Working Together to Build a Modular CI Ecosystem

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

• Introduction
• Issues
• Proposals
Standards...

How standards proliferate:
(see: A/C chargers, character encodings, instant messaging, etc)

Situation: There are 14 competing standards.

14?! Ridiculous! We need to develop one universal standard that covers everyone's use cases. Yeah!

Soon:

Situation: There are 15 competing standards.

Source: https://xkcd.com/927/  License: see https://xkcd.com/license.html
Reason for Open Source CI standards

- Many test framework systems are monolithic
  - Or at least tied closely to specific sub-components
    - e.g. Jenkins, ttc, LAVA, Beaker, buildbot, labgrid, etc.
- Want to mix and match components
- Want an ecosystem of modular CI components
- Allow for collaboration and specialization
- Reduce work!
A software stack
Need similar APIs
Need similar APIs
Need similar APIs
Key issues

- Module boundaries
- Nature of the APIs
- Language
- How to share and re-use code?
  - Install-time issues (how to access, where to install)
  - Sharing configuration data
  - Discoverability
- How to share data?
  - Common place to share objects?
  - Common formats?
Major modules and repositories

Modules
- Test manager
  - Job definition front-end
  - Job manager front-end
- Test scheduler
- Board manager
- Lab equipment manager
- Notification generator
- Report generator
- Results visualization front-end

Servers/Repositories
- Test definition repository
- Build artifact server
  - SUT image repository
  - Test binary package repository
- Job request server
- Results artifact server
- Results database
Smaller modules or pieces

- bisection tool
- testlog output parser
- provisioning system
- serial port manager
- power control manager
- expect tool
Need way to incorporate other system's pieces (modules) into our frameworks
  • Need to define modules

Definition:
  • Responsibilities
  • Interface (module APIs)
Accessing modules from other frameworks

• How?
  • Do we have to install multiple frameworks?
    • If I want to use CKI triggers, LAVA provisioning, Labgrid lab management, Fuego tests, and LKFT reporting, how would this work?
  • Do we need to split out modules as separate components?
    • Does this make things harder for our own users?
  • Can we import modules from other frameworks’ git repositories, for our framework’s users?
API options

- `<language X> library (LIB)`
  - C, python, go, Haskell, java, ruby, etc.
- Linux command line (CLI)
- Linux IPC (IPC)
- Network API (NET)
CLI-style proposal

• git-style interfaces: `<toolname> <verb> <args>`
  • Using standardized operation verbs and args
• Result data output in JSON
  • With exceptions (not json) for single-value or bulk data
• For async operations:
  • Use start/stop/collect verbs
  • With data going to a file
• Question:
  • Input as environment variables, command line args, or JSON?
CLI-style rationale

• Can easily wrapper LIB, IPC or NET interfaces with CLI
• Operations are not time-critical
  • Overhead of CLI invocation is small compared to duration of operations
• Many systems already have an existing CLI
  • But args are not standardized
    • To support common verbs and args, can extend existing tool or create thin wrapper
CLI proposal details:

- Filesystem discoverability:
  - `/usr/lib/testing` prefix
  - Propose a "<module>.d" interface, with a `<name>-<module>` program name
  - Could end in nothing, or standard executable extension (.sh, .py)
  - Examples:
    - ttc-power-control, pdud-power-control
    - grabserial-access-serial
    - lava-provision, r4d-provision
    - kciddb-results, kernelci-results, squad-results
How to get from here to there

- Take existing systems, without breaking them
- Good presentation at LinuxCon Japan (keynote)
  - Monolithic monster
  - Can’t break system while refactoring it
    - Need to break system apart slowly
    - Take a little piece at a time
- Ability to use a feature from a test framework without importing the whole system
Chicken and egg problem

- No incentive for framework author to change until benefits are available
  - e.g. not worth creating a board management API if no systems use it
  - and not worth changing test system to use a generic board management API until multiple systems provide it
- Someone has to go first
  - Actually, multiple people have to do one side of a layer for there to be benefits
    - Is this true – are the other benefits from modularity?
- Danger of locking in a bad interface
Pieces that could be isolated

- Results parser
- Smart diff (for easier expected value)
  - seddiff
- Expect (tcl-less)
  - for program control, firmware control
    - Small footprint, simplified
- bisect tool
- aggressive rmdir (see Dmitry email)
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
Conclusions

• Fragmentation makes it difficult to collaborate
• Need to identify modules, and boundaries between modules
• Start working on creating modules
  • Create internal APIs, data structures and protocols
  • Without changing functionality
• Need to decide common IPC
• Proposal:
  • Data format=json
  • Schema= <to be determined per object>
  • IPC=Linux command line
• Propose to use KernelCI for shared repositories
Thanks

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