Software Testing

Calle Rosenquist

Software Testing
Outline

- *Automated* Testing Support
  - Unit & Integration Tests
- Difference
- Tools
- Examples
- Recommendations

Software Testing
Comparison

Unit Tests

- Test scope small
- Build-phase
- Runtime env: Builder
- Fast
- Language specific
- Repro Difficulty: Easy

Integration Tests

- Test scope arbitrarily big
- Post-installation
- Runtime env: User
- Language agnostic
- Tests make use of
  - IPC
  - Services
  - Simulators
  - Hardware
- Repro Difficulty: Harder

Why Unit Test?

Why Integration Test?

Unit Tests

- Built and executed (`waf test`)
  - Reuses Dependency Information
  - Creates Test Environment
    - LD_LIBRARY_PATH/PYTHONPATH
- Test Runners/Frameworks
  - Google Test for C++
  - Nose for Python
  - *Qt Test for Qt*

Unit Tests

- Tests in waf module

```
rad/cpp/core$
|-- src
|   |-- AnyEvent.cpp
|   |-- ...
|   `-- Timer.cpp
|-- test
|   |-- TestEvents.hpp
|   |-- ...
|   `-- TestTimer.cpp
`-- wscript
```

```
py/rad$
|-- src
|   `-- rad
|-- test
|   |-- test_dispatcher.py
|   |-- ...
|   `-- test_subscriber.py
`-- wscript
```

Unit Tests

```
Output from test "/tmp/rad.git/build/rad/cpp/sm/radsm_test" (exitcode: 0)
Running main() from gmock_main.cc
[==========] Running 27 tests from 8 test cases.
[----------] Global test environment set-up.
[----------] 1 test from TestActionGroup
[ RUN      ] TestActionGroup.GetAndSetId
[       OK ] TestActionGroup.GetAndSetId (0 ms)
[----------] 1 test from TestActionGroup (0 ms total)

Output from test "/tmp/rad.git/scxml4py/src/scxml4py/__init__.py" (exitcode: 0)

.....................................................................
---------------------------------------------------------------------
Ran 71 tests in 0.822s

OK

execution summary
  tests that pass 9/9
    /tmp/rad.git/build/rad/cpp/core/radcore_test
    /tmp/rad.git/build/rad/cpp/mal/radmal_test
    /tmp/rad.git/build/rad/cpp/services/radservices_test
    /tmp/rad.git/build/rad/cpp/utils/radutils_test
    /tmp/rad.git/build/scxml4cpp/engine/scxmlengine_test
    /tmp/rad.git/build/scxml4cpp/parser/scxmlparser_test
    /tmp/rad.git/build/rad/cpp/sm/radsm_test
    /tmp/rad.git/rad/py/rad/src/rad/__init__.py
    /tmp/rad.git/scxml4py/src/scxml4py/__init__.py
  tests that fail 0/9
'test' finished successfully (1m1.015s)
```

Google Test

- Creates test runner application
- Module sources in /src and /test compiled together:
  - src/main.cpp will be filtered out

Google Test Features

- Test Suites/Fixtures w/ setup/teardown
- Test Cases
- Expectations
- Assertions
- Mocking

Unit Tests for C++

Google Test Example

```
class Fixture : ::testing:Test {
  void SetUp() override {
     m_vec.push_back(2);
     m_vec.push_back(2);
  }
  void TearDown() override {}

  std::vector<int> m_vec;
};

TEST_F(Fixture, TestAccumulate) {
  ASSERT_EQ(2, m_vec.size());
  EXPECT_EQ(4, std::accumulate(std::begin(m_vec),
                               std::end(m_vec),
                               0);
}
```

Unit Tests for C++

Google Test Example

```
TEST(TestAccumulate) {
  std:vector<int> vec;
  ...

  EXPECT_EQ(2, vec.size()) << "vector size not what is expected";
  ...
}
```

Mock Objects

- Replaces real objects
- Can be programmed with expectations

Unit Tests for C++

Google Mock

```
class Validator {
  virtual bool IsValid(std::string_view);
};

class Parser {
  Parser(Validator&);

  bool Parse(std::string_view input) {
    ...
    if (!m_validator.IsValid(input)) {
        return false;
    }
    ...
  }
};
```

