Integration testing framework for YaST modules

Rodion Iafarov
riafarov@suse.com

Agenda

- Testing on different development phases
  - Levels of software testing
  - Software lifecycle vs software testing
- YaST modules and libyui
  - Current approach and challenges
  - RSpec
- YUI REST API
  - Features
  - Implementation
  - Further steps

Levels of software testing

- Unit testing
- Integration testing
- System testing
- Acceptance testing

Software lifecycle in V-Model

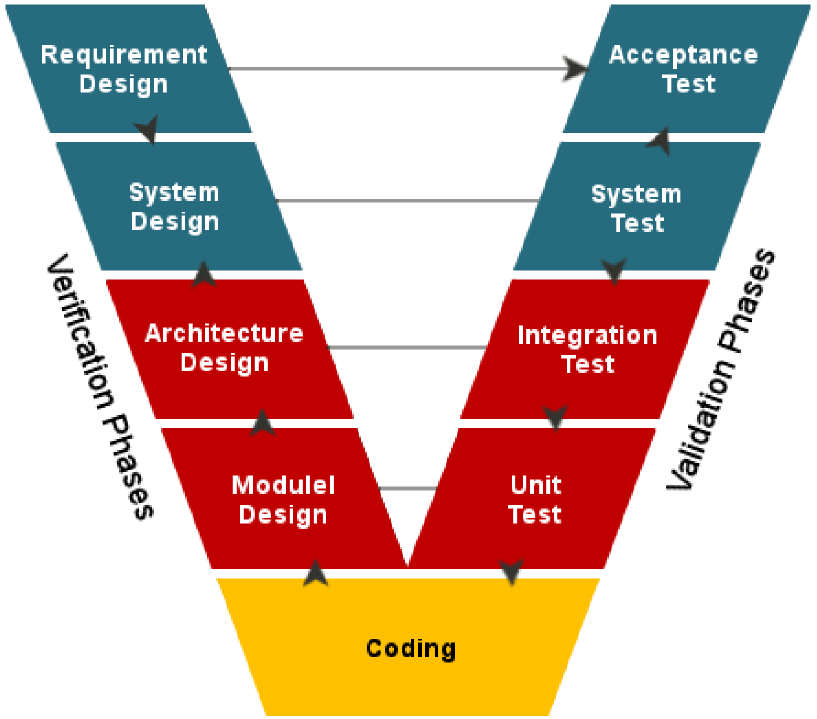


Figure 1. “What is V Model in Software Testing?”, received from https://www.testbytes.net/blog/what-is-v-model-in-software-testing/

Motivation

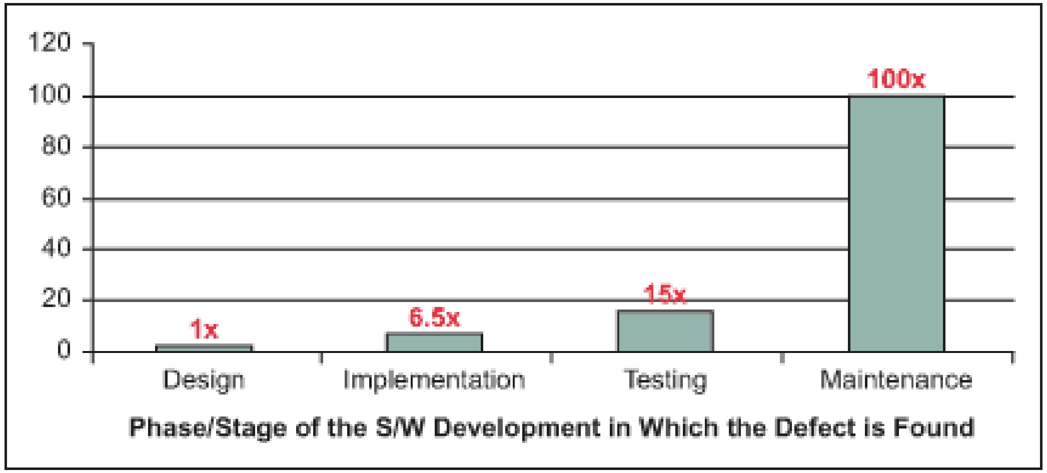


Figure 2: Relative Costs to Fix Software Defects, source: IBM Systems Sciences Institute

