



Rutema

one test tool to rule them all

Vassilis Rizopoulos - damphyr

www.braveworld.net/riva

Euruko 2008, Prague
rutema: one test tool to rule them all



The Problem

Lots of test tools

Euruko 2008, Prague
rutema: one test tool to rule them all

1/13

We use a lot of tools for testing, to do unit tests, GUI tests, web tests, load tests, system tests. We also write our own custom tools/scripts when the need arises. So there is a proliferation of test tools and consequently a proliferation of ways to document, execute and manage all these tests. Quite often all these different ways happen within the same project.



What Is Rutema?

A set of tools and practices to manage

- specification
- execution
- results

of tests.

It drives test tools and does the dirty work.

All in Ruby

It's A solution to the stated problem.

Rutema does not test. It helps you organise all the different tools in a simple and hopefully sane way.

The dirty work would be to take care of gathering output from all the test steps, determine success and archive all the results.



How To Go About Organizing Everything?

Specifications, Runs, Results

Some definitions:

A run is a single execution of one or more tests.

As for the results? Ideally they will be gathered, archived, reported on and all with the minimum of hassle or repetitive work.

Oh, and they should be available the next time the PM asks for them.

Test Specifications

Test specifications should be

- Unambiguous
- Human and machine readable
- Domain specific

Unambiguous so that consecutive runs on the same codebase yield identical results.

If you can't trust your tests better not have any.

Human readable to make the creation/maintenance easy and machine readable to make automation easy.

Domain specificity is a human issue: it eases the learning curve and makes adoption easier.

Rutema Specifications

rutema uses XML for test specifications.

```
<specification name="T001">
  <title>Basic parser elements</title>
  <description>This shows off the basic parser elements</description>
  <scenario>
    <echo>Hello Testing World</echo>
    <command cmd="ls"/>
    <prompt>Are you there?</prompt>
  </scenario>
</specification>
```

More examples: [sm301.spec](#), [ms.spec](#)

We have the human part of the test specification on top - to use in reports and fancy PDFs , some tracing information - always useful to know what you have covered and keep your PM happy
The juice is in the scenario where the sequence of steps that comprise the test is defined.

All steps must complete successfully for the test to be successful.

It's a mini XML DSL. Since we are talking simple sequences here XML actually helps as it is familiar and in this setting extremely simple.

Which test will run, which parser to use, setup and teardown specification, location of executables

to use as well as context information is defined in the configuration (which is valid ruby code). rutemax captures and stores all output (stdout and stderr), run/test/step durations along with context information defined in the configuration.

Domain Specific Specs

```
<scenario>
  <mstest assembly="../../Builds/Tests/SomeTests.dll"/>
</scenario>
```

Extending The Parser

Build your own parser to make specs domain specific. [It's easy!](#)

```
class DemoParser < Rutema::MinimalXMLParser
  def element_mstest step
    unless step.has_assembly?
      raise Rutema::ParserError, "Missing required 'attribute' assembly"
    end
    if File.exists?(step.assembly)
      cmdline="#{@configuration.tools.mstest} /testcontainer:#{step.assembly}"
      step.cmd=Patir::ShellCommand.new(:cmd=>cmdline,:working_directory=>Dir.pwd)
    else
      raise Rutema::ParserError, "Cannot find assembly file '#{step.assembly}' in mstest step"
    end
  end
end
```

This is the complete code with error handling and everything Patir is the base library. Part of it is a Command abstraction that we use for the executable part of a test step.

Demo:Spec

```
<specification name="T001">
  <title>T001</title>
  <description>An echo test</description>
  <scenario>
    <echo>Hello Testing World</echo>
  </scenario>
</specification>
```

Demo: Run

rutemax -c database.rutema specs/T001.spec

```
[20080326 21:32:33] INFO: Configuration loaded from test/distro_test/config/database.rutema
[20080326 21:32:33] INFO: Connecting with database '/Users/riva/Projects/mine/rubyforge/patir/rutema/trunk/test/distro_test/config/sample.db'
[20080326 21:32:33] INFO: Run started in mode '/Users/riva/Projects/mine/rubyforge/patir/rutema/trunk/test/distro_test/specs/T001.spec'
[20080326 21:32:33] INFO: Running T001 - T001
[20080326 21:32:33] INFO: Scenario for T001
Hello Testing World
[20080326 21:32:33] INFO: 1 - echo - success
[20080326 21:32:33] INFO: Run completed in 0.006709s
[20080326 21:32:33] INFO: Report:
Parsed 1 files
Parse errors: 0
Current run:
Scenarios: 1
[20080326 21:32:33] INFO: All tests successful
```

Demo: Text Report

rutemah -c database.rutemah T001


```
[20080326 21:33:29] INFO: Configuration loaded from database.rutemah
[20080326 21:33:29] INFO: Configuration loaded from test/distro_test/config/database.rutemah
[20080326 21:33:29] INFO: Connecting with database '/Users/riva/Projects/mine/rubyforge/patir/rutema/trunk/test/distro_test/config/sample.db'
```

| run_id | name | status | attended | version | start_time | stop_time |
|--------|------|---------|----------|---------|--------------------------------|--------------------------------|
| 1 | T001 | success | no | N/A | Wed Mar 26 21:26:13 +0100 2008 | Wed Mar 26 21:26:13 +0100 2008 |
| 4 | T001 | success | no | N/A | Wed Mar 26 21:32:33 +0100 2008 | Wed Mar 26 21:32:33 +0100 2008 |

Demo: Web Interface

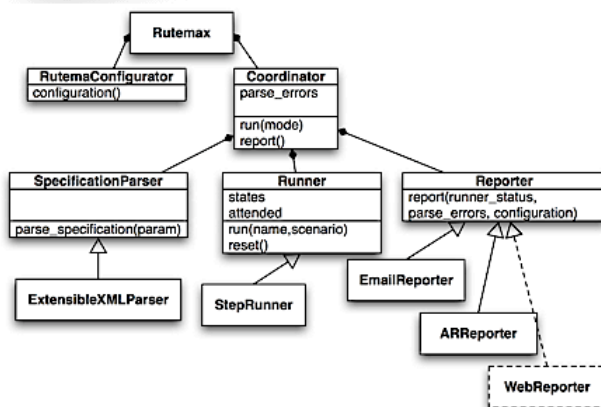
Summary

[Run 4](#)
[Run 3](#)
[Run 2](#)
[Run 1](#)

| status | name | version | start_time | stop_time |
|---|------|---------|-------------------|-------------------|
|  | T001 | N/A | 080326 - 21:32:33 | 080326 - 21:32:33 |

I Don't Like It!

Change it! Built your own parser, add your own reporters.



Built your own parser, add your own reporters.

Strong Points

- Regression tests
- Smoke tests
- Tests for regulated development
- Acceptance tests

Do your exploratory testing first. Keep the good ones. Leave the rinse&repeat to rutema.

It's not the be all and end all of testing. Not everything needs to be specified like this

What's Coming?

- Centralized result repository
- More work on the web interface
- Documentation on advanced configuration topics (context and parameter passing for parsers and reporters)
- TFS integration (creation of work items) - Windows only at first
- Rubot

rutemaweb will gain a REST entry point where a Reporter implementation can post test results. We need to work on the web interface to the DB.

rubot is rutema's cousin project, a ruby distributed build management system – still vaporware, code is available at the rubyforge repository in its early alpha stages.



Resources And Credits

- patir.rubyforge.org/rutema
- Web Design: [RedTie](#), from www.oswd.org
- Icons: [Fast Icon](#)