Unit Tests for C++

Google Mock

```
class MockValidator : public Validator {
  MOCK_METHOD1(bool, IsValid, (std::string_view));
};

TEST(ParseShouldFailIfValidationFail) {
  // Setup
  MockValidator mockValidator;

  // Expect
  EXPECT_CALL(mockValidator, IsValid(_))
    .WillRepeatedly(Return(false));

  // Run
  Parser parse(mockValidator);
  EXPECT_FALSE(parse.Parse("test"));
}
```

_ = Match any

Unit Tests for C++

Google Test Mocking

```
EXPECT_CALL(mock-object, method (matchers)?)
     .With(multi-argument-matcher)  ?
     .Times(cardinality)            ?
     .InSequence(sequences)         *
     .After(expectations)           *
     .WillOnce(action)              *
     .WillRepeatedly(action)        ?
     .RetiresOnSaturation();        ?
```

https://github.com/google/googletest/blob/master/googlemock/docs/cheat_sheet.md

Unit Tests for Python

Nose

- Nose (or soon maybe pytest)
  - Supports doctest and built-in unit tests and mocking
    - doctests are enabled with attribute `` `with_doctest=True` ``
  - Tests (files, directories and class names) are discovered with regexp: `"(?:\b|_)[Tt]est"`
    - `class TestFoo: pass # test`
    - `class Also_Test: pass # test`

Unit Tests for Python

Nose

```
class TestWithFixture(unittest.Testcase):
  """Using standard library unittest"""
  def setUp(self):
    pass

  def tearDown(self):
    pass

  def test_split(self):
    s = 'hello world'
    self.assertEqual(s.split(), ['hello', 'world'])
```

Unit Tests for Python

Nose

```
def test_add():
  """ Using nose test discovery"""
  assert 2 + 2 = 4

class TestWithFixture:
  """Using fixture with Nose"""
  def setup(self):
    pass

  def teardown(self):
    pass

  def test_split(self):
    s = 'hello world'
    nose.assert_equal(s.split(), ['hello', 'world'])
```

Unit Tests for Python

Nose

```
def Add(a, b):
    """Adds a and b

    >>> Add(1, 2)
    3
    >>> Add('foo', 'bar')
    'foobar'
    """
    return a + b
```

Unit Tests for Python

Mocking

```
>>> from unittest.mock import *
>>> mock = MagicMock(spec=['foo', 'bar'])
>>> mock.foo
<MagicMock name='mock.foo' id='140649922465520'>
>>> mock.baz
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/opt/anaconda3/lib/python3.5/unittest/mock.py", line 580, in
__getattr__
    raise AttributeError("Mock object has no attribute %r" % name)
AttributeError: Mock object has no attribute 'baz'
```

Unit Tests for Python

Mocking

```
>>> from unittest.mock import *
>>> class Foo:
...     attr = 'attr'
...     def bar(self):
...         print('bar')
...
>>> mock = create_autospec(Foo)
>>> mock.attr
<NonCallableMagicMock name='mock.attr' spec='str' id='14064...'>
>>> mock.bar
<MagicMock name='mock.bar' spec='function' id='14064...'>
```

Unit Tests
Recommendations & Tips

- Do
  - Write small test cases
  - Write testable code (TDD)
  - Reduce boiler plate code with fixtures
  - Log additional information on failures
- Don't
  - Over constrain mocks
  - Assume a specific timing
  - Assume a DISPLAY

Integration Testing
Robot Framework

- Test Automation Framework
- Keyword Driven

```
*** Settings ***
Documentation     A test suite with a single test for valid login.
...
...               This test has a workflow that is created using keywords in
...               the imported resource file.
Resource          resource.robot

*** Test Cases ***
Valid Login
    Open Browser To Login Page
    Input Username    demo
    Input Password    mode
    Submit Credentials
    Welcome Page Should Be Open
    [Teardown]    Close Browser
```

