Apex.AI

launch_testing in ROS 2

Presenter: Sumanth Nirmal
Authors: Peter Baughman,
William Woodall, Michel Hidalgo

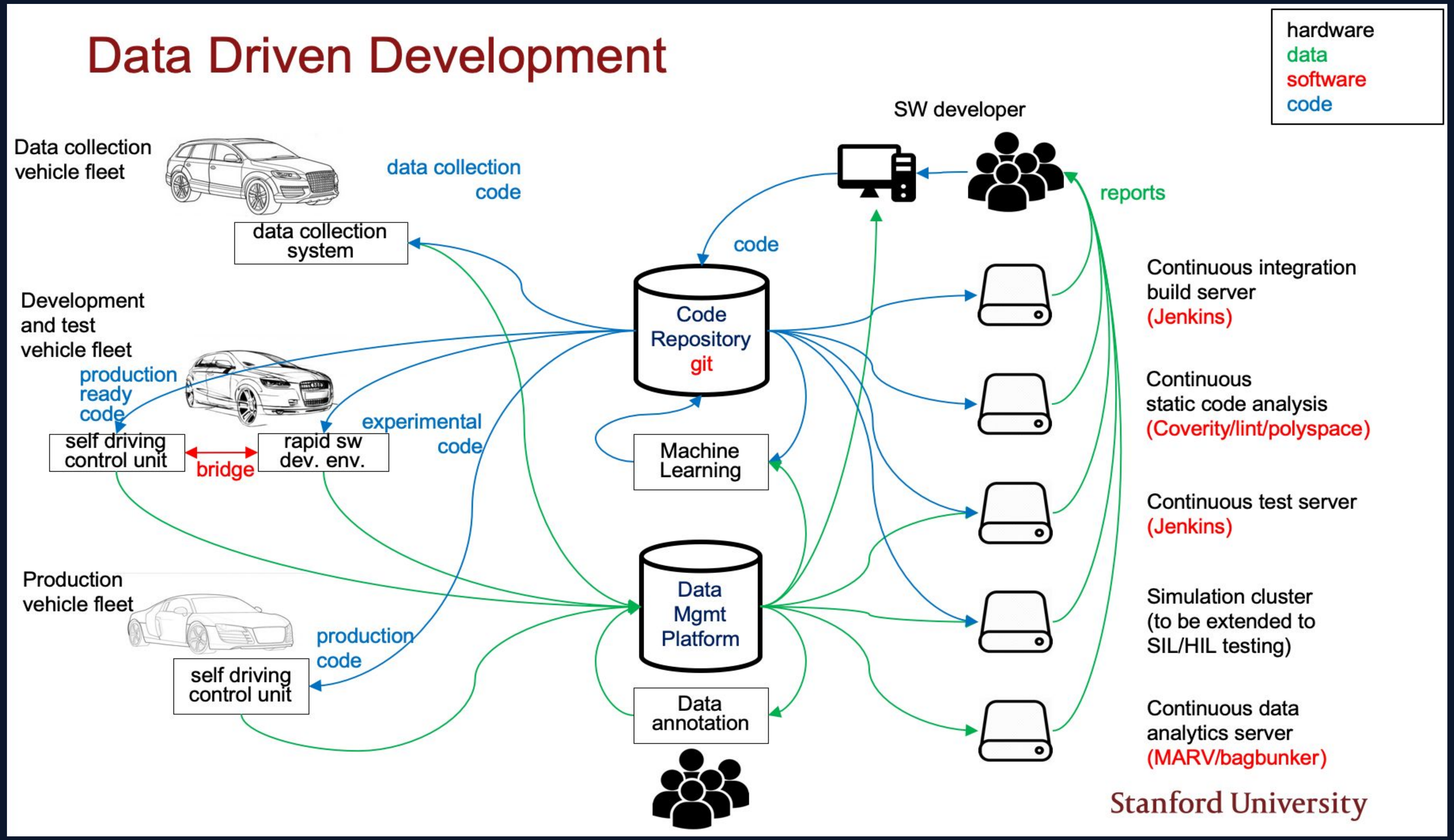
Agenda

- Integration testing
- *rostest*
- ROS 2 launch
- *rostest* improvements
- *launch_test*
- *launch_test* example
- *launch_test* demo

Motivation for Integration Testing

Integration testing is where software modules are integrated logically and tested as a group

