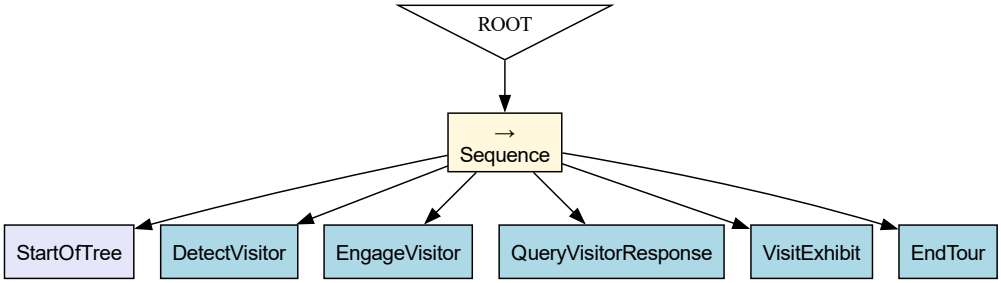

Figure 1: Behavior Tree Diagram of the "TourGuide" Subtree

```
<BehaviorTree ID="TourGuide">
  <Sequence>
    <StartOfTree/>
    <SubTree ID="DetectVisitor"/>
    <SubTree ID="EngageVisitor"/>
    <SubTree ID="QueryVisitorResponse"/>
    <SubTree ID="VisitExhibit"/>
    <SubTree ID="EndTour"/>
  </Sequence>
</BehaviorTree>
```

Listing 1: XML Representation of the "TourGuide" Subtree

2.3.2 DetectVisitor Subtree

The DetectVisitor subtree is dedicated to identifying when a visitor is present. Before initiating visitor detection, however, the robot first retrieves the list of exhibits to be visited and initializes its own starting position within the environment. It then utilizes various sensors available to the robot to continuously monitor its surroundings. The robot is animated to appear lively, actively scanning its environment to reliably localize and track a visitor before initiating an interaction.

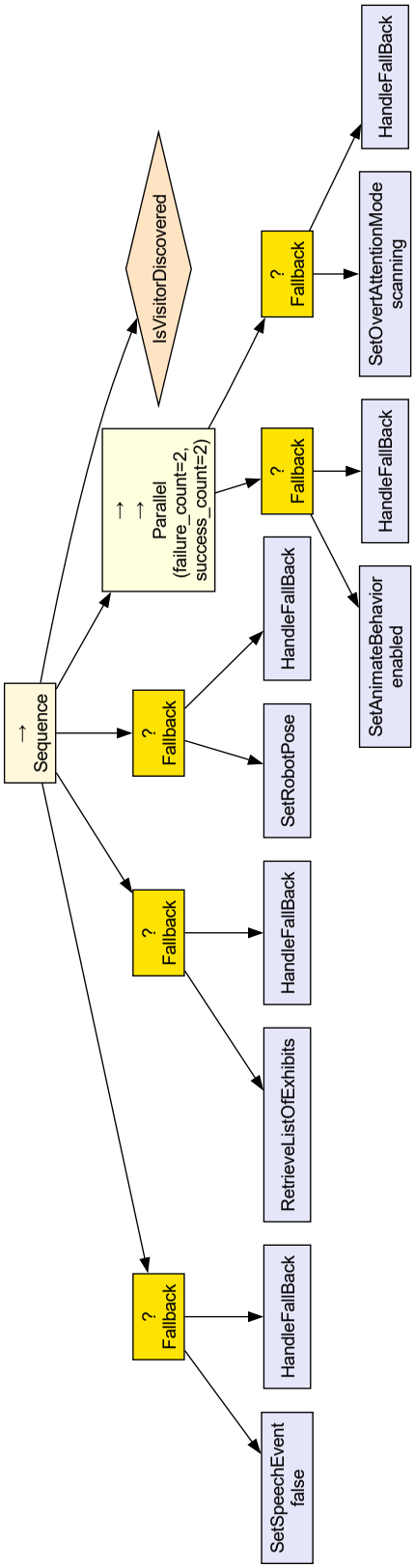

Figure 2: Behavior Tree Diagram of the "DetectVisitor" Subtree

```
<BehaviorTree ID="DetectVisitor">
    <Sequence>
      <Fallback>
        <SetSpeechEvent name="false"/>
        <HandleFallBack/>
      </Fallback>
      <Fallback>
        <RetrieveListOfExhibits/>
        <HandleFallBack/>
      </Fallback>
      <Fallback>
        <SetRobotPose/>
        <HandleFallBack/>
      </Fallback>
      <Fallback>
        <SetAnimateBehavior name="enabled"/>
        <HandleFallBack/>
      </Fallback>
      <Fallback>
        <SetOvertAttentionMode name="scanning"/>
        <HandleFallBack/>
      </Fallback>
      <IsVisitorDiscovered/>
    </Sequence>
  </BehaviorTree>
```