Integration Testing
Robot Framework

- Python
- Extensible

```
*** Settings ***
Library    EtrUtils

*** Test Cases ***

    ${port} =  Get Random Port
    ...
```

Integration Testing
Robot Framework

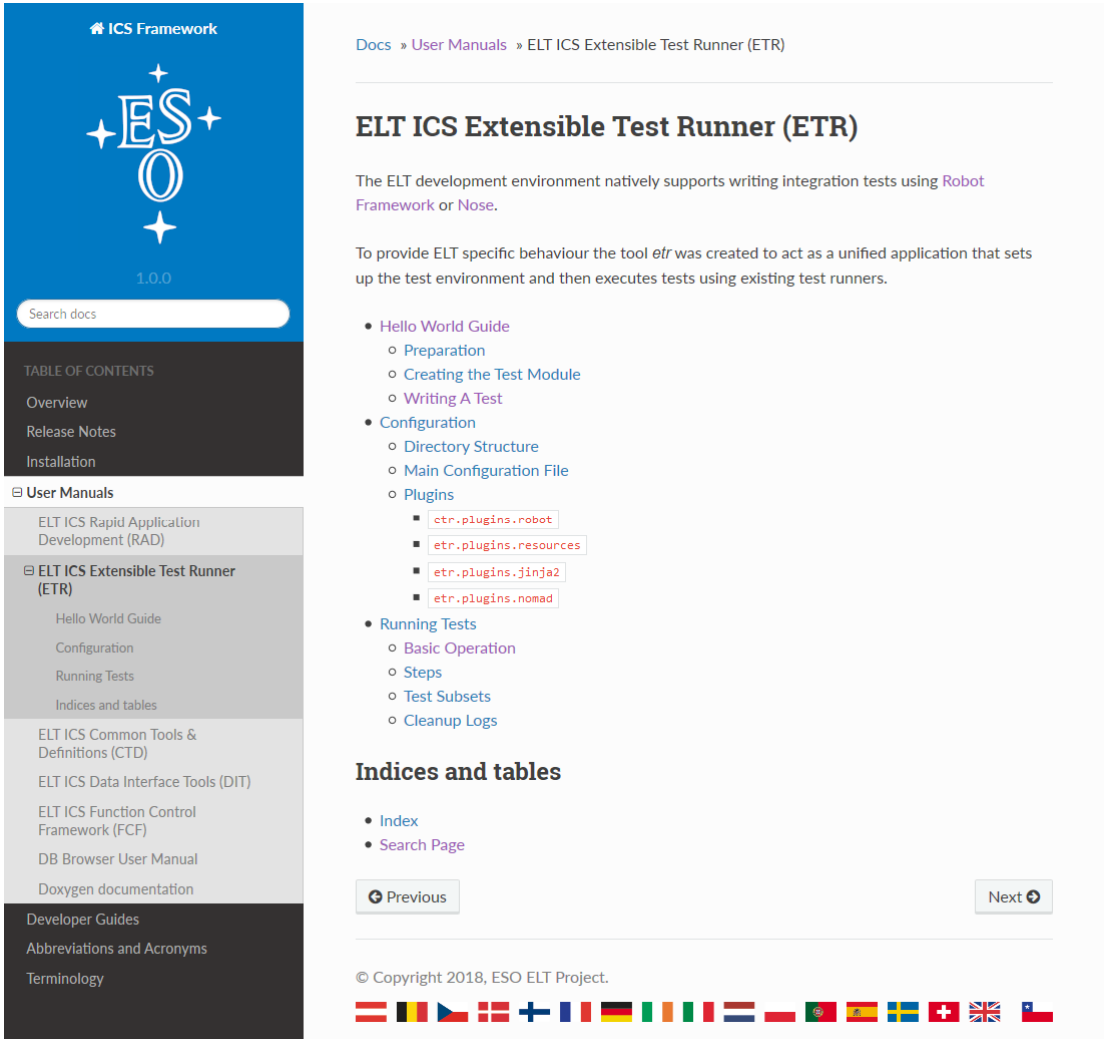
```
import socket
import logging
from contextlib import closing

class EtrUtils:
    """Robot Framework Library of utilities provided by etr
    """
    def get_random_port(self):
        """Returns a random free port
        """
```

Integration Testing

Extensible Test Runner `etr`

- Test Framework Abstraction
- Global Test Setup
  - Acquire Test Resources
  - Software Deployment
- Supports
  - Robot Framework
  - Nose

ETR



ICS Framework

1.0.0

Search docs

TABLE OF CONTENTS

- Overview
- Release Notes
- Installation
- User Manuals
  - ELT ICS Rapid Application Development (RAD)
  - ELT ICS Extensible Test Runner (ETR)
    - Hello World Guide
    - Configuration
    - Running Tests
    - Indices and tables
  - ELT ICS Common Tools & Definitions (CTD)
  - ELT ICS Data Interface Tools (DIT)
  - ELT ICS Function Control Framework (FCF)
  - DB Browser User Manual
  - Doxygen documentation
- Developer Guides
- Abbreviations and Acronyms
- Terminology

Docs » User Manuals » ELT ICS Extensible Test Runner (ETR)

ELT ICS Extensible Test Runner (ETR)

The ELT development environment natively supports writing integration tests using Robot Framework or Nose.

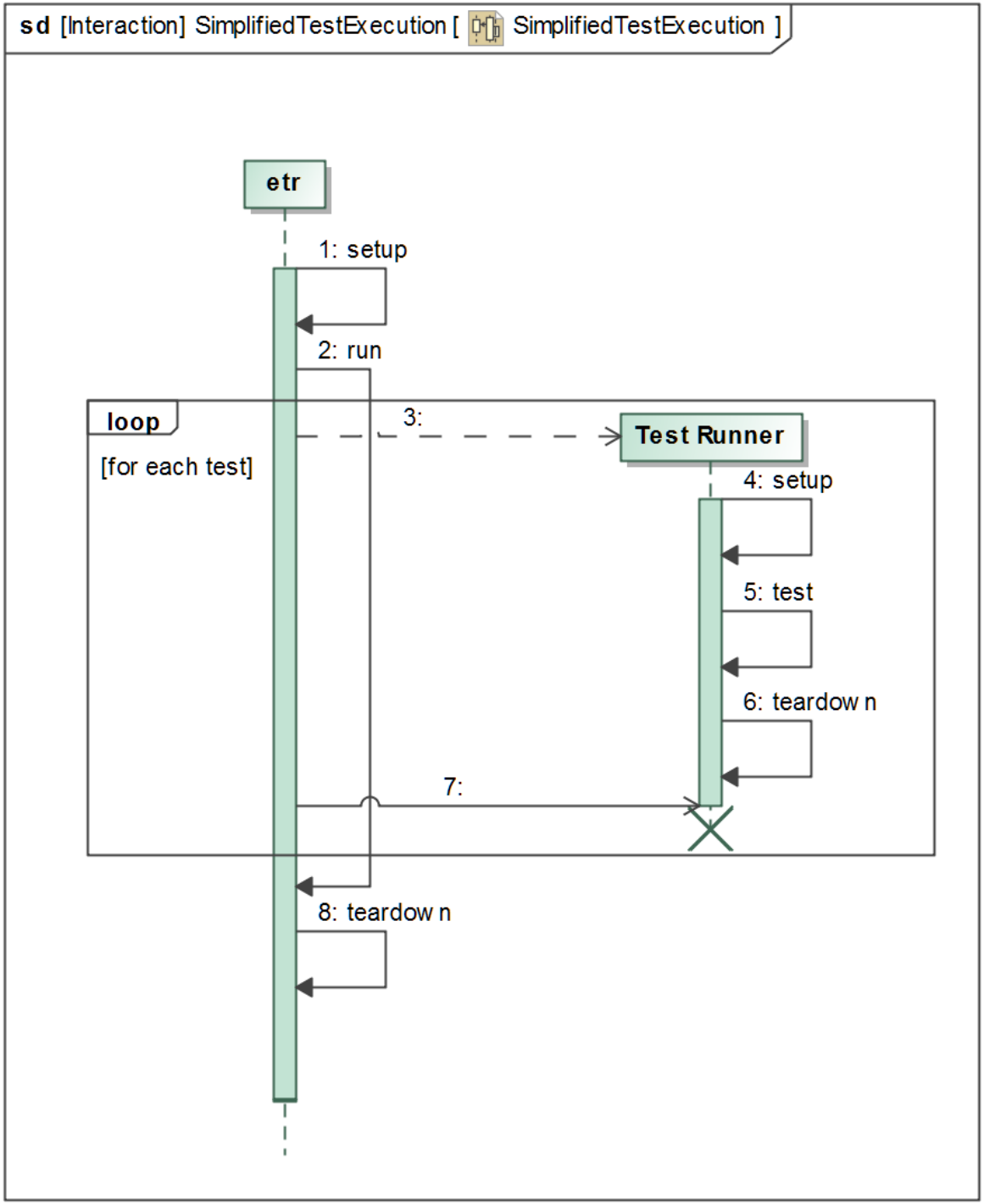
To provide ELT specific behaviour the tool *etr* was created to act as a unified application that sets up the test environment and then executes tests using existing test runners.