- Clear API boundary between processes being tested
- Integration tests checks the real program flow
- Integration tests check the real user interactions
- Unit tests often uses bespoke configurations and can miss issues with real configurations
- Unit tests are often single-threaded, and don't expose complex race conditions
- Performance testing



Data Driven Development
hardware
data
software
code
SW developer
Data collection vehicle fleet
data collection code
data collection system
reports
code
Development and test vehicle fleet
production ready code
self driving control unit
bridge
rapid sw dev. env.
experimental code
Code Repository git
Machine Learning
Continuous integration build server (Jenkins)
Continuous static code analysis (Coverity/lint/polyspace)
Continuous test server (Jenkins)
Production vehicle fleet
production code
self driving control unit
Data Mgmt Platform
Data annotation
Simulation cluster (to be extended to SIL/HIL testing)
Continuous data analytics server (MARV/bagbunker)
Stanford University

ME 302B: Stanford University; Jan Becker

Lack of Integration tests!
2 Unit tests, 0 Integration tests

rostest (from ROS 1)

- It is an integration test suite based on ***roslaunch***
- Has <*test*> tag in the xml file, which specifies the test nodes to run
- *rostest's roscore* is restarted for each test
- By default *rostest* uses random ports, so many *rostest's* can run in parallel and in isolation
- Launches all of the processes from a launch (xml) file, then waits for the test process to finish
- The final exit code is based on the exit code of the test process

launch and launch_ros in ROS 2

- ***launch*** is a ROS-agnostic tool to launch processes
- ***launch_ros*** is a tool to launch ROS nodes
- ROS 2 launch is exposed as a Python API
- ROS 2 launch API is extendable and supports adding custom actions and events
- ROS masters are no longer required for nodes to communicate
- ***launch*** can use events for feedback and have actions that create other actions
- ***launch*** descriptions can be introspected

New Challenges:

- ROS 2 launch descriptions can be arbitrarily complex and they themselves need to be tested

Rough Edges In rostest

- Tests might run before the processes under test were ready
  - ➢ This results in a test failure, which might be hard to reproduce
- Processes under test might crash, and tests would not notice this crash
  - ➢ Results in a test failure, which would be difficult to reproduce
- Debug output like *stdout* or *stderr* from tests might be hard to find
  - ➢ This makes it difficult to analyse the *stdout* or *stderr* streams to align output with the test cases

