Towards using Behaviour Trees for long-term social robot behaviour

Sara Cooper
PAL Robotics
Barcelona
sara.cooper@pal-robotics.com

Séverin Lemaignan
PAL Robotics
Barcelona
severin.lemaignan@pal-robotics.com

***Abstract*—This paper introduces a Behaviour Tree based design of long-term social robot behaviour in the context of SHAPES project, using ROS-compatible libraries, specifically two types of behaviours: a robot idle behaviour where the human approaches and begins the interaction, and a second behaviour where the robot actively navigates and searchers for a specific user to deliver a reminder. The behaviours will be tested on-site as part of SHAPES pilots and adjusted based on feedback and needs and is focused on long-term robot acceptance.**

***Index Terms*—human-robot interaction, older care, social robot**

I. Introduction

A. Behaviour trees to model social tasks

Modelling the diversity and complexity of behaviours of a social robot is a challenging task. In the past, most robots were one-task-specific, but as problem complexity increases, it is necessary for their internal software architecture to be able to easily extend, adapt to changes, and for that, modularity is key.

Behaviour Trees (BTs) have been used in the past to model different tasks or robot behaviours [1]–[3], proposed initially Mateas and Stern [4]. A Behaviour Tree (BT) is a formalism to structure the switching between different tasks in an autonomous agent, such as a robot or a virtual entity in a computer game [5].

A Behaviour Tree is a hierarchical tree of nodes, where leave nodes are executable programs that correspond to robot actions: pick up an object, say something, do a gesture. The tree is executed or "ticked" from left to right, where each child node that has been ticked may return a result of SUCCESS, FAILURE or RUNNING (node has not finished executing). There are three main types of control flow nodes:

1) *Sequence*: every time the control node receives a tick, it ticks the first of his children, once it returns SUCCESS it will tick the next child. Once all the children of the sequence return success, the parent will also return SUCCESS. It is the most common type used to do consecutive actions;
2) *Decorators*: they have only one child, such as *Repeat* or *Retry*;
3) *Fallbacks*: ticks the first child, if it is SUCCESS it returns SUCCESS; if FAILURE, it ticks the second child. For example, if no user is recognised after a given time, we can have a fallback action that makes the robot go to its initial position;
4) *Parallel*: it runs two child nodes at the same time. If one of the child returns FAILURE or SUCCESS, it will stop the execution of the other child. The number of failures or successes is often dependent on a threshold. For example, if we want to navigate around while detecting faces.

In addition, there are two execution or leaf nodes: *Action* nodes, that execute a task without checking a condition (e.g. pick an object, go to dock station, say something); and *Condition* nodes, that check a condition (e.g. face detected yes or not).

In contrast to other alternatives such as Finite State Machines (FSMs), Behaviour Trees make it easy to switch between different tasks of an autonomous robot, offer higher modularity and reactiveness, that lead to their use in Robotics and AI applications. This is a key requirement for dynamic environments as typically found in HRI and social robotics. For instance, if the robot is moving towards a person and another person enters the planned trajectory, it should adapt its behaviour quickly [6].

Specifically, some advantages of Behaviour Trees in contrast to FSM include [1]:

- Modularity: separate into building blocks, as such also offering possibility to extend the trees compose behaviours of sub-behaviours. For example, we can make "Go to Poin A" as sub-tree of "Look for User";
- Reactivity: quickly enables reacting to changes;
- Enable separation of tree structure from node implementation, option to reuse a node several times;
- Can run several nodes in parallel, instead of one state at a time;
- Expressivity: easier to understand behaviour tree workflow and express complex flows, also with possibility to add custom nodes;
- Easily reusable code and maintainable.

B. The SHAPES project

H2020 SHAPES project[1] is a European project that aims to create the first European open ecosystem that enables the

[1]https://shapes2020.eu/

large-scale deployment of a broad range of digital solutions to support and extend healthy and independent living among older individuals. Specifically, PAL Robotics ARI robot [7] is being used for one of the pilot deployment [8], that aims to use the robot at clinics in Mallorca (Spain), with older adults between 70-90 at their sheltered apartments. The robot is expected to autonomously deliver reminders, play games, monitor temperature, or fill the daily menu. The robot additionally integrates a set of technologies shared accross the whole SHAPES ecosystem, like face recognition, detection, authentication mechanisms, chatbot and emotion recognition. Figure 1 shows some example screens.

