Unit & Integration Testing in ROS 2

Arne Baeyens

FOSDEM '25

intermodalics
from software to robots

Unit testing: gtest

test/test_kinematics.cpp

```cpp
#include <gtest/gtest.h>
#include <math.h>

TEST(dummysuite, dummytestcase) {
    // 2^2 == 4
    EXPECT_EQ(pow(2, 2), 4);
}

int main(int argc, char **argv) {
    ::testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

gtest macro

assert tested function

Unit testing: gtest

test/test_kinematics.cpp

```cpp
#include <gtest/gtest.h>
#include <math.h>

TEST(dummysuite, dummytestcase) {
    // 2^2 == 4
    EXPECT_EQ(pow(2, 2), 4);
}

int main(int argc, char **argv) {
    ::testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

gtest macro

assert tested function

CMakeLists.txt

```
########
# test #
########
if (BUILD_TESTING)
  # C++ unit tests
  find_package(ament_cmake_gtest REQUIRED)
  ament_add_gtest(test_kinematics test/test_kinematics.cpp)
  target_link_libraries(test_kinematics ${PROJECT_NAME})
  ament_target_dependencies(test_kinematics ${dependencies})
endif()
```

test source

register test

Integration testing:
launch_testing

Integration testing:
launch_testing

= ROS Python launchfile

test/test_simulation.py

```python
def generate_test_description():
    return launch.LaunchDescription([
        ## Gazebo simulator
        IncludeLaunchDescription(
            XMLLaunchDescriptionSource([os.path.join(
                get_package_share_directory('simulation'),
                'launch/simulator.launch.xml')]),
        ),
        ## Robot description
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(os.path.join(
                get_package_share_directory('robot_description'),
                'robot_description.launch.py'))),
        ## Hexapod gait generator
        launch_ros.actions.Node(
            package='app',
            executable='follow_velocity_rectangle',
            name='follow_velocity_rectangle',
            parameters=[
                ('use_sim_time', 'true'),
            ],
        ),
        # Launch tests
        launch_testing.actions.ReadyToTest(),
    ])
```

Gazebo simulator

code under test

Integration testing:
launch_testing

test/test_simulation.py

= ROS Python launchfile

```python
def generate_test_description():
    return launch.LaunchDescription([
        ## Gazebo simulator
        IncludeLaunchDescription(
            XMLLaunchDescriptionSource([os.path.join(
                get_package_share_directory('simulation'),
                'launch/simulator.launch.xml')]),
        ),
        ## Robot description
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(os.path.join(
                get_package_share_directory('robot_description'),
                'robot_description.launch.py'))),
        ## Hexapod gait generator
        launch_ros.actions.Node(
            package='app',
            executable='follow_velocity_rectangle',
            name='follow_velocity_rectangle',
            parameters=[
                ('use_sim_time', 'true'),
            ],
        ),
        # Launch tests
        launch_testing.actions.ReadyToTest(),
    ])
```

Gazebo simulator

code under test

```python
# Active tests
class TestHexapod(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        rclpy.init()

    @classmethod
    def tearDownClass(cls):
        rclpy.shutdown()

    def test_gazebo_started(self, proc_output):
        """Did Gazebo start properly?"""
        proc_output.assertWaitFor(
            'JointPositionController subscribing to Actuator',
            timeout=10, stream='stdout')

    def test_messages_published(self, proc_output):
        """Does the simulator publish odometry?"""
        wait_for_topics = launch_testing_ros.WaitForTopics(
            [('odom', nav_msgs.msg.Odometry)], timeout=10)
        assert wait_for_topics.wait()
        wait_for_topics.shutdown()

    def test_walk_forward(self, proc_output):
        """Does our pet walk forward sufficiently quickly?"""
        self.node = rclpy.create_node('test_hexapod')
        # publish twist reference and check pose changes
        # ... omitted for brevity
```

Integration testing:
launch_testing

= ROS Python launchfile

test/test_simulation.py

```python
def generate_test_description():
    return launch.LaunchDescription([
        ## Gazebo simulator
        IncludeLaunchDescription(
            XMLLaunchDescriptionSource([os.path.join(
                get_package_share_directory('simulation'),
                'launch/simulator.launch.xml')]),
        ),
        ## Robot description
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(os.path.join(
                get_package_share_directory('robot_description'),
                'robot_description.launch.py'))),
        ## Hexapod gait generator
        launch_ros.actions.Node(
            package='app',
            executable='follow_velocity_rectangle',
            name='follow_velocity_rectangle',
            parameters=[
                ('use_sim_time', 'true'),
            ],
        ),
        # Launch tests
        launch_testing.actions.ReadyToTest(),
    ])
```

