Open Source QA -
What will it take
to get to the next level?

Tim Bird
Fuego Test System Maintainer
Sr. Staff Software Engineer, Sony Electronics
Problem

- We can do better with Linux QA, as an industry and community
Problem  Opportunity

- We can do better with Linux QA, as an industry and community
- Not saying we’re doing something wrong
  - Lots of great projects and efforts
- I think that we’re missing opportunities to do better
Outline

Attributes of Open Source
Status of Open Source QA
Obstacles to Sharing
Solutions and Next Steps
Outline

Attributes of Open Source
Status of Open Source QA
Obstacles to Sharing
Solutions and Next Steps
Attributes of Open Source

- Openly available
- Easy to contribute to
- Generalized
  - Applies to a broad range of uses
- Has a development community
- Community effect
  - Build on the work of others
  - As contributor pool increases, better ideas are found
  - Feedback loops
- The essence of Open Source is “exchange” or sharing
Outline

Attributes of Open Source
Status of Open Source QA
Obstacles to Sharing
Solutions and Next Steps
Status of Open Source QA

- Software Landscape
  - Tools
  - Tests
- Hardware
- Industry QA efforts
  - In-house
  - Ad-hoc solutions
- Result = lots of unshared stuff
What tools in QA are Open Source

- Systems:
  - Buildbot, Jenkins, LAVA, LKFT, Fuego, KernelCI, CKI
- Lab/board management: Labgrid, SLAV, libvirt, r4d
- Harnesses: pytest, ptest (Yocto Project), ktest
- Services: 0-day, Phoronix, CKI?
- Tests suites: LTP, kselftest
- Tests
  - See next page
Open Source tests

- kernel testing
  - LTP – actually multiple test suites
  - sysbench, unixbench, hackbench, dhrystone, etc.
- system testing:
  - lsb-test, yocto ptest, debian autopkgtest
- filesystem – iozone, xfstest, bonnie, dbench
- realtime – cyclicittest, pitest
- network – iperf, netperf
- security – vuls
- vertical tests – Android Compatibility Test Suite (CTS)?
Test hardware

- Some off-the-shelf components:
  - PDUs (power distribution units)
  - Commercial multi-boards: ACME
  - Video capture: Numato Opsis, Lenking LKV373a HDMI extender

- Some open hardware:
  - Multiboards: ACME, MuxPi, Sony debug board
  - Analyzers: Sigrok-based boards (e.g. BeagleLogic)

- Different labs use different combinations of things
  - Often manually manage the hardware

- Companies have lots of in-house custom hardware solutions
  - e.g. Sony debug board has custom USB switching capability
Industry QA Efforts

- In-house
- Ad-hoc
- Lots of legacy manual testing
Unshared

- Many QA artifacts are not even tangible
  - Knowledge specific to the QA objectives
    - What tests to run?
    - Dependencies
    - Expected values – How to interpret results?
    - What metrics are important?

- Hardware testing is extremely silo-ed

- Exceptions:
  - Vendor-provided: Android compliance testing (CTS)
An analogy:
Today’s QA software = Yesterday’s RTOS
Embedded OS landscape 20 years ago

- Fragmented
- In-house
- Ad-hoc
- Unshared

Some exceptions:
- Commercial offerings: VxWorks, pSOS
- Regional industry standard: µTron
- Some open source solutions: mbed, ecos, etc.
Open Source Software vs. Testing

- Samsung, LG, Sony all produce TV sets
- 80% of software stacks in TVs are Open Source
- What percentage of QA software is Open Source?
- Which of these companies:
  - Contribute to Open Source test projects?
  - Share their TV functionality tests?
Outline

Attributes of Open Source
Status of Open Source QA
Obstacles to Sharing
Solutions and Next Steps
Silos... Why did it have to be silos...?
A tale of

- Fragmentation
- Specialization
It’s worse than just silos; The silos are all different
Obstacles to sharing

- Fragmentation
- Custom hardware
- Unique lab configurations
- Dependency on test framework (different APIs)
- Unique software
- Different testing goals
- Organizational inertia
- Licensing
Fragmentation of Test hardware API

• No consistent API for lab hardware with similar functionality

• Example: PDUs
  • There are many different ways to control power to a board in a lab
    • Serial-controlled power devices
      • APC devices, YKush, ...
    • Network-controlled power switches
      • Digital Loggers web power switch, Devantech devices
      • Custom-built network relays
    • USB-controlled power devices
      • Sony debug board, PowerUSB, etc.

• No single API for all of them
Specialization: Lab configuration

- Even with off-the-shelf hardware, labs use different hardware
  - Everyone seems to have a different PDU (Power Distribution Unit)
  - Different external power measurement devices
- Labs mix-and-match equipment
- Each lab ends up with unique combinations
- Tests written for one lab don’t work in another
Specialization: Test Framework

- Different test frameworks have different APIs, test models

<table>
<thead>
<tr>
<th></th>
<th>Fuego</th>
<th>LAVA</th>
<th>Yocto Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>test driven from</td>
<td>host</td>
<td>target</td>
<td>target</td>
</tr>
<tr>
<td>target lifecycle</td>
<td>multiple tests per boot</td>
<td>re-provision every test</td>
<td>n/a</td>
</tr>
<tr>
<td>languages</td>
<td>host: bash, python, yaml, json(target: sh)</td>
<td>sh, awk, yaml</td>
<td>python</td>
</tr>
<tr>
<td>APIs</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>dependencies</td>
<td>permission, kconfig, mem, storage</td>
<td>permission, packages</td>
<td>packages</td>
</tr>
</tbody>
</table>
Specialization: Unique software

