ReBeT: Architecture-based Self-adaptation of Robotic Systems through Behavior Trees

Elvin Alberts
Vrije Universiteit Amsterdam & Delft University of Technology

Ilias Gerostathopoulos
Vrije Universiteit Amsterdam

Vincenzo Stoico
Vrije Universiteit Amsterdam

Patricia Lago
Vrije Universiteit Amsterdam

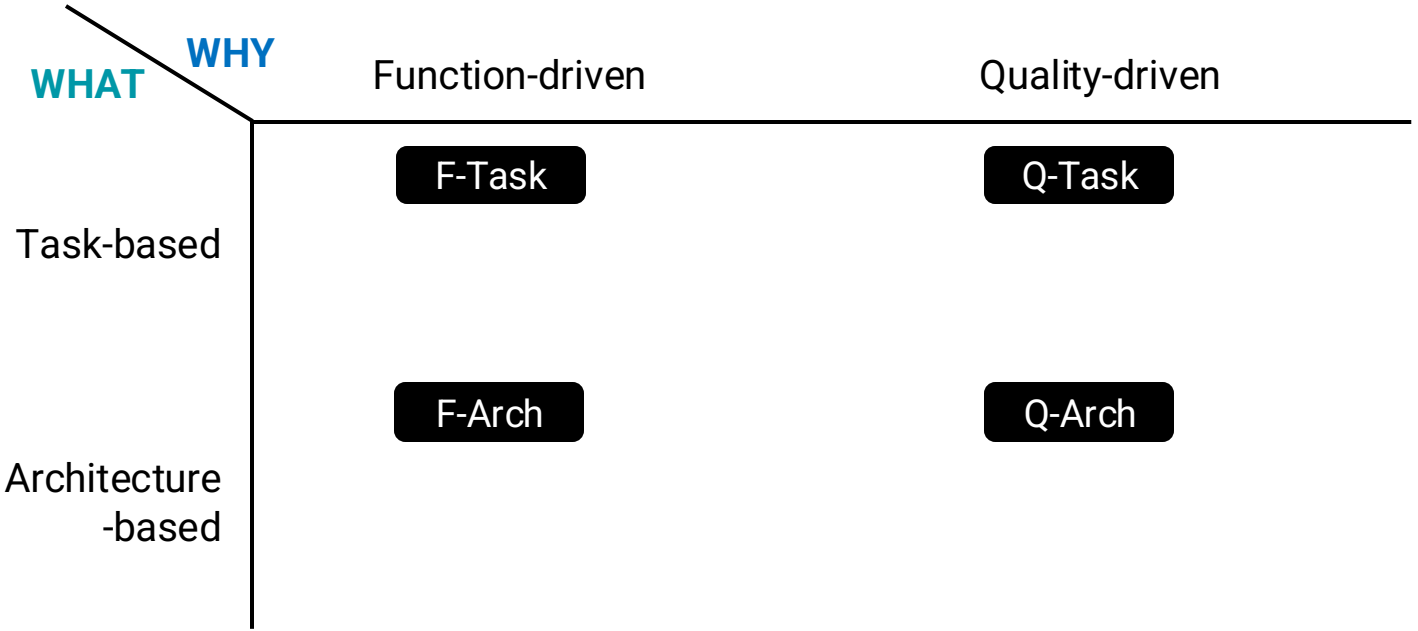
Robots need to become self-adaptive.

Background

Behavior Trees

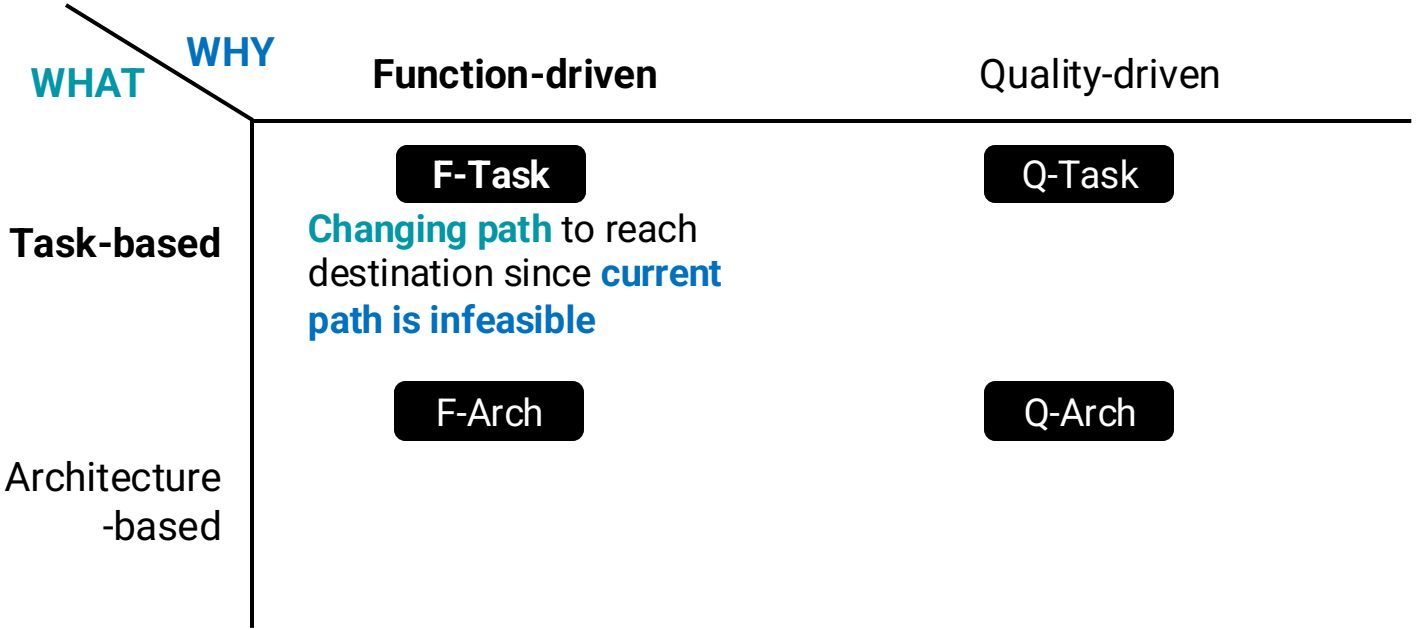
- Kind of like a hierarchical state machine
- The tree is ‘ticked’ at the root, and this is propagated down.
- Each node is either in a state of ‘Success’, ‘Failure’ or ‘Running’
- There is quite nice library non-specific to robotics called BT.CPP

Background



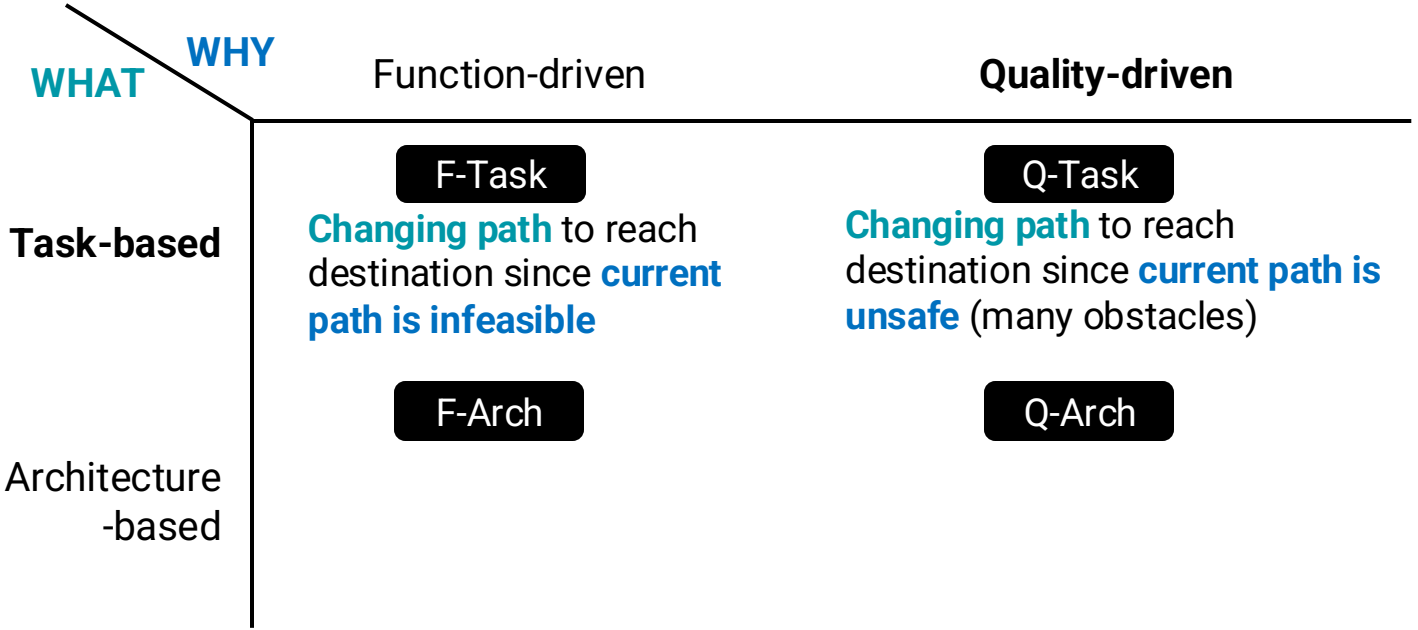
Based on Alberts et al. "Software Architecture-Based Self-Adaptation in Robotics."

Background



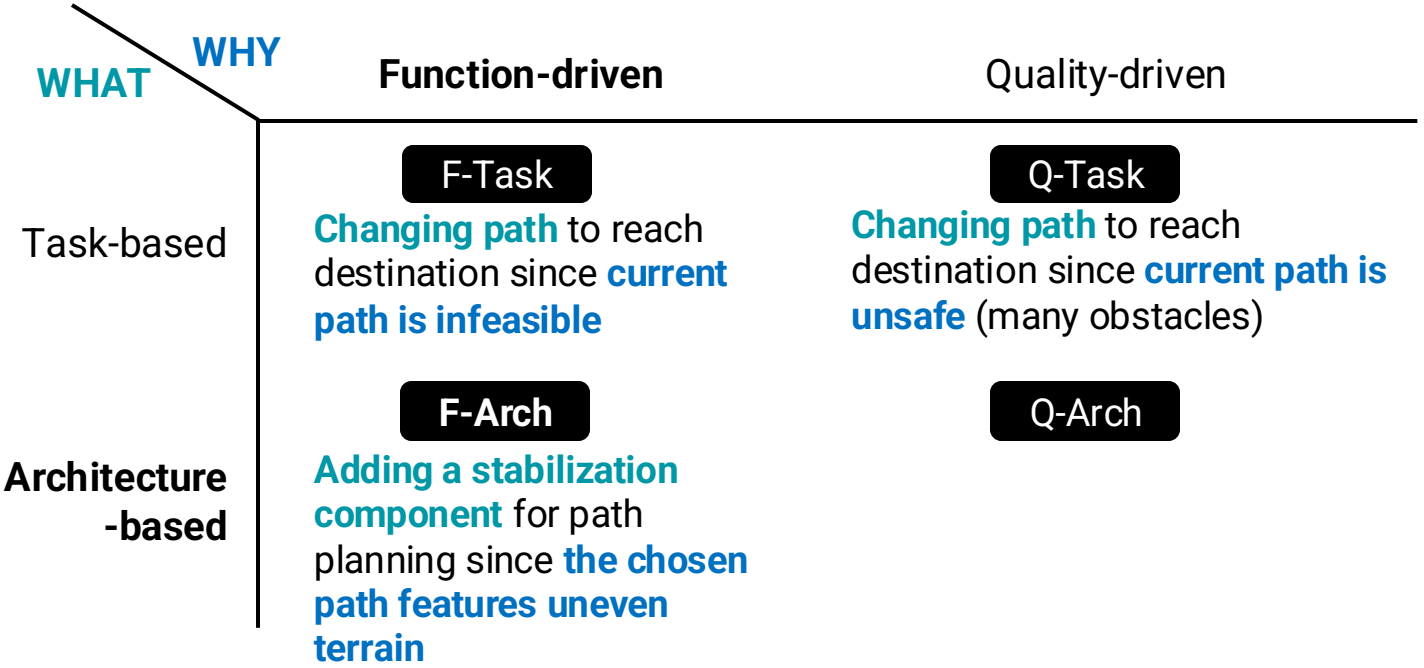
Based on Alberts et al. "Software Architecture-Based Self-Adaptation in Robotics."

Background



Based on Alberts et al. "Software Architecture-Based Self-Adaptation in Robotics."

