Standalone applications testing and automation

Desktop QA team
Red Hat, Inc.

2014-02-02
Who we are

Red Hat's Desktop QE team

Quality assurance for:
  Desktop hardware
    Wireless, graphics, audio...
  Desktop application stack
    Desktop environments
    GUI management tools
    Office applications
Distributions under test

RHEL 7.0 Desktop

RHEL 5-6 updates

Fedora Rawhide/RHEL next

Latest versions testing:

NetworkManager

Evolution
Distributions under test

RHEL 7.0 Desktop

RHEL 5-6 updates

Fedora Rawhide/RHEL next

Latest versions testing:

NetworkManager

Evolution
Distributions under test

RHEL 7.0 Desktop
RHEL 5-6 updates
Fedora Rawhide/RHEL next

Latest versions testing:
NetworkManager
Evolution
Distributions under test

- RHEL7.0 Desktop
- RHEL 5-6 updates
- Fedora Rawhide/RHEL next

Latest versions testing:
- NetworkManager
- Evolution
Infrastructure bits

**Beaker** manages available machines and distributions

Simple Test Harness task fetches test automation code and runs required tests

**Behave** controls automation execution

**Dogtail/pexpect** executes actions on UI/CLI

Code review in **Gerrit**

Test case management - **Nitrate**
Infrastructure bits

Beaker manages available machines from the pool and distributions

**Simple Test Harness** task fetches test automation code and runs required tests

Behave controls automation execution

Dogtail/pexpect executes actions on UI/CLI

Code review in Gerrit

Test case management - Nitrate
**Infrastructure bits**

**Beaker** manages available machines from the pool and distributions.

**Simple Test Harness** task fetches test automation code and runs required tests.

**Behave** controls automation execution.

**Dogtail/pexpect** executes actions on UI/CLI.

Code review in **Gerrit**.

Test case management - **Nitrate**.
Infrastructure bits

Beaker manages available machines from the pool and distributions

Simple Test Harness task fetches test automation code and runs required tests

Behave controls automation execution

Dogtail/pexpect executes actions on UI/CLI

Code review in Gerrit

Test case management - Nitrate
Infrastructure bits

Beaker manages available machines from the pool and distributions

Simple Test Harness task fetches test automation code and runs required tests

Behave controls automation execution

Dogtail/pexpect executes actions on UI/CLI

Code review in Gerrit

Test case management - Nitrate
Beaker

http://beaker-project.org

Automation and task execution system for labs of test computers

VM / bare-metal machines support

Multiarchitecture support

Flexible permissions system
Beaker

Most notable tasks used:

/desktop/rhel7/install

Installs the desktop components (GNOME/KDE)

/desktop/simpletestharness

fetches automation code from git/gzip sources
starts specified tests to be executed
stores test output, reports and artifacts
Beaker

Most notable tasks used:

`/desktop/rhel7/install`
Installs the desktop components (GNOME/KDE)

`/desktop/simulestestharness`
fetches automation code from git/gzip sources
starts specified tests to be executed
stores test output, reports and artifacts
Dogtail

Python GUI automation framework taking advantage of Accessibility technologies

Based on AT-SPI – toolkit-neutral technology, used by GTK, Qt, Mozilla, LibreOffice
Dogtail

Dogtail-based upstream test suites

PiTiVi

GNOME Software

Evince
Sniff is a GUI for AT-SPI structure
Scripts and unit tests

Scripts → Unittests → BDD

BDD – Behaviour Driven Development
Gherkin – Business Readable
Domain Specific Language

Test scenarios are human-readable list of steps to be performed

Steps are matched to python procedures, called step definitions
Gherkin

Core keywords: Feature, Background, Scenario

Scenarios can be grouped by feature or using tags
Behave

https://pypi.python.org/pypi/behave

Python implementation of BDD approach

Flexible output formatters (HTML, plain etc.)

Setup / breakdown is implemented via before_* / after_* procedures
**Behave**

**Scenario**: Add contact in new addressbook
- **Create a new contact**
- **Set "Full Name..." in contact editor to "Adam Doe"**
- **Set "Where:" in contact editor to "...Local Contacts"**
- **Save the contact**
- **Refresh addressbook**
- **Select "Doe, Adam" contact**
- **Open contact editor for selected contact**
- **Then "Full Name..." property is set to "Adam Doe"**

```python
@step(u'Create a new contact list')
def create_new_contact_list(context):
    context.app.instance.menu('File').click()
    context.app.instance.menu('File').menu('New').point()
    context.app.instance.menu('File').menu('New').menuItem("Contact List").click()
    context.execute_steps(u"Then Contact List editor window is opened")

@then(u'Contact editor window is opened')
@then(u'Contact editor window with title "{title}" is opened')
def contact_editor_with_title_is_opened(context, title="Contact Editor"):
    context.app.contact_editor == context.app.instance.dialog(title)
    context.app.contact_editor != context.app.instance.dialog("Contact Editor wasn't found")
    context.app.contact_editor.showing == context.app.instance.dialog("Contact Editor didn't appear")
```
Tips and Tricks

HTML report with screenshots after each step and logs from journalctl

Screencast recording

Detect app crashes via abrt
Benefits of BDD approach

Automated test scenarios are human-readable

Can be used as instructions for manual tests

Easy to update / enhance

Scenarios can be written by designers

draft new features

document app behaviour

Steps can be reused across several projects:

Gnome Online Accounts handling

Working with GNOME open / save file dialogs
Benefits of BDD approach

Automated test scenarios are human-readable
Can be used as instructions for manual tests
Easy to update / enhance
Scenarios can be written by designers
draft new features
document app behaviour
Steps can be reused across several projects:
Gnome Online Accounts handling
Working with GNOME open / save file dialogs
Benefits of BDD approach

Automated test scenarios are human-readable

Can be used as a instructions for manual tests

Easy to update / enhance

Scenarios can be written by designers

draft new features

document app behaviour

Steps can be reused across several projects:

Gnome Online Accounts handling

Working with GNOME open / save file dialogs
Benefits of BDD approach

UI abstraction in scenarios:

Same scenarios can be used to test various frontends: e.g. NetworkManager's nmcli / nmtui

Evolution tests with minimal adjustments can be used for RHEL 6, RHEL 7, Fedora 20 and Fedora Rawhide

Due to grouping scenarios by feature we can easily run regression check for affected feature
Benefits of BDD approach

UI abstraction in scenarios:

Same scenarios can be used to test various frontends: e.g. NetworkManager's nmcli / nmtui

Evolution tests with minimal adjustments can be used for RHEL 6, RHEL 7, Fedora 20 and Fedora Rawhide

Due to grouping scenarios by feature we can easily run regression check for affected feature
Success stories: NM, Evolution

Evolution testing across available distributions

Proposed patch testing

Network Manager

Uses pexpect instead of dogtail

Scenarios can be re-used to test various NM frontends: nmcli, nmtui (ncurses-based)

Sharing steps with GNOME Control Center
Success stories: NM, Evolution

Evolution testing across available distributions

Proposed patch testing

Network Manager

Uses pexpect instead of dogtail

Scenarios can be re-used to test various NM frontends: nmcli, nmtui (ncurses-based)

Sharing steps with GNOME Control Center
Future plans

Working with Fedora QA to have a similar process for Fedora

Execute tests directly in upstream CI:

- GNOME-continuous
- Build.kde.org
Future plans

Working with Fedora QA to have a similar process for Fedora

Execute tests directly in upstream CI:

- GNOME-continuous
- Build.kde.org
Questions