COLLABORA

Plan your testing

Ekaterina Gerasimova
kat@collabora.co.uk
kat@gnome.org
2018-08-11

What is testing?

- Verification: are we building the product right?
  - Adherence to standards
  - Does the job
  - Is complete
- Validation: are we building the right product?
  - Meets requirements
  - Works how we want

Why do you test?

Photo by Emily Morter on Unsplash

Why do you test?

- Consistency and reproducibility?
- Ready for release?
- Quality?
- Confidence in your product?

What type of testing do you do on your project?

Photo by David Travis on Unsplash

What type of testing do we do in GNOME?

- Automated

What type of testing do we do in GNOME?

- ~~Automated~~ (implementation, not type)

What type of testing do we do in GNOME?

- ~~Automated~~ (implementation, not type)
- Usability? (Jim Hall & design team)

What type of testing do we do in GNOME?

- ~~Automated~~ (implementation, not type)
- Usability? (Jim Hall & design team)
- Performance?
- Regression?
- Accessibility?

Testing level: component testing

- Also known as: unit testing
- Tests code within each module works how it should (example, test parts of code in a library or application)
- Possible to track coverage (how much of the library or application is tested)

Testing level: integration testing

- Tests how two or more modules work together (example, application with library)
- Tests interfaces and interactions
- May include performance testing

Testing level: system testing

- Tests the whole system together (example, test GNOME with all dependencies and core applications)
- Tests behaviour & configuration of a whole system
- What the user is going to “see”
- What the system does, performance, usability and more!

Other common levels of testing

- **System integration**: how well system works with other systems (e.g. application with server)
- **Acceptance**: check if the system is ready
- **Regression**: add tests after something breaks to make sure it does not break again
- ...and more

Is your testing consistent?

Photo by rawpixel on Unsplash

How can you achieve consistent testing?

- Add reproducibility
  - Plan what you will test
  - Automate where practical
- Add structure
  - Define what you test in test cases

How can you achieve consistent testing?

- Add reproducibility
  - **Plan what you will test**
  - Automate where practical
- Add structure
  - Define what you test in test cases

Making decisions

- What?
- Why?
- How?
- When?
- Who?

When to test?

- Waterfall model:
  design, build, test, repeat!

- V-model:
  different testing for each development stage

How can you achieve consistent testing?

- Add reproducibility
  - Plan what you will test
  - **Automate where practical**
- Add structure
  - Define what you test in test cases

Effort in testing

- Automated
  - Can be high setup cost
  - Low human running cost
  - High reproducibility
- Manual *
  - Usually low setup cost
  - High running cost
  - Human error?

Effort of testing

* Most testing can be automated with enough effort

Effort of testing

* Most testing can be automated with enough effort

... but is the effort worth it?

Some tools to help you...

- GitLab – CI support, can run automated tests with every commit (high processing cost)
- LAVA – deploys and tests system on physical and virtual hardware
- OpenQA – GUI testing (compares UIs)
- ...

How do you organise your testing?

How do you organise your testing?

- Randomly?
- From experience?
- On the back of an envelope?
- Do you have a bus factor of one?

Test cases

- Uniquely identify your test
- Describe what you are doing and why
- Give input parameters
- Give precise instructions
- Tell you what result to expect

Benefits from test cases

- You are always testing the same thing
- Anyone can run the tests
- Easy to tell if something stops working