Background



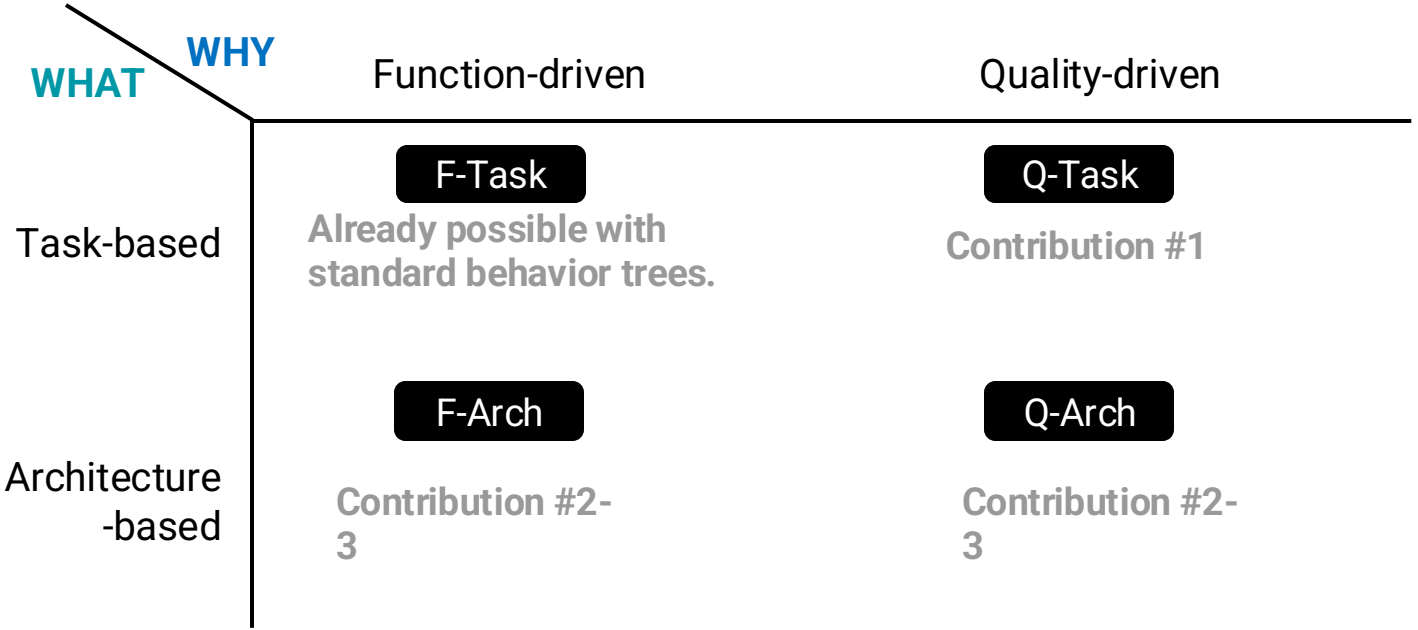
Based on Alberts et al. "Software Architecture-Based Self-Adaptation in Robotics."

Background

WHAT \ WHY	Function-driven	**Quality-driven**
Task-based	**F-Task** **Changing path** to reach destination since **current path is infeasible**	**Q-Task** **Changing path** to reach destination since **current path is unsafe** (many obstacles)
Architecture -based	**F-Arch** **Adding a stabilization component** for path planning since **the chosen path features uneven terrain**	**Q-Arch** **Utilizing an external rather than onboard camera feed** for path planning since the **external camera is more accurate.**

Based on Alberts et al. "Software Architecture-Based Self-Adaptation in Robotics."

Background


WHY
WHAT
Function-driven
Quality-driven
Task-based
Architecture -based
F-Task
Already possible with standard behavior trees.
Q-Task
Contribution #1
F-Arch
Contribution #2-3
Q-Arch
Contribution #2-3

Standard Behavior Trees (F-Task)

Three types of BT nodes:

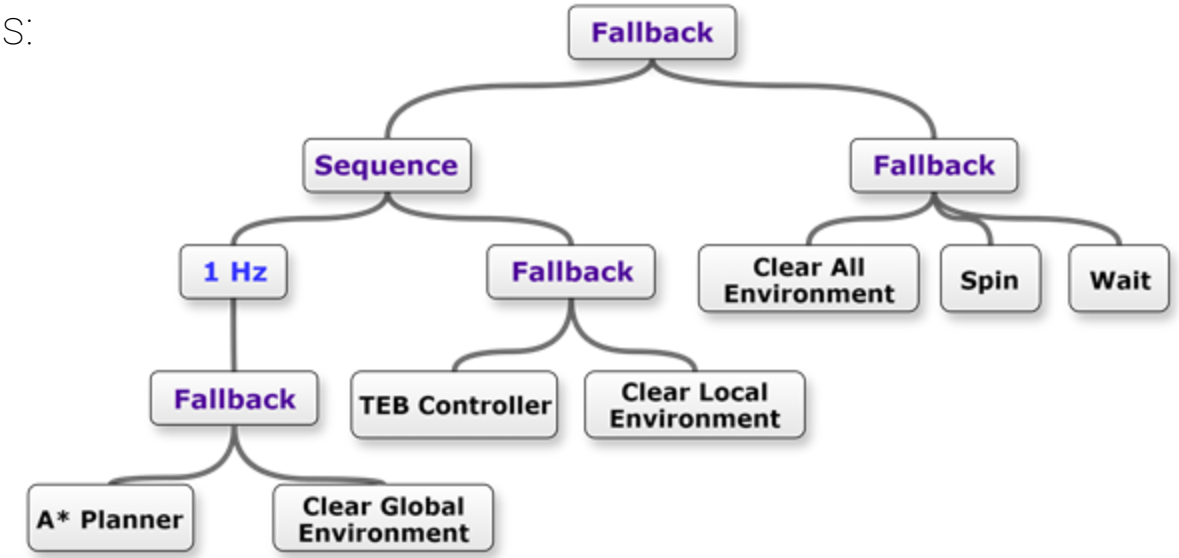


Diagram adapted from Macenski et al. "The marathon 2: A navigation system." *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

Standard Behavior Trees (F-Task)

Three types of BT nodes:

- **Control**

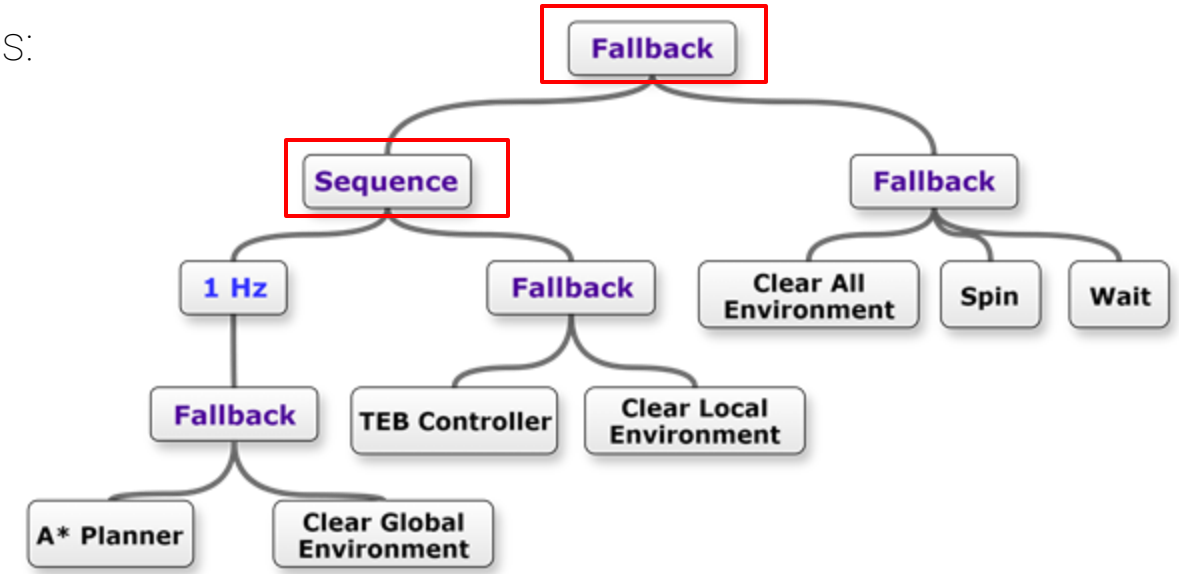


Diagram adapted from Macenski et al. "The marathon 2: A navigation system." *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

Standard Behavior Trees (F-Task)

Three types of BT nodes:

- Control
- **Decorators**

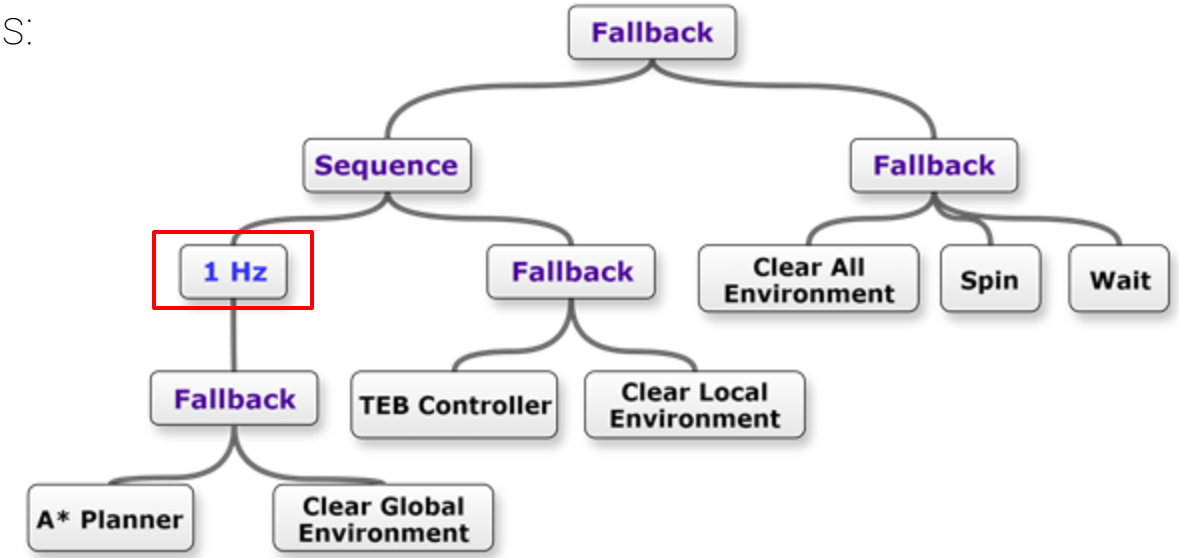


Diagram adapted from Macenski et al. "The marathon 2: A navigation system." *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

Standard Behavior Trees (F-Task)

Three types of BT nodes:

- Control
- Decorators
- **Actions**

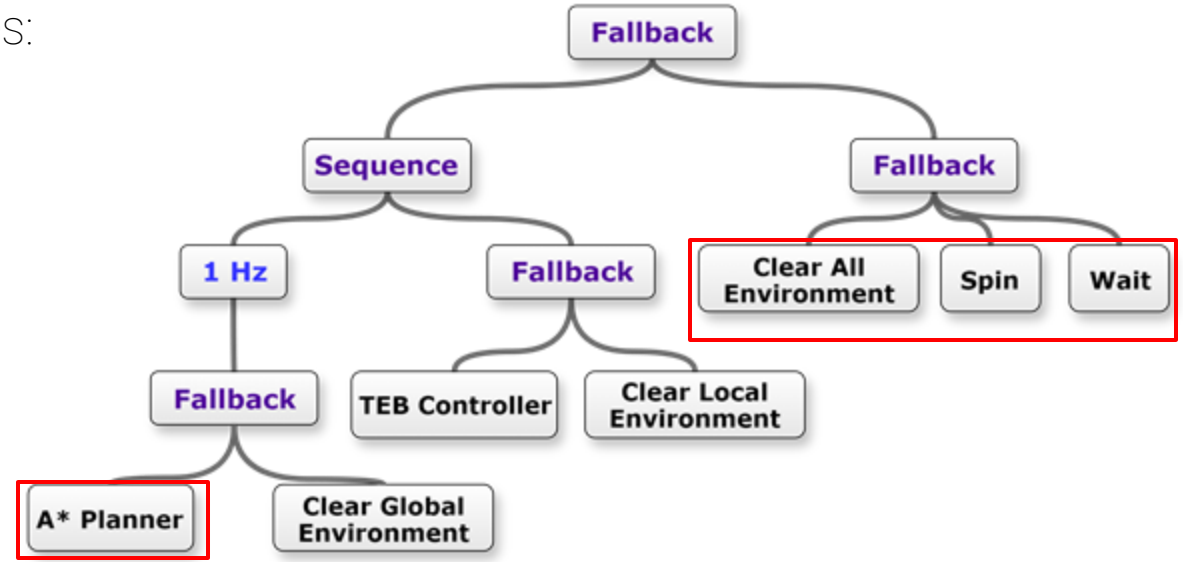
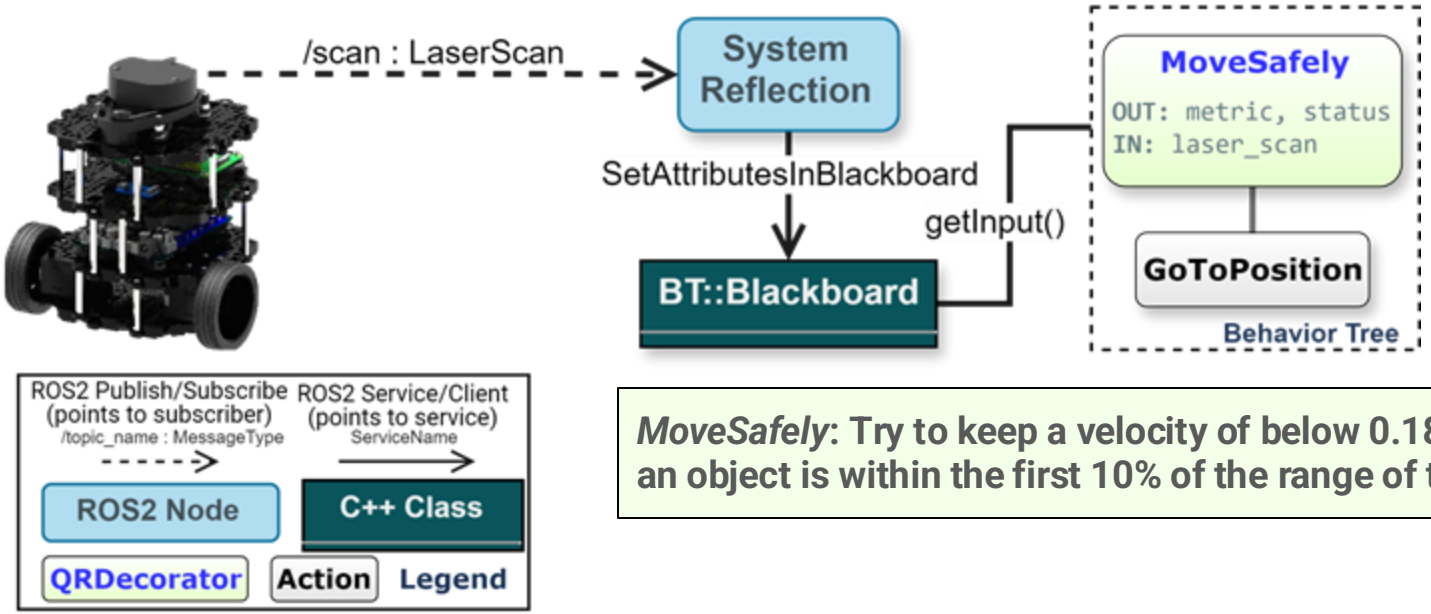


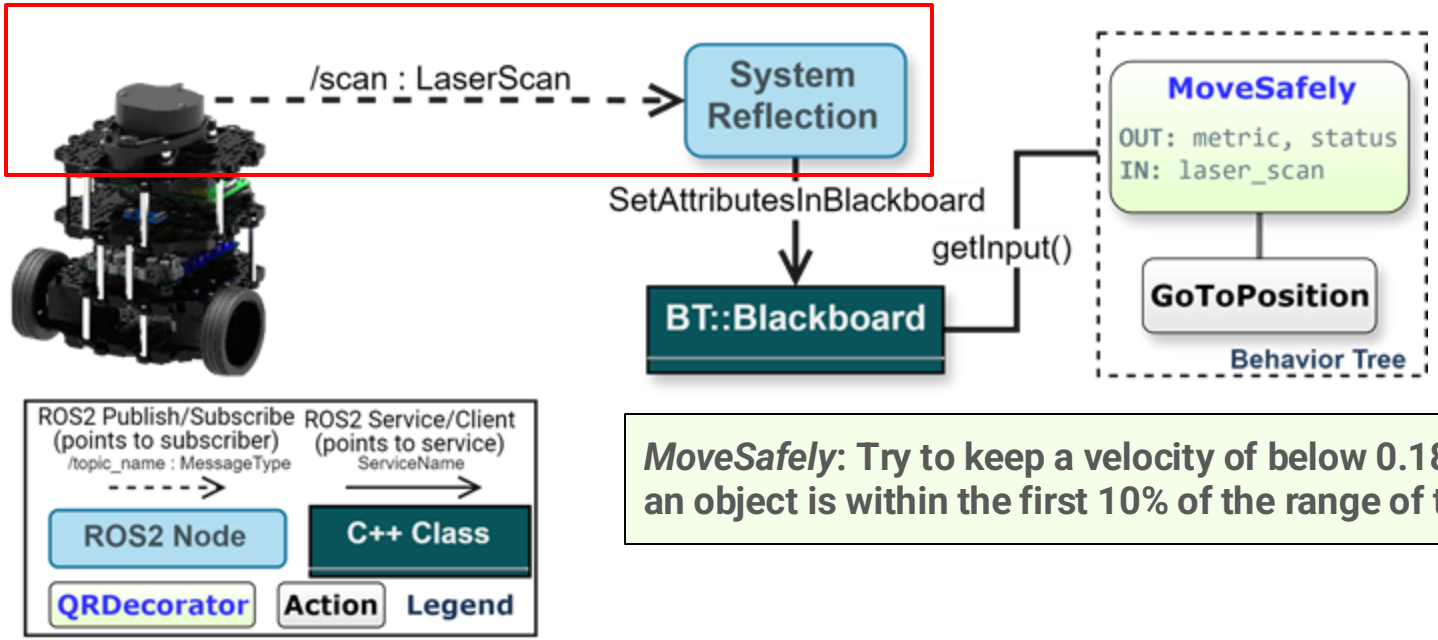
Diagram adapted from Macenski et al. "The marathon 2: A navigation system." *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

Quality Requirement Decorators (Q-Task)



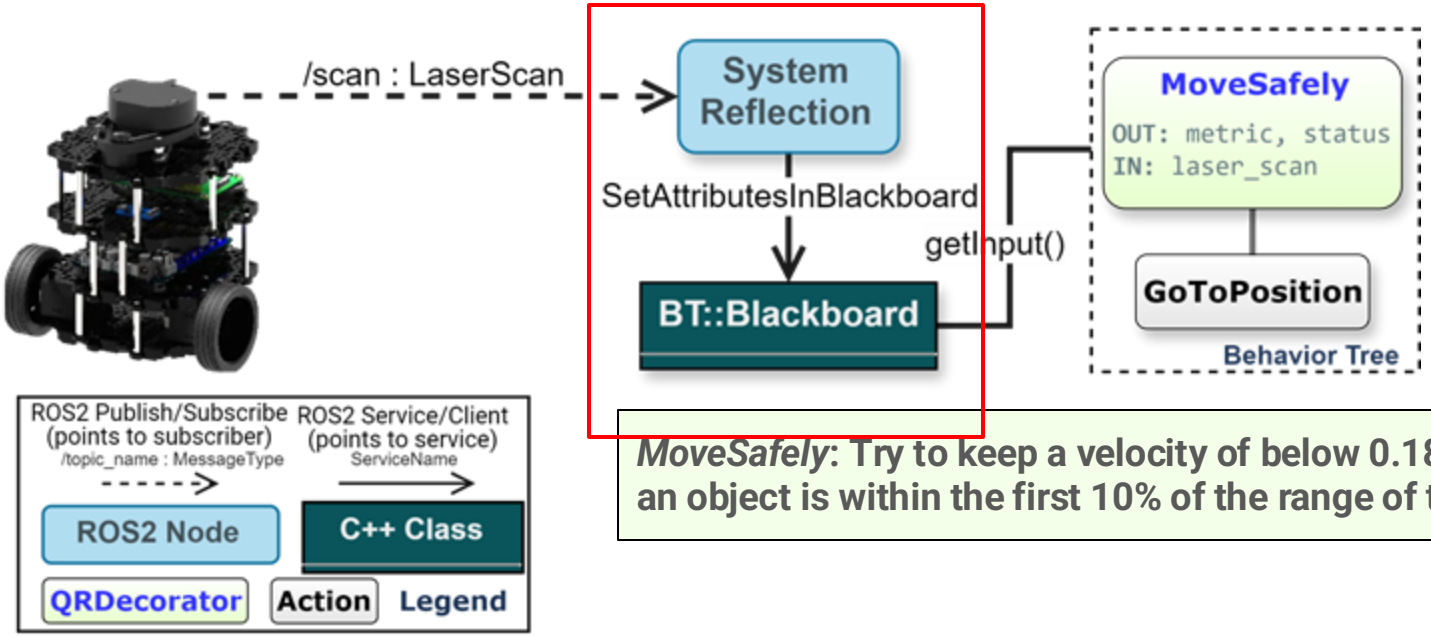
***MoveSafely*: Try to keep a velocity of below 0.18m/s while an object is within the first 10% of the range of the laser.**

Quality Requirement Decorators (Q-Task)



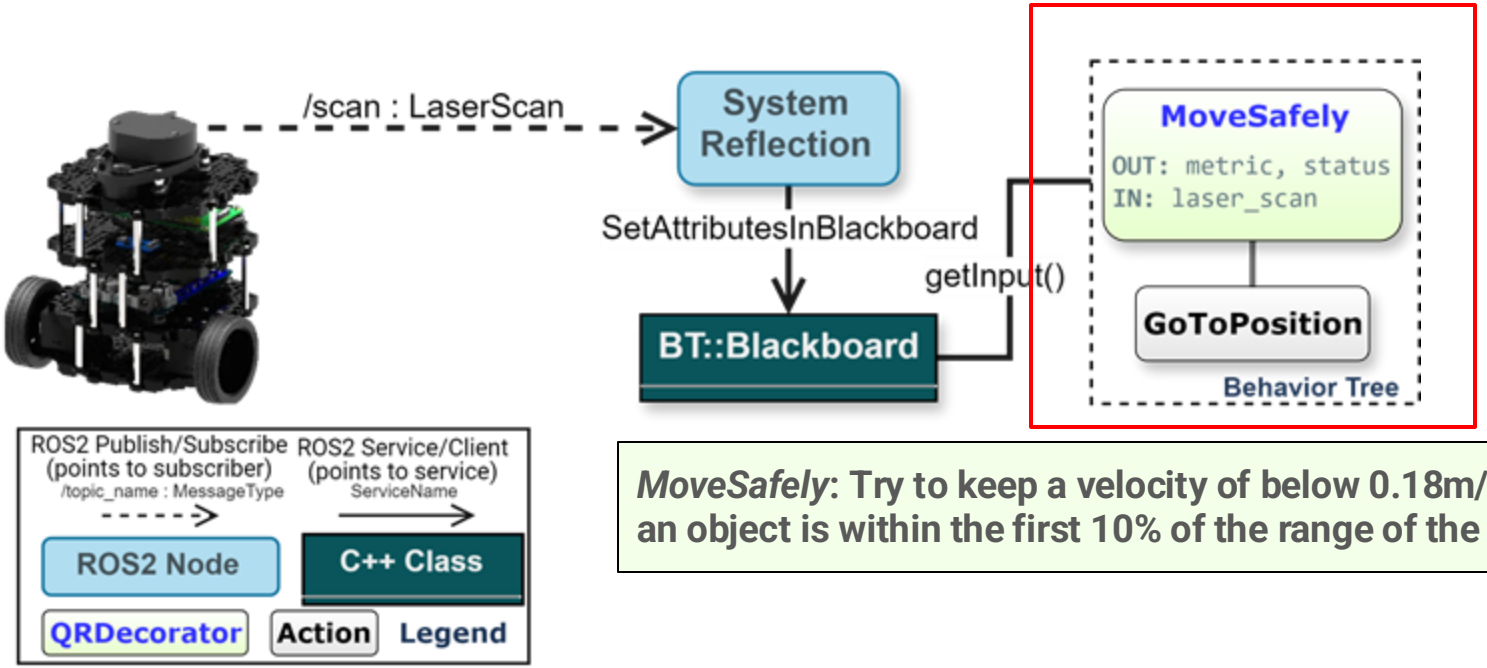
***MoveSafely*: Try to keep a velocity of below 0.18m/s while an object is within the first 10% of the range of the laser.**

Quality Requirement Decorators (Q-Task)



MoveSafely: **Try to keep a velocity of below 0.18m/s while an object is within the first 10% of the range of the laser.**

Quality Requirement Decorators (Q-Task)