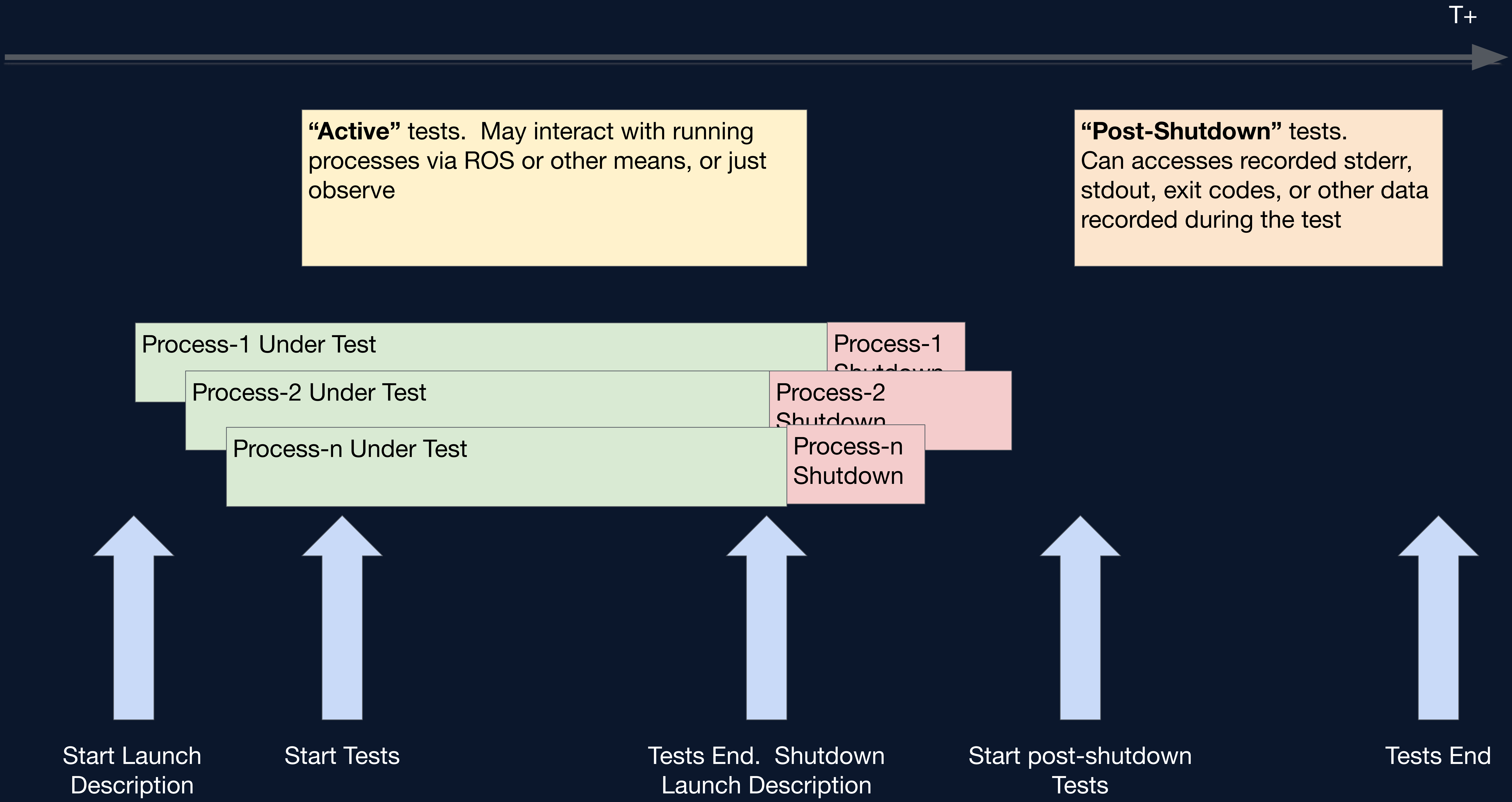
Improvements on rostest Functionality

- What if there was a way for the test author to explicitly coordinate when the test runs with the processes under test?
  - ➢ ***launch_testing*** has a ***ReadyToTest()*** action that coordinates test start with the rest of the launch
- What if tests could access process information like the exit codes of launched processes?
  - ➢ ***launch_testing*** has a **ProcInfoHandler** object, which has exit code information for processes that were launched
- What if tests could access process information like *stdout* or *stderr* of launched processes?
  - ➢ ***launch_testing*** has an **IOHandler** object, which has *stdout/stderr* information for processes that were launched
- What if we could automatically generate test cases for the launched processes?
  - ➢ Ability to introspect the launch description makes it easy to programmatically generate test cases

launch_testing

- ***launch_testing*** is a framework for integration testing based on the ROS 2 launch system
- The exit codes, *stdout* and *stderr* of all processes launched are available to the tests
- The command-line used to launch the processes are available to the tests
- Tests can run concurrently with the launched processes, or can run after the launched processes are shut down
- Output looks like regular unit-test output; each test case gets a line
- Parametrize the launch description
- ***launch_testing_ros*** can automatically generate a unique *ROS_DOMAIN_ID* to isolate tests from one-another to be able to run the tests in parallel and in isolation

Timeline of a launch_test Command


T+
"Active" tests. May interact with running processes via ROS or other means, or just observe
"Post-Shutdown" tests. Can accesses recorded stderr, stdout, exit codes, or other data recorded during the test
Process-1 Under Test
Process-1 Shutdown
Process-2 Under Test
Process-2 Shutdown
Process-n Under Test
Process-n Shutdown
Start Launch Description
Start Tests
Tests End. Shutdown Launch Description
Start post-shutdown Tests
Tests End

Using the launch_testing API

- Auto-generate test cases from documentation
  - Check that launch files in documentation continue to work as documented
  - Check that process arguments and values continue to work as documented
  - Check that documented topics are present in a launched system
  - Check that output generated by processes matches what's in the documentation
- Auto-generate tests that always run for every process tested automatically
  - Check output for errors
  - Check exit codes of all processes
- Automatically remap topics and insert nodes that fuzz the data before republishing, or drop messages

Possible Future Development

- Add ROS 1 style test processes
  - Run *gtests* in a separate process
  - Run *pytests* in a separate process
  - Run an arbitrary process that can pass/fail based on an exit code
- Include tests in the launch description
  - These tests could easily introspect the launch context to help test the launch description itself
- *rosbag* integration to record data for analysis in post-shutdown tests