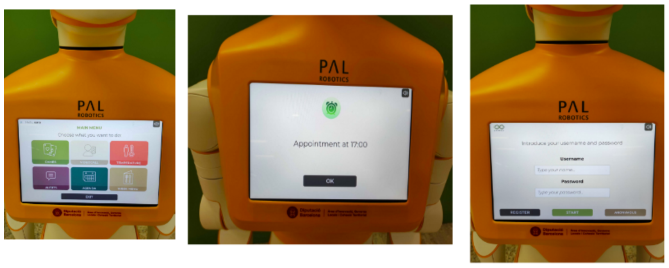

Fig. 1. SHAPES apps menu, reminders' screen, and authentification screen

Modelling a suitable human-robot interaction behaviour is a key requirement to achieve robot acceptance in this scenario. As the pilots will run across different phases, and will be re-adjusted based on feedback, the architecture should be easily extensible, building on reusable modules. At this stage only Behavior Trees will be used as no further need has been identified to use FSM, however, as pilots progress, they may be also used.

II. Related Work

Behaviour Trees have been used in different robotics sectors such as autonomous vehicles or industrial robotics [3], [6], [9]. In case of industrial robotics, a major reason has been the shift between single-task industrial applications, to collaborative robots that need to perform in unstructured environments.

For robot modelling additional architectures used include SMACH [2] [10], a ROS independent Python library that builds hierarchical state machines. However, for unstructured tasks such as social interactions, it falls short.

There is some related work done that applies Behaviour Trees to model robot behaviours [1], including virtual agents [5], but little when it comes to modelling social behaviour. Axelsson [3] used BT for dialogue management of interactive robots, tested in SciRoc challenge, where robots such as Pepper [11] and TIAGo [12] where used int he context of taking an elevator.

III. Proposed Approach

ROS (Robotics Operating System) is used as a basis for implementing Behaviour Trees of SHAPES user-cases. The `BehaviourTree.CPP`[3] library has been used, which enables easy integration with the ARI robot due to its full ROS support.

Implementation wise, the advantage of using Behaviour Trees observed is that any kind of ROS action, service or topic can be wrapped as a BT node and also saving and passing values accross the tree nodes. For instance:

- `ShapesFaceIdentifConditional`: checks the topic `/facerecognised`, of type Boolean, and outputs a `String` that is stored as `username`, name of the identified person, in the Blackboard of the tree; that can be used in another node;
- `GoToWebAction`: publishes the web URL indicated as input port to the node to a `/web/goto` topic;
- `TTSAction`: triggers the ROS action `/tts` with the text to be said by the robot

It also makes it easy to use different face recognizers, as as long as they are both publishing to the same ROS topic, the same BT node can be used.

The interactive behaviour of the robot is implemented as two independent Behaviour Trees, that consist of two XML files: idle behaviour, and deliver reminders.

These behaviour trees were designed with Groot[4].

A. Modelling Robot idle behaviour

In this behaviour, the ARI robot remains at the dock station, positioned at a side of the house, in a "sleeping" mode e.g. head down. For a first test, this has been modelled by lowering the head down, and keeping still.

Figure 2 shows the complete behaviour tree sketched using Groot. As it can be observed, it will be constantly repeating the tree using a `Repeat` node, as the goal is to have long-term interaction where the robot starts from a "sleeping" position, interacts when a user comes, and goes back to "sleeping" position when done.

The `Sequence` starts by "tick"ing the first child on the left, which is a `TTSAction` Control Action node that triggers robot Text-To-Speech with movements. The type of robot motion to do is indicated in tags `<mark name='doTrick trickName=sleep'/>`.

Next, the conditional node `InteractionStatusConditional` waits to detect any human interaction. This is defined as one of the following:

- Face has been detected, OR
- Touch-screen has been pressed, OR
- Speech has been detected.