• Vendors have different software stacks above their Open Source layers
  • e.g. Samsung’s TV stack doesn’t look like Sony’s TV stack
• More and more software is open source, and therefore common
  • Estimate is that for many products, 80% of the software is Open Source
• Lots of legacy test software
• Target distribution may have different installed software
  • e.g. may or may not have ‘expect’, ‘awk’, ‘sed’, or even ‘grep’.
Fragmentation: Different test goals

- Developers and testers with different roles:
  - System software developers (Kernel developers)
  - Distribution developers (e.g. AGL, Android, CIP, other stacks)
  - Product developers (hardware/software integrators)

- Testing to:
  - Find software regressions
  - Find integration problems
  - Meet criteria for shipping

- Interested in different parts of the system
- Different threshold of sensitivity to bugs
- Focus on local remediation vs. upstream reporting and fixing
Organizational inertia

- QA department is not yet interacting with OSS community
  - Some companies still working to have their software teams learn to interact with OSS communities
- There’s a learning curve going from using to contributing
- Testing has not historically been an open activity
  - Concern that tests reveal product info prematurely
Licensing

- License may not require sharing
- Requirement to publish is not invoked
  - QA software is not distributed by a company
  - There’s no trigger for license publication requirement
Outline

Attributes of Open Source
Status of Open Source QA
Obstacles to Sharing
Solutions and Next Steps
Solutions

- Create objects that can be shared
  - Common CI reference model
  - Standardize test definitions
    - Common APIs
    - Common languages
  - Standardize APIs to lab hardware
- Create places where objects can be shared
- Test generalization
- Interchange
A software stack
Need similar APIs
Need similar APIs
Need similar APIs
Common CI reference model

• Like protocol stack
  • Define discrete entities and interfaces
• Organizes operations and responsibilities into layers or modules
• Standard interfaces allow for interchangeable implementations
• At recent Automated Testing Summit, we produced a draft reference model
Sharable objects

- Create sharable objects
  - Test definitions usable in multiple frameworks
  - Sharable pass criteria
  - Sharable (mineable) results
    - Common result format
Sharable objects
Test definitions

• Meta-data and instructions for a running a test
• Elements:
  • Pre-requisites and dependencies
  • Information about a test
  • Instructions for test execution
  • Output parsing or conversion
  • Customizations (knobs and dials)
  • Visualization control
• See my presentation later today
Define standards for lab environment

- Board control (power, bus control, multiplexing)
- Multi-machine tests (servers, peers, simulators)
- External hardware (monitors, analyzers)
- API above and below lab controller entities
- Goal is to create ecosystems of plug-and-play modules
Place to share objects

• Create:
  • Project neutral site for collecting/disseminating objects
  • or...
  • Agreement to consolidate tests in one repository

• Issues:
  • Peer-to-peer test sharing
    • Eliminate gatekeeping for collaboration in testing community
  • Allow ad-hoc test customization and enhancement
    • For diagnosing problems
  • Support diagnosis by remote developer who has hardware needed for test
Test generalization

• Allow tests to be generalized
  • by making tests more customizable

• Ways to make a test customizable:
  • Skiplists (control of testcases)
    • Use dependencies to automate
  • Test variables
  • Expected values/Test outcomes (pass criteria)
    • Localized results interpretation
  • Data files for different use cases
    • e.g. filesystem workload - For example, dbench supports custom “loadfiles” which specify the set of operations to perform

• Preferably do automatic customization
  • e.g. Set benchmark value threshold based on previous results
Generalization vs. specialization
Generalization

- Take our pirate pieces, and turn them into LEGO\textregistered s
- Or make them compatible with LEGO\textregistered s
Next steps

- Experimenting
- Creating standards
- Cross-system interactions
- More face-to-face meetings
Next steps – Standards

• At ATS 2018:
  • Agreed to use pdudaemon as standard Power controller
    • Is the first lab API to be standardized
      • Need to actually document and standardize the API

• Working towards standardizing:
  • Test definition
  • Results definition
  • Backend API – KernelCI vs. Squad
Next steps – Interchange

- Utilize Test Stack Survey
  - See https://elinux.org/Test_Sack_Survey
- Prototype cross-use of results
  - Between Fuego and Linaro projects (LAVA, LKFT, Squad)
    - Unified results format
- Prototype cross-use of tests
  - Between Fuego and Linaro projects
  - Fuego running Linaro tests
  - LAVA running Fuego tests
Next steps – More meetings

- Meetings this week
- Some meeting this fall:
  - Automated Testing Summit #2, or
  - Plumbers Microconference
  or both
Advice from my Uber driver
Vision – super high level

Do for testing what open source has done for coding

Promote the sharing of automated CI components, artifacts, and results, the way code is shared now

• Allow components to specialize
• Support collaboration between projects
TL; DR for the talk...

- Being Open Source is not enough
- Also have to be non-fragmented
- And we have to generalize
  - That is, we have to work together

Let’s try to reduce our silos
Image sources

• tux crash test from LTP project
• Ralls_texas silos image from Wikipedia, User Leaflet CC BY SA 3.0
• silo-coal from heiligbv.com
• white-silo from liberaldictionary.com
• Port_Giles silo from wiktionary.com
• muxpi - Pawel Wieczorek, Samsung
• Lego, pirate toys, blocks, ladder chair toys from respective toy company web sites and marketing materials
• Uber driver - http://vietnamnews.vn/tags/51757/taxi-operator.html
• china dynasty picture – by Thomas Lessman (http://www.thomaslessman.com/)
• Fire images, sony debug board - by Tim Bird
Thanks

Tim Bird
Fuego Test System Maintainer
Sr. Staff Software Engineer, Sony Electronics