Gazebo simulator

code under test

```python
# Active tests
class TestHexapod(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        rclpy.init()

    @classmethod
    def tearDownClass(cls):
        rclpy.shutdown()

    def test_gazebo_started(self, proc_output):
        """Did Gazebo start properly?"""
        proc_output.assertWaitFor(
            'JointPositionController subscribing to Actuator',
            timeout=10, stream='stdout')

    def test_messages_published(self, proc_output):
        """Does the simulator publish odometry?"""
        wait_for_topics = launch_testing_ros.WaitForTopics(
            [('odom', nav_msgs.msg.Odometry)], timeout=10)
        assert wait_for_topics.wait()
        wait_for_topics.shutdown()

    def test_walk_forward(self, proc_output):
        """Does our pet walk forward sufficiently quickly?"""
        self.node = rclpy.create_node('test_hexapod')
        # publish twist reference and check pose changes
        # ... omitted for brevity

# Post-shutdown tests
@launch_testing.post_shutdown_test()
class TestHexapodSimShutdown(unittest.TestCase):
    def test_exit_codes(self, proc_info):
        """Check whether all processes exited normally."""
        launch_testing.asserts.assertExitCodes(proc_info)
```

Test isolation

! launch_testing: no test isolation !

CMakeLists.txt

```
1  ########
2  # test #
3  ########
4  if(BUILD_TESTING)
5    # Integration tests
6    find_package(launch_testing_ament_cmake REQUIRED)
7    add_launch_test(test/test_in_simulation.py)
8  endif()
```

register integration test

all tests same ROS_DOMAIN_ID
=> crosstalk

Test isolation

! launch_testing: no test isolation !

CMakeLists.txt

```
1  ########
2  # test #
3  ########
4  if(BUILD_TESTING)
5    # Integration tests
6    find_package(launch_testing_ament_cmake REQUIRED)
7    add_launch_test(test/test_in_simulation.py)
8  endif()
```

register integration test

all tests same ROS_DOMAIN_ID
=> crosstalk

CMakeLists.txt

```
11 ########
12 # test #
13 ########
14 if(BUILD_TESTING)
15   # Integration tests
16   find_package(ament_cmake_ros REQUIRED)
17   find_package(launch_testing_ament_cmake REQUIRED)
18   function(add_ros_isolated_launch_test path)
19     set(RUNNER "${ament_cmake_ros_DIR}/run_test_isolated.py")
20     add_launch_test("${path}" RUNNER "${RUNNER}" ${ARGN})
21   endfunction()
22   add_ros_isolated_launch_test(test/test_in_simulation.py)
23 endif()
```

test runner ensures unique
ROS_DOMAIN_ID

Test isolation

! launch_testing: no test isolation !

CMakeLists.txt

```
1  ########
2  # test #
3  ########
4  if(BUILD_TESTING)
5    # Integration tests
6    find_package(launch_testing_ament_cmake REQUIRED)
7    add_launch_test(test/test_in_simulation.py)
8  endif()
```

register integration test

all tests same ROS_DOMAIN_ID
=> crosstalk

CMakeLists.txt

```
11 ########
12 # test #
13 ########
14 if(BUILD_TESTING)
15   # Integration tests
16   find_package(ament_cmake_ros REQUIRED)
17   find_package(launch_testing_ament_cmake REQUIRED)
18   function(add_ros_isolated_launch_test path)
19     set(RUNNER "${ament_cmake_ros_DIR}/run_test_isolated.py")
20     add_launch_test("${path}" RUNNER "${RUNNER}" ${ARGN})
21   endfunction()
22   add_ros_isolated_launch_test(test/test_in_simulation.py)
23 endif()
```

test runner ensures unique ROS_DOMAIN_ID

WIP to robustly run parallel tests: [testjob_coordinator]

Let’s run tests

```
$ colcon test
```

sudo play the video

View test results

each test => xUnit file

```
$ xunit-viewer -r build -c
```

View test results

each test => xUnit file

```
$ xunit-viewer -r build -c
 app.test_in_simulation.launch_tests[forward]
     ✓ test_exit_codes[forward] time=0.0
     ✓ test_gazebo_started[forward] time=2.518
     ✓ test_messages_published[forward] time=0.186
     ✓ test_walk_around[forward] time=6.041

 hexapod_kinematics
     disabled=0
     timestamp=2025-01-26T16:03:15.274

     ✓ parse_robot_description time=0.031
     ✓ run_inverse_kinematics time=0.009

6 passed
Written to: /home/arend/hexapod/index.html _
```

fancy web page

Unit & Integration Testing in ROS 2

Example repo’s

github.com/abaeyens/ros2-integration-testing-examples

github.com/abaeyens/hexapod-MPC

Unit & Integration Testing in ROS 2

Example repo's

github.com/abaeyens/ros2-integration-testing-examples

github.com/abaeyens/hexapod

Official ROS 2 docs

docs.ros.org/en/rolling/Tutorials/Intermediate/Testing/Cpp.html

docs.ros.org/en/rolling/Tutorials/Intermediate/Testing/Integration.html