Apertis

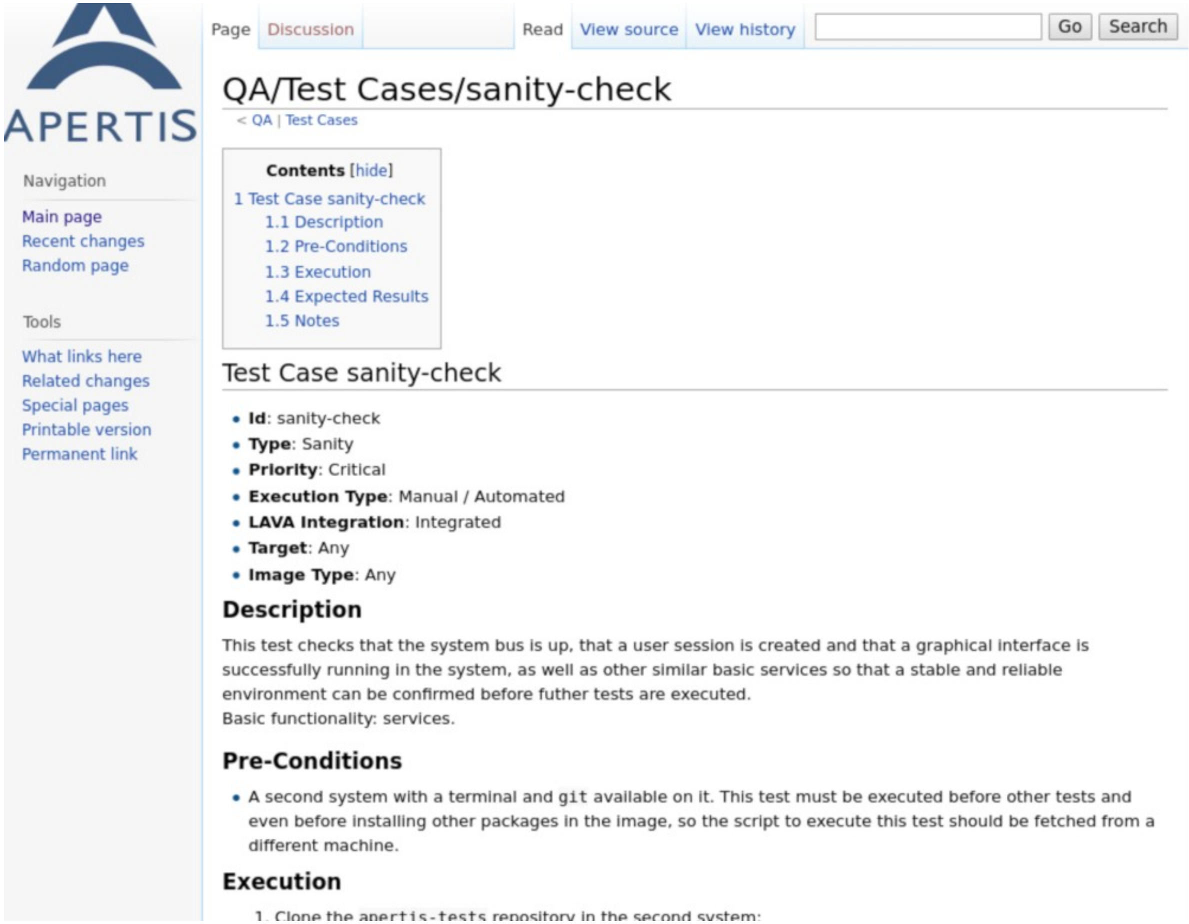
- GNOME-derived distribution
- Made for cars, but...
- Can be used in many areas

- Website: https://apertis.org

Managing test cases and results

- Lots of proprietary tools
- F/LOSS tools?
  - Nitrate
  - SQUAD (for LAVA)
- Manually?

Manual test case management is… unpleasant



QA/Test Cases/sanity-check

< QA | Test Cases

Contents [hide]

Test Case sanity-check

- **Id**: sanity-check
- **Type**: Sanity
- **Priority**: Critical
- **Execution Type**: Manual / Automated
- **LAVA Integration**: Integrated
- **Target**: Any
- **Image Type**: Any

Description

This test checks that the system bus is up, that a user session is created and that a graphical interface is successfully running in the system, as well as other similar basic services so that a stable and reliable environment can be confirmed before futher tests are executed.
Basic functionality: services.