Example: Launch Test

# Part 1/3

# Package: launch/launch_testing/examples/good_proc.test.py

```
$ # can be run with:
$ launch_test launch_testing/examples/good_proc.test.py
```

```
def generate_test_description():
  return launch.LaunchDescription([
      launch.actions.ExecuteProcess(
          cmd=[path_to_process],
      ),

      # Start tests right away - no need to wait for anything in this example.
      # In a more complicated launch description, we might want this action happen
      # once some process starts or once some other event happens
      launch_testing.actions.ReadyToTest()
  ])
```

Example: Launch Test (con't)

# Part 2/3

# Active test

```
# These tests will run concurrently with the test in process.  After all these tests are done,
# the launch system will shut down the processes that it started up
class TestGoodProcess(unittest.TestCase):

  def test_count_to_four(self, proc_info, proc_output):
      # This will match stdout from any process.  In this example there is only one process
      # running
      proc_output.assertWaitFor('Loop 1', timeout=10)
      proc_output.assertWaitFor('Loop 2', timeout=10)
      proc_output.assertWaitFor('Loop 3', timeout=10)
      proc_output.assertWaitFor('Loop 4', timeout=10)
```

Inherits from ***unittest.TestCase***

ProcInfoHandler object

IOHandler object

Waits for a particular output on ***proc_info***

Example: Launch Test (con't)

# Part 3/3

# Post ShutDown tests

Decorated with ***post_shutdown_test*** descriptor

Inherits from ***unittest.TestCase***

```
# These tests will run after with the test in process is shut-down.
@launch_testing.post_shutdown_test()
class TestProcessOutput(unittest.TestCase):

  def test_exit_code(self, proc_info, proc_output):
      # Check that all processes in the launch (in this case, there's just one) exit
      # with code 0
      launch_testing.asserts.assertExitCodes(proc_info)

  def test_out_of_order(self, proc_info, proc_output):
      # This demonstrates that we notice out-of-order IO
      with self.assertRaisesRegexp(AssertionError, "'Loop 2' not found"):
          with assertSequentialStdout(proc_output, <process>) as cm:
              cm.assertInStdout('Loop 1')
              cm.assertInStdout('Loop 3')
              cm.assertInStdout('Loop 2')  # This should raise
```

Asserts that the specified process exited with a particular exit code.

Asserts that *stdout* was seen in a particular order.

Asserts that a message is found in the *stdout* of the process

Using CMake to run launch_test

- Declare a dependency on ***launch_testing_ament_cmake*** in *package.xml*

```
<test_depend>launch_testing_ament_cmake</test_depend>
```

- In *CMakeLists.txt* file, add

```
find_package(launch_testing_ament_cmake)
add_launch_test(test/name_of_test.test.py)
```

- Optionally, arguments can also be passed to the tests:

```
add_launch_test(
  test/test_with_args.test.py
  ARGS "arg1:=foo"
)
```