- Hello World Guide
  - Preparation
  - Creating the Test Module
  - Writing A Test
- Configuration
  - Directory Structure
  - Main Configuration File
  - Plugins
    - `etr.plugins.robot`
    - `etr.plugins.resources`
    - `etr.plugins.jinja2`
    - `etr.plugins.nomad`
- Running Tests
  - Basic Operation
  - Steps
  - Test Subsets
  - Cleanup Logs

Indices and tables

- Index
- Search Page

Previous Next


sd [Interaction] SimplifiedTestExecution [SimplifiedTestExecution]
etr
1: setup
2: run
loop
[for each test]
3:
Test Runner
4: setup
5: test
6: teardow n
7:
8: teardow n

ETR
Test "Module"

```
test $
|-- src
|   |-- mytest.robot
|   |-- deployment.nomad
|   `-- ...
`-- etr.yaml
```

```
# etr.yaml
version: "1.0"
etr:
    plugins:
       - etr.plugins.robot
       - etr.plugins.nomad
robot:
    tests:
         - "src/mytest.robot"
nomad:
    jobs:
        - "src/deployment.nomad"
```

ETR
Output

```
/tmp/etr.git/test/etr  etr
------------------------------------------------------------------------
running test suite "src/service-plugins.robot"
Service-Plugins.test that etr can use resources plugin and report failures ... OK
Service-Plugins.test that etr do not acquire already acquired resources ... OK
Service-Plugins.test that etr can use resources plugin with local resources ... OK
Service-Plugins.test that etr can retry on transient error from resources plugin ... OK
Service-Plugins.test that etr can use nomad for deployment ... OK
running test suite "src/etr.robot"
Etr.fake test to see if etr actually started robot the test runner ... OK
Etr.test that etr can use nose plugin and report failures ... OK
Etr.test sending signal to etr will abort test execution and run teardown to release resources and c
leanup files ... OK
Etr.test that etr can render templates with jinja2 and cleans up after ... OK
Etr.test that etr logs robot output if verbose ... OK
Etr.test that etr can execute a subset of tests ... OK
Etr.test etr randomization ... OK
Etr.test that etr cleans up with the clean command ... OK
------------------------------------------------------------------------
Ran 13 tests in 1m 2s

OK
```

REPORT

Devmgr Log

Generated
20191030 15:31:08 UTC+01:00
9 days 22 hours ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	32	32	0	00:02:06	
All Tests	32	32	0	00:02:06	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
TestAdc	4	4	0	00:00:16	
TestDrot	8	8	0	00:00:49	
TestLamp	5	5	0	00:00:12	
TestMotor	5	5	0	00:00:21	
TestShutter	2	2	0	00:00:02	
TestSM	8	8	0	00:00:26	

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Devmgr	32	32	0	00:02:40	

Test Execution Errors

20191030 15:29:00.910 WARN Multiple test cases with name 'Switch Lamp Off' executed in test suite 'Devmgr'.

