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 Quality Assurance (QA), as an industry and community
• 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:** LAVA, 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 – cyclictest, 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-function boards: ACME
  • Video capture: Numato Opsis, Lenking LKV373a HDMI extender

• Some open hardware:
  • Multi-function boards: ACME, MuxPi, Sony debug board
  • Analyzers: Sigrok-based boards (eg. 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
  - Each test harness is different

- Exceptions:
  - KernelCI is defacto lab standard for kernel build/boot testing
    - They have something like 10 labs now
  - Vendor-provided: Android compliance testing (CTS)
An analogy:

Today’s QA software = Yesterday’s’s RTOS
Embedded OS landscape 20 years ago

- Fragmented
- In-house
- Ad-hoc
- Unshared
- Some exceptions:
  - Commercial offerings: VxWorks, pSOS
  - Regional industry standard: μItron
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
Obstacles to sharing

- Custom hardware
- Unique lab configurations
- Dependency on test framework (different APIs)
- Unique software
- Different product use cases
- 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
- Example: power measurement hardware
  - There is no standard
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
  - For many products, 80% of the software is Open Source (estimated)
- 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 use cases

- Different products have different testing needs
  - Ex: Enterprise database server vs. mobile phone
- Big difference in hardware and usage, but...
- Both need to test:
  - syscalls
  - IPC performance
  - filesystems
  - networking
- Can they use the same tests?
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 ways to share dis-similar objects
  - Translation layers or converters
- Create places where objects can be shared
- Support test customization
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 results
    - In a format that would be mineable
Test definitions

- Meta-data and instructions for a running a test
- Elements:
  - Pre-requisites and dependencies
  - Information about a test
  - Instructions for test execution
  - Customizations (knobs and dials)
  - Output parsing or conversion
  - Results analysis
  - Visualization control
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

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

Possible uses:
- Peer-to-peer test sharing
  - Eliminate gatekeeping for collaboration in testing community
- Allow customization and enhancement of ad-hoc tests
  - For diagnosing problems
- Apply tests to board that have hardware needed for test
  - Give access to developer who does not have hardware
Test customization

- Allow generalization, by making tests customizable

Ways to make a test customizable:
- Skip lists (control of test cases)
  - 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
Next steps

- Creating standards
- Cross-system interaction (interchange)
- Experimentation
- Continued communications
  - 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
      • There is work in progress to document and standardize the API

• Working towards standardizing:
  • Test definitions
  • Results formats
  • Backend API – KernelCI vs. Squad
  • Board management API
    • pdudaemon is just a start
Next steps – Interchange

• Prototyping sharing of results
  • Between Fuego and Linaro projects (LAVA, LKFT, Squad)
    • Unified results format
  • Linaro proposal for shared results repository using Google BigQuery
    • See https://lists.yoctoproject.org/pipermail/automated-testing/2019-May/000417.html

• Prototyping cross-use of tests
  • Between Fuego and Linaro projects
    • Fuego running Linaro tests
    • LAVA running Fuego tests
  • Next target:
    • ptest (better integration), CKI?, PTS?, 0-day?
Next steps - Experimentation

- There are issues for which an approach has yet to be decided:
  - How to integrate different systems?
    - At the source level (if so, using what languages?)
    - As binary package level, containerized commands, network services?
  - Where does data live?
    - Does the user need to store information in multiple frameworks?
    - In multiple formats?
  - How to break apart currently monolithic systems?
  - What is the API between components
- Need to experiment with integration to see what approaches work
Next steps – More communication

• Using the Automated Testing mailing list
  • https://lists.yoctoproject.org/listinfo/automated-testing

• There is now a monthly call to discuss subjects
  • See https://elinux.org/Automated_Testing#Conference_call

• Face-to-face meetings
  • Plumbers “Testing and Fuzzing microconference”
    • September, Lisbon, Portugal
  • RedHat CKI Hackfest
    • September, Lisbon, Portugal
  • Automated Testing Summit 2019
    • October, Lyon, France (CFP is still open)
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
Thanks

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