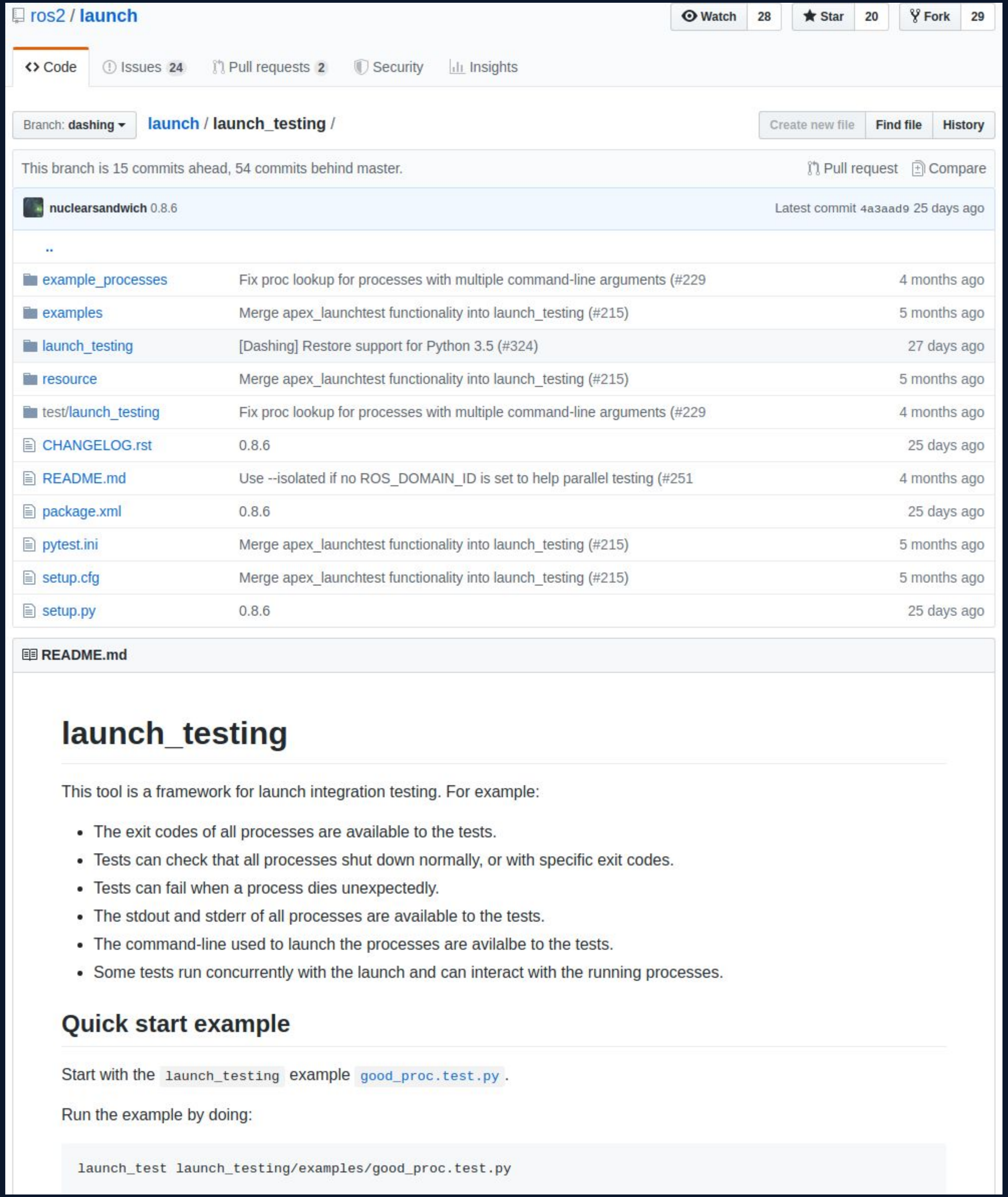
Launch arguments to be passed to the launch test

Live Demo: Launch Test

Quick demonstration of launch test

Documentation and the ROS Buildfarm

Documentation:



ros2 / launch

Watch 28 | Star 20 | Fork 29

Code | Issues 24 | Pull requests 2 | Security | Insights

Branch: dashing ▾ launch / launch_testing /

Create new file | Find file | History

This branch is 15 commits ahead, 54 commits behind master. Pull request | Compare

nuclearsandwich 0.8.6 Latest commit 4a3aad9 25 days ago

..

example_processes	Fix proc lookup for processes with multiple command-line arguments (#229	4 months ago
examples	Merge apex_launchtest functionality into launch_testing (#215)	5 months ago
launch_testing	[Dashing] Restore support for Python 3.5 (#324)	27 days ago
resource	Merge apex_launchtest functionality into launch_testing (#215)	5 months ago
test/launch_testing	Fix proc lookup for processes with multiple command-line arguments (#229	4 months ago
CHANGELOG.rst	0.8.6	25 days ago
README.md	Use --isolated if no ROS_DOMAIN_ID is set to help parallel testing (#251	4 months ago
package.xml	0.8.6	25 days ago
pytest.ini	Merge apex_launchtest functionality into launch_testing (#215)	5 months ago
setup.cfg	Merge apex_launchtest functionality into launch_testing (#215)	5 months ago
setup.py	0.8.6	25 days ago

README.md

launch_testing

This tool is a framework for launch integration testing. For example:

- The exit codes of all processes are available to the tests.
- Tests can check that all processes shut down normally, or with specific exit codes.
- Tests can fail when a process dies unexpectedly.
- The stdout and stderr of all processes are available to the tests.
- The command-line used to launch the processes are avilalbe to the tests.
- Some tests run concurrently with the launch and can interact with the running processes.