If any of above is detected, the node returns SUCCESS, and will tick the next node that makes ARI raise the head and look around.

Once interaction has began, the tree enters a *Parallel* sequence. The advantage of *Parallel* control flows is that it is possible to run two tasks at the same. Specifically, the robot tries to detect a face (`ShapesFaceConditional` node), but if none is detected after a specific time (*Delay* decorator

[2]http://wiki.ros.org/smach
[3]https://www.behaviourtree.dev/, developed by Davide Faconti.

[4]https://github.com/BehaviourTree/Groot

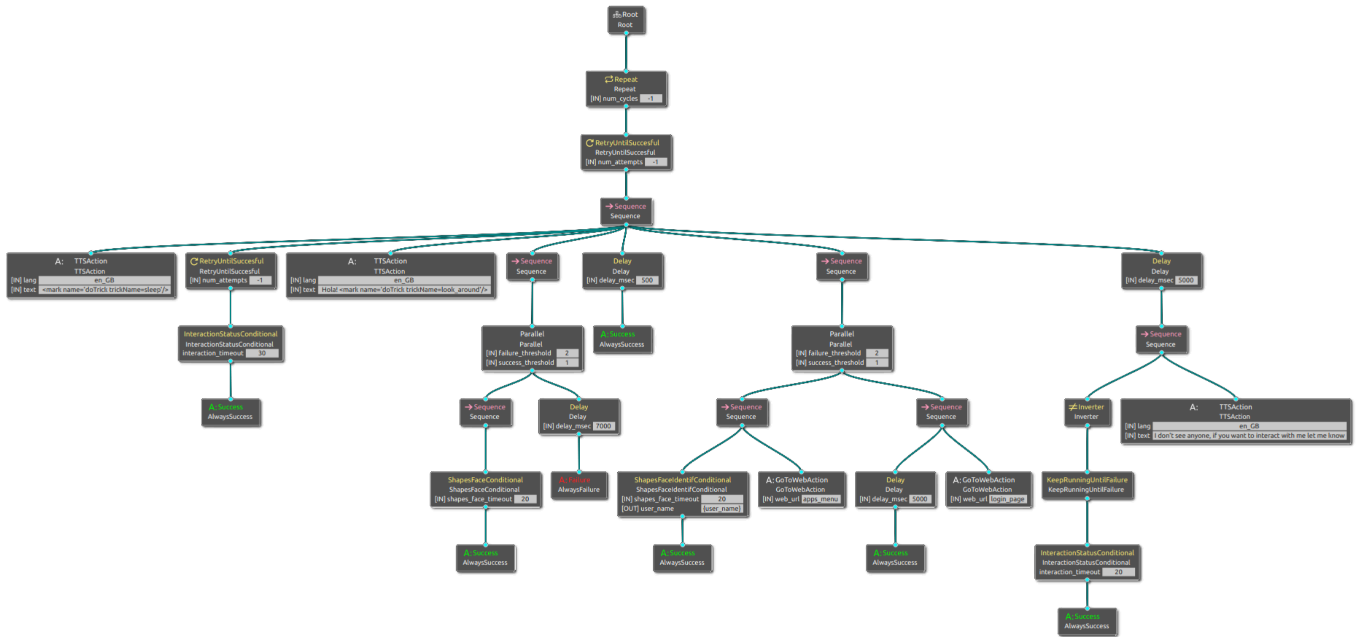
Root Root
Repeat Repeat [IN] num_cycles -1
RetryUntilSuccesful RetryUntilSuccesful [IN] num_attempts -1
Sequence Sequence
A: TTSAction TTSAction [IN] lang en_GB [IN] text <mark name='doTrick trickName=sleep'/>
RetryUntilSuccesful RetryUntilSuccesful [IN] num_attempts -1
InteractionStatusConditional InteractionStatusConditional interaction_timeout 30
A: Success AlwaysSuccess
A: TTSAction TTSAction [IN] lang en_GB [IN] text Hola! <mark name='doTrick trickName=look_around'/>
Sequence Sequence
Parallel Parallel [IN] failure_threshold 2 [IN] success_threshold 1
Sequence Sequence
ShapesFaceConditional ShapesFaceConditional [IN] shapes_face_timeout 20
A: Success AlwaysSuccess
Delay Delay [IN] delay_msec 7000
A: Failure AlwaysFailure
Delay Delay [IN] delay_msec 500
A: Success AlwaysSuccess
Sequence Sequence
Parallel Parallel [IN] failure_threshold 2 [IN] success_threshold 1
Sequence Sequence
ShapesFaceIdentifConditional ShapesFaceIdentifConditional [IN] shapes_face_timeout 20 [OUT] user_name {user_name}
A: Success AlwaysSuccess
A: GoToWebAction GoToWebAction [IN] web_url apps_menu
Sequence Sequence
Delay Delay [IN] delay_msec 5000
A: Success AlwaysSuccess
A: GoToWebAction GoToWebAction [IN] web_url login_page
Delay Delay [IN] delay_msec 5000
Sequence Sequence
Inverter Inverter
KeepRunningUntilFailure KeepRunningUntilFailure
InteractionStatusConditional InteractionStatusConditional interaction_timeout 20
A: Success AlwaysSuccess
A: TTSAction TTSAction [IN] lang en_GB [IN] text I don't see anyone, if you want to interact with me let me know

Fig. 2. Complete behaviour tree for the robot's *Idle* behaviour

node), the right branch will return FAILURE, together with the left branch (no face detected, resulting in another FAILURE). As in the Parallel a failure threshold of 2 has been defined, the Sequence will return FAILURE, and the tree would start again thanks to the `RetryUntilSuccessfull`, by putting the robot to sleep.

In contrast, if a face is detected, the left branch of the `Parallel` will return SUCCESS, and thanks to the success threshold of 1, the tree will continue ticking the next node.

Once a face has been detected, the tree follows a similar pattern to try to recognise the face. The Parallel node in this case offers two options to return a SUCCESS:

- Face is recognised, where `ShapesFaceIdentifConditional` returns SUCCESS), so the robot displays the SHAPES Apps Menu to the user using the `GoToWebAction` Control Action node. The left branch is successful.
- No face is recognised after 5 seconds, in such case the robot displays the SHAPES login page so the user can authenticate using username and password login using the same `GoToWebAction` node.

Finally, it is important for the robot to know when an interaction is finished. For this the `InteractionStatusConditional` Decorator node is checked continuously until it returns FAILURE, that is, no face, speech or touch-screen action is detected. The behaviour then ends by going back to sleep and starting from the beginning.

B. *Modelling robot look for user behaviour*

The second behaviour requires the robot to:

- Stay at the dock station
- Undock when it is time to deliver a reminder
- Navigate accross a series of points while detecting a face
- Stop if a face is detected
- Recognise the needed user
- Deliver the reminder using speech and touch screen
- Carry out additional interactions if needed
- Return back to dock station

Figure 3 shows a top-view of an example map the robot can navigate around, indicating the location of the dock station. In purple indicated are the Points of Interests (POI) that the robot will navigate through, until a face is detected.

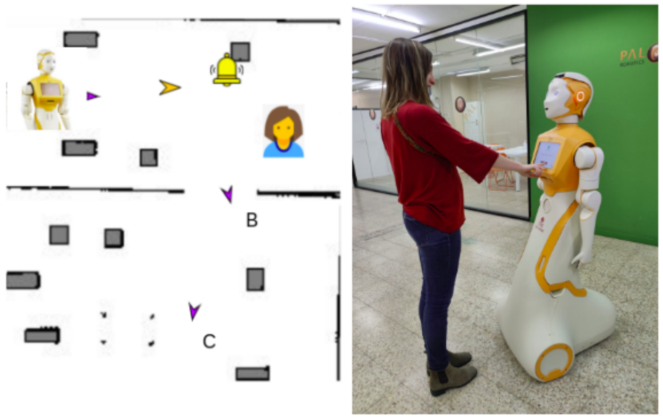

Fig. 3. Example robot map with Points of Interests and Dock Station. As the robot navigates across, it stops once a user is detected and identified to deliver the reminder

Unlike the previous case, here the behaviour is not executed repeatedly, only when a reminder is sent. To start the BT, the variables `name` and `text` are inputted to the Blackboard, indicating who the robot should look for, and what to show on the touch-screen.

Firstly, as seen in Figure 4 there is an `UndockAction` Control Action node, that triggers the needed ROS Action to leave the dock station. Then, similar to the idle behaviour, there is a Parallel structure. - The left branch tries to detect a face, if any is detected, the robot will say "Hello" - The right branch will continue running until it returns FAILURE. Specifically, it sends the robot to a list Points of Interests (POI), that are previously defined in the map of the robot: PointA, PointB and PointC (stored in the BT blackboard as `ListPois` variable). The `SelectPoiAction` pops the first POI, e.g. PointA to the `POIgoal` variable, and uses the `GoToPOIAction` to send the robot to the target poi. Every time the robot reaches the goal, it says "I am looking for `name`, I have a reminder". Once the robot goes through all points, the right branch will return FAILURE.

Note that if no face has been detected after checking all the points of interest, the tree has a *Fallback* node, making the robot go back to the dock station. If a face is detected while the robot is moving, the robot will stop.

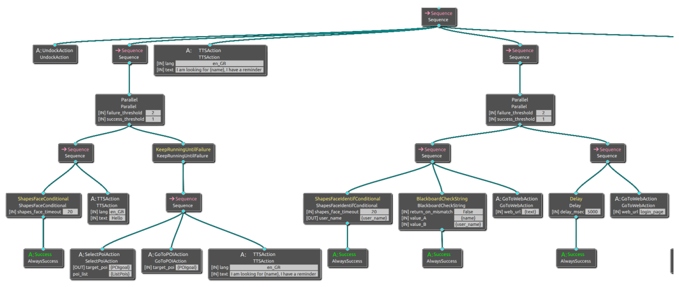
Fig. 4. *Look for user* behaviour 1: undock and go through a series of Points of Interest until a face is detected. Check if face recognised matches user to be found and deliver reminder if needed

Next is the process of identifying the face, similar to the idle behaviour. The difference here as seen in Figure 5 is that the robot checks if the username returned by the recogniser matches with the `name` it should look for (`BlackBoardCheckString`). If this is the case, it will display the reminder needed seen in Figure 1. If not, it will ask the user to authenticate using credentials.

Ending the interaction is similar as well, with the difference that when no interaction is detected, the robot is sent to its docking station using the *DockAction*, after alerting using speech.

IV. Discussion

Especially in the context of human-robot interaction, reactivity is highly important, as robots are in unstructured environments where humans may enter and exit the scene at any time, and for human safety, robots need to be able to quickly stop, as is the case of looking for a user. To this end, BT has been regarded a useful tool for these pilots.

While Behaviour Trees offer higher modularity than Finite State Machines, it is observed that for situations with high

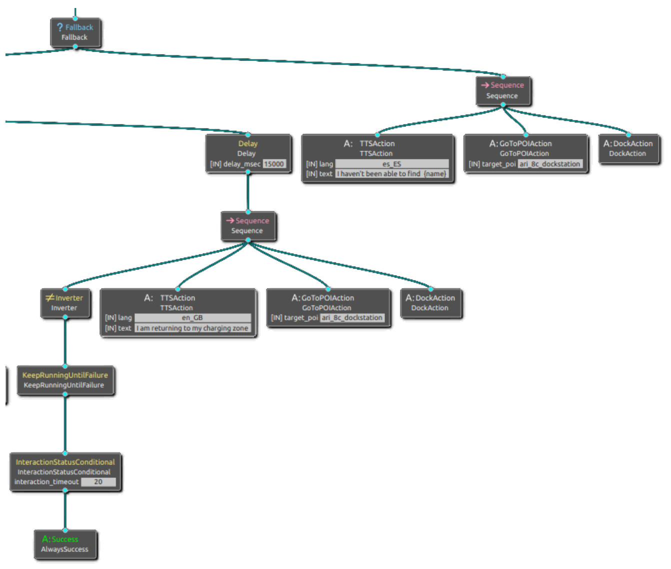

Fig. 5. *Look for user* behaviour 2: return to dock station when no interaction is detected

variability or complexity, they could pose similar limitations. To this extend, it would be considered better to use BT exclusively for tasks that form a plan, but use a higher-level planning approach to combine them.