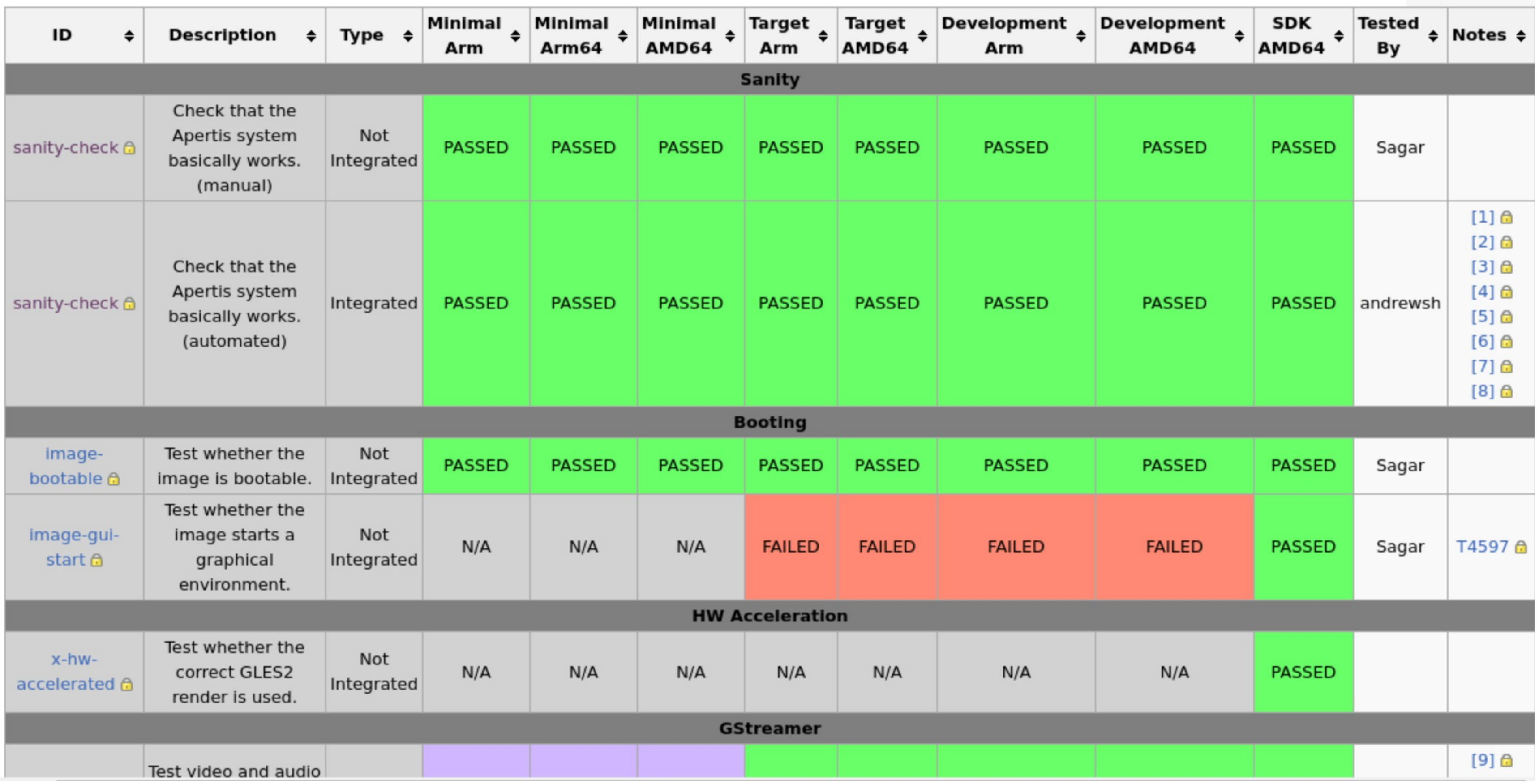
Pre-Conditions

- A second system with a terminal and `git` available on it. This test must be executed before other tests and even before installing other packages in the image, so the script to execute this test should be fetched from a different machine.

Execution

1. Clone the `apertis-tests` repository in the second system:

Manual test result management is… even worse



ID	Description	Type	Minimal Arm	Minimal Arm64	Minimal AMD64	Target Arm	Target AMD64	Development Arm	Development AMD64	SDK AMD64	Tested By	Notes
Sanity												
sanity-check	Check that the Apertis system basically works. (manual)	Not Integrated	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	Sagar	
sanity-check	Check that the Apertis system basically works. (automated)	Integrated	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	andrewsh	[1] [2] [3] [4] [5] [6] [7] [8]
Booting												
image-bootable	Test whether the image is bootable.	Not Integrated	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	Sagar	
image-gui-start	Test whether the image starts a graphical environment.	Not Integrated	N/A	N/A	N/A	FAILED	FAILED	FAILED	FAILED	PASSED	Sagar	T4597
HW Acceleration												
x-hw-accelerated	Test whether the correct GLES2 render is used.	Not Integrated	N/A	N/A	N/A	N/A	N/A	N/A	N/A	PASSED		
GStreamer												
	Test video and audio											[9]

