UML for Robotic Systems

Ansgar Radermacher / Asma Smaoui
ansgar.radermacher@cea.fr

Acknowledgments – contains material from Matteo Morelli and https://www.behaviortree.dev/

Agenda

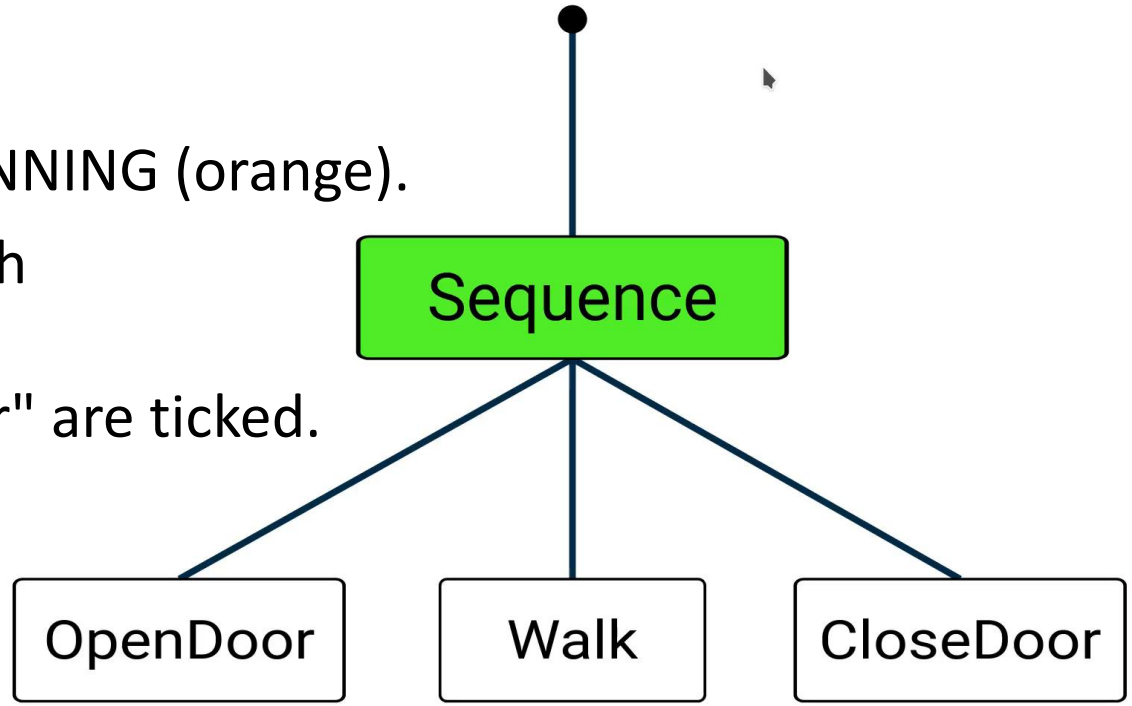
- Behavior Trees
- Papyrus for Robotics

Behavior Trees

- A “**tick**" is sent to the root node. It propagates through the tree until it reaches a leaf node.
- A **non-leaf node** is responsible for propagating the tick to its children. Node types may have different rules on how to do that.
- A node has three different states
  - SUCCESS – has run successfully
  - FAILURE – has run, returned fail
  - RUNNING – is still running
- **Leaf nodes** execute behavior interacting with the rest of the system

Behavior Trees

- Minimal example, based on “Sequence” node
- Execute children (from left to right),
  return SUCCESS if **all** Succeed
- The first tick sets the Sequence node to RUNNING (orange).
- Sequence ticks first child, "OpenDoor“ which
  returns SUCCESS.
  → second child "Walk" and later "CloseDoor" are ticked.
- Completion of last child
  → entire sequence switches from
  RUNNING to SUCCESS.

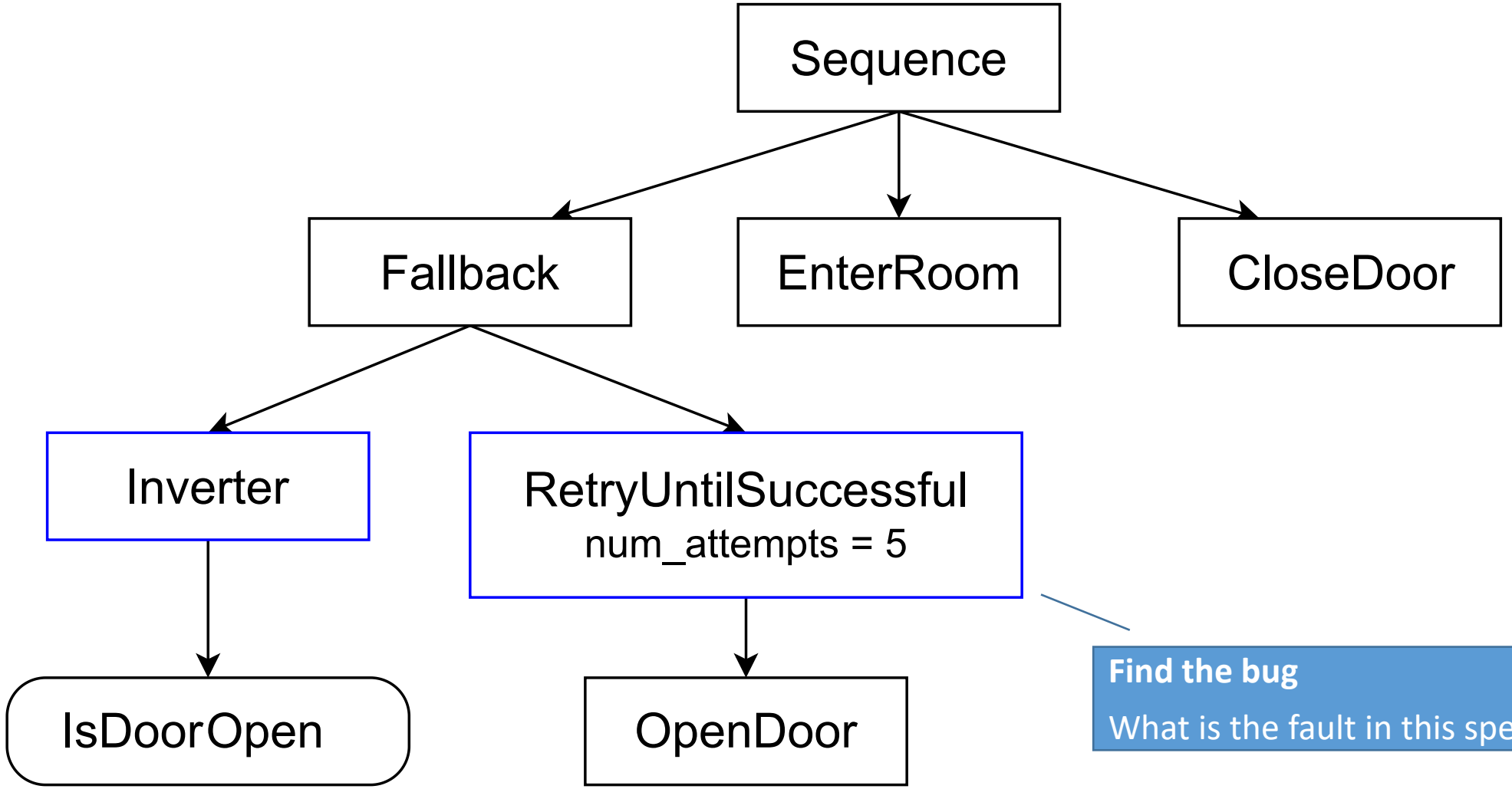


source https://www.behaviortree.dev/docs/learn-the-basics/BT_basics

Behavior Trees – Node types

Node type	Children Count	Notes
ControlNode	1...N	Usually, ticks a child based on the result of its siblings or/and its own state.
DecoratorNode	1	Among other things, it may alter the result of its child or tick it multiple times.
ConditionNode	0	Should not alter the system. Shall not return RUNNING.
ActionNode	0	This is the node that *“does something”* (typically a leaf node)

Behavior Trees – Find the bug

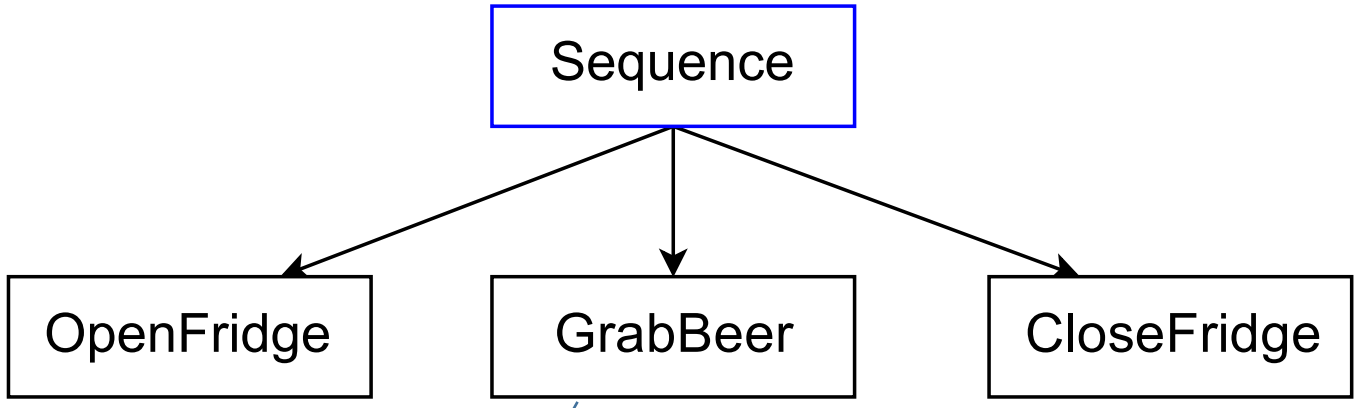


Find the bug

What is the fault in this specification?

source https://www.behaviortree.dev/docs/learn-the-basics/BT_basics

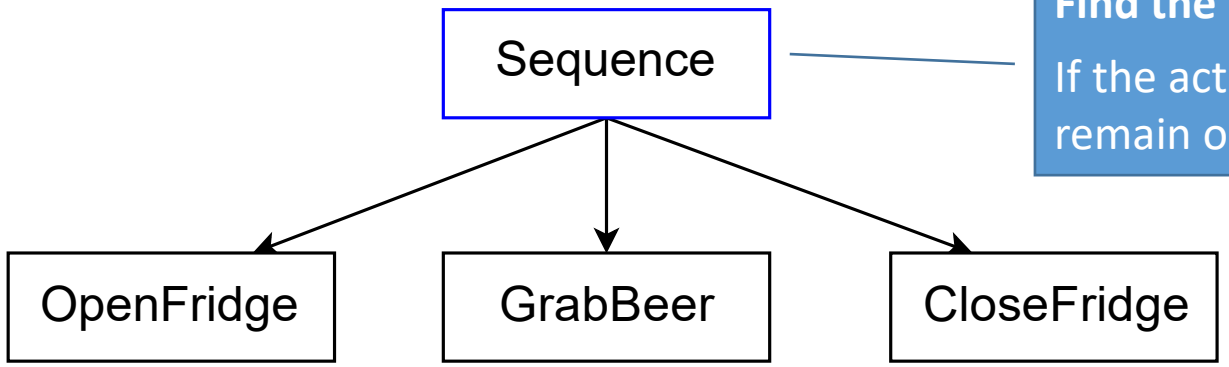
Behavior Trees – Find the bug continued



Find the bug

If the action **GrabBeer** fails, the door of the fridge will remain open since the last action **CloseFridge** is skipped.

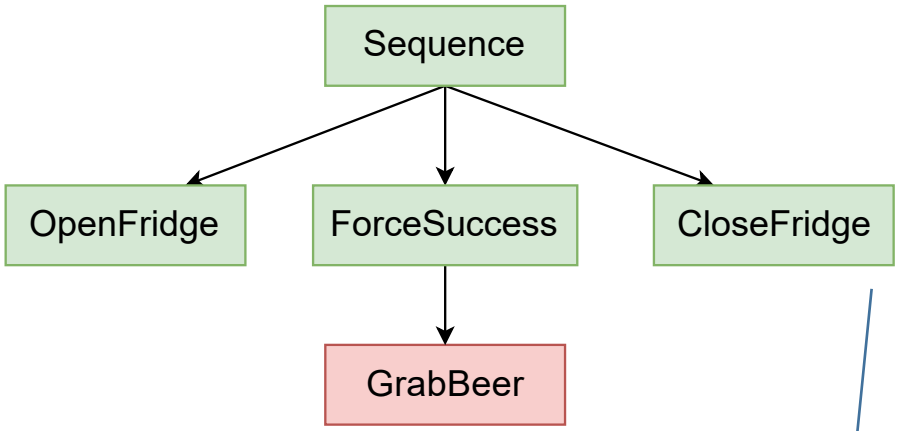
Behavior Trees – Find the bug continued

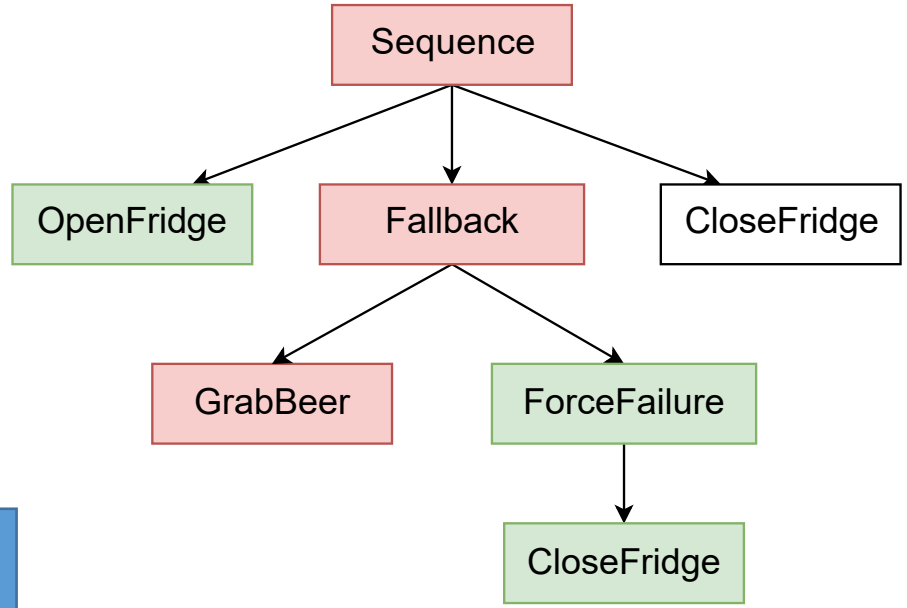


Find the bug

If the action **GrabBeer** fails, the door of the fridge will remain open since the last action **CloseFridge** is skipped.

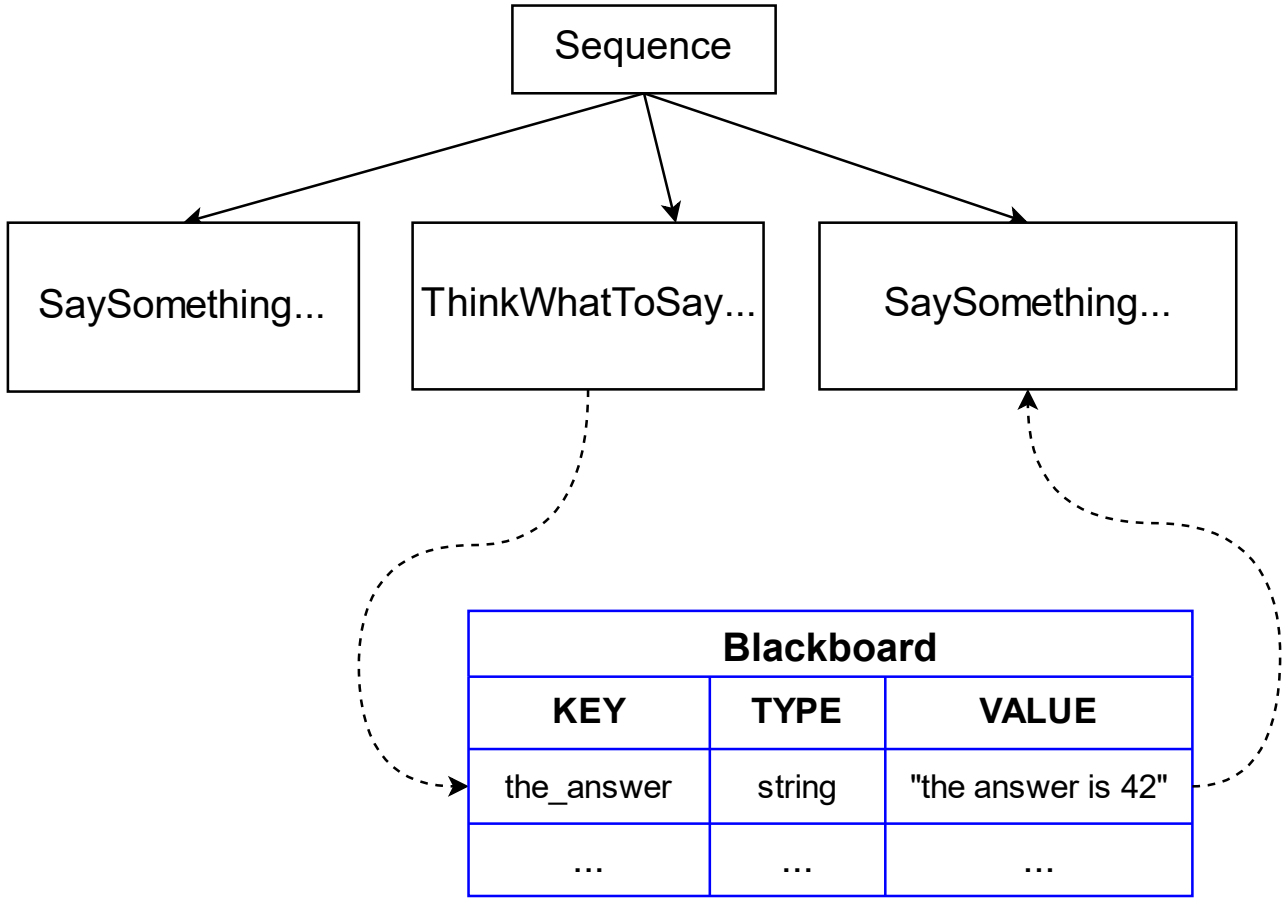
Source
https://www.behaviortree.dev/docs/learn-the-basics/BT_basics





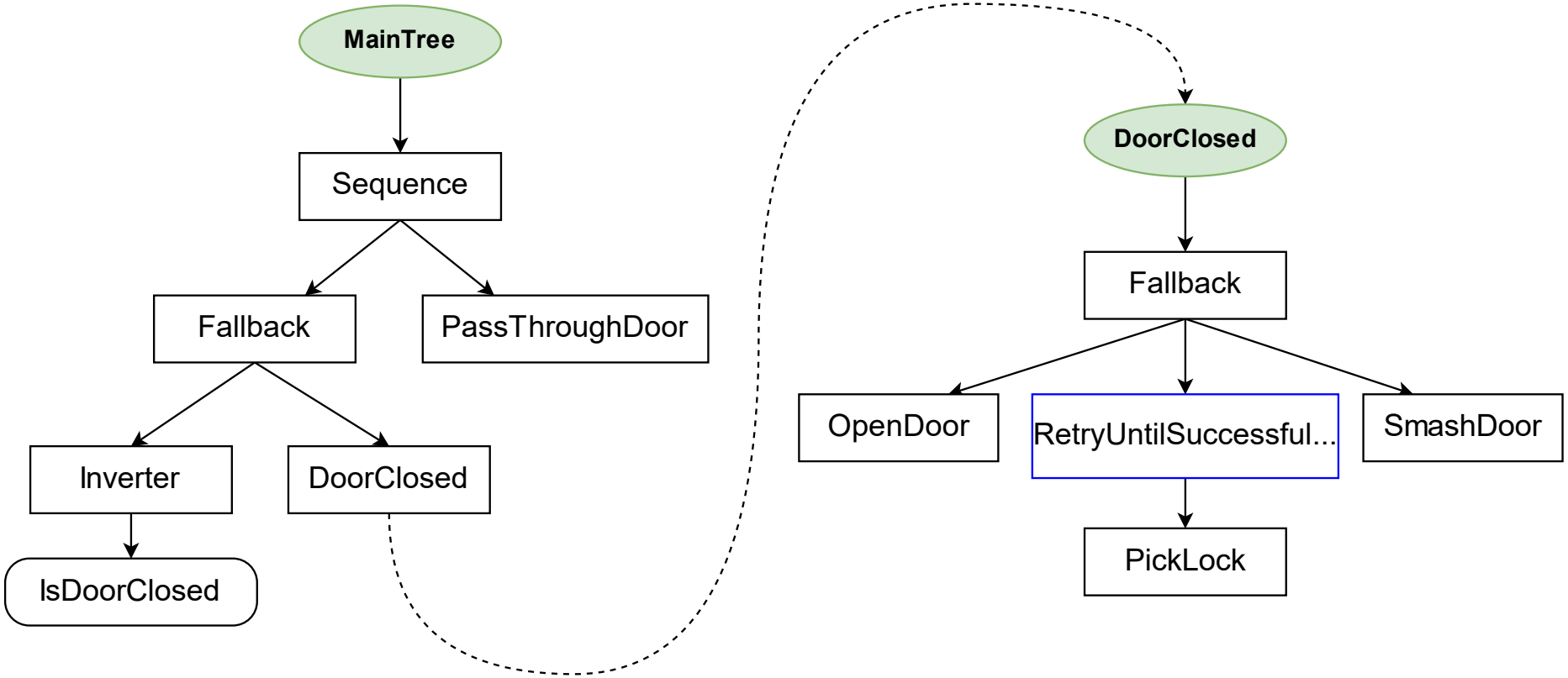
Two possible fixes, not trivial any more. Can you tell the difference between the two options?

Blackboard and ports



Key/value pair

Subtrees – Handle bigger trees



Behavior Trees – Summary

- Specify high-level behaviors
- Slightly similar to state machines, but more reactive
- Used in Gaming and Robotics

Papyrus for Robotics – Domain specific modeling

1. **Developed in the context of a European project**
2. **Aligned with the middleware ROS 2**
3. **Behavior Tree Support**
   - **Transform Behavior Tree into XML format**
   - **Include BT execution engine in deployed code**

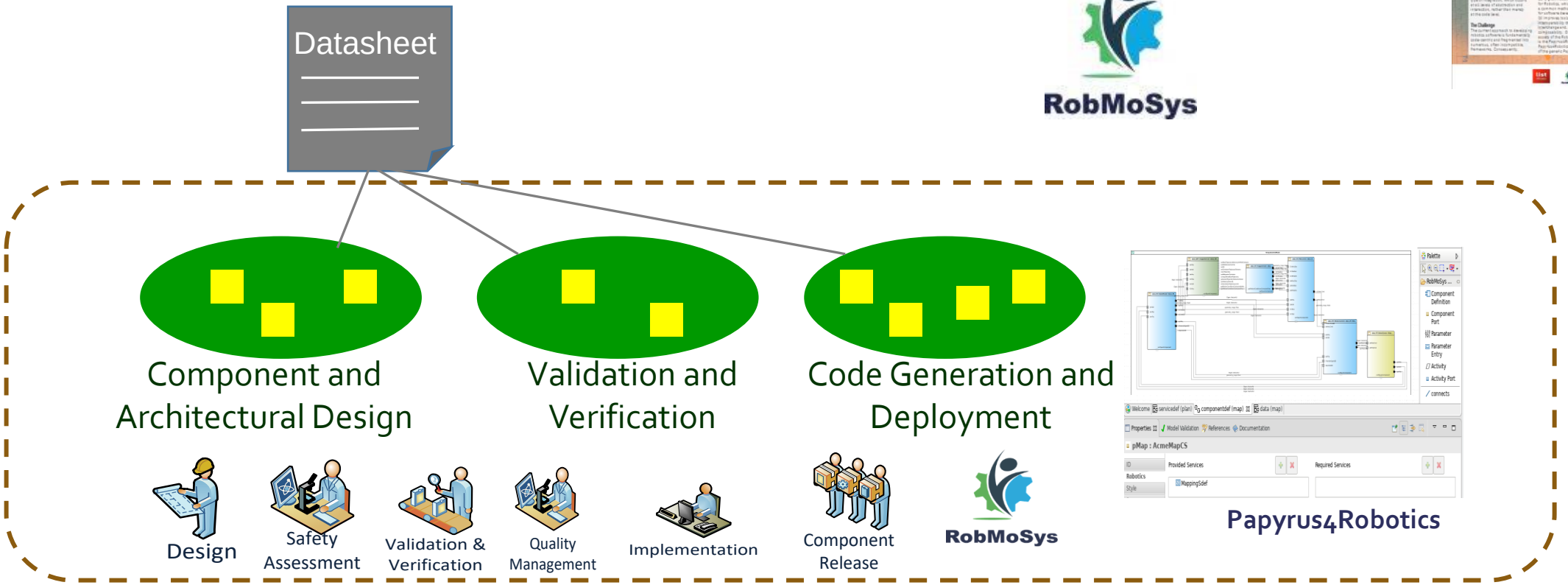
Papyrus for Robotics

Customization for the robotics domain

https://eclipse.org/papyrus/components/robotics

Youtube channel: https://www.youtube.com/c/PapyrusEclipseUML

RobMoSys aligned – European project https://robmosys.eu/



The Robot Operating System (ROS)

- https://index.ros.org/doc/ros2/
- Set of software libraries and tools for building robot applications.
- Wide range (drivers, algorithms, visualization tools …), open source.
- ROS 1 was started in 2007
- ROS 2 – reduced footprint, based on DDS middleware, better real-time support
- microROS for resource-constrained systems

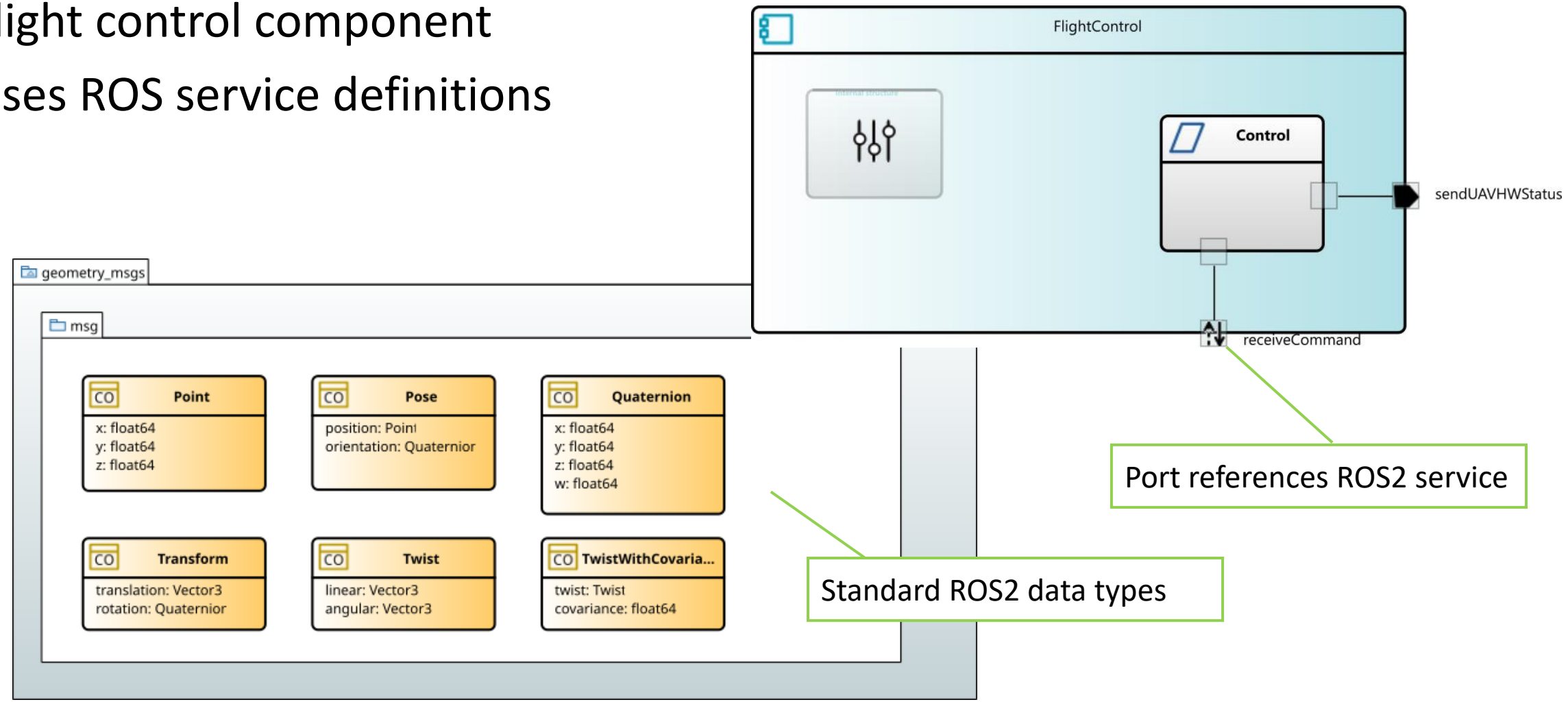
⇨ Growing number of companies migrating to ROS2
⇨ Papyrus for Robotics supports code generation for ROS2

ROS-MDD

ROSin

Examples in Papyrus for Robotics

- Flight control component
- Uses ROS service definitions

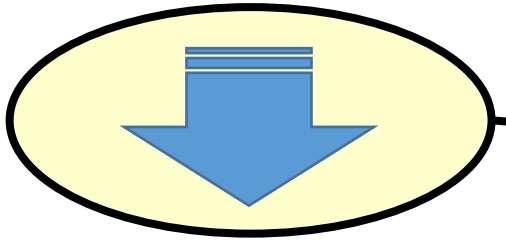


Component code-generation

Component Developer

Component Definition

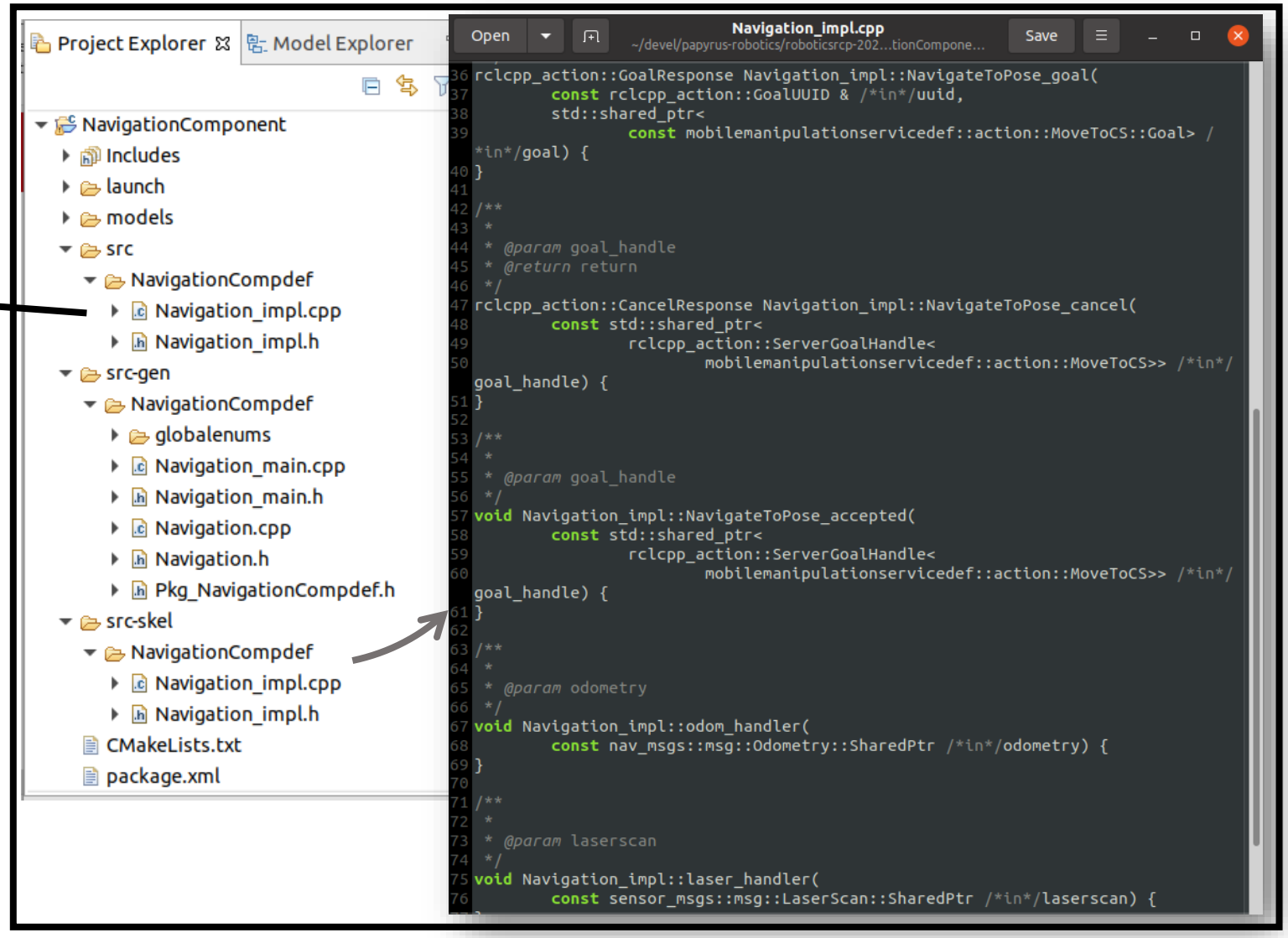
Programmer



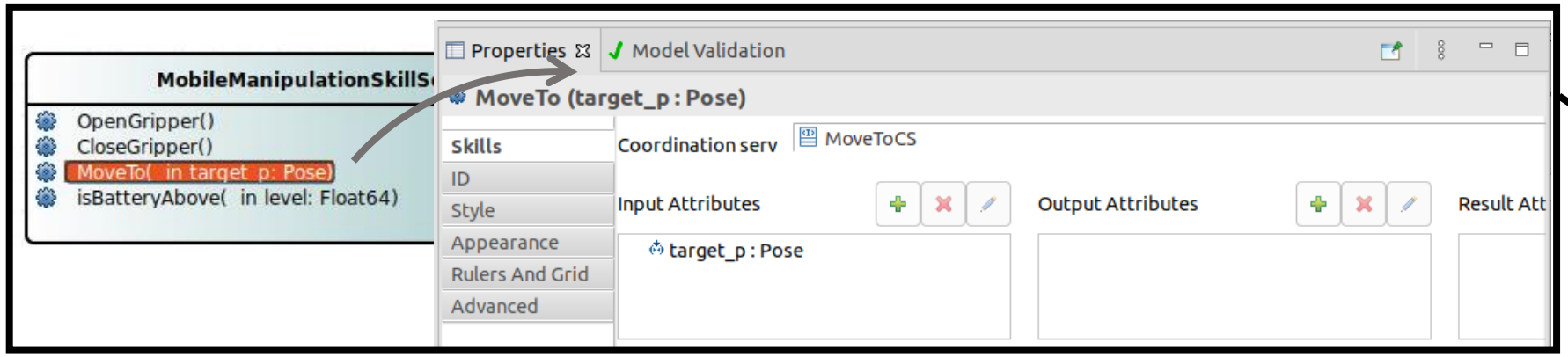
code-generation

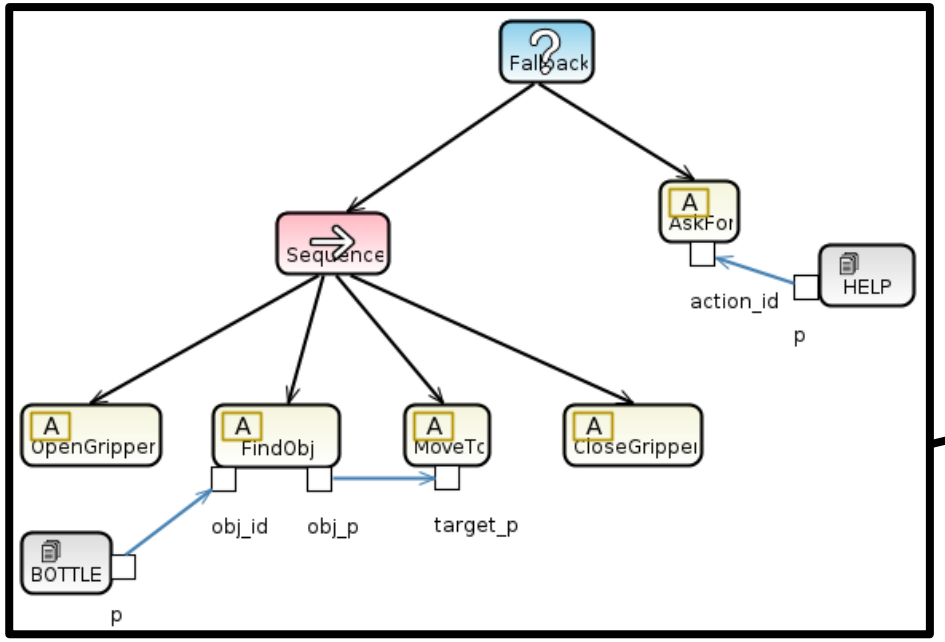
binds data from component interface to internals (algorithms, etc.)

- `const auto goal = goal_handle->get_goal();`
- `res = do_navigate_algo ( goal->position.x, goal->position.y, … );`
- `if (res == 1) { // algo OK } else {algo KO }`



Skills and behaviors





Skill Definition

Skills Designer

Tier 2

Tier 3

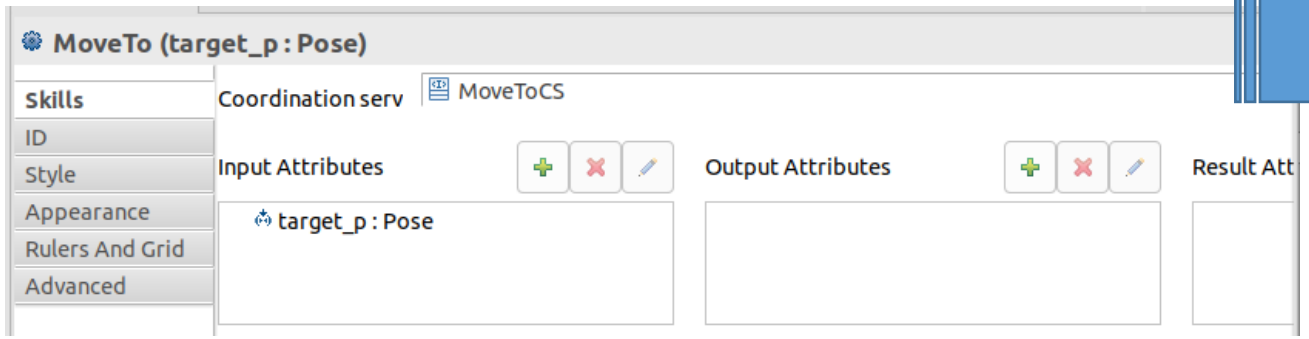
Behavior Development phase

Task Modeling

Behavior Developer

Skill code-generation

code-generation



move_to_action.cpp

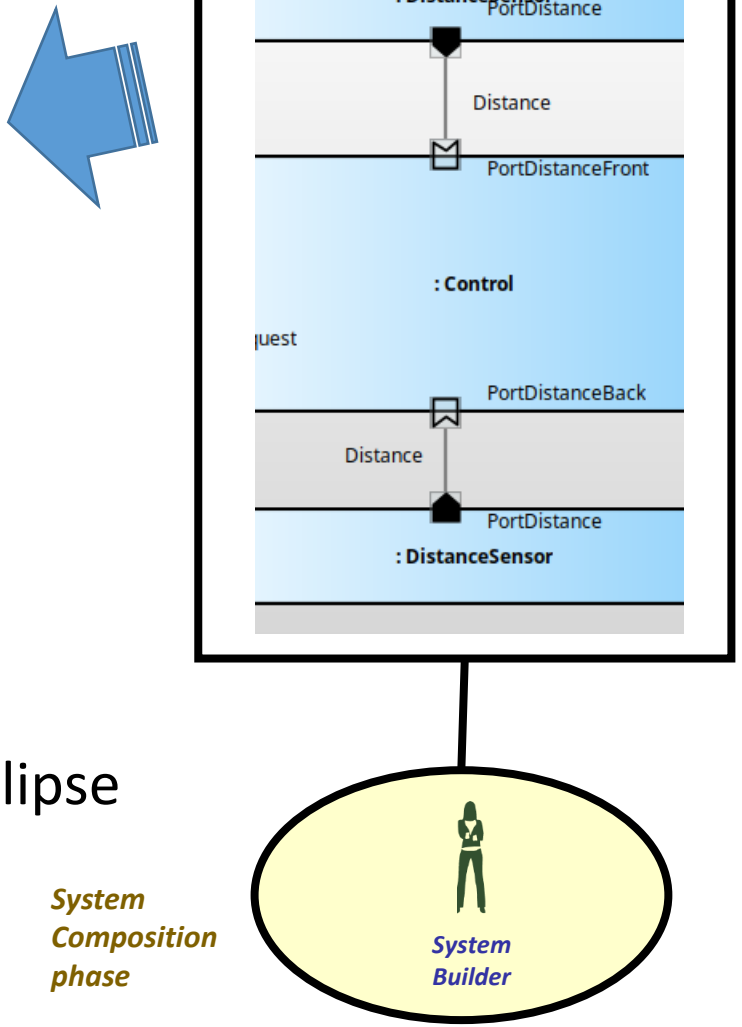
```cpp
// Generated by Papyrus4Robotics
//

#include "geometry_msgs/msg/pose.hpp"
#include "mobilemanipulationservicedef/action/move_to_cs.hpp"
#include "nav2_behavior_tree/bt_action_node.hpp"

class MoveToAction : public
nav2_behavior_tree::BtActionNode<mobilemanipulationservicedef::action::MoveToCS>
{
public:
  MoveToAction(
      const std::string& name,
      const std::string & action_name,
      const BT::NodeConfiguration& conf)
  :
nav2_behavior_tree::BtActionNode<mobilemanipulationservicedef::action::MoveToCS>(
action_name, conf)
    {
    }

  void on_tick() override
  {
    geometry_msgs::msg::Pose target_p;
    getInput("target_p", target_p);
    goal_.position = target_p.position;
    goal_.orientation = target_p.orientation;
  }


  // MoveTo has in/out parameters => must provide a providedPorts method
  static BT::PortsList providedPorts()
  {
    return{
      BT::InputPort<geometry_msgs::msg::Pose>("target_p")
    };
  }
};

#include "behaviortree_cpp_v3/bt_factory.h"
BT_REGISTER_NODES(factory)
{
  BT::NodeBuilder builder =
    [](const std::string & name, const BT::NodeConfiguration & config)
```

Generation of build and launch files

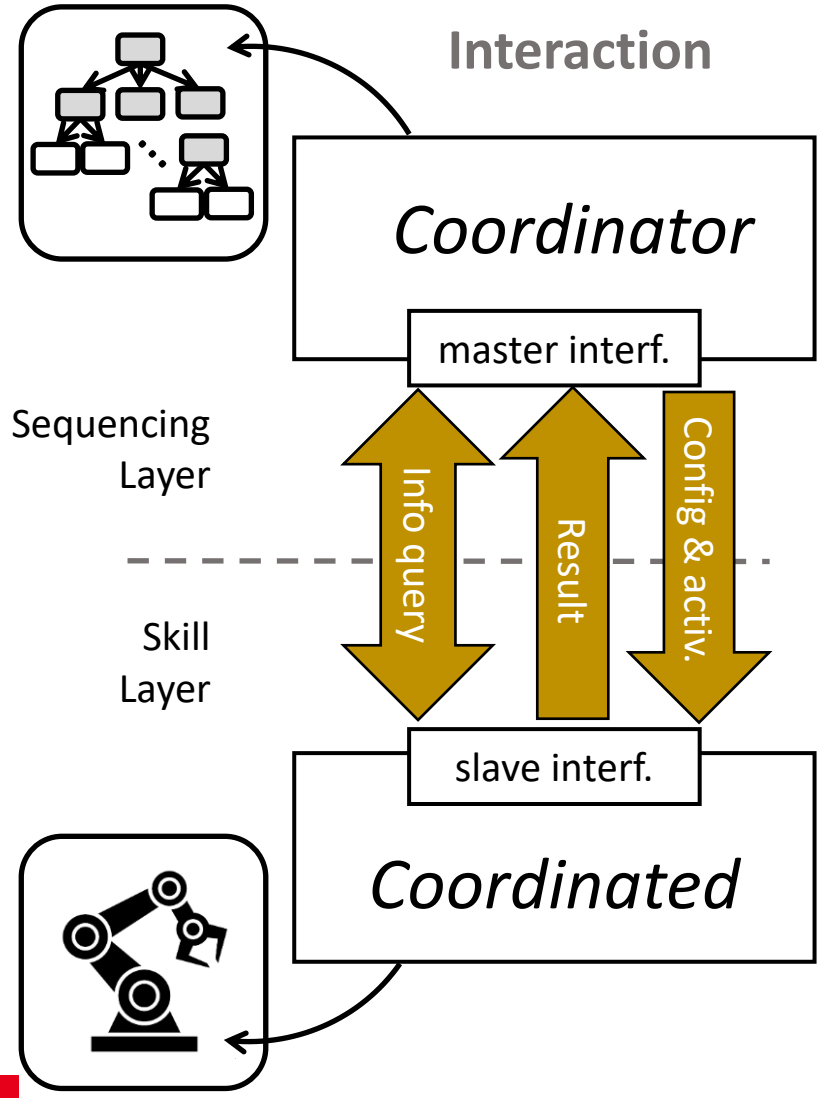
- package.xml and CMakelists.txt
  configure Eclipse CDT to use colcon
- launch script with re-mappings according to composition,
  activate components automatically
- YAML files for parameter configuration
  default value overridden per instance

Eclipse integration

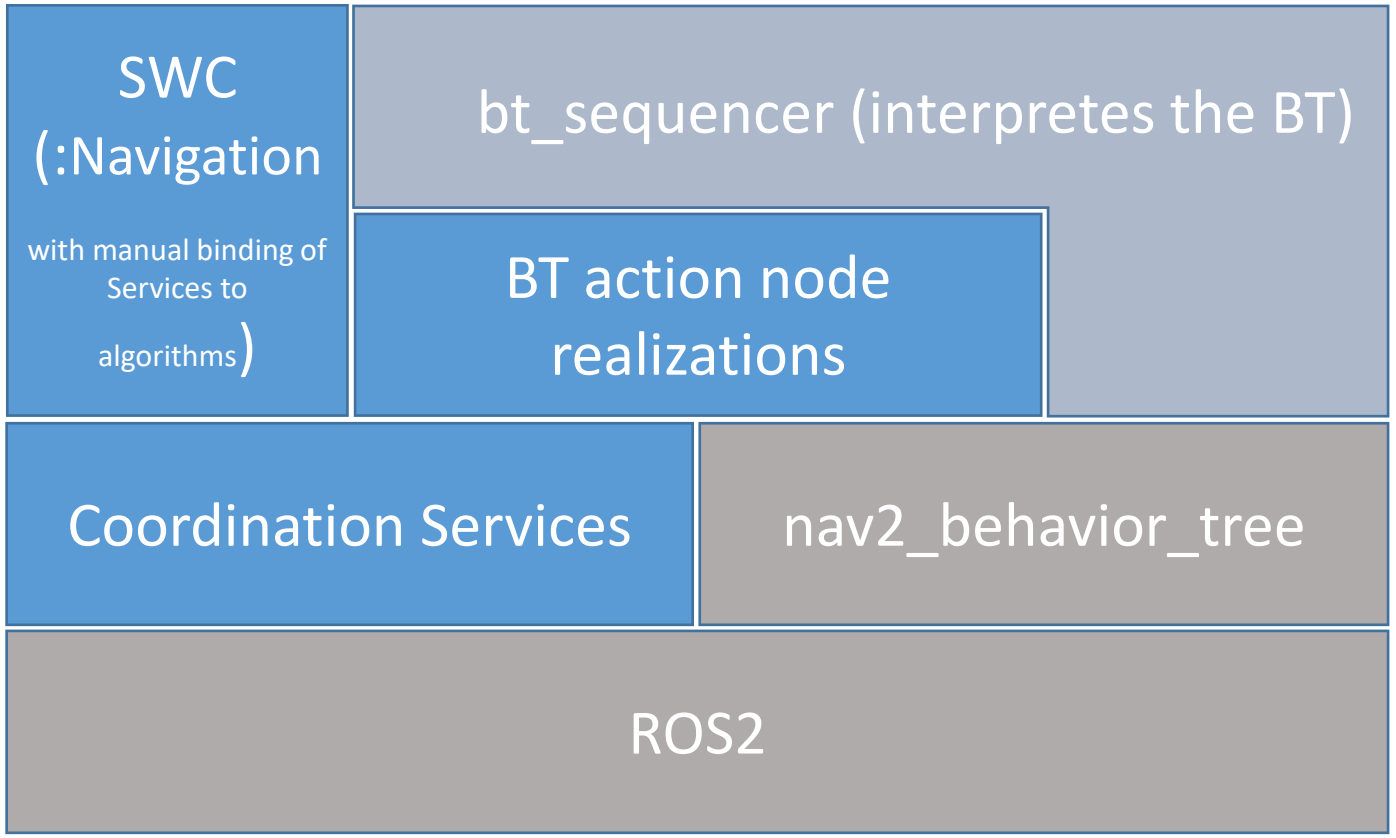
- Can launch and debug (CDT debugger) a component from Eclipse
  https://youtu.be/kWkUpKcJq48
- Use of Eclipse lauch configurations / console



Task execution



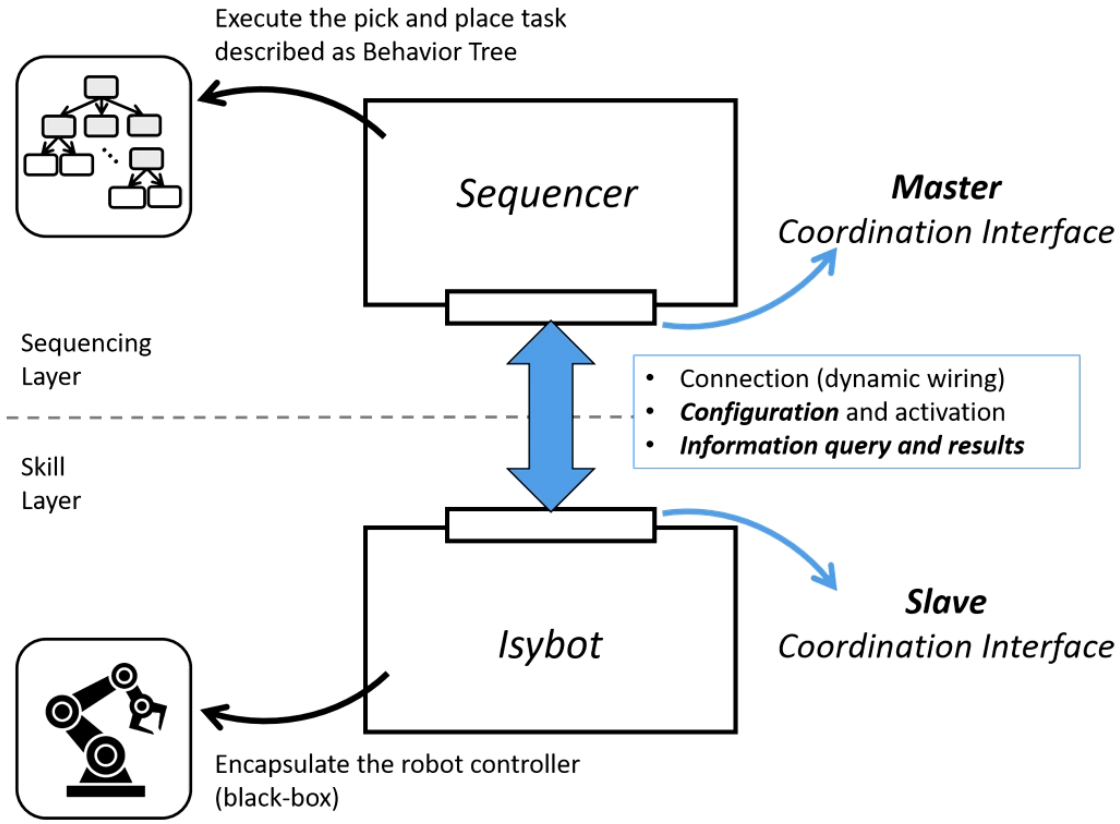
Run-time architecture



- Supporting modules (from standard ROS2)
- Supporting modules (customized from standard ROS2)
- Generated modules

Example video

- Context: simulation of the collaborative robot arm Isybot performing a pick and place task
- System composed of 2 components -- the **Sequencer** executes the **behavior tree** specification of pick -and-place task by opportune **configuration and activation of** the **Isybot component**
- The **coordination interface** conforms to the “Architectural Pattern for Component Coordination” [1] introduced in RobMoSys and is ***generated from models*** *of the behavioral and system specification*



[1] https://robmosys.eu/wiki/general_principles:architectural_patterns:component-coordination

Gosselin, F. et al, “Robot Companion, an Intelligent Interactive Robot Coworker for the Industry 5.0”, IROS 2022