Listing 2: XML Representation of the “DetectVisitor” Subtree

2.3.3 EngageVisitor Subtree

Once a visitor is detected, the robot transitions to actively engaging them. In this phase, the robot makes a welcoming gesture, greets the visitor verbally, and introduces itself as the tour guide. The engagement involves adjusting its body language to establish a friendly and approachable interaction.

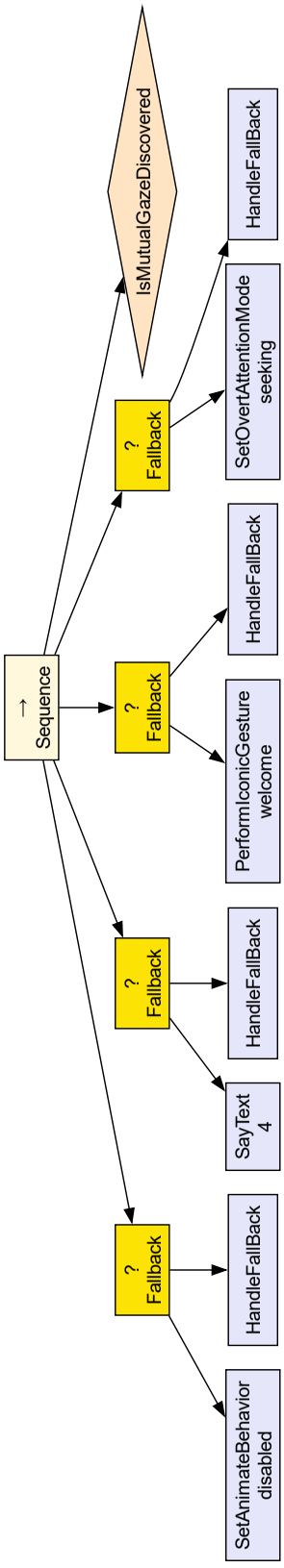

Figure 3: Behavior Tree Diagram of the "EngageVisitor" Subtree

```
<BehaviorTree ID="EngageVisitor">
  <Sequence>
    <Fallback>
      <SetAnimateBehavior name="disabled"/>
      <HandleFallBack/>
    </Fallback>
    <Fallback>
      <SayText name="4"
               _description="Welcome speech"/>
      <HandleFallBack/>
    </Fallback>
    <Fallback>
      <PerformIconicGesture name="welcome"/>
      <HandleFallBack/>
    </Fallback>
    <Fallback>
      <SetOvertAttentionMode name="seeking"/>
      <HandleFallBack/>
    </Fallback>
    <IsMutualGazeDiscovered/>
  </Sequence>
</BehaviorTree>
```

Listing 3: XML Representation of the “EngageVisitor” Subtree

2.3.4 QueryVisitorResponse Subtree

After the initial greeting, the robot seeks confirmation from the visitor about whether they would like to take the tour. This segment handles both speech-based and tablet-based responses. The robot asks a clear question, and waits a certain amount of time for an "affirmative" response (using either automatic speech recognition or a visual touch interface as input). If the response is positive, the robot proceeds to the next segment. Otherwise, it provides a polite response and ends the interaction.