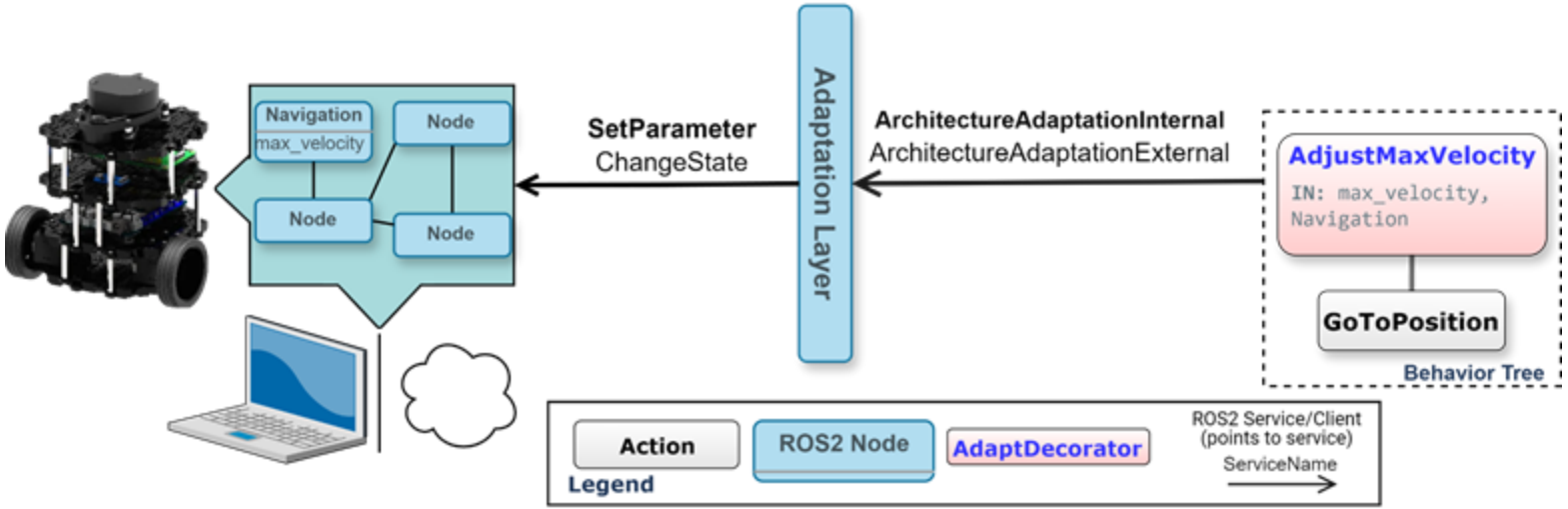


MoveSafely: Try to keep a velocity of below 0.18m/s while an object is within the first 10% of the range of the laser.

Software Architectural Adaptation (F-Arch)

We allow modifying the software architecture of a ROS2 system in several ways:

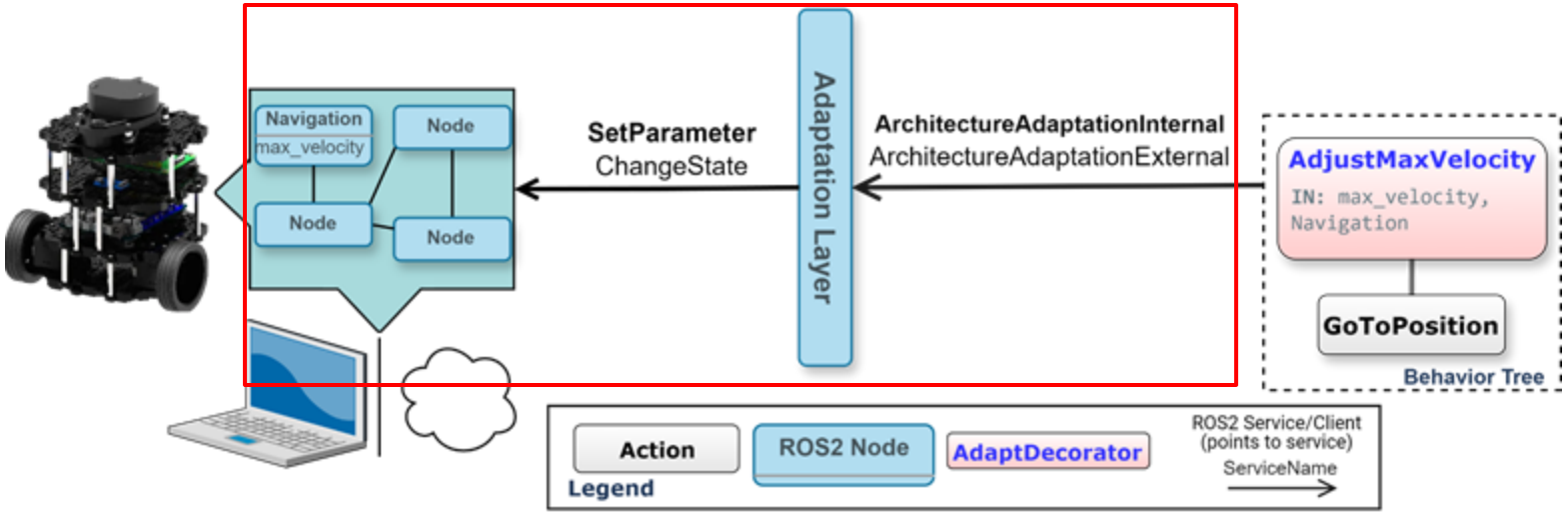
- Removal and/or addition of components
- A change in parameters of components
- A change in the interconnections of components



Software Architectural Adaptation (F-Arch)

We allow modifying the software architecture of a ROS2 system in several ways:

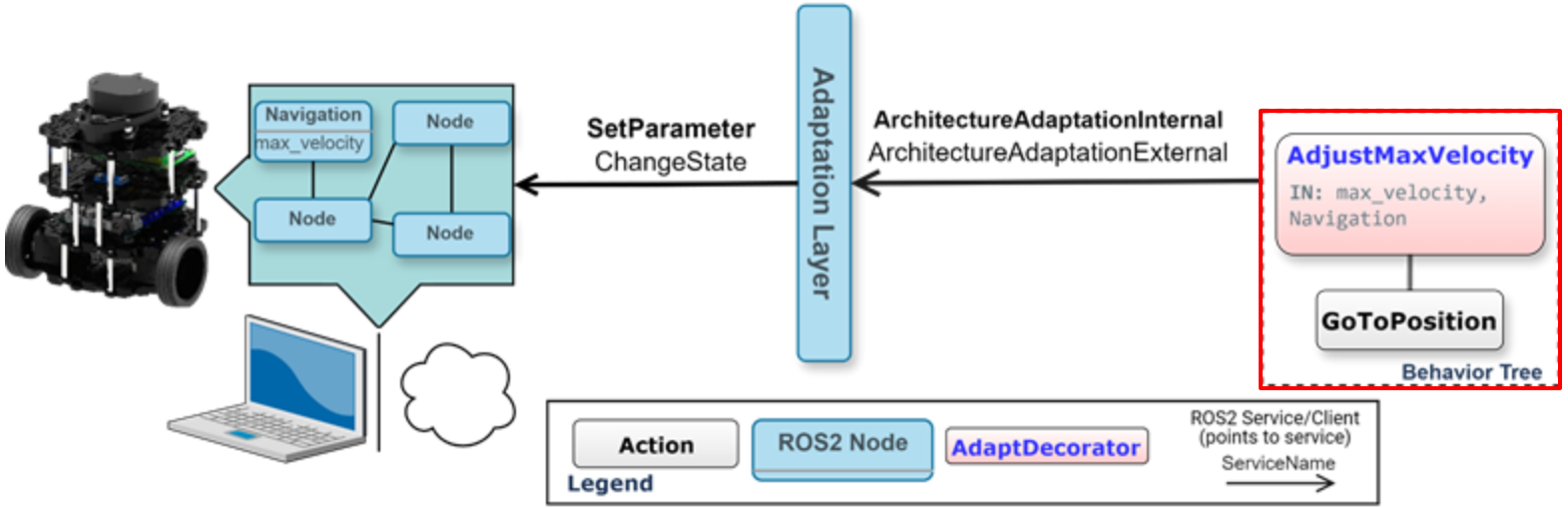
- Removal and/or addition of components
- A change in parameters of components
- A change in the interconnections of components



Software Architectural Adaptation (F-Arch)

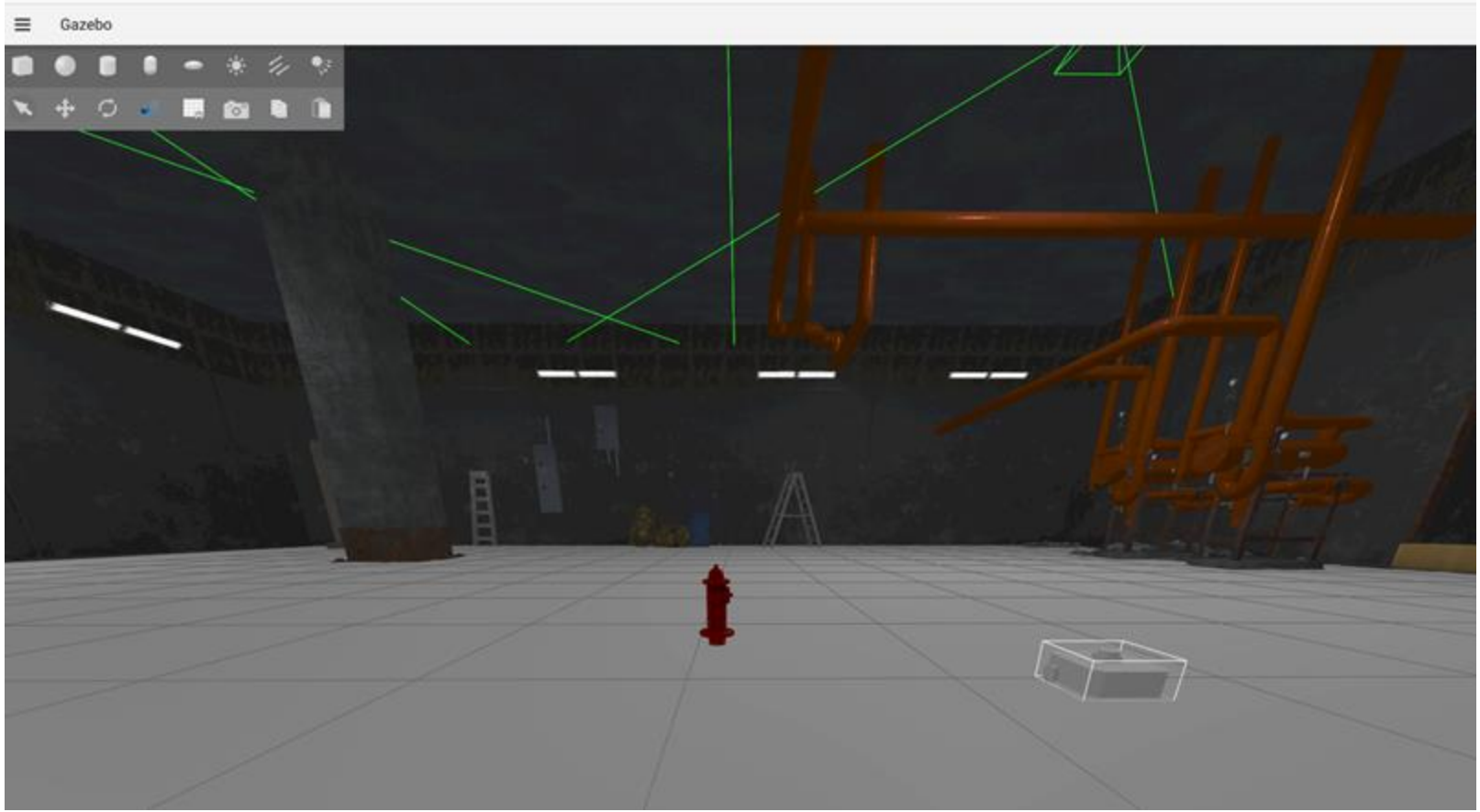
We allow modifying the software architecture of a ROS2 system in several ways:

- Removal and/or addition of components
- A change in parameters of components
- A change in the interconnections of components