When it comes to behaviour modelling, empathetic robots are demonstrated to increase acceptability, by making idle motions when idle [13] - such as moving arms and head slightly, and a higher degree of politeness and when the needs to interrupt a person [14]. For example, by the robot saying to this end,"Hello, sorry for disturbing, but are you Sara? I have a reminder for Sara". Both of these may be added to the Behaviour Tree as nodes.

V. Future Work

During the SHAPES initial pilots in Can Granada residence [5] and Clínica Humana [6], the preliminary behaviours will be tested, and will be adjusted based on their performance and user preference. Some improvements may include better defining when to end interaction or trigger a reminder. Additional extensions planned include adding a voice-based chatbot.

Thanks to the modularity and hierarchical nature of the behaviour trees, it will be possible to extend the robot behaviour in the future. For example, once the robot finds a user at Point C, instead of only delivering a reminder, it could then continue to find another person or help the user to another room. If needed the usage of FSMs will also be considered.

Acknowledgment

This work was supported by the SHAPES project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 857159

[5]https://www.cangranada.com/
[6]https://www.clinicahumana.es/

References

[1] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI*. CRC Press, Jul. 2018. [Online]. Available: https://doi.org/10.1201/9780429489105

[2] R. Ghzouli, T. Berger, E. B. Johnsen, S. Dragule, and A. Wasowski, "Behavior trees in action: A study of robotics applications," in *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering*, ser. SLE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 196–209. [Online]. Available: https://doi.org/10.1145/3426425.3426942

[3] N. Axelsson and G. Skantze, "Modelling adaptive presentations in human-robot interaction using behaviour trees," in *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*. Stockholm, Sweden: Association for Computational Linguistics, Sep. 2019, pp. 345–352. [Online]. Available: https://aclanthology.org/W19-5940

[4] M. Mateas and A. Stern, "A behavior language for story-based believable agents," *IEEE Intelligent Systems*, vol. 17, no. 4, pp. 39–47, 2002.

[5] I. Hasegawa, T. Hasegawa, K. Kurosaka, A. Kishi, A. Iwasawa, and Y. Miyake, "How to build a fantasy world based on reality: A case study of final fantasy xv: Part ii," in *SIGGRAPH Asia 2017 Courses*, 2017, pp. 1–149.

[6] F. Martín, F. J. Rodríguez Lera, J. Ginés, and V. Matellán, "Evolution of a cognitive architecture for social robots: Integrating behaviors and symbolic knowledge," *Applied Sciences*, vol. 10, no. 17, p. 6067, 2020.

[7] S. Cooper, A. Di Fava, C. Vivas, L. Marchionni, and F. Ferro, "Ari: The social assistive robot and companion," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 745–751.

[8] S. Cooper, A. Di Fava, Ó. Villacañas, T. Silva, V. Fernandez-Carbajales, L. Unzueta, M. Serras, L. Marchionni, and F. Ferro, "Social robotic application to support active and healthy ageing," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 1074–1080.

[9] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of behavior trees in robotics and ai," *arXiv preprint arXiv:2005.05842*, 2020.

[10] J. Bohren and S. Cousins, "The smach high-level executive [ros news]," *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 18–20, 2010.

[11] A. K. Pandey and R. Gelin, "A mass-produced sociable humanoid robot: Pepper: The first machine of its kind," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 40–48, 2018.

[12] J. Pages, L. Marchionni, and F. Ferro, "Tiago: the modular robot that adapts to different research needs," in *International workshop on robot modularity, IROS*, 2016.

[13] R. H. Cuijpers and M. A. Knops, "Motions of robots matter! the social effects of idle and meaningful motions," in *International Conference on Social Robotics*. Springer, 2015, pp. 174–183.

[14] P. E. McKenna, I. Keller, J. L. Part, M. Y. Lim, R. Aylett, F. Broz, and G. Rajendran, "" sorry to disturb you" autism and robot interruptions," in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 360–362.