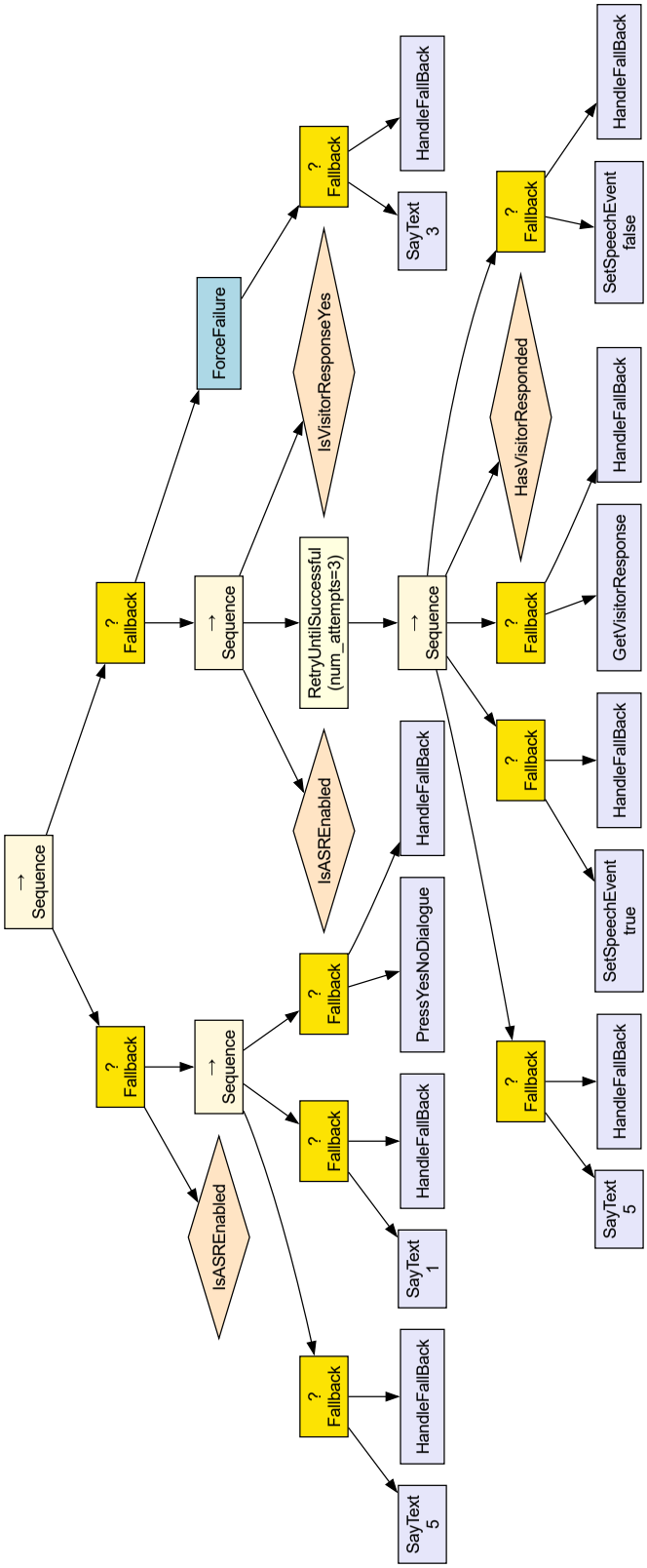

Figure 4: Behavior Tree Diagram of the "QueryVisitorResponse" Subtree

```
<BehaviorTree ID="QueryVisitorResponse">
    <Sequence>
      <Fallback>
        <IsASREnabled/>
        <Sequence>
          <Fallback>
            <SayText name="5"
                     _description="Query Tour Speech"/>
            <HandleFallBack/>
          </Fallback>
          <Fallback>
            <SayText name="1"
                     _description="Press &quot;yes&quot; or
                        &quot;no&quot; speech"/>
            <HandleFallBack/>
          </Fallback>
          <Fallback>
            <PressYesNoDialogue/>
            <HandleFallBack/>
          </Fallback>
        </Sequence>
      </Fallback>
      <Fallback>
        <Sequence>
          <IsASREnabled/>
          <RetryUntilSuccessful num_attempts="3">
            <Sequence>
              <Fallback>
                <SayText name="5"
                         _description="Query Tour Speech"/>
                <HandleFallBack/>
              </Fallback>
              <SetSpeechEvent name="true"/>
              <Sequence>
                <Delay delay_msec="2000">
                  <Fallback>
                    <GetVisitorResponse/>
                    <HandleFallBack/>
                  </Fallback>
                </Delay>
              </Sequence>
              <HasVisitorResponded/>
              <SetSpeechEvent name="false"/>
            </Sequence>
          </RetryUntilSuccessful>
          <IsVisitorResponseYes/>
        </Sequence>
        <ForceFailure>
          <Fallback>
```

```
            <SayText name="3"
                     _description="Maybe another time speech"/>
            <HandleFallBack/>
          </Fallback>
        </ForceFailure>
      </Fallback>
    </Sequence>
```

Listing 4: XML Representation of the “QueryVisitorResponse” Subtree

2.3.5 VisitExhibit Subtree

With a positive response, the tour moves into the exhibit visit segment. Here, the robot guides the visitor from one exhibit to another. For each exhibit, the robot retrieves and announces information about the exhibit from a knowledge base, navigates to the location, checks for visual contact to verify continuation, uses gestures such as pointing to highlight key aspects of the exhibit, and provides descriptive commentary about what is being shown. The core part of this subtree is designed to be repeatable for each exhibit along the tour route. Once all exhibits have been visited, the robot navigates to tis "Home" location, communicates that the tour has ended, expresses gratitude and hope that the visitor enjoyed the tour.

