Introduction to Unit Testing

León Jaramillo

softserve

Test Automation

- **Automating testing** is using software tools to control and manage:
  - Tests execution.
  - Comparing results obtained vs. those expected.
  - Defining preconditions.
  - Reporting results.

- Test automation have some **advantages**, such as:
  - Reducing QA costs.
  - Reducing human error-proneness.
  - Reducing the difference of test quality between different individuals.
  - Reducing regression testing costs.

- It involves automating **test cases' execution**.

Test Cases

A **test case** is an artifact which typically comprises:

- Test case values
- Results expected
- Prefix values
- Postfix values
  - Verification values
  - Exit values

Test Automation Frameworks

- A **test automation framework** is a set of concepts and tools that support test automation.

- Most of them support:
  - **Assertions**, to assess actual results vs. those expected.
  - To enable **sharing common data** between different tests.
  - **Test cases** to organize and execute tests easily.
  - Manage the execution of tests either using a **CLI** or a **UI**.

- A **test driver** is just the wrapper/mechanism that organizes the tests, runs them, and handles their output.

softserve

Test Automation Frameworks

xUnit.net

Cucumber

MOCHA chai

PHPUnit

JUnit 5

nunit

cypress

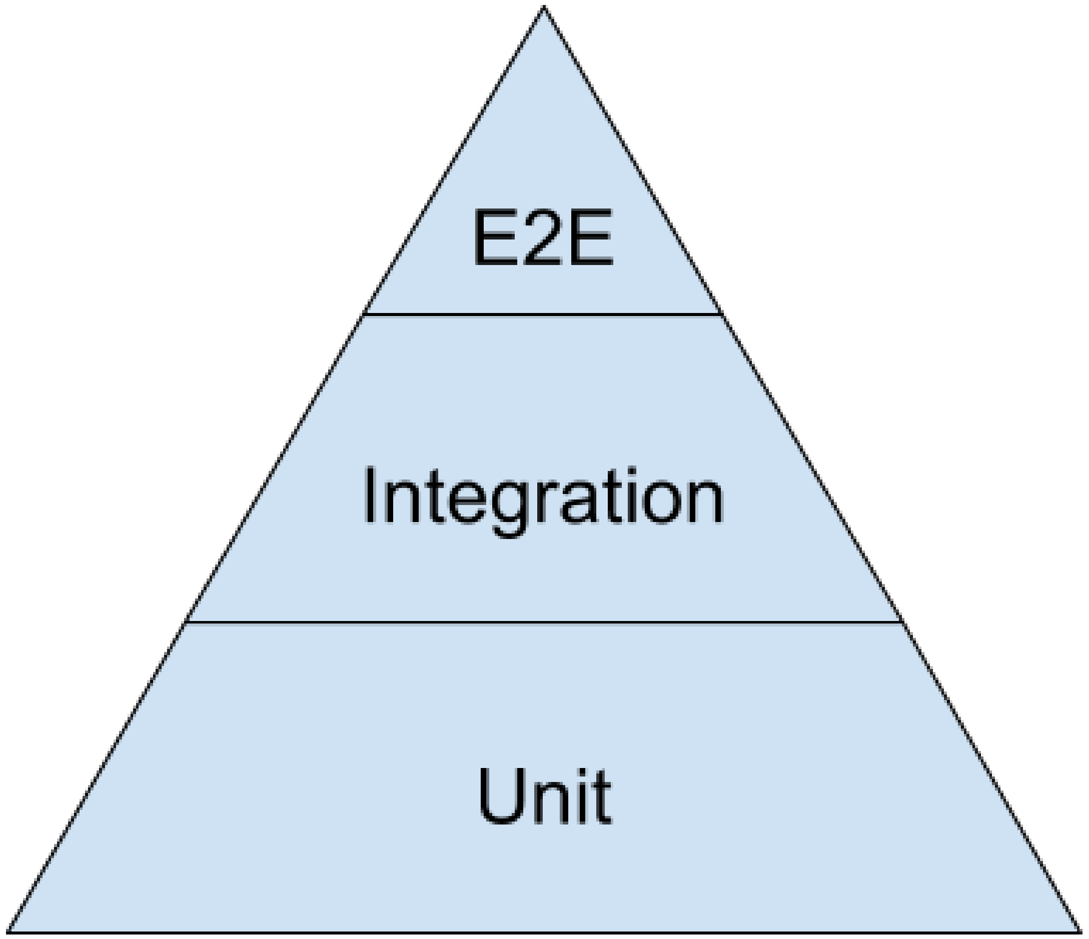
Selenium

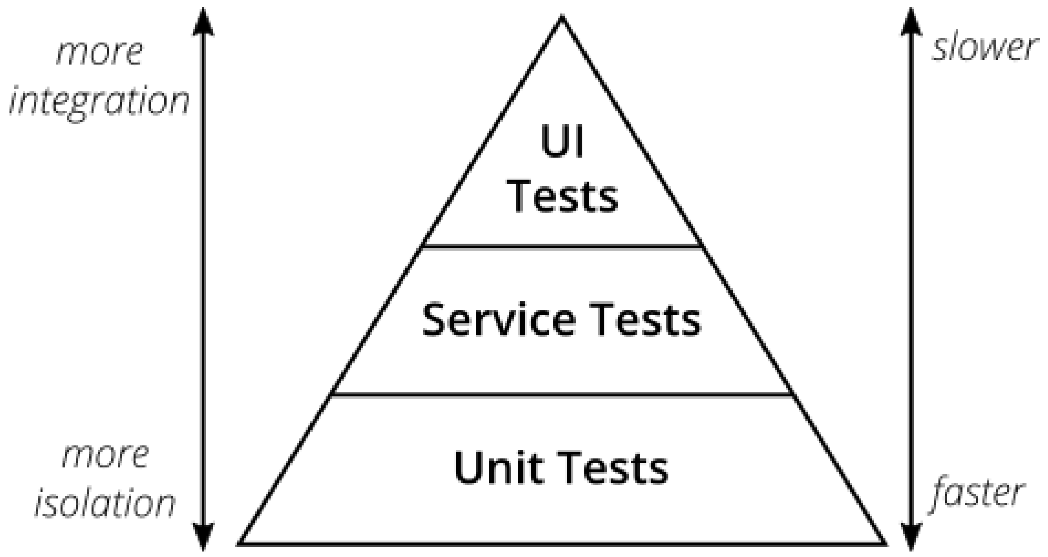
Jest

Test::Unit

softserve

How many tests should we do?

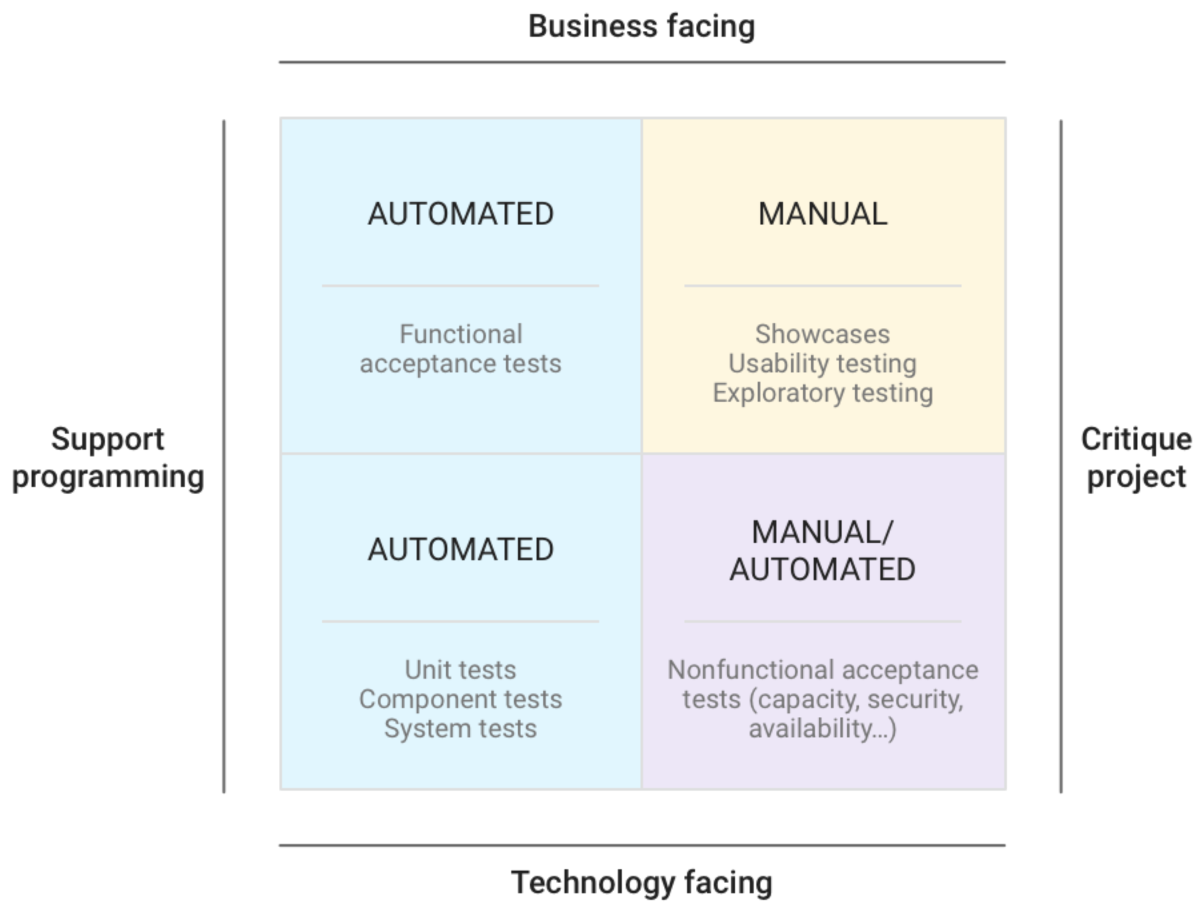




From: Google Testing Blog

softserve

What kinds of tests should be automated?



From: Google Testing Blog

softserve

Test::Unit

- **Test::Unit** is a unit test automation framework for Ruby.
- You can **set-up**, **organize**, and **run** tests using this tool.
- Test::Unit can go along with **other testing frameworks**, such as Ruby on Rails tests, Selenium, or Cucumber.
- Test::Unit uses **assertions** to assess whatever we need to test.
- There are different ways to determine what is a “unit” in these unit tests.

Example

```
class Calculator
  def add(a, b)
    return a+b
  end

  def multiply(a, b)
    return a*b
  end
end
```

```
class TestCalculator < Test::Unit::TestCase
  def setup
    @calc = Calculator.new
  end

  def test_add
    assert_equal(1, @calc.add(1, 0))
    assert_equal(0, @calc.add(0, 0))
    assert_equal(5, @calc.add(2, 3))
  end

  def test_nan

assert_raise(TypeError){@calc.multiply("Hola",
"mundo")}
  end

  def test_multiply
    assert_equal(0, @calc.multiply(1, 0))
    assert_equal(0, @calc.multiply(0, 0))
    assert_equal(6, @calc.multiply(2, 3))
    assert_equal(5, @calc.multiply(5, 1))
  end
end
```

softserve

Some Assertions to Consider

`assert( boolean, [message] )`	True if boolean
`assert_equal( expected, actual, [message] )`	True if expected == actual
`assert_not_equal( expected, actual, [message] )`	
`assert_match( pattern, string, [message] )`	True if string =~ pattern