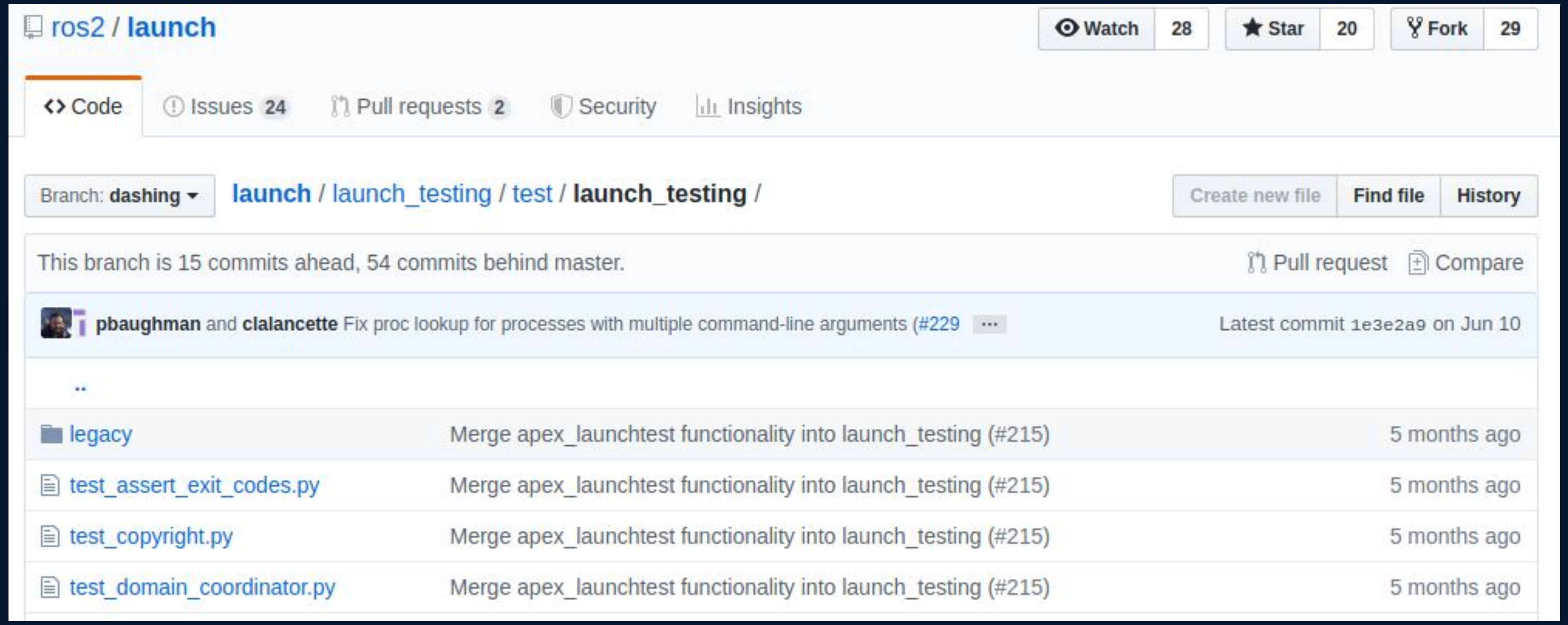
Quick start example

Start with the `launch_testing` example `good_proc.test.py`.

Run the example by doing:

```
launch_test launch_testing/examples/good_proc.test.py
```

Example packages using *launch_testing* in ROS Buildfarm:



ros2 / launch

Watch 28 | Star 20 | Fork 29

Code | Issues 24 | Pull requests 2 | Security | Insights

Branch: dashing ▾ launch / launch_testing / test / launch_testing /

Create new file | Find file | History

This branch is 15 commits ahead, 54 commits behind master. Pull request | Compare

pbaughman and clalancette Fix proc lookup for processes with multiple command-line arguments (#229 … Latest commit 1e3e2a9 on Jun 10