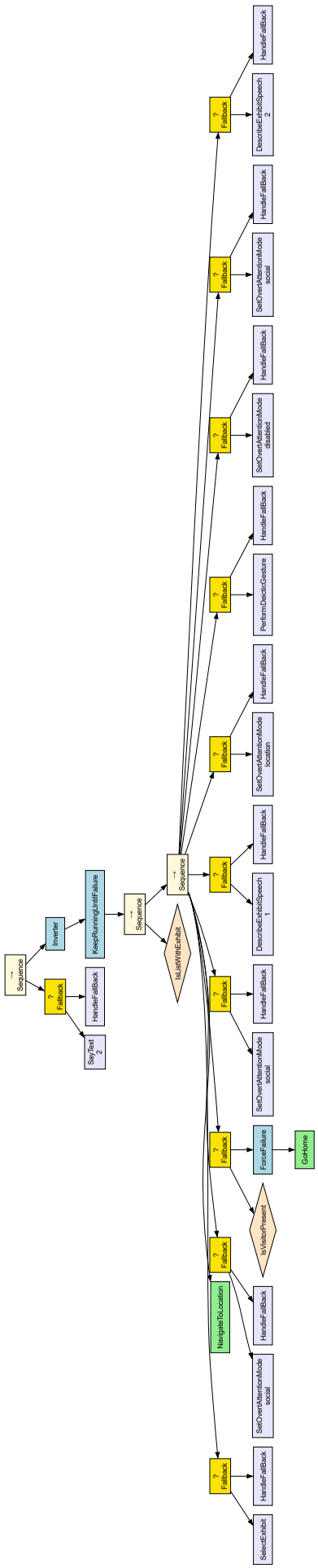

Figure 5: Behavior Tree Diagram of the "VisitExhibit" Subtree

```
<BehaviorTree ID="VisitExhibit">
  <Sequence>
    <Fallback>
      <SayText name="2"
               _description="Follow me speech"/>
      <HandleFallBack/>
    </Fallback>
    <Inverter>
      <KeepRunningUntilFailure>
        <Sequence>
          <IsListWithExhibit/>
          <Sequence>
            <Fallback>
              <SelectExhibit/>
              <HandleFallBack/>
            </Fallback>
            <SubTree ID="NavigateToLocation"/>
            <Fallback>
              <SetOvertAttentionMode name="social"/>
              <HandleFallBack/>
            </Fallback>
            <Fallback>
              <IsVisitorPresent/>
              <ForceFailure>
                <SubTree ID="GoHome"/>
              </ForceFailure>
            </Fallback>
            <Fallback>
              <DescribeExhibitSpeech name="1"/>
              <HandleFallBack/>
            </Fallback>
            <Fallback>
              <SetOvertAttentionMode name="location"/>
              <HandleFallBack/>
            </Fallback>
            <Fallback>
              <PerformDeicticGesture/>
              <HandleFallBack/>
            </Fallback>
            <Fallback>
              <SetOvertAttentionMode name="disabled"/>
              <HandleFallBack/>
            </Fallback>
            <Fallback>
              <SetOvertAttentionMode name="social"/>
              <HandleFallBack/>
            </Fallback>
            <Fallback>
```

```
                <DescribeExhibitSpeech name="2"/>
                <HandleFallBack/>
              </Fallback>
            </Sequence>
          </Sequence>
        </KeepRunningUntilFailure>
      </Inverter>
    </Sequence>
  </BehaviorTree>
```

Listing 5: XML Representation of the "VisitExhibit" Subtree

2.3.6 EndTour Subtree

The final segment concludes the tour experience. Once all the exhibits have been visited and the robot has navigated back to its "Home" location, it finally says goodbye while performing a farewell gesture. This segment ensures a polite and complete wrap-up of the interaction.