`assert_no_match( pattern, string, [message] )`	
`assert_nil( object, [message] )`	True if object == nil
`assert_not_nil( object, [message] )`	
`assert_in_delta( expected_float, actual_float, delta, [message] )`	True if (actual_float - expected_float).abs <= delta
`assert_instance_of( class, object, [message] )`	True if object.class == class
`assert_kind_of( class, object, [message] )`	True if object.kind_of?(class)
`assert_same( expected, actual, [message])`	True if actual.equal?(expected).
`assert_not_same( expected, actual, [message] )`	
`assert_raise( Exception,... ) {block}`	True if the block raises (or doesn't) one of the listed exceptions.
`assert_nothing_raised( Exception,...) {block}`	
`assert_throws( expected_symbol, [message] ) {block}`	True if the block throws (or doesn't) the expected_symbol.
`assert_nothing_thrown( [message] ) {block}`	
`assert_respond_to( object, method, [message] )`	True if the object can respond to the given method.
`assert_send( send_array, [message] )`	True if the method sent to the object with the given arguments return true.
`assert_operator( object1, operator, object2, [message] )`	Compares the two objects with the given operator, passes if true

softserve

Useful Resources

- Ruby Programming / Unit Testing: https://en.wikibooks.org/wiki/Ruby_Programming/Unit_testing

- Module Test::Unit Documentation: https://ruby-doc.org/stdlib-3.1.0/libdoc/test-unit/rdoc/Test/Unit.html

softserve

COLOMBIA DC

Thanks! Any question?

softserve