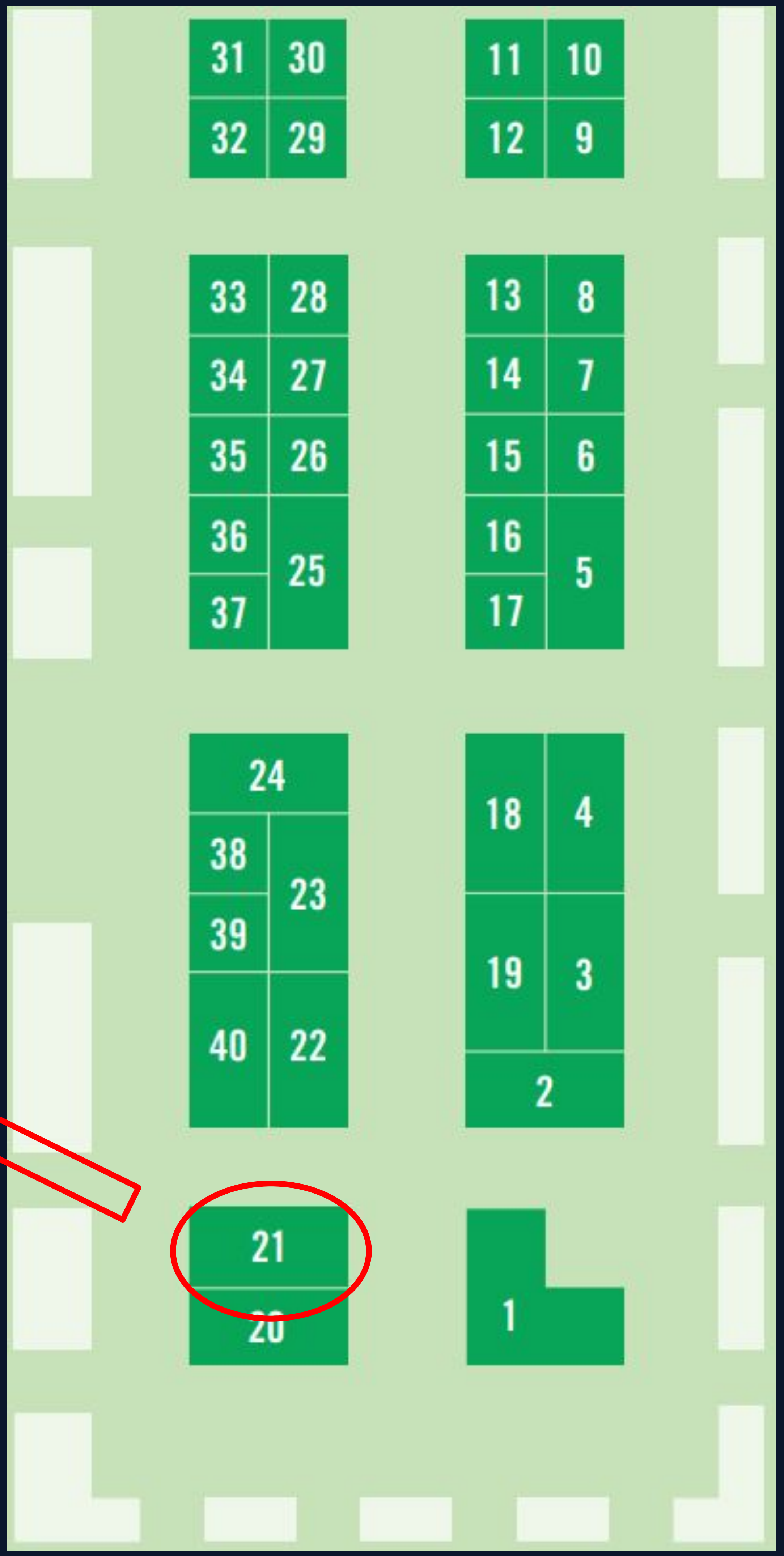
..

legacy	Merge apex_launchtest functionality into launch_testing (#215)	5 months ago
test_assert_exit_codes.py	Merge apex_launchtest functionality into launch_testing (#215)	5 months ago
test_copyright.py	Merge apex_launchtest functionality into launch_testing (#215)	5 months ago
test_domain_coordinator.py	Merge apex_launchtest functionality into launch_testing (#215)	5 months ago

1. https://github.com/ros2/rcutils/blob/dashing/test/test_logging_output_format.py
2. https://github.com/ros2/rcl/blob/dashing/rcl/test/rcl/test_two_executables.py.in
3. https://github.com/ros2/demos/blob/dashing/demo_nodes_cpp/test/test_executables_tutorial.py.in

More About Us

Apex.AI®


31
30
11
10
32
29
12
9
33
28
13
8
34
27
14
7
35
26
15
6
36
25
16
5
37
17
24
18
4
38
23
39
19
3
40
22
2
21
20
1