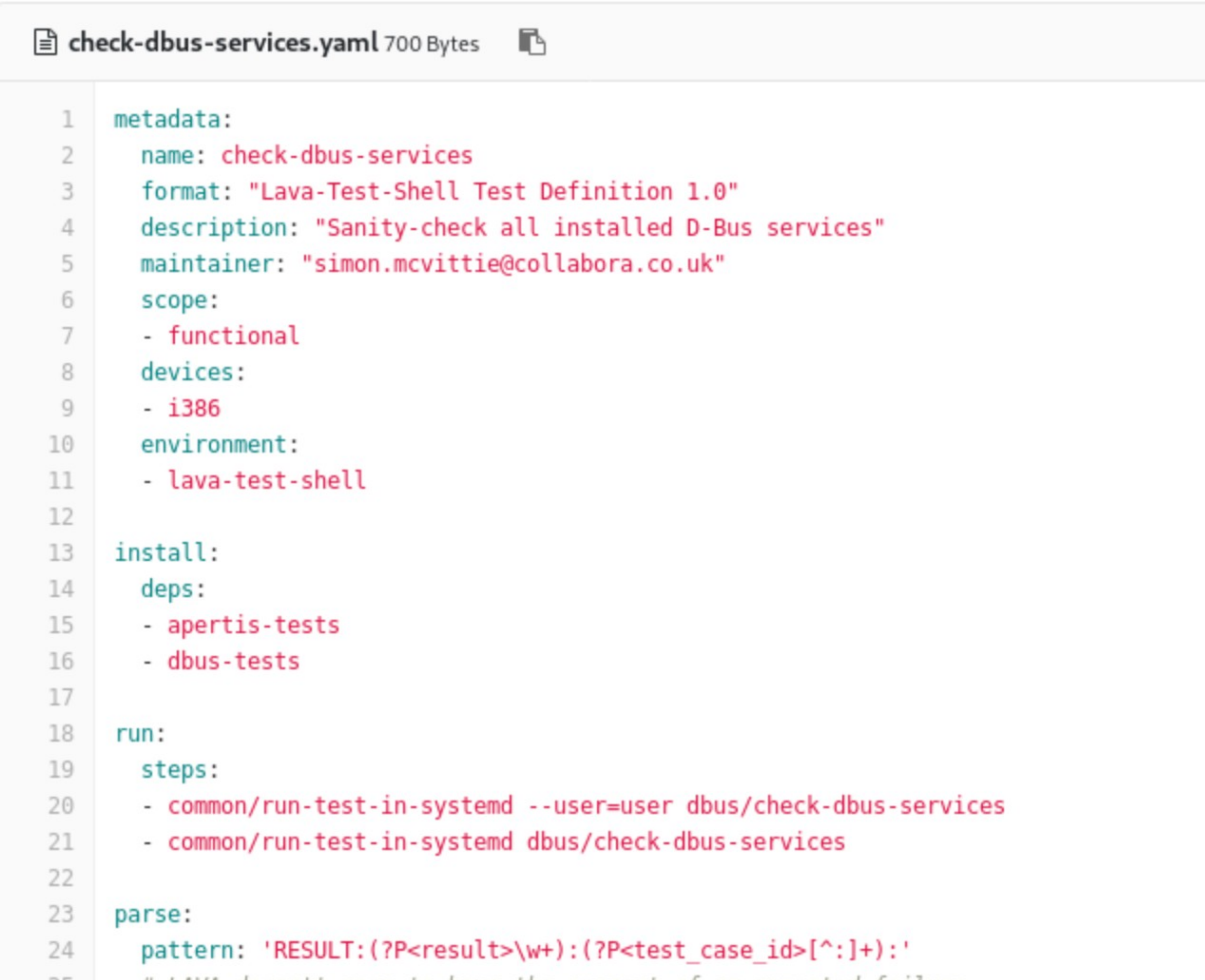
What are the problems?

- Keeping everything up to date
- Tracking failures between test runs
- Analysing failure trends
- Tracking regressions

What is our solution?

- Store all test cases in git, in the same format as the automated test recipes
- Web UI to integrate with automated test system to extract results and to allow manual entry of manual results

What is our solution?



check-dbus-services.yaml 700 Bytes

```yaml
metadata:
  name: check-dbus-services
  format: "Lava-Test-Shell Test Definition 1.0"
  description: "Sanity-check all installed D-Bus services"
  maintainer: "simon.mcvittie@collabora.co.uk"
  scope:
  - functional
  devices:
  - i386
  environment:
  - lava-test-shell

install:
  deps:
  - apertis-tests
  - dbus-tests

run:
  steps:
  - common/run-test-in-systemd --user=user dbus/check-dbus-services
  - common/run-test-in-systemd dbus/check-dbus-services

parse:
  pattern: 'RESULT:(?P<result>\w+):(?P<test_case_id>[^:]+):'
```

Conclusion: our aims

- Make sure that our software does what it’s supposed to do
- Find out if something breaks before release
- Make sure the same problems don’t happen again
- Develop faster
- Let you reproduce our testing results

Thank you!

Collabora is hiring!

Contact me at kat@gnome.org or kat@collabora.co.uk