Test Execution Log

SUITE **Devmgr** 00:02:40.457

Full Name: Devmgr
Source: /home/jenkins/workspace/ICS/ifw-hl-integration/test/fcf/src/devmgr.robot
Start / End / Elapsed: 20191030 15:28:28.261 / 20191030 15:31:08.718 / 00:02:40.457
Status: 32 critical test, 32 passed, 0 failed
32 test total, 32 passed, 0 failed

SETUP BuiltIn . **Run Keyword** Setup 00:00:04.012

TEARDOWN BuiltIn . **Run Keyword** Teardown 00:00:30.253

TEST **Init Devices** 00:00:23.173

Full Name: Devmgr.Init Devices
Tags: TestSM
Start / End / Elapsed: 20191030 15:28:32.311 / 20191030 15:28:55.484 / 00:00:23.173
Status: PASS (critical)

KEYWORD Process . **Run Process** devmgrClient, ${CMD_TOUT}, localhost, ${CMD_PORT}, modif.ReqEnable, \"\", alias=cli 00:00:00.014
KEYWORD ${result} = Process . **Get Process Result** cli, stdout=yes, stderr=yes 00:00:00.001
KEYWORD ${result} = BuiltIn . **Convert To String** ${result} 00:00:00.000
KEYWORD BuiltIn . **Should Match** ${result}, *ERROR: Request rejected in state NotReady/NotOperational/On/* 00:00:00.001
KEYWORD Process . **Run Process** devmgrClient, ${CMD_TOUT}, localhost, ${CMD_PORT}, modif.ReqDisable, \"\", alias=cli 00:00:00.010
KEYWORD ${result} = Process . **Get Process Result** cli, stdout=yes, stderr=yes 00:00:00.000
KEYWORD ${result} = BuiltIn . **Convert To String** ${result} 00:00:00.000
KEYWORD BuiltIn . **Should Match** ${result}, *ERROR: Request rejected in state NotReady/NotOperational/On/* 00:00:00.001
KEYWORD Process . **Run Process** devmgrClient, ${CMD_TOUT}, localhost, ${CMD_PORT}, modif.ReqRecover, \"\", alias=cli 00:00:00.010
KEYWORD ${result} = Process . **Get Process Result** cli, stdout=yes, stderr=yes 00:00:00.001
KEYWORD ${result} = BuiltIn . **Convert To String** ${result} 00:00:00.000
KEYWORD BuiltIn . **Should Match** ${result}, *ERROR: Request rejected in state NotReady/NotOperational/On/* 00:00:00.001

ETR

Some tips and tricks

- Re-run tests quickly:
  - `etr --step=setup`
  - `etr --step=run`
    - Modify test and repeat
    - Debug
  - `etr --step=teardown`
- Run selected tests:
  - `etr --test=src/foo* --test=src/baz.robot`

- Robustness

```
*** Keywords ***
Setup
        Remove Environment Variable    DISPLAY
        ${port} =  Get Random Port
        Set Global Variable  ${flask_port}  ${port}
        Start Process  python  -m  flask  run  -p  ${port} alias=flask
        Sleep   2s      Give time for flask to start the web server.
```

Robustness

```
*** Keywords ***
Start Nomad
        Start Process    nomad  agent  -dev  alias=nomad
        Run Keyword    Wait for Nomad

Wait for Nomad
        :FOR  ${index}  IN RANGE  10
        \       Sleep  1s
        \       ${r} =  Run Process    nomad  status
        \       Log Many    ${r.rc}  ${r.stdout}  ${r.stderr}
        \       Run Keyword If    '${r.rc}' == '0'          Exit For Loop
        Run Keyword If    '${r.rc}' != '0'    Fail  Nomad failed to
start, aborting test.
```

Software Testing
Best Practices

- Do
  - Make etr test suites independent/self contained
  - Make tests robust
  - Log output
  - Run in CI or other reference system
- Don’t
  - Hard code resources (e.g. service ports, addresses)
  - Assume a specific timing (`sleep(N)`)
    - E.g. poll for a service to start up
  - Assume there's a display

Questions?