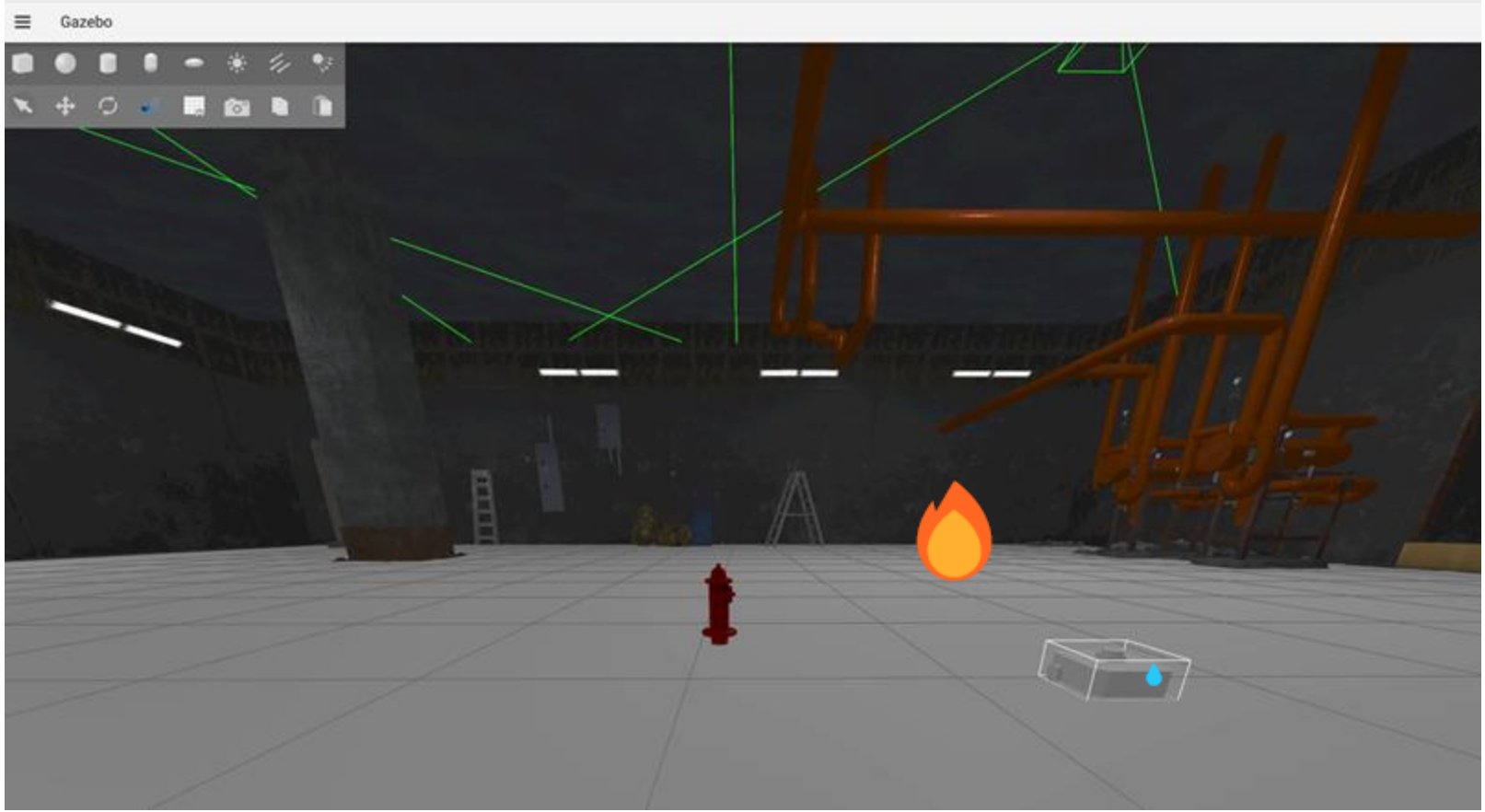


Let’s put it all together concretely.

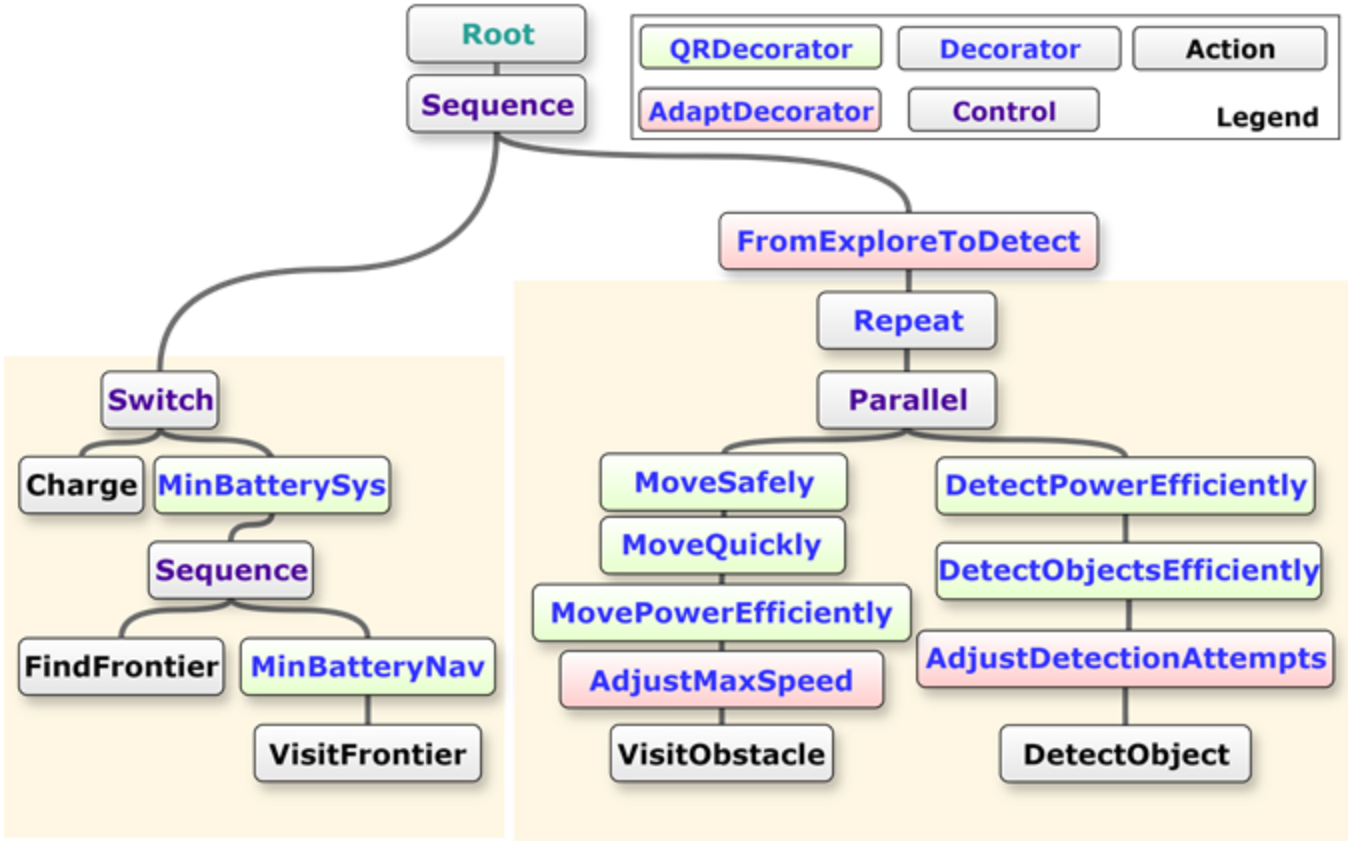
FROG



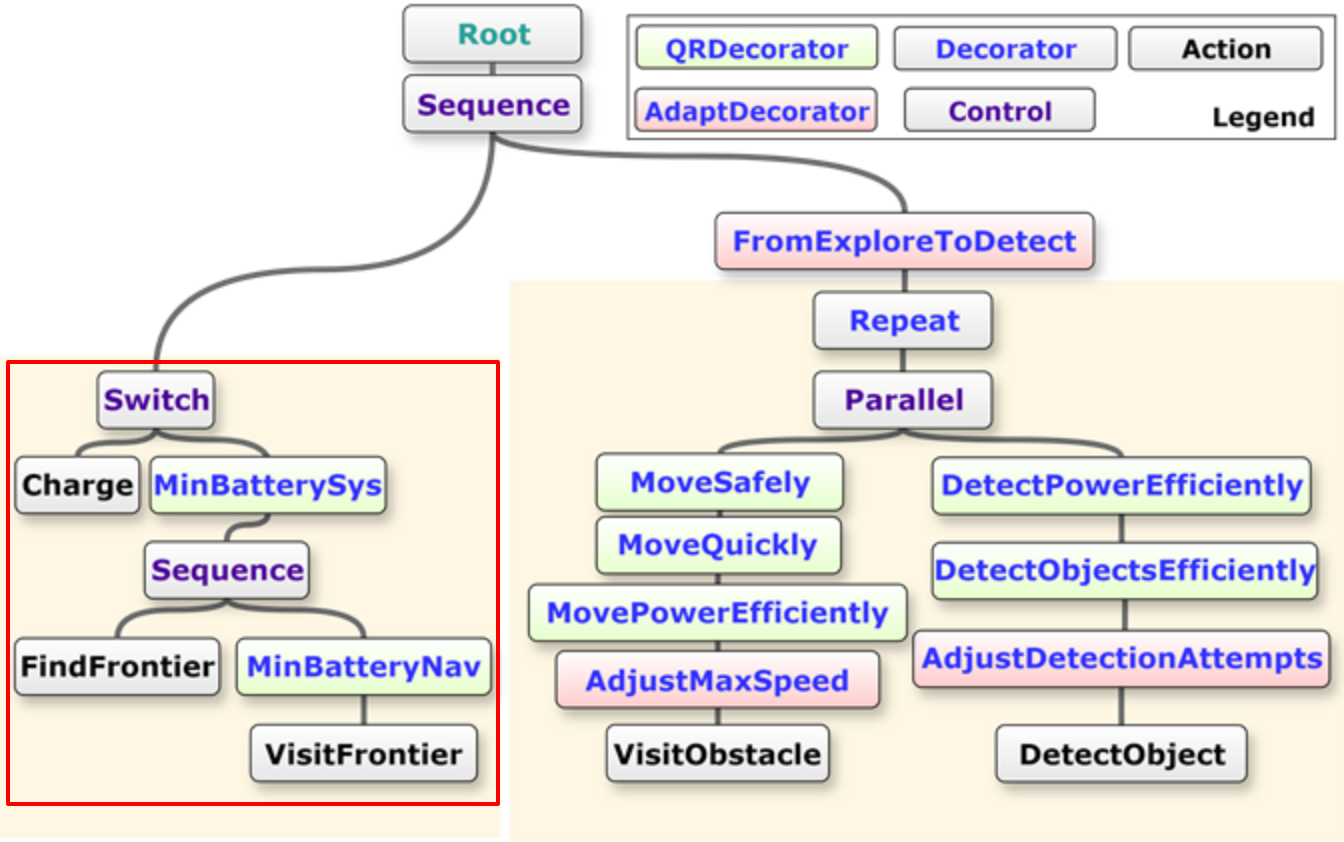
FROG and his newfound mission



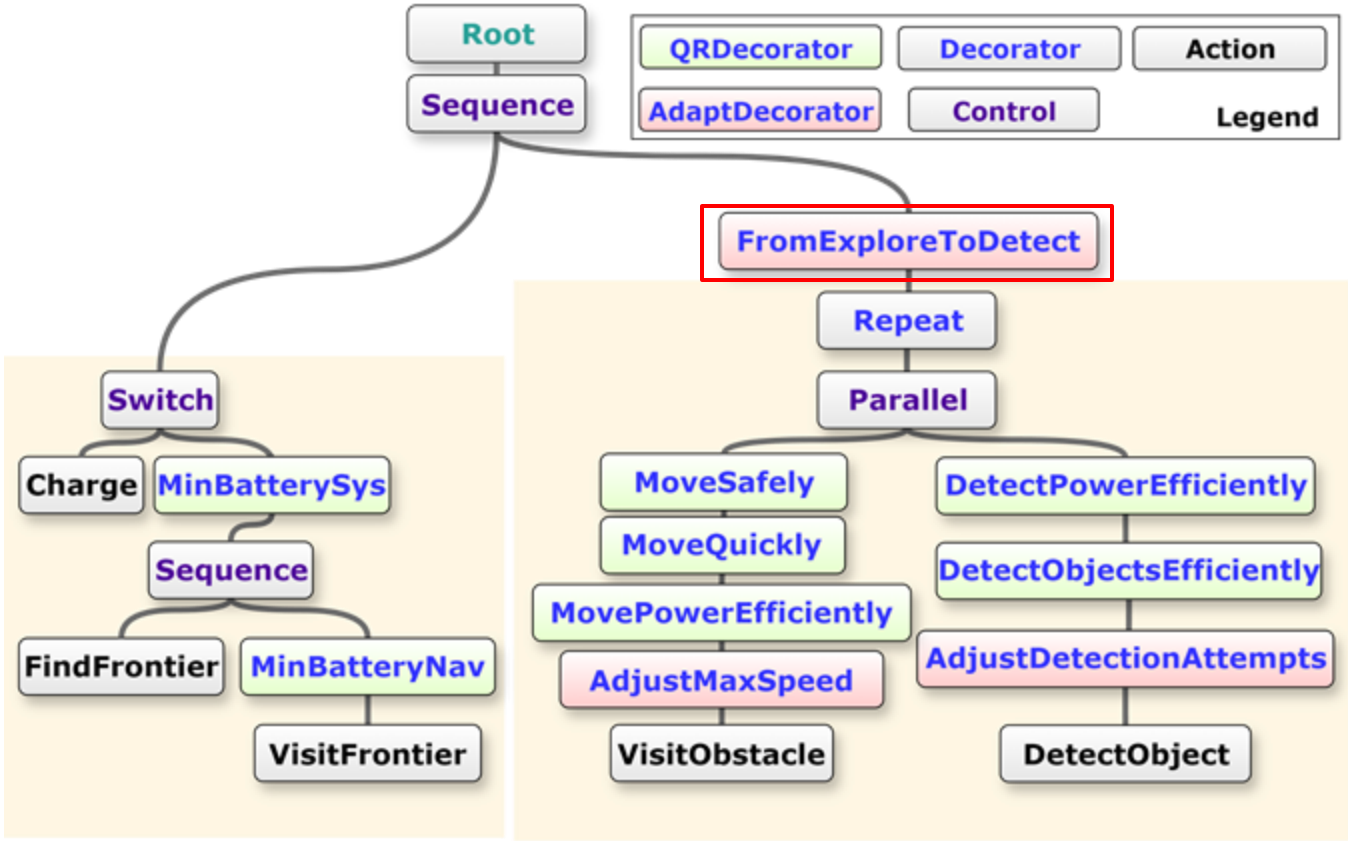
A tree for our mission



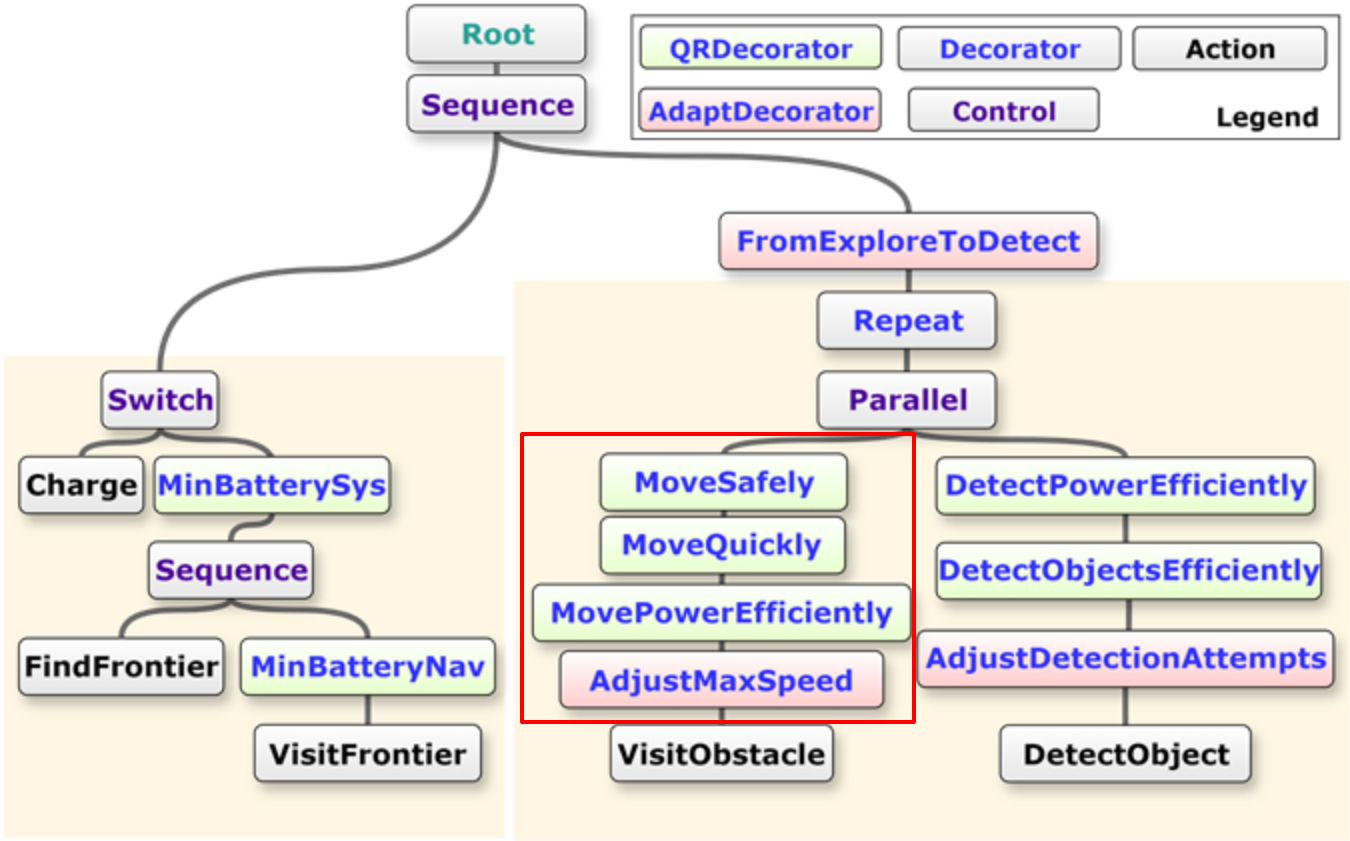
A tree for our mission



A tree for our mission



A tree for our mission

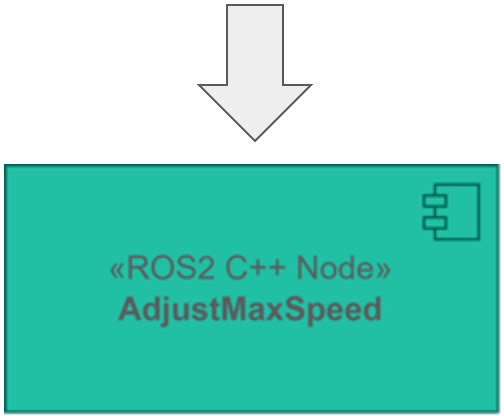


AdjustMaxSpeed

Behind the scenes

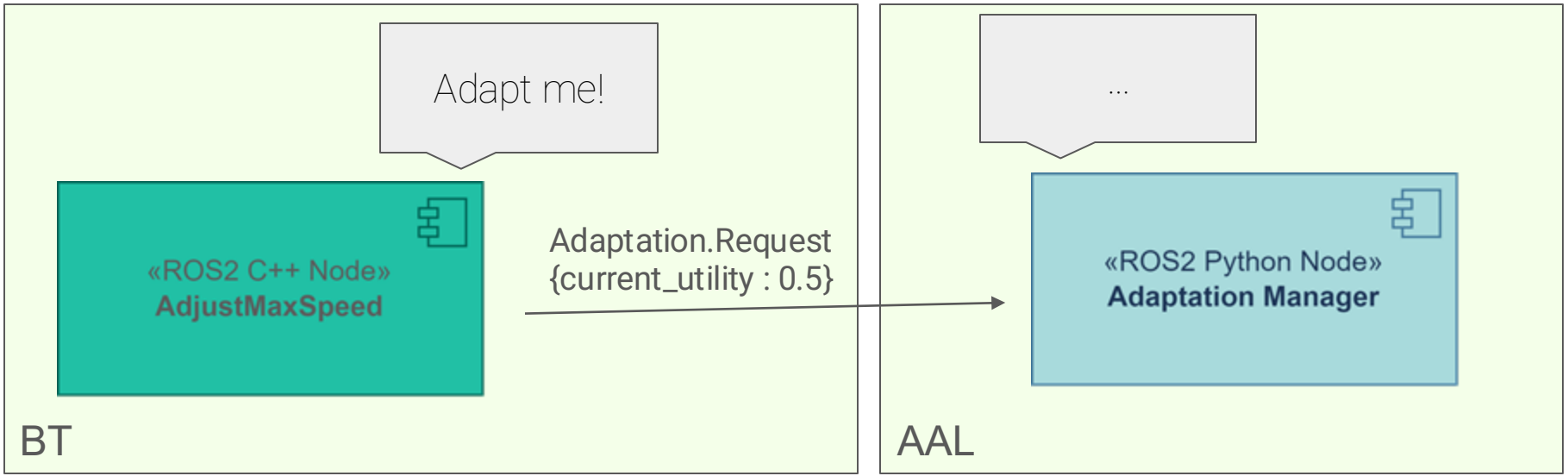
```
<AdjustMaxSpeed
adaptation_strategy="ucb_strategy" adaptation_options="0.10;0.18;0.26"
adaptation_subject="max_velocity" subject_location="velocity_smoother">
```

XML



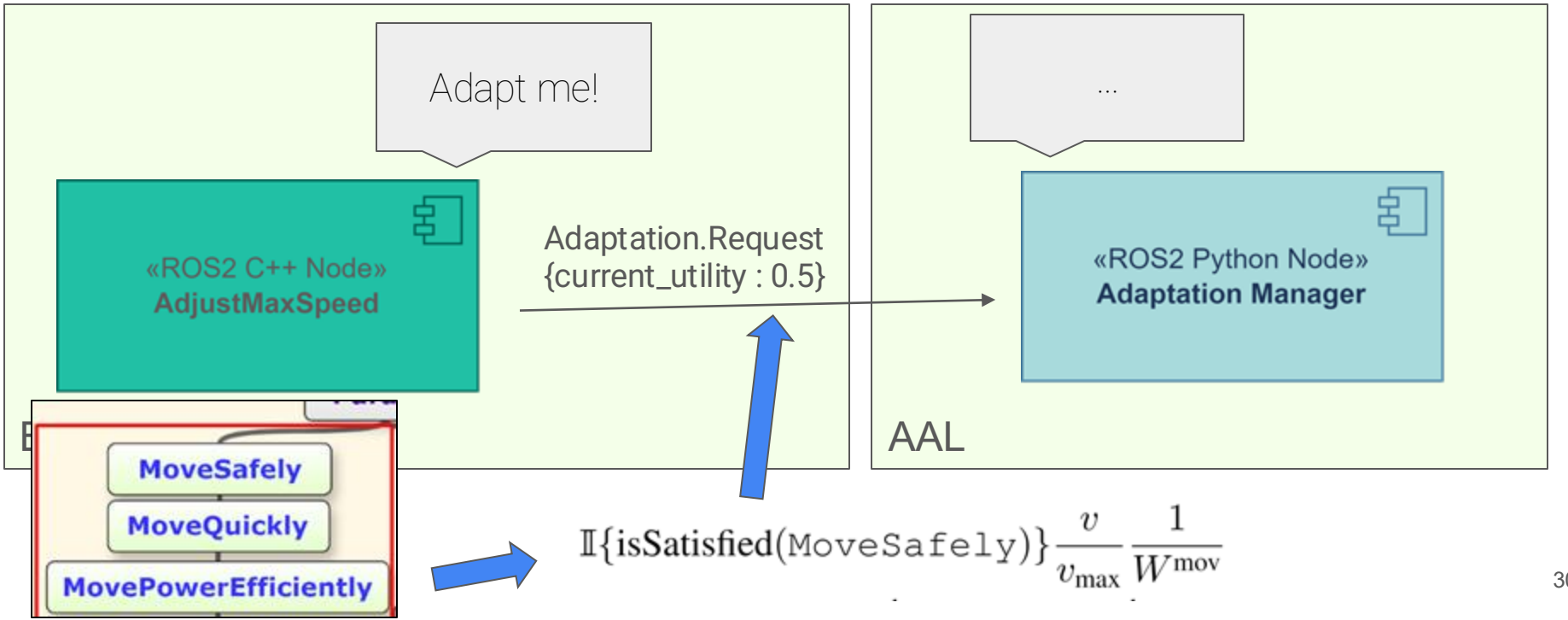
AdjustMaxSpeed

Behind the scenes



AdjustMaxSpeed

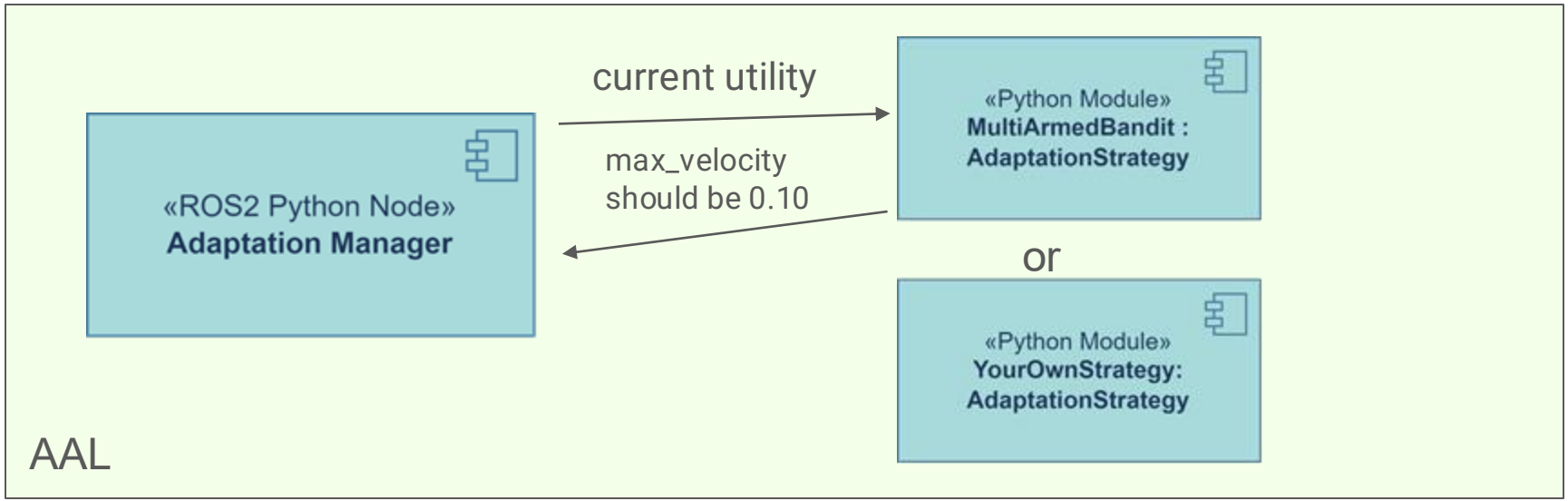
Behind the scenes



AdjustMaxSpeed

Behind the scenes

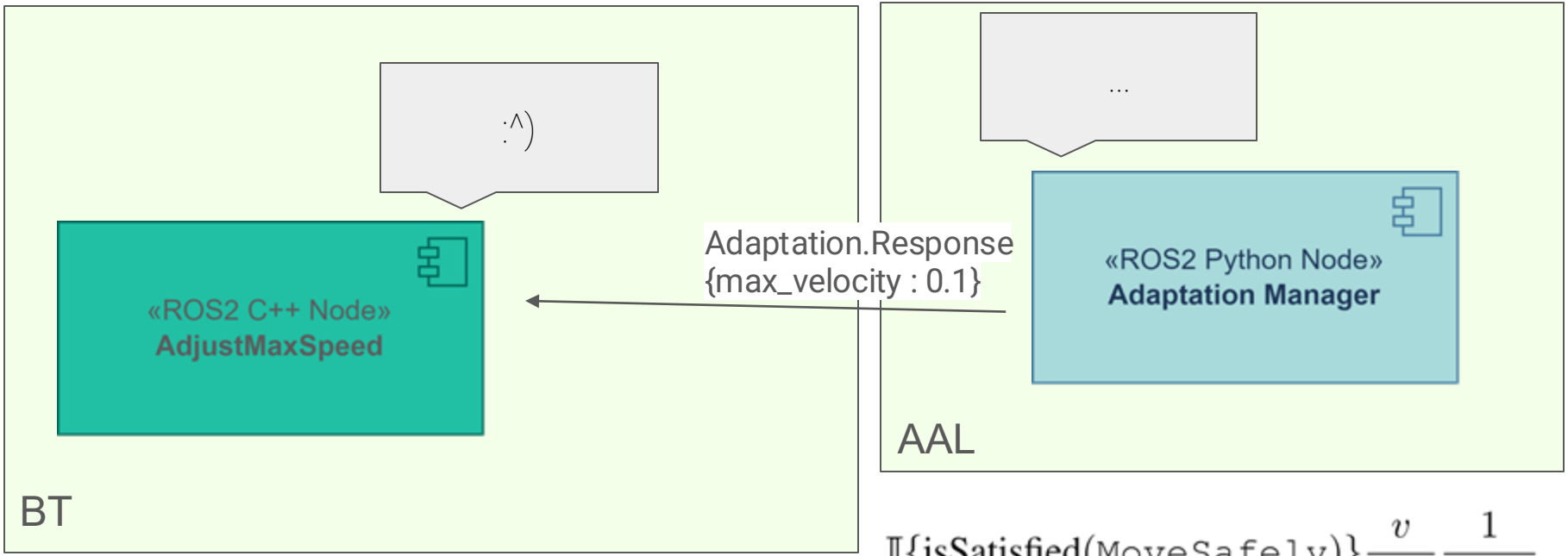
Two ROS2 Nodes walk into a bar…



AdjustMaxSpeed

Behind the scenes

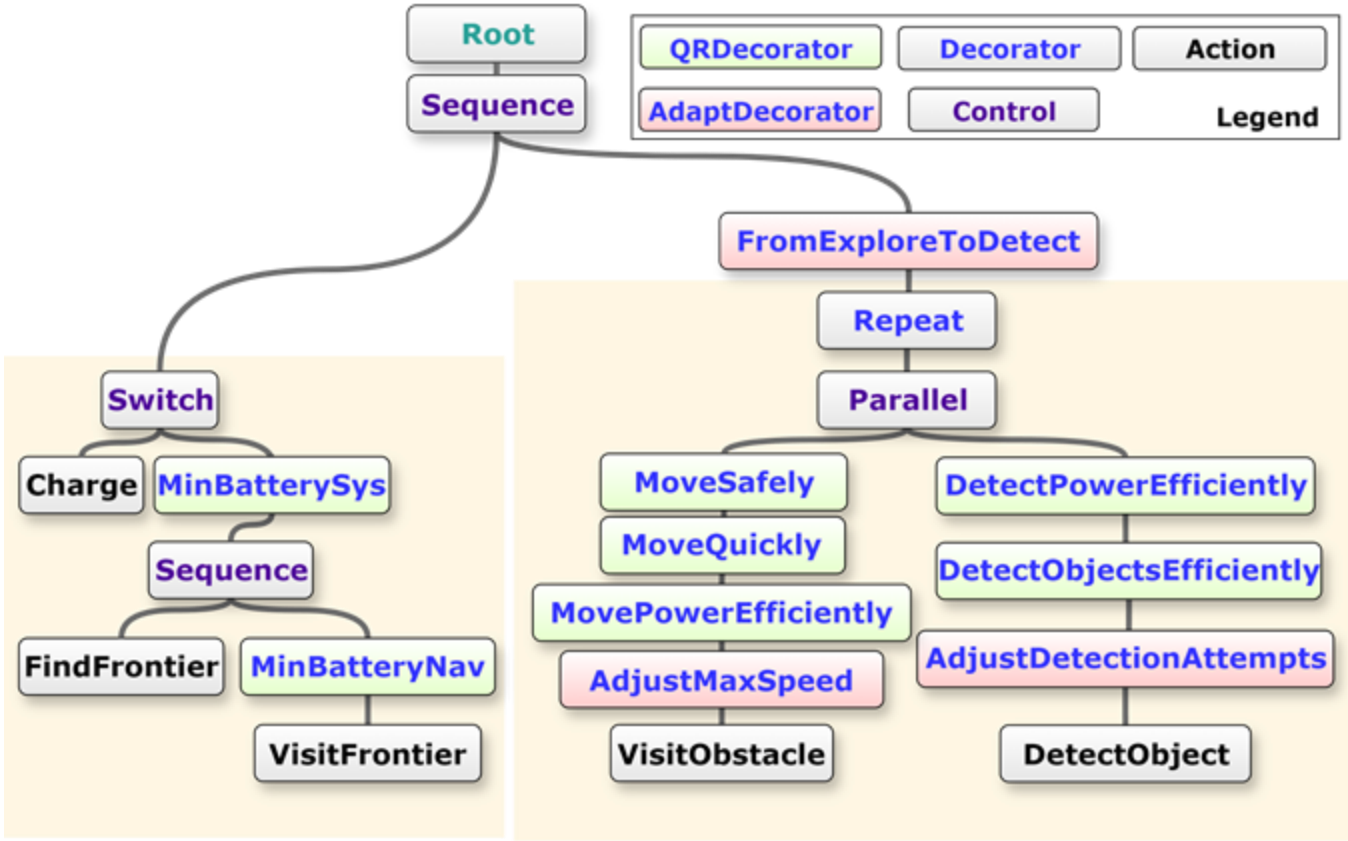
Two ROS2 Nodes walk into a bar...



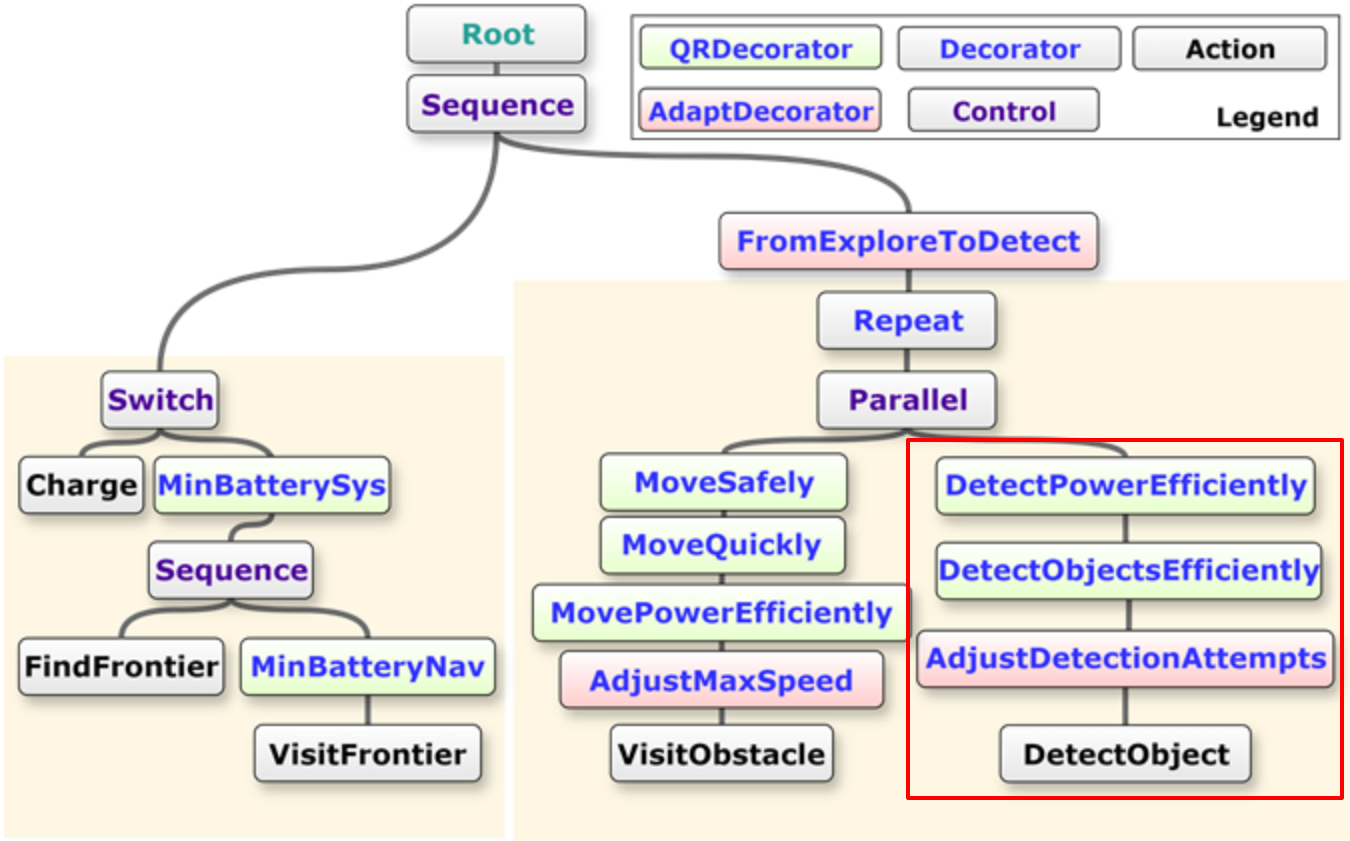
$$\mathbb{I}\{\text{isSatisfied}(\texttt{MoveSafely})\}\frac{v}{v_{\max}}\frac{1}{W^{\text{mov}}}$$

Hopefully things can often be simpler.

