Yocto QA Introduction

- Yocto Overall Testing Plan
- Automation Overview
Yocto Overall Testing Plan

• **Currently, following project components are covered:**
  – Core Build System (oe-core, Bitbake, meta-yocto)
  – GUI tools - Hob, Toaster, Eclipse-plugin, Build Appliance
  – Runtime - Core BSPs (qemu - all archs, genericx86, beagle bone , etc)
  – Compliance testing (LTP, POSIX)
  – Build performance and stress testing
  – ADT
  – **Distro testing**
  – Ptest

Current testing plan: https://wiki.yoctoproject.org/wiki/Yocto_1.6_Overall_Test_Plan
Automation Overview

- Image-tests
- Oe-selftest
- Ptest
- Toaster (backend)
Image tests

- Automated tests for QEMU images run by the build system
- Based on Python unittest module
- All tests are run on target over ssh
- Runs on Public AB (including systemd tests)
Oe-selftest

- Used for testing the poky build system
- Based on Python unittest module
- Simulates poky external usage patterns
- New layer with generic/specific metadata used by tests
- Runs on Public AB
Ptest (package test)

- Test suites included in packages (e.g. systemd)
- Simple or complex tests that run a binary (or set of binaries) and checking their output
- Usually written by the package developers
- Can be added to a build to test your image
- To be added to Image tests (in Yocto 1.7)
Toaster

• **Web-based interface to Bitbake and Poky**

• **Backend testing**
  – API testing
  – Data collection rate
  – Data quality collected through the UI

• **Frontend**
  – Manual testing in the first stage
Future: Automated HW testing

- Automatically test releases against a rack of existing hardware
- Currently (v1.6): Tools which enable remote testing (accