Advanced Remanufacturing and Technology Centre

ARTC

ROS-INDUSTRIAL DEVELOPERS MEETING JUNE SESSION

9th June 2020

Meeting Agenda

1) New ROS Software Quality Guidelines

2) Industrial Continuous Integration (CI)

3) Experiences with MoveIt2

4) Questions and Answers

Summarizing **REP-2004**: Package Quality Categories

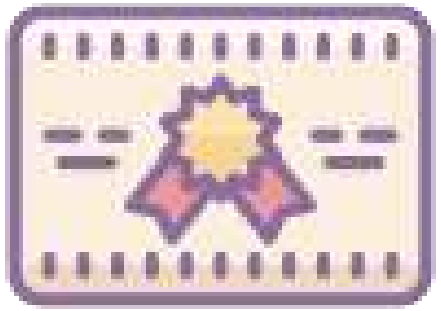
NEW ROS SOFTWARE QUALITY GUIDELINES

Presenter: Bey Hao Yun, Research Engineer, ROS-Industrial Consortium Asia Pacific
Date: 9th June 2020

Content Outline

- What is **REP-2004**: Package Quality Categories?
- Why **REP-2004**?
- How to Use **REP-2004** [For Package Users]?
- Software Quality Metrics Used
  - Version Policy
  - Change Control Process
  - Documentation
  - Testing
  - Dependencies
  - Platform Support
  - Security
- 5 Quality Levels In a Nutshell
- References

What is REP-2004: Package Quality Categories?



- REP stands for **ROS Enhancement Proposal**.
- **REP-2004** provides standard guidelines regarding **package quality**.
- It states **5 different Quality Levels** which **a ROS package** can have, setting expectations for both users and maintainers.

https://github.com/ros-infrastructure/rep/blob/master/rep-2004.rst

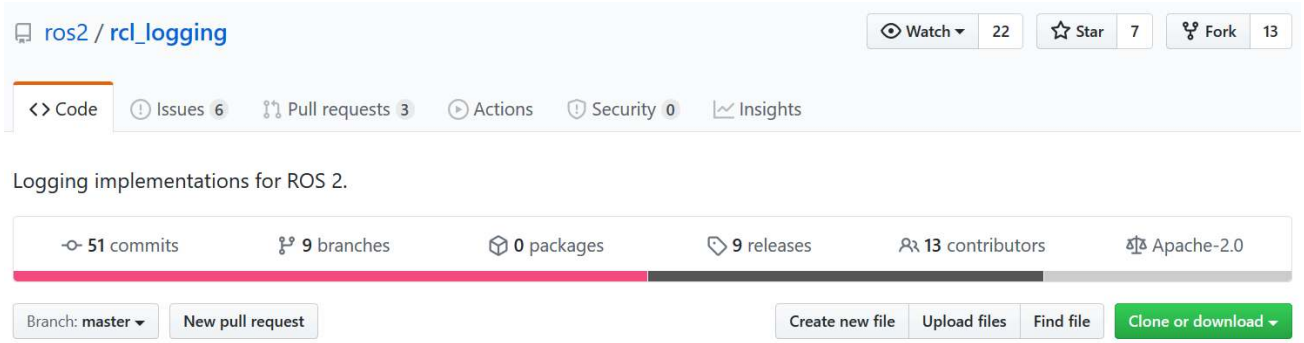
Why REP-2004?

- For **ROS package users**, it allows you to:
  - Determine whether a package is **suitable for your project**, at a glance.

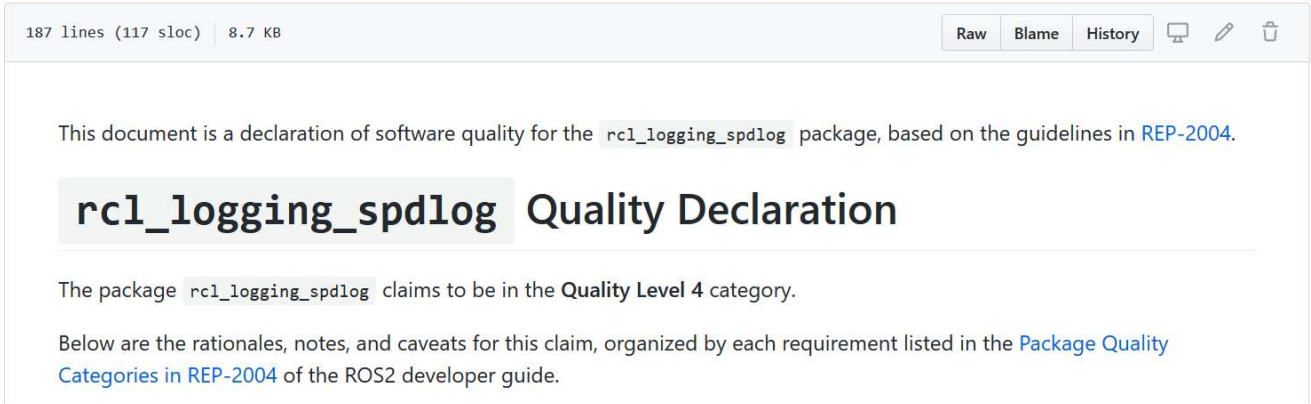
- For **ROS package maintainers**, it allows you to:
  - Adopt structured **software quality management**.
  - Improve **ease of the use** of the package among the global community.

How to Use REP-2004? [For Package Users]

- Go to the online repository which has your ROS package of interest.
  - Eg. https://github.com/ros2/rcl_logging



- Look for a file called **QUALITY_DECLARATION.md**
  - This document states how a package adheres to its declared quality level.



Software Quality Metrics Used [1/8] - Overview

- There are **5** different **Quality Levels**.
  - 1 being the highest quality.
  - 5 being the lowest.
- There are **7 metrics** used to distinguish between different Quality levels.

1	2	3	4	5	6	7
						
Version Policy	Change Control Process	Documentation	Testing	Dependencies	Platform Support	Security

Software Quality Metrics Used [2/8] – Version Policy

Format:
MAJOR.MINOR.PATCH

Eg. Ubuntu 18.04.5

- **Version Policy** refers to the way you label different versions of your software to reflect changes to Application Programming Interface (**API**) or Application Binary Interface (**ABI**).
- It is advisable to follow the **semver** or **Semantic Versioning >= 1.0.0** for your policy.

[1]

1	Requirements
i.	Must have a version policy. (Eg. semver)
ii.	Must be a stable version. (e.g. for semver that means version >= 1.0.0)
iii.	Must have a strictly declared public API.
iv.	Must have a policy for API stability.
xxii.	Must have a policy for ABI stability
vi.	Must have a policy that keeps API and ABI stability within a released ROS distribution

Software Quality Metrics Used [3/8] – Change Control Process

- **Change Control Process** refers to the way you control and track changes to the software over time.
- This involves the use of version control platforms like GitHub and Continuous Integration (CI) tools.

[2]

2	Requirements
i.	Must have all code changes occur through a change request (e.g. pull request, merge request, etc.)
ii.	Must have confirmation of contributor origin (e.g. DCO, CLA, etc.)
iii.	Must have peer review policy for all change requests (e.g. require one or more reviewers)
iv.	Must have Continuous Integration (CI) policy for all change requests
xxii.	Must have documentation policy for all change requests

Software Quality Metrics Used [4/8] – Documentation

- **Documentation** refers to the instruction manuals that guide users on how to use your software. (Eg. API library, README.md)
- It also states the license of your software for any 3rd parties who wish to use it.

[3]

3	Requirements
i.	Must have documentation for each "feature" (e.g. for rclcpp: create a node, publish a message, spin, etc.)
ii.	Must have documentation for each item in the public API (e.g. functions, classes, etc.)
iii.	Must have a declared license or set of licenses
iv.	Must state copyrights within the project and attribute all authors
xxii.	Must have a "quality declaration" document, which declares the quality level and justifies how the package meets each of the requirements • *For more details on this, please refer to the **official REP-2004 .rst documentation**.*

Software Quality Metrics Used [5/8] – Testing

- **Testing** refers to the use of static and dynamic analysis quality assurance tools (Eg. pytest, googletest, linters).

[4]

4	Requirements
i.	Must have system tests which cover all items in the "feature" documentation.
ii.	Must have system, integration, and/or unit tests which cover all of the public API.
iii.	**Code Coverage**: *a) Must have code coverage tracking for the package* *b) Must have and enforce a code coverage policy for new changes*
iv.	**Performance**: a) Must have performance tests (exceptions allowed if they don't make sense to have) b) Must have a performance regression policy (i.e. blocking either changes or releases on unexpected performance regressions)
xxii.	**Linters and Static Analysis**: *a) Must have a code style and enforce it* *b) Must use static analysis tools where applicable*

Software Quality Metrics Used [6/8] – Dependencies

- **Dependencies** refers to providing justification for the use of each external libraries which your package depends on.
  - Eg. Your package should depend on direct runtime “ROS” dependencies which share the same Quality Level you declared.
  - Eg. Provide justification on the perceived Quality Level of direct runtime “non-ROS” dependencies (Eg. Opencv, PointCloud Library (PCL))

[5]

5	Requirements
i.	Must not have direct runtime "ROS" dependencies which are not at the same level as the package in question ('Level N'), but...
ii.	May have optional direct runtime "ROS" dependencies which are not 'Level N', e.g. tracing or debugging features that can be disabled
iii.	Must have justification for why each direct runtime "non-ROS" dependency is equivalent to a 'Level N' package in terms of quality

Software Quality Metrics Used [7/8] – Platform Support

- **Platform Support** refers to what combination of Operating System (OS), architecture and ROS Middleware (**rmw**) your package must be built on to run properly.
- For more details, please refer to **REP-2000**.

[6]

6	Requirements
i.	Must support all target platforms for the package's ecosystem. • For ROS 2 this means supporting all tier 1 platforms, as defined in REP-2000

Software Quality Metrics Used [8/8] – Platform Support

- **Security** refers to having a **Vulnerability Disclosure Policy**.
- This policy document should outline **how** a user, who has identified a security flaw in your package, can:
  - Submit the bug report.
  - Expect a timely response on how developers of the package is addressing the issue.

[7]

7	Requirements
i.	Must have a declared Vulnerability Disclosure Policy and adhere to a response schedule for addressing security vulnerabilities

https://ros.org/reps/rep-2006.html

5 Quality Levels In A Nutshell

- The table summarizes Quality Levels 1 to 5, based on the numbering scheme used for requirements in previous slides.

✓	required
●	recommended
○	optional

- Referenced from **Quality Level Comparison Chart in REP-2004**.

	Level 1	Level 2	Level 3	Level 4	Level 5
1.i	✓	✓	✓	●	
1.ii	✓	✓	✓		
1.iii	✓	✓	●		
1.iv	✓	✓	✓		
1.v	✓	✓	✓		
1.vi	✓	✓	●		
2.i	✓	✓	✓	●	
2.ii	✓	✓			
2.iii	✓				
2.iv	✓	✓	✓		
2.v	✓				
3.i	✓	✓			
3.ii	✓				
3.iii	✓	✓	✓	✓	●
3.iv	✓	✓	✓	✓	
3.v	✓	✓	○		
3.v.a	✓	✓	✓		
3.v.b	●	●	○		
3.v.c	✓	✓	✓		
4.i	✓	✓	●	●	
4.ii	✓				
4.iii.a	✓	✓			
4.iii.b	✓				
4.iv.a	✓				
4.iv.b	✓				
4.v.a	✓	✓			
4.v.b	✓	✓			
5.i	✓	✓			
5.ii	○	○	○		
5.iii	✓	✓			
6.i	✓	✓	✓	●	
7.i	✓	✓	●		

References

- REP-2004
  - https://ros.org/reps/rep-2004.html
- REP-2000 – For Platform Support details
  - https://www.ros.org/reps/rep-2000.html#support-tiers
- REP-2004 GitHub Pull Request – For keep track of updates.
  - https://github.com/ros-infrastructure/rep/pull/218
- Semantic Versioning 2.0.0 – Regarding Version Policy
  - https://semver.org/

Automation in Quality Assurance

INDUSTRIAL CONTINUOUS INTEGRATION (CI)

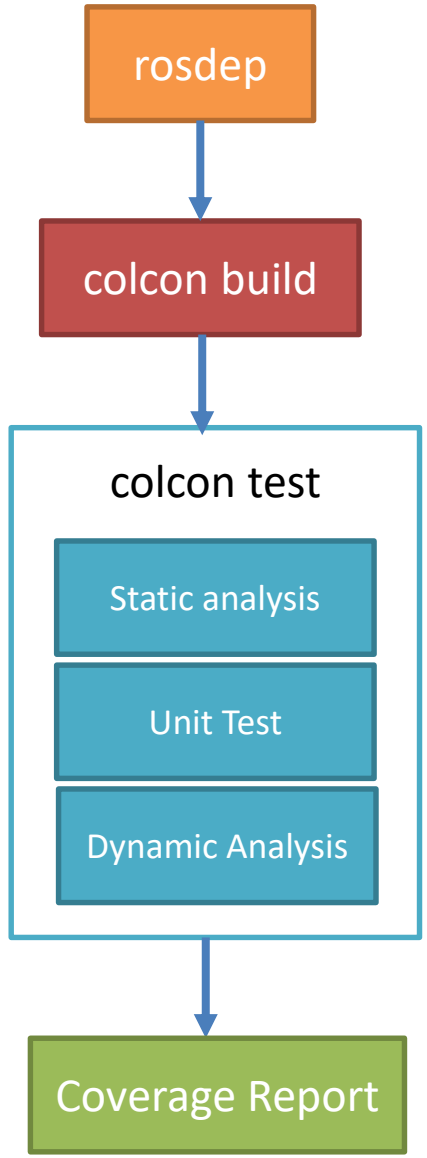
Presenter: Chen Bainian, Development Engineer, ROS-Industrial Consortium Asia Pacific
Date: 9th June 2020

Overview

- Setup Quality Assurance Pipeline on local PC
- Introduction to industrial_ci
- Quality Assurance on industrial_ci
- Future Development

Setup Quality Assurance Pipeline on local PC

- Configure Dependencies
- Static Analysis and Unit Tests
- Dynamic Analysis
- Code Coverage



Configure Dependencies

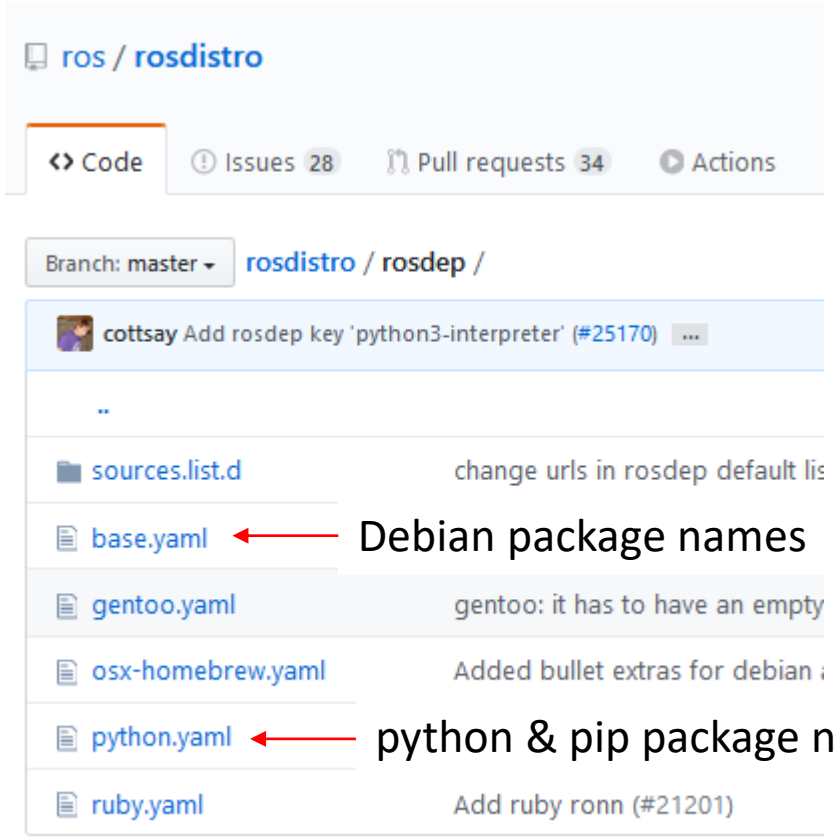
rosdep (best)

python, pip, debian packages

Define dependency in **package.xml**
Package name Link:
https://github.com/ros/rosdistro

Command:

```
cd <workspace-directory>
rosdep install --from-paths src --ignore-src -y --rosdistro $ROS_DISTRO
```



Configure Dependencies

vcstool

Online repository (git, hg, svn)
Colcon supported builder
  pip, cmake, ament(ros2), catkin

Define dependency in **.repos** file
Documentation Link:
https://github.com/dirk-thomas/vcstool

Command

```
cd <workspace-directory>
vcs import --recursive src < my.repos
```

Sample **.repos** file

```
repositories:
  rmf_core:
    type: git
    url: https://github.com/osrf/rmf_core.git
    version: master

 rmf_schedule_visualizer:
   type: git
   url: https://github.com/osrf/rmf_schedule_visualizer.git
   version: master

 traffic_editor:
   type: git
   url: https://github.com/osrf/traffic_editor.git
   version: master
```

Static Analysis

ament_lint

Repo:
https://github.com/ament/ament_lint

ROS2 static analysis library
Run during **colcon test** along with gtest and/or gmock

ament_lint_auto
The auto-magic functions for ease to use of the ament linters in CMake.

CPP (CMakeLists.txt)

```
if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for copyrights
  # uncomment the line when a copyright and license is not present in all source files
  #set(ament_cmake_copyright_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()
```

- copyright
- cppcheck
- cpplint
- lint_cmake
- uncrustify
- xmllint

Static Analysis

ament_lint

Repo:
https://github.com/ament/ament_lint

ROS2 static analysis library
Run during **colcon test** along with pytest/unittest

- copyright
- flake8
- pep257
- xmllint

Python

test_flake8.py 775 Bytes

```python
# Copyright (c) 2019 ROS-Industrial Consortium Asia Pacific
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

from ament_flake8.main import main
import pytest


@pytest.mark.flake8
@pytest.mark.linter
def test_flake8():
    rc = main(argv=[])
    assert rc == 0, 'Found errors'
```

Unit Tests

CPP

ament_gtest (gtest)
ament_gmock (gmock)

Python

pytest (recommended)
unittest
nosetest

System test

lauch_testing
(https://github.com/ros2/launch/tree/master/launch_testing)

Dynamic Analysis (Address & Thread Sanitizer)

Address Sanitizer

To build

```
colcon build --build-base=build-asan --install-base=install-asan \
    --cmake-args -DCMAKE_BUILD_TYPE=Debug \
                 -DCMAKE_C_FLAGS='-fsanitize=address' \
                 -DCMAKE_CXX_FLAGS='-fsanitize=address'
## You can also run the mixin alternative to relace the CMake flags
#               --mixin asan-gcc
```

To test

```
colcon test --build-base=build-asan --install-base=install-asan \
            --event-handlers sanitizer_report+
```

Thread Sanitizer

To build

```
colcon build --build-base=build-tsan --install-base=install-tsan \
    --cmake-args -DCMAKE_BUILD_TYPE=Debug \
                 -DCMAKE_C_FLAGS='-fsanitize=thread -O2 -g -fno-omit-frame-pointer' \
                 -DCMAKE_CXX_FLAGS='-fsanitize=thread -O2 -g -fno-omit-frame-pointer'
## You can also run the mixin alternative to relace the CMake flags
#               --mixin tsan
```

To test

```
colcon test --build-base=build-tsan --install-base=install-tsan \
            --event-handlers sanitizer_report+
```

More information

https://github.com/colcon/colcon-sanitizer-reports

Code Coverage

```
colcon build --cmake-args -DCMAKE_C_FLAGS='--coverage' \
                          -DCMAKE_CXX_FLAGS='--coverage'
## You can also run the mixin alternative
# colcon build --mixin coverage-gcc
```

```
colcon test --pytest-with-coverage \
            --pytest-args --cov-report=term \
            --event-handlers console_direct+
## You can also run the mixin alternative
# colcon test --mixin coverage-pytest \
#             --event-handlers console_direct+
```

← Python xml & html report

Other type of unit tests using **coverage.py**
https://github.com/colcon/colcon-coveragepy-result

```
colcon lcov-result
```

← CPP html report

TODO: Combine cpp and python coverage report locally

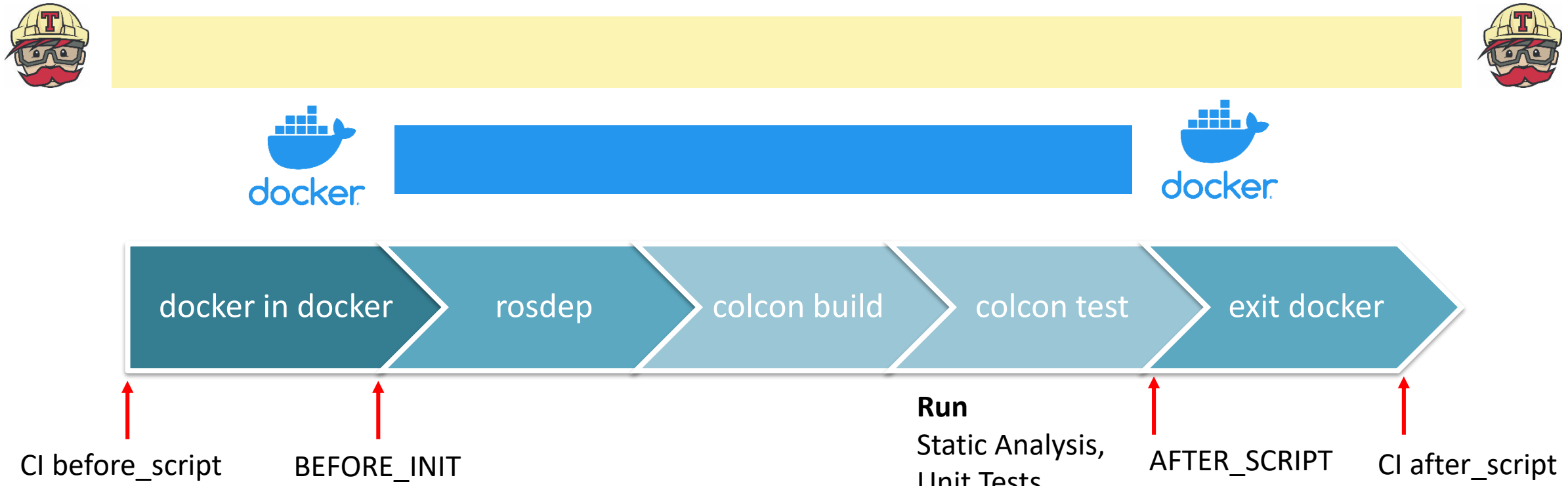
Industrial CI

This package contains CI (Continuous Integration) bash scripts that any ROS-powered packages can commonly use. Some notable feature:

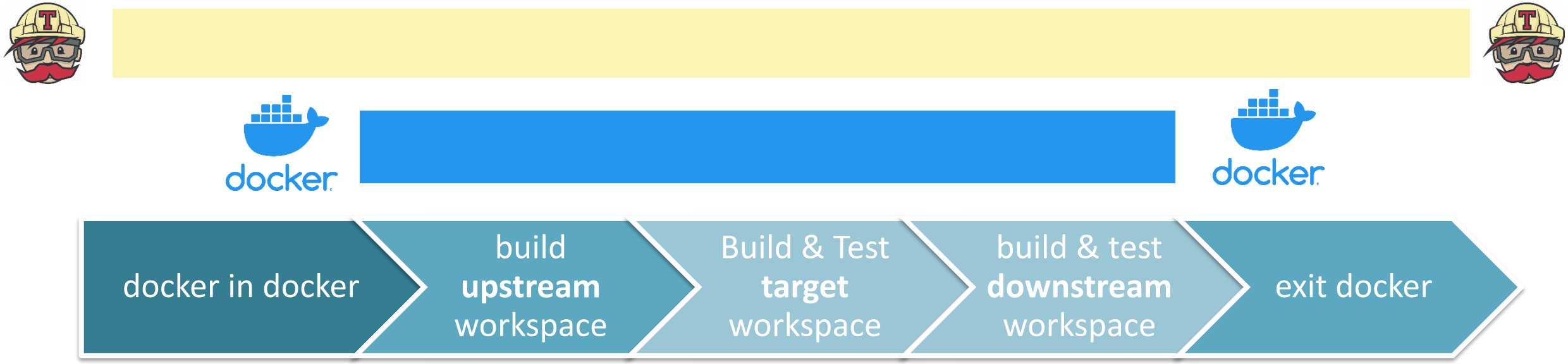
- Checks if your package builds, installs without issues. If unit/system tests are defined run them. ROS Prerelease Test(ROS 1), ABI check can optionally be run.
- Proven to cover the general requirements of the ROS-based robotics repositories. Easily configurable.
- Users can add custom pre/post processes.
- Covers **ROS1** Indigo, Jade, Kinetic, Lunar, Melodic and **ROS2** distributions.
- This repo provides scripts for **Bitbucket CI**, **Gitlab CI, GitHub Actions** and **Travis CI** only, but it can be easily adapted for other CI services.

Repo: https://github.com/ros-industrial/industrial_ci/tree/master

Industrial CI ROS2 Source Test Workflow



Industrial CI ROS2 Source Test Workflow



Industrial CI ROS2 Source Test Setup

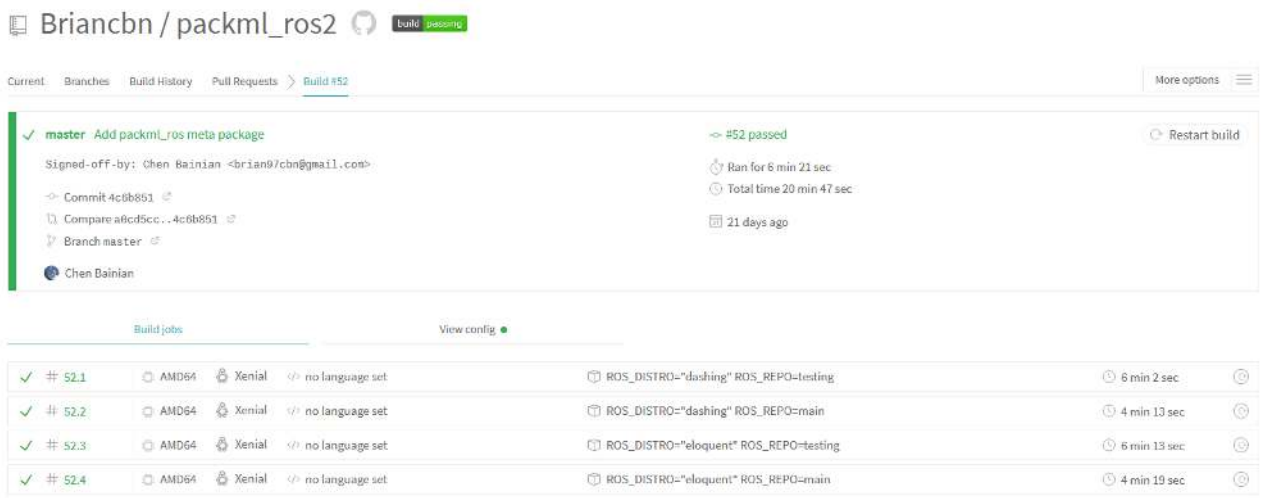
- Activate CI

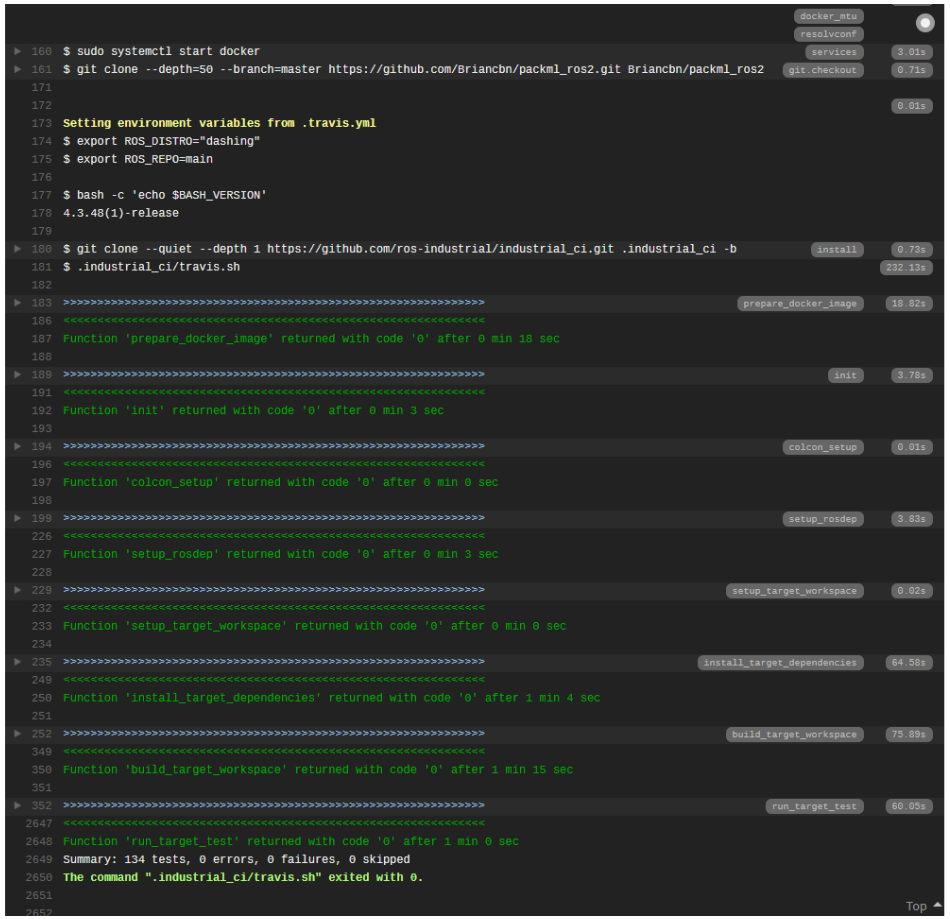
- Add in config file (sample **.travis.yml** for Travis CI)

```
language: generic
services:
  - docker

env:
  global:
    - PARALLEL_TESTS=true  # Global Variable

  matrix:
    - ROS_DISTRO="dashing" ROS_REPO=testing
    - ROS_DISTRO="dashing" ROS_REPO=main
    - ROS_DISTRO="eloquent" ROS_REPO=testing
    - ROS_DISTRO="eloquent" ROS_REPO=main
    - ROS_DISTRO="foxy" ROS_REPO=testing
    - ROS_DISTRO="foxy" ROS_REPO=main

install:
  - git clone --quiet --depth 1 https://github.com/ros-industrial/industrial_ci.git .industrial_ci -b master
script:
  - .industrial_ci/travis.sh
```

Industrial CI ROS2 Source Test Setup

- Commit and Push





Industrial CI ROS2 Source Test Setup

- Other useful Configuration

DOCKER_IMAGE/DOCKER_FILE: Custom Image from Dockerhub or from local file

UPSTREAM_WORKSPACE/DOWNSTREAM_WORKSPACE: .repos, .rosinstall or links to additional packages

ROSDEP_SKIP_KEYS: dependency ignored by rosdep

CMAKE_ARGS / TARGET_CMAKE_ARGS / UPSTREAM_CMAKE_ARGS / DOWNSTREAM_CMAKE_ARGS:

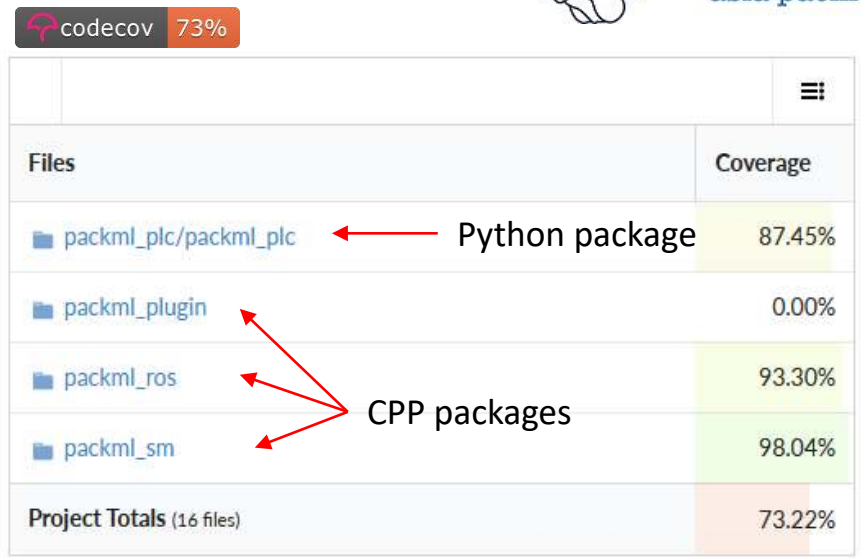
Additional CMake arguments for respective workspace

More Information

https://github.com/ros-industrial/industrial_ci/blob/master/doc/index.rst#optional-environment-variables

Future Development

- CODE_COVERAGE
  - ☒ Run code coverage
  - ☒ Upload report to codecov.io (**CODE_COVERAGE**=codecov.io)
  - ☐ Upload report to coveralls.io
- RUN_ASAN
  - ◉ Run Address Sanitizer
- RUN_TSAN
  - ◉ Run Thread Sanitizer
- Acceptance Criteria
  - ☐ Check syntax check, style check, unit tests and dynamic analysis were included.
  - ☒ All tests passed
  - ☒ Code Coverage > 90%
  - ☐ Quality Assurance Badge



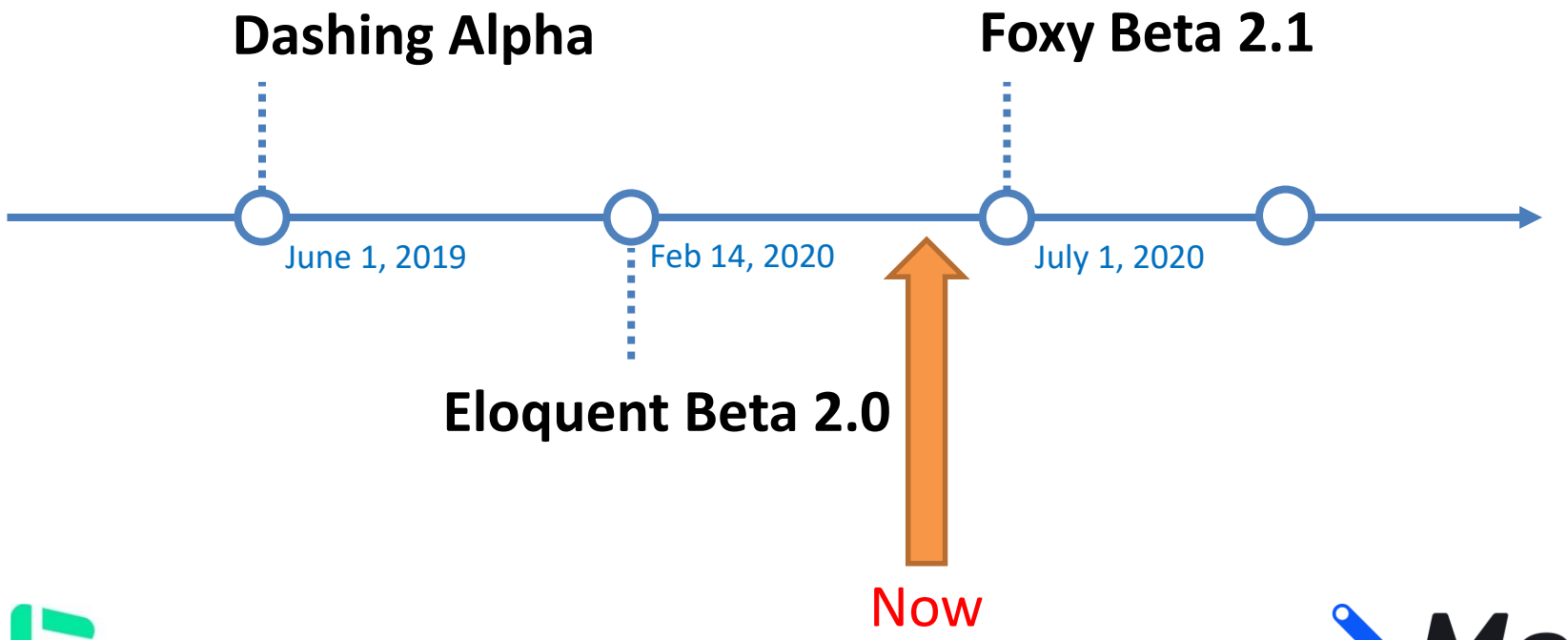
EXPERIENCES WITH MOVEIT2

Presenter: Chen Bainian, Development Engineer, ROS-Industrial Consortium Asia Pacific
Date: 9th June 2020

Overview

- Moveit2 Development Roadmap
- Experience with Alpha Release
- First Beta Release
- What to expect in July Release

Moveit2 Development Roadmap



PICKNIK

MoveIt2

Moveit2 Development Roadmap

Milestone 1

Straight Port to ROS 2

- Fully migrate existing MoveIt packages to ROS 2
  - Wrap up Acutronic's work porting core MoveIt functionality
- Leverage ROS 2:
  - Build type (ament), middleware, logging, parameters
- Cleanup MoveIt 2 codebase

Milestone 2

Realtime Support

- Reactive, closed-loop control to sensor input
  - Visual servoing, faster octomap updates
  - Preempt motion if new collision detected
- Seperate global and local planner (hybrid planning)
  - Global planner (full collision checking): ~30hz
  - Local planner (IK-based, field-based): ~300hz
- Zero-memory copy integration to controllers (ros_control)
  - Tighter integration to ros_control
- Integrate pilz_industrial_motion

Milestone 3

Fully Leverage ROS 2

- Lifecycle management of MoveIt nodes
  - Deterministic startup, reset, & shutdown sequences
- Leverage ROS2 component nodes
  - Ability to run MoveIt as single or multi-process
  - Replace pluginlib with components
- Cleanup API
  - More generic and standalone interfaces

PICKNIK

Foxy Beta 2.1

MoveIt2

Moveit2 Development Roadmap

Future Milestones

Determinism	Improved Interfaces / State Machines	Machine Learning
Out of box / default planners return reliable paths	Deprecate the Pick and Place pipeline	Neural-network based motion planning - new plugins
Run multiple planners in parallel to address limitations	Fully support the MoveIt Task Constructor	General near-optimal heuristics for path planning
Further optimize paths, improve path quality	First class support of state machines	e.g. MPNet
Default use TOTG, TOPP time parameterization	Non-ROS C++ API	ML to generate DMPs for contact-rich interactions
Use post-processing optimization (STOMP, TrajOpt)	Similar to MoveGroup but without middleware	ML for grasp synthesis
Fully featured Cartesian Planner		
Similar to Descartes but fully integrated as MoveIt planner		
Force-torque control		

PICKNIK

MoveIt2

Detail Information: https://moveit.ros.org/documentation/contributing/roadmap/
Code Sprint: https://moveit.ros.org/documentation/contributing/future_projects/

Experience with Alpha Release

November: A Trying Experience with Alpha Release >.<

Sources: https://giphy.com/gifs/l2qJnXbUzQAeY
https://news.efinancialcareers.com/cn-en/272698/the-most-frustrated-people-in-finance

Experience with Alpha Release

Problem 1: Parameters (Since Dashing)

In Crystal:

```
auto node = rclcpp::Node::make_shared("example_params");
std::string planning_plugin =
  node_->get_parameter("planning_plugin").get_value<std::string>();
```

Since Dashing:

```
auto node = rclcpp::Node::make_shared("example_params");
node->declare_parameter("planning_plugin", "ompl_interface/OMPLPlanner");
std::string planning_plugin =
  node_->get_parameter("planning_plugin").get_value<std::string>();
```

Advantages:

- Better Security
- Better Parameter Control
- Parameter Descriptor

```
rcl_interfaces::msg::ParameterDescriptor format_description;
format_description.name = "format";
format_description.type = rcl_interfaces::msg::ParameterType::PARAMETER_STRING;
format_description.description = "Compression method";
format_description.read_only = false;
format_description.additional_constraints = "Supported values: [jpeg, png]";
node->declare_parameter("format", kDefaultFormat, format_description);
```

An example of parameter descriptor

Experience with Alpha Release

Problem 1: Parameters (Since Dashing)

Other solution **(Not recommended)**:

allow_undeclared_parameters: allow getting/setting undeclared parameters
automatically_declare_parameters_from_overrides:
allow parameters override on launch from .yaml files

```
196  196      rclcpp::init(argc, argv);
197       -   auto node = rclcpp::Node::make_shared(move_group::NODE_NAME);
     197  +   rclcpp::NodeOptions node_option;
     198  +   node_option.allow_undeclared_parameters(true);
     199  +   node_option.automatically_declare_parameters_from_overrides(true);
     200  +   auto node = rclcpp::Node::make_shared(move_group::NODE_NAME, node_option);
```

Experience with Alpha Release

Problem 1: Parameters (Since Dashing)

ROS2 .yaml parameter format:

```yaml
move_group:
  ros__parameters:
    controller_list:
      name: ""
      action_ns: follow_joint_trajectory
      type: FollowJointTrajectory
      joints: [shoulder_pan_joint, shoulder_lift_joint, elbow_joint, wrist_1_joint, wrist_2_joint, wrist_3_joint]
```

ROS1 .yaml parameter format:

```yaml
controller_list:
  - name: ""
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    joints:
      - shoulder_pan_joint
      - shoulder_lift_joint
      - elbow_joint
      - wrist_1_joint
      - wrist_2_joint
      - wrist_3_joint
```

Experience with Alpha Release

Problem 2: .launch.py

.launch.xml (Since Eloquent)

ROS2 .launch.py example:

```
import sys
import os

from ament_index_python.packages import get_package_share_directory, get_package_prefix
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, IncludeLaunchDescription
from launch.actions import SetLaunchConfiguration
from launch.substitutions import LaunchConfiguration
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch_ros.actions import Node
import xacro
import tempfile
```

```
def to_urdf(xacro_path, urdf_path=None):
    """Convert the given xacro file to URDF file.
    * xacro_path -- the path to the xacro file
    * urdf_path -- the path to the urdf file
    """
    # If no URDF path is given, use a temporary file
    if urdf_path is None:
        urdf_path = tempfile.mktemp(prefix="%s_" % os.path.basename(xacro_path))

    # open and process file
    doc = xacro.process_file(xacro_path)
    # open the output file
    out = xacro.open_output(urdf_path)
    out.write(doc.toprettyxml(indent='  '))

    return urdf_path  # Return path to the urdf file
```

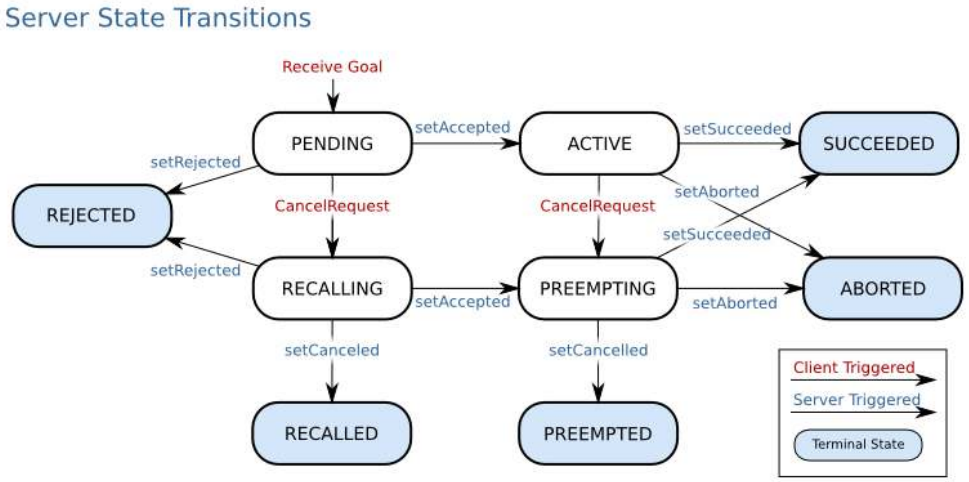
```
def generate_launch_description():
    xacro_path = os.path.join(get_package_share_directory('lab_cell_description'), 'urdf/', 'workcell' + '.xacro')
    srdf = os.path.join(get_package_share_directory('lab_cell_moveit_config'), 'config', 'myworkcell' + '.srdf')
    controller_list = os.path.join(get_package_share_directory('lab_cell_moveit_config'), 'config', 'ros_controllers' + '.yaml')
    kinematics_config = os.path.join(get_package_share_directory('lab_cell_moveit_config'), 'config', 'kinematics.yaml')
    assert os.path.exists(xacro_path)
    assert os.path.exists(srdf)
    assert os.path.exists(controller_list)
    assert os.path.exists(kinematics_config)
```

```
ld = LaunchDescription([
    DeclareLaunchArgument('allow_trajectory_execution', default_value='True'),
    DeclareLaunchArgument('fake_execution', default_value='False'),
    DeclareLaunchArgument('info', default_value='True'),
    DeclareLaunchArgument('debug', default_value='False'),


    # Given the published joint states, publish tf for the robot links
    Node(
        package='robot_state_publisher',
        node_executable='robot_state_publisher',
        node_name="robot_state_publisher",
        output='screen', arguments=[to_urdf(xacro_path)]),

    # Launching move_group
    Node(
        package='moveit_ros_move_group',
        node_executable='move_group',
        # node_name='ur3e_move_group', this will currently force the transform_listener to launch under the same name as move_group
        output='screen',
        parameters=[
            {'allow_trajectory_execution': LaunchConfiguration('allow_trajectory_execution')},
            {'moveit_manage_controllers': True},
            {'planning_plugin': "ompl_interface/OMPLPlanner"},
            {"moveit_controller_manager": "moveit_simple_controller_manager/MoveItSimpleControllerManager"},
            controller_list,
            kinematics_config
            ]
        ),
```

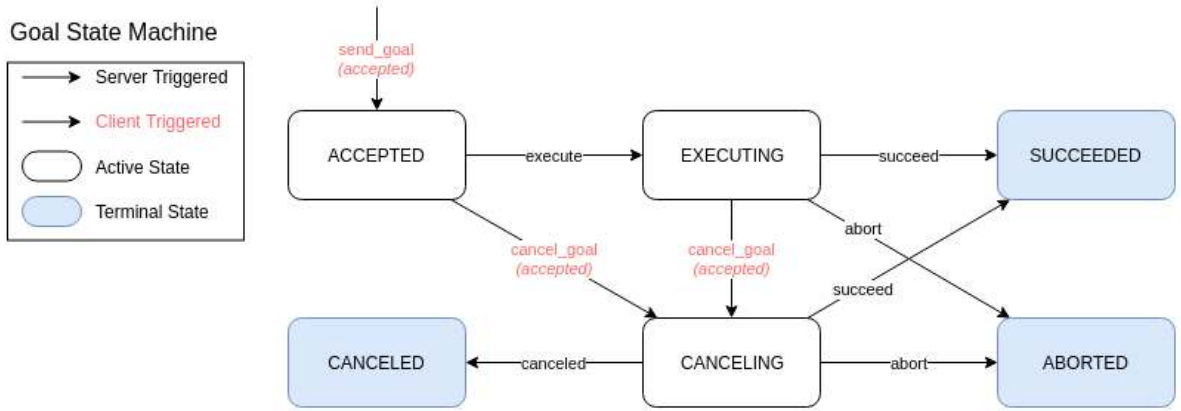
Experience with Alpha Release

Problem 2: Action interface and ROS1 Bridge

ROS1:



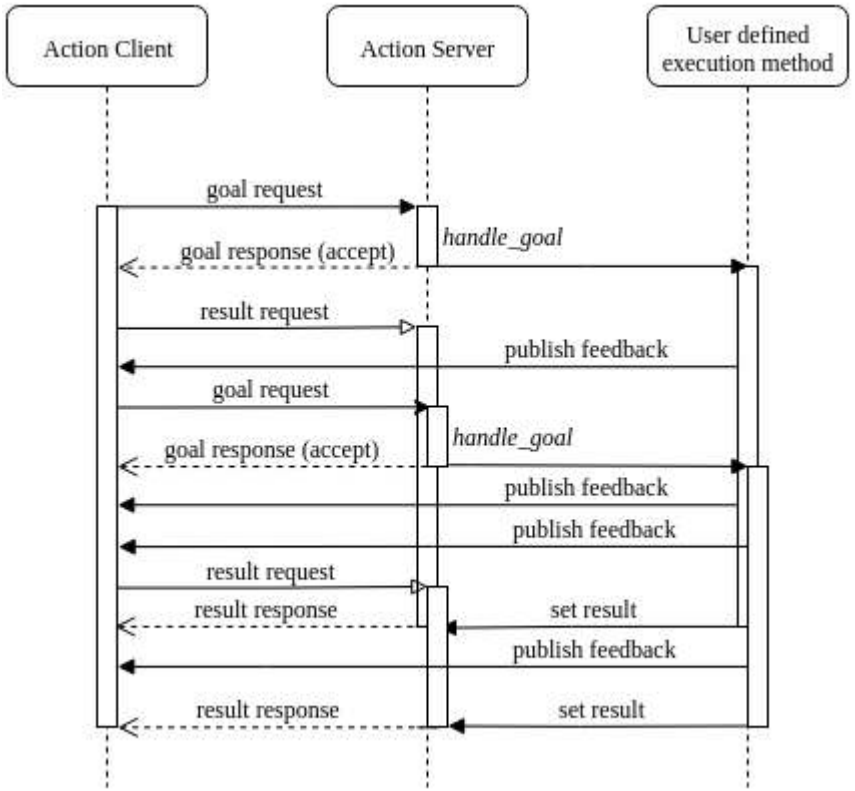
ROS2:



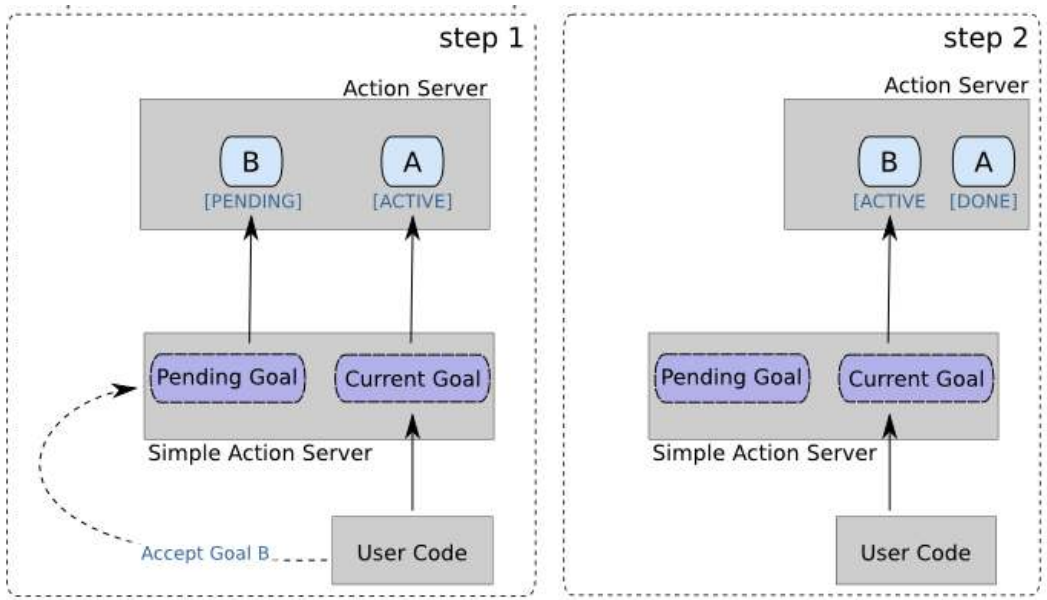
Experience with Alpha Release

Problem 2: Action interface and ROS1 Bridge

ROS2 Action Server:



ROS1 Simple Action Server:



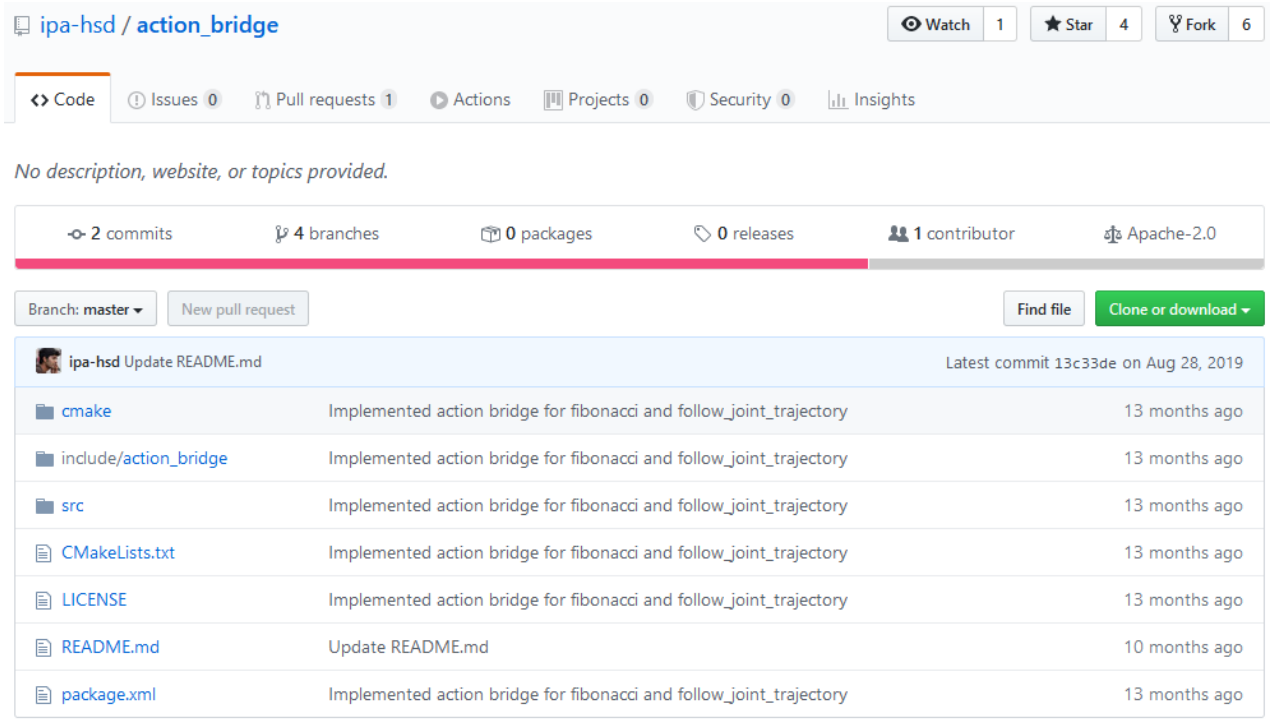
Experience with Alpha Release

Problem 3: Action interface and ROS1 Bridge

1. Goal Handler UUID
2. No topics
3. Not supported by ROS1 Bridge

ROS1 action_bridge repository:

- FollowTrajectoryAction
- FibonacciAction



https://github.com/ipa-hsd/action_bridge

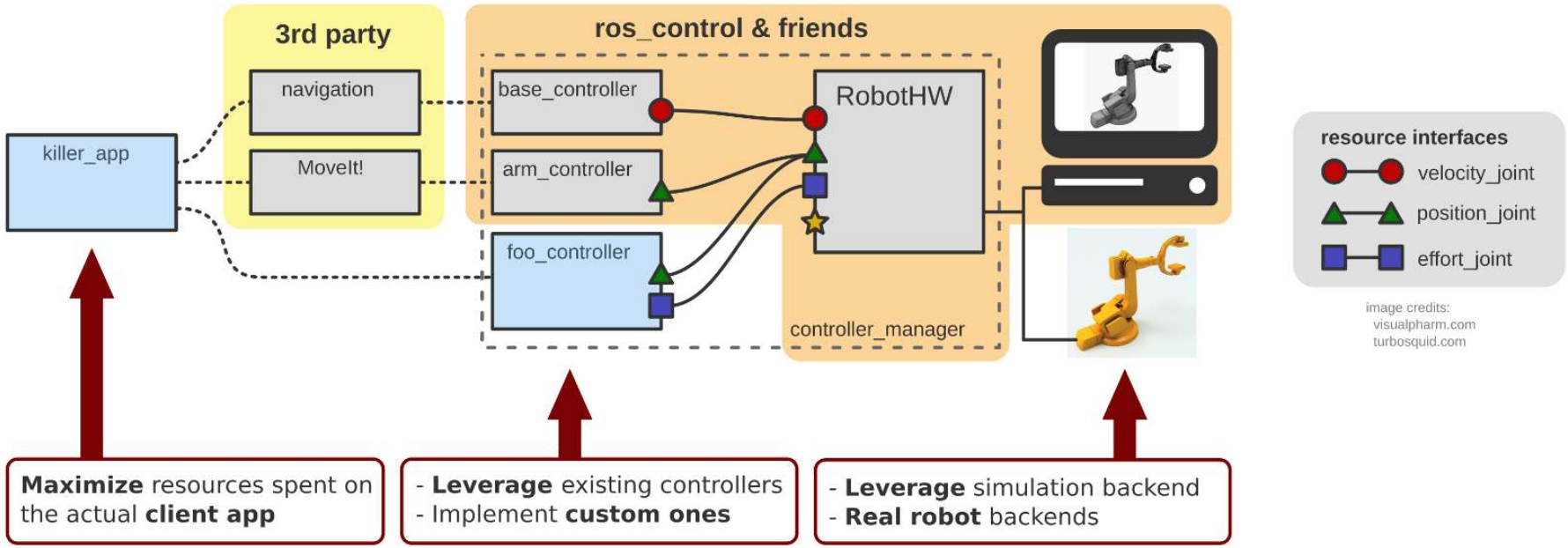
Experience with Alpha Release

Problem 4: Missing ros2_control

1. No drivers
2. No Gazebo Simulation Plugins
3. No Moveit ros2_control interface

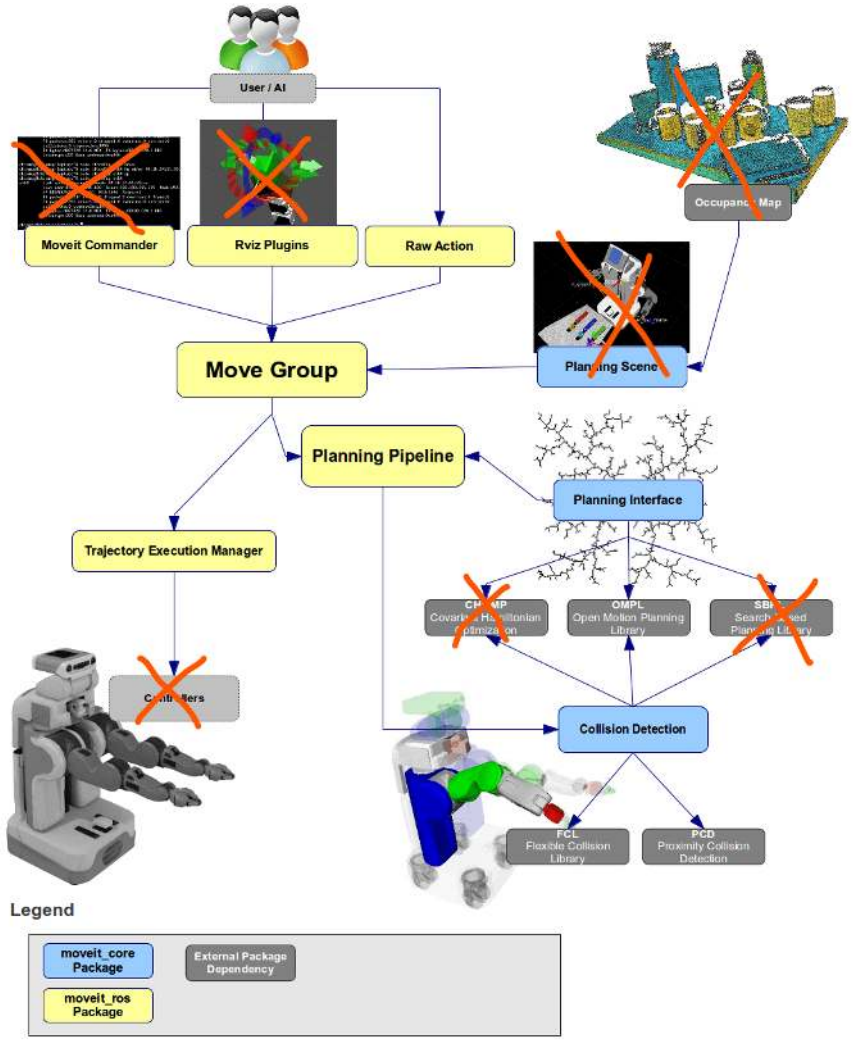
Solution:

- ros1_bridge
- action_bridge



Experience with Alpha Release

Problem 5: Other unfinished modules and missing APIs

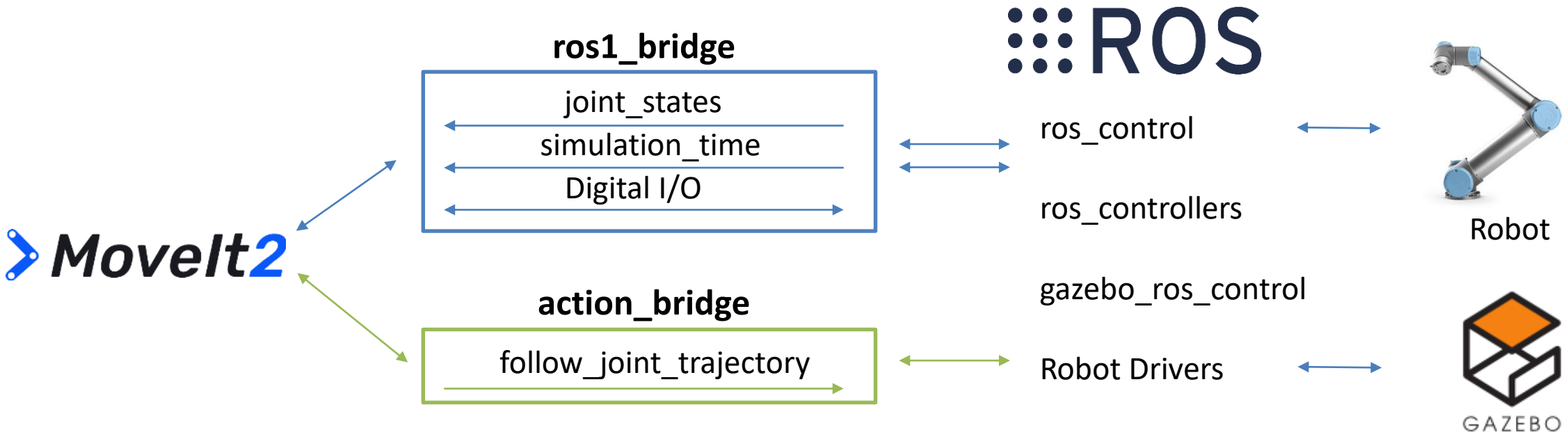


Missing APIs

- Change target reference frame
- Cartesian planning
- Named target planning
- blocked execution
- Python interface
- rviz plugins
- moveit_setup_assistance

Experience with Alpha Release

Software Architecture



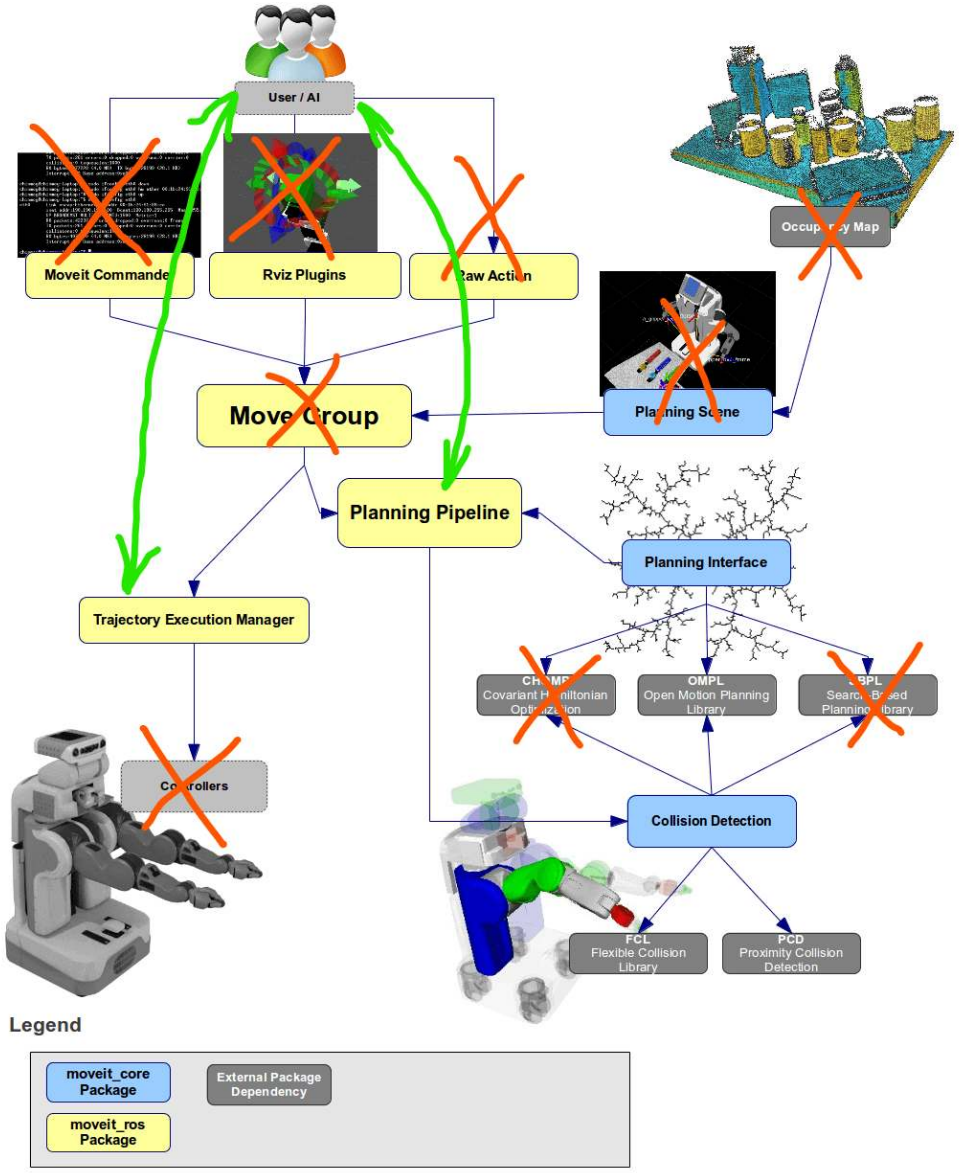
Beta 2.0 Release

Improvement

- More complete and stable APIs
- Better code quality
- Fixes

Missing APIs

- ros2_control
- move_group
- ~~Python interface~~
- rviz plugins
- moveit_setup_assistance



Beta 2.0 Release

Verdict

- Good for proof-of-concept work and showcase

- Long-term code API development needs to wait for the July

Foxy Moveit 2.1 Beta Release and an official ros2_control release

with proper action interface support .

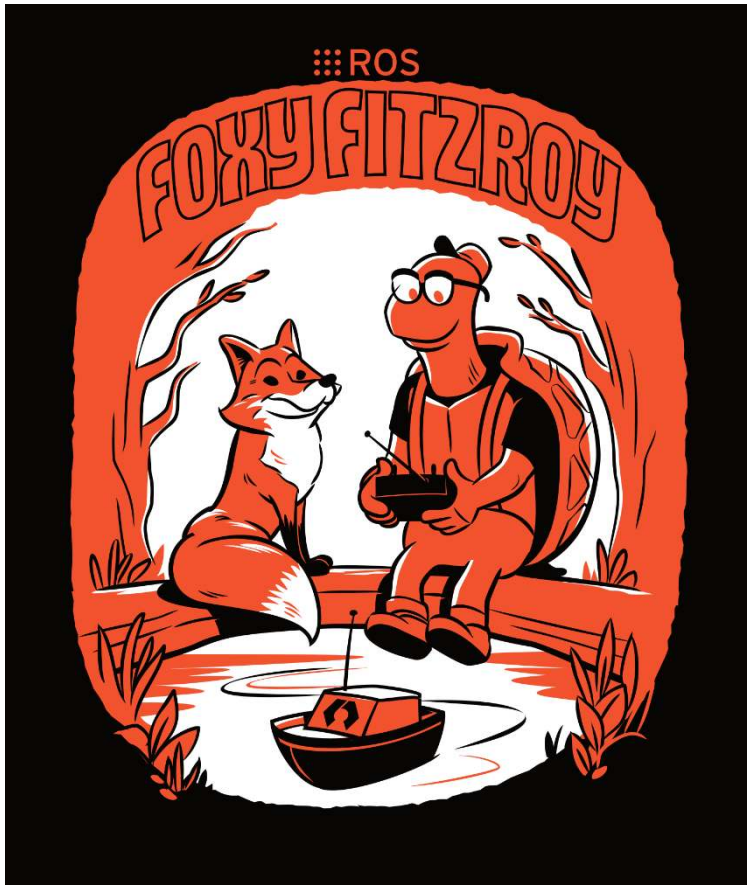
What to expect in July Release

Feature lists

- Full ROS1 Migration (move_group, rviz_plugins etc.)
- ros2_control, gazebo_ros_control
- Tutorials

Verdict

- Good for proof-of-concept work and showcase
- Good for driver development
- Start for long-term API development



Q&A

CREATING GROWTH, ENHANCING LIVES

Advanced Remanufacturing and Technology Centre
ARTC

THANK YOU

www.a-star.edu.sg