A tree for our mission



A tree for our mission



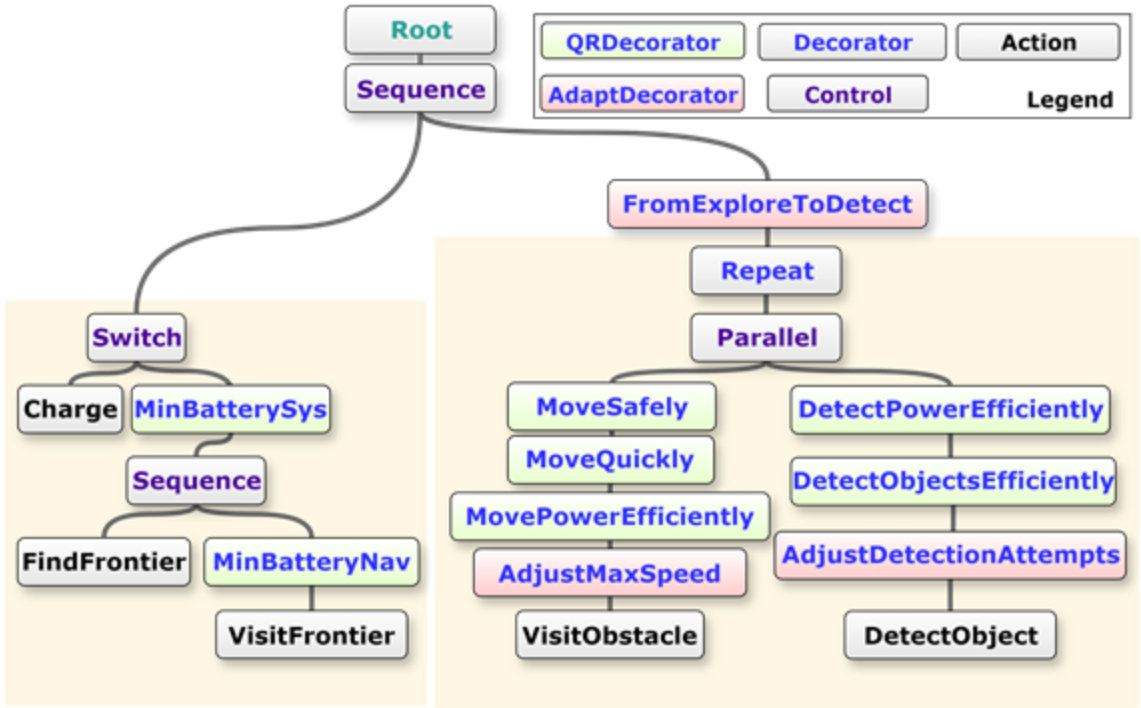
Behind the scenes

```
virtual bool evaluate_condition() override
{
    if(remaining_power_budget < 0.0) {
      return change_camera_feed(ALTERNATE_CAMERA);
    }
    if((current_darkness) > 0.70) {
        if(remaining_power_budget >= power_to_be_used &&
remaining_power_budget > 0.0) {
          return increase_picture_rate() || change_camera_feed(ROBOT_CAMERA);
        }
        else {
          return change_camera_feed(ALT_CAMERA_TOPIC);
        }
    }
    else {
      return (reduce_picture_rate() || change_camera_feed(ROBOT_CAMERA));
    }
}
```

Behind the scenes

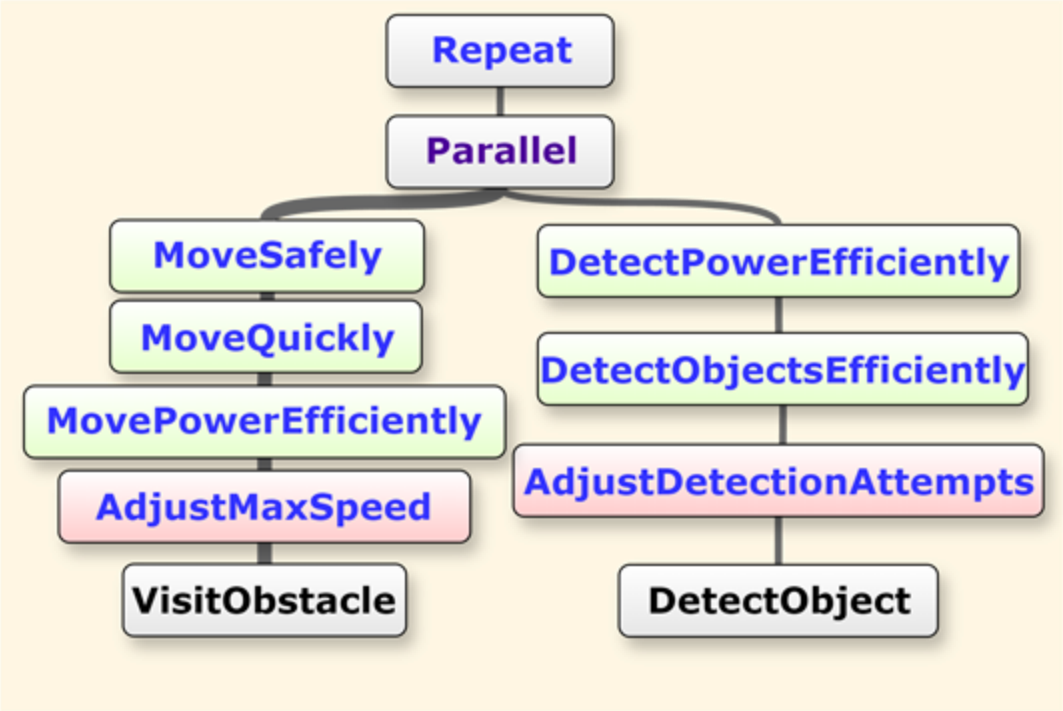
```
virtual bool evaluate_condition() override
{
    if(remaining_power_budget < 0.0) {
      return change_camera_feed(ALTERNATE_CAMERA);
    }
    if((current_darkness) > 0.70) {
        if(remaining_power_budget >= power_to_be_used &&
remaining_power_budget > 0.0) {
          return increase_picture_rate() || change_camera_feed(ROBOT_CAMERA);
        }
        else {
          return change_camera_feed(ALTERNATE_CAMERA);
        }
    }
    else {
      return (reduce_picture_rate() || change_camera_feed(ROBOT_CAMERA));
    }
}
```

Something Interesting, and Future Work



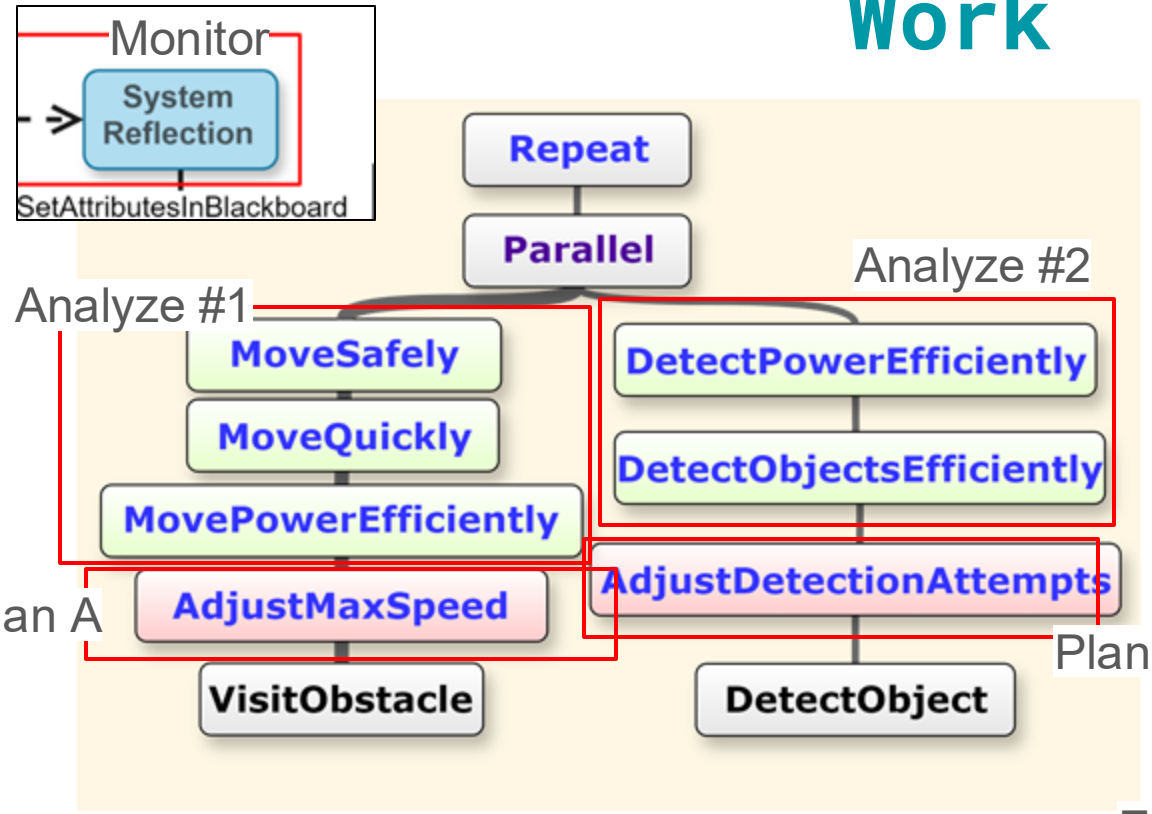
There is no tree.

Something Interesting, and Future Work

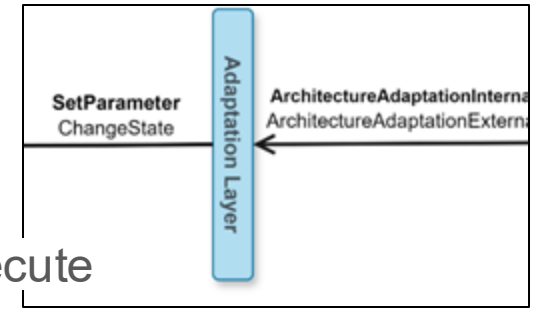


There is no tree.

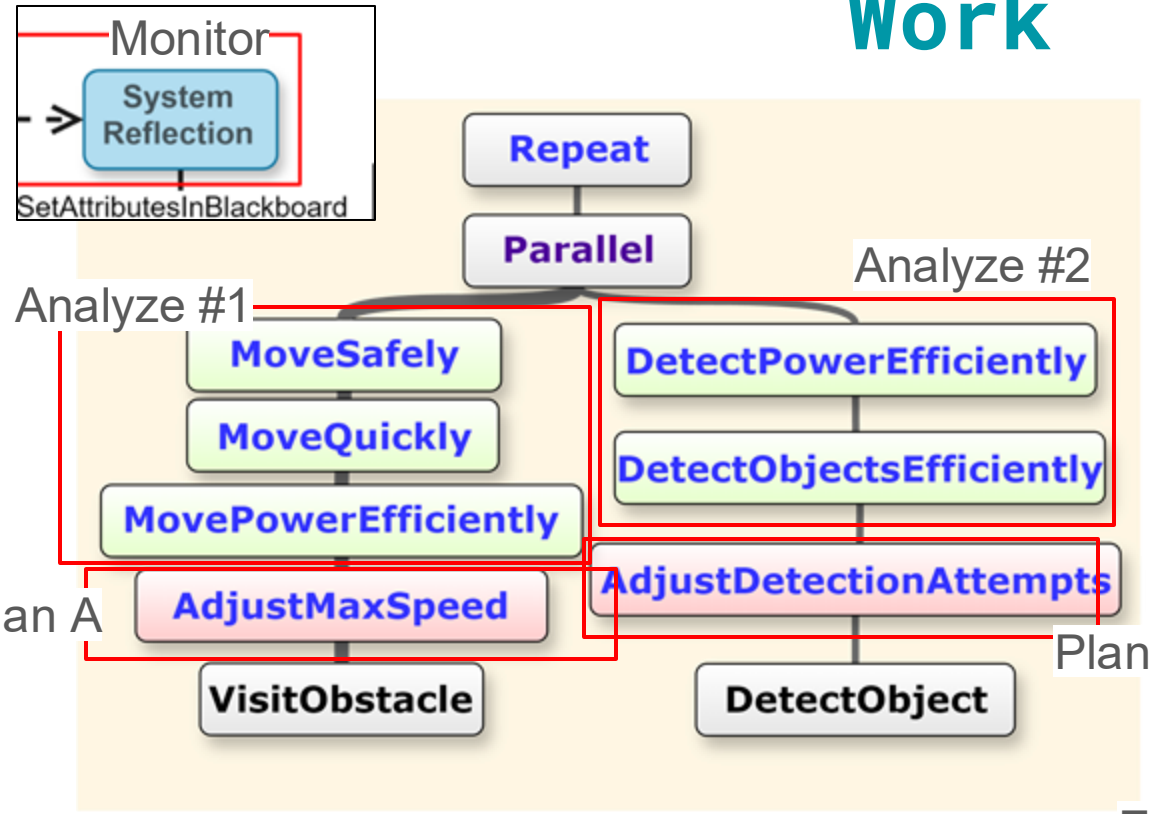
Something Interesting, and Future Work



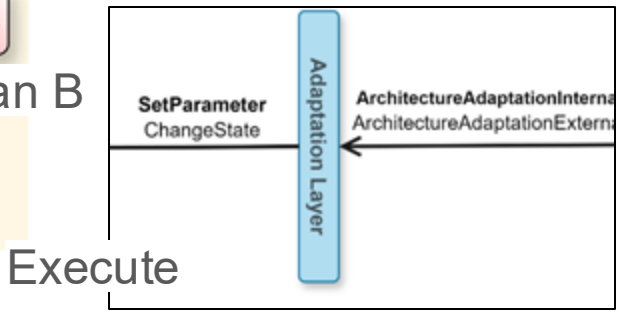
There is no tree.



Something Interesting, and Future Work



Each branch of the tree entails its own realization of a MAPE-K loop. The management of these separate loops can be expressed through the logic already provided by BTs.



Something Interesting, and Future Work

- Our implementation of QR metrics is quite rudimentary.
- More types of architectural adaptation are still to be implemented for ROS2 e.g. redeployment of components.

We have an artifact