YaST and libyui

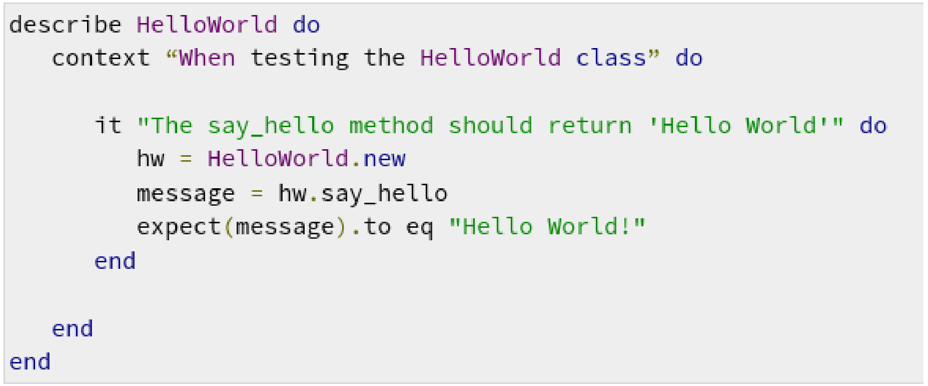
- YaST
  - Set of installation and configuration tools
  - YaST modules use libyui
- libyui is a user interface engine
  - Qt
  - Gtk
  - text based user interfaces (ncurses)

How YaST components are tested

- Unit tests using RSpec
- Integration and system testing in openQA
- Problems:
  - No automated integration tests on pull requests
  - No integration tests to test changes for regressions quickly
  - High maintenance costs of the screen based tests

RSpec

- Behavior Driven Development
- Can be used for unit and integration testing
- Built-in reporting capabilities



```
describe HelloWorld do
   context “When testing the HelloWorld class” do

      it "The say_hello method should return 'Hello World'" do
         hw = HelloWorld.new
         message = hw.say_hello
         expect(message).to eq "Hello World!"
      end

   end
end
```

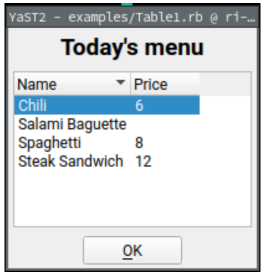
Figure 3. RSpec hello world example, received from https://www.tutorialspoint.com/rspec/rspec_basic_syntax.htm

libyui REST API Server side

- Developed in C++
- Dynamically loaded plugins
- Separate implementations for qt and ncurses
- Generates events to simulates users input
- Provides capabilities for reading widgets properties

UI Properties example

```
...
  {
    "class" : "YTable",
    "columns" : 2,
    "hasMultiSelection" : false,
    "header" :
    [
      "Name",
      "Price"
    ],
    "hstretch" : true,
    "icon_base_path" : "",
    "immediate_mode" : false,
    "items" :
    [
      {
        "labels" :
        [
          "Chili",
          "6"
        ],
        "selected" : true
      },
      {
        "labels" :
        [
          "Salami Baguette",
          ""
        ]
      },
...
```

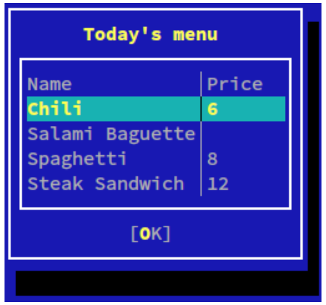


YaST2 - examples/Table1.rb @ ri-…

Today's menu

Name	Price
Chili	6
Salami Baguette	
Spaghetti	8
Steak Sandwich	12

Figure 4. Sample YaST application in qt



Today's menu

Name	Price
Chili	6
Salami Baguette	
Spaghetti	8
Steak Sandwich	12

Figure 5. Sample YaST application in ncurses

libyui REST API Ruby Client

- Provides interfaces to operate UI applications
- Easily integrates with rspec
- gem is published to RubyGems
- Single implementation for qt and ncurses applications
- Widgets support is in sync with server side

Further steps

- Further improvements to server side:
  - https support
  - support for more widgets
  - Use body in POST requests instead of URL query parameters
- Extending testing coverage
- Executing integration tests on early stages of the development
- Perl client side framework

References

- libyui: https://github.com/libyui/
- YaST: https://github.com/yast/
- libyui-rest-api: https://github.com/libyui/libyui-rest-api
- ruby yui rest client: https://github.com/qe-yast/ruby-yui-rest-client
- RSpec tutorials: https://www.tutorialspoint.com/rspec/rspec_basic_syntax.htm

Finish

Thank You