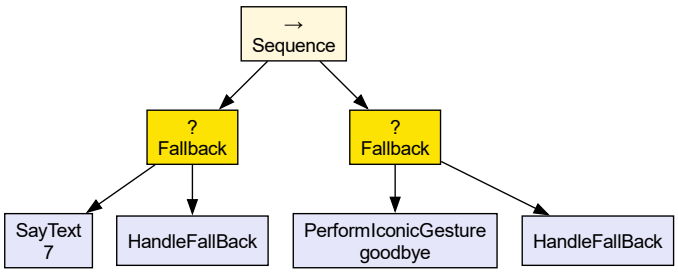

Figure 6: Behavior Tree Diagram of the "EndTour" Subtree

```
  <BehaviorTree ID="EndTour">
  <Sequence>
    <Fallback>
      <SayText name="7" _description="Say Goodbye speech"/>
      <HandleFallBack/>
    </Fallback>
    <Fallback>
      <PerformIconicGesture name="goodbye"/>
      <HandleFallBack/>
    </Fallback>
  </Sequence>
</BehaviorTree>
```

Listing 6: XML Representation of the "EndTour" Subtree

2.3.7 NavigateToLocation Subtree

The `NavigateToLocation` encapsulates a sequence of behaviors that are executed whenever the robot performs a navigation task. The main mission node in this subtree is the **Navigate** action node. This node is the one that's directly responsible for navigating the robot to a specified location. The robot uses its localization and mapping capabilities to plan a path to the target location and execute the navigation. But, before the robot navigates to a location, it must first disable **overt attention mode**. This is necessary since the robot doesn't need to set its gaze anywhere but right in front of it. The robot must focus on the navigation task and ensure it reaches the target location successfully.

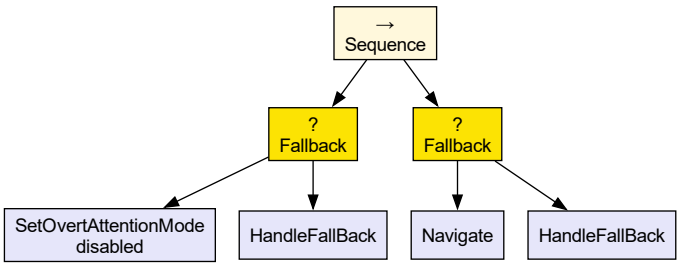

Figure 7: Behavior Tree Diagram of the "NavigateToLocation" Subtree

```
  <BehaviorTree ID="NavigateToLocation">
  <Sequence>
    <Fallback>
      <SetOvertAttentionMode name="disabled"/>
      <HandleFallBack/>
    </Fallback>
    <Fallback>
      <Navigate/>
      <HandleFallBack/>
    </Fallback>
  </Sequence>
</BehaviorTree>
```

Listing 7: XML Representation of the "NavigateToLocation" Subtree

2.3.8 GoHome Subtree

The `GoHome` subtree encapsulates a set of behaviors that are consistently executed whenever the robot is required to return to its designated "Home" location. This location corresponds to the point where the robot initially began the tour. The process begins

with the `RetrieveInitialLocation` action node, which retrieves the coordinates of the "Home" location. Subsequently, the robot invokes the `NavigateToLocation` subtree to execute the navigation behavior. The implementation details of the navigation logic can be found in Section 2.3.7.

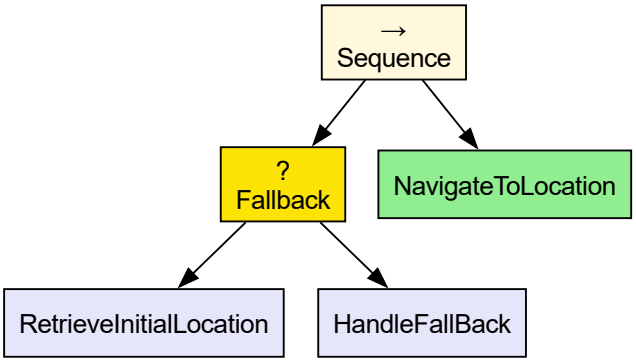

Figure 8: Behavior Tree Diagram of the "GoHome" Subtree

```
<BehaviorTree ID="GoHome">
  <Sequence>
    <Fallback>
      <RetrieveInitialLocation/>
      <HandleFallBack/>
    </Fallback>
    <SubTree ID="NavigateToLocation"/>
  </Sequence>
</BehaviorTree>
```

Listing 8: XML Representation of the "GoHome" Subtree

2.4 Mission Nodes

The leaf nodes, which include both action and condition nodes, are where the custom functionality is implemented. Combined with control flow nodes, these building blocks enable the definition of the desired behaviors. A total of 21 action and condition nodes were defined for the "Lab Tour" scenario, with many of these nodes reused multiple times throughout the behavior tree. These custom nodes are comprehensively listed in Table **??**, which provides the name, type, and description of each node. Note that the actual logic and implementation of these nodes are not detailed here; they are encapsulated within the robot mission interpreter, which executes the behavior tree, as described in Deliverable D5.4.3 Robot Mission Interpreter.

Table 1: High-Level Mission Node Descriptions

Node	Type	Description
DescribeExhibitSpeech	Action	Delivers an auditory description of the selected exhibit to inform and engage the visitor.
GetVisitorResponse	Action	Captures and processes a response from the visitor, used after a prompt or question.
HandleFallBack	Action	Handles unexpected states or errors by triggering fallback mechanisms in the mission logic.
HasVisitorResponded	Condition	Checks if a visitor has provided a response or not.
IsASREnabled	Condition	Checks whether the automatic speech recognition (ASR) system is enabled and operational.
IsListWithExhibit	Condition	Determines whether there is an available list of exhibits to present or navigate to.
IsMutualGazeDiscovered	Condition	Detects whether mutual gaze between the robot and the visitor has been established.
IsVisitorDiscovered	Condition	Checks if a visitor is currently detected within the interaction range of the robot.
IsVisitorResponseYes	Condition	Determines whether the visitor has provided an affirmative response to a previous query.
Navigate	Action	Directs the robot to move toward a specified target location.
PerformDeicticGesture	Action	Performs a pointing gesture to draw the visitor's attention to a specific area or exhibit.
PerformIconicGesture	Action	Executes a recognizable gesture such as a goodbye wave or a welcome motion to visually complement the interaction.
PressYesNoDialogue	Action	Initiates a Yes/No interaction via buttons or a GUI to capture explicit visitor responses.
RetrieveInitialLocation	Action	Retrieves the robot's starting location.
RetrieveListOfExhibits	Action	Collects and stores the list of exhibits that will be part of the guided tour.
SayText	Action	Enables the robot to vocalize a given text string.
SelectExhibit	Action	Chooses the next exhibit from the list retrieved using "RetrieveListOfExhibits"
SetAnimateBehavior	Action	Sets the state of the animation behaviors to enhance the robot's lifelike presence.
SetOvertAttentionMode	Action	Sets the robot's overt attention mode, influencing how it visually engages with its environment.
SetRobotPose	Action	Assigns a specific pose or position to the robot in the environment.
SetSpeechEvent	Action	Activates or deactivates the automatic speech recognition feature.

Node	Type	Description
`StartOfTree`	Action	Used for debugging and initializing mission wide variables.

References

[1] R. Ghzouli, T. Berger, E. B. Johnsen, A. Wasowski, and S. Dragul. Behavior trees and state machines in robotics applications. *IEEE Transactions on Software Engineering*, 49(9):4243 – 4267, 2023.

[2] E. Dortmans and T. Punter. Behavior trees for smart robots practical guidelines for robot software development. *Journal of Robotics*, 2022.

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).

Tsegezeab Tefferi, Carnegie Mellon University Africa.
David Vernon, Carnegie Mellon University Africa.

Document History

Version 1.0

First version created by moving and reorganizing Section 3 from Deliverable D5.4.2.
David Vernon.
18 April 2025.

Version 1.1

Updated the list of high-level mission nodes, along with their descriptions to reflect the simplified mission specification.
Updated the XML output of the mission specification and revised the descriptions of each subtree to reflect the new behavior tree structure.
Replaced the large, comprehensive behavior tree diagram with individual diagrams for each subtree
Resolved various issues noted in the previous version.
Tsegazeab Tefferi.
05 May 2025

Version 1.2

Updated the XML outputs, the diagrams and the list of high-level mission nodes to reflect the current version of the behavior tree.
Updated the diagrams to be centered and to fit within the page margins.
Revised the description of the mission specification to clarify that it is not an exact replication of the scenario outlined in D2.1, but rather a functional implementation.
Tsegazeab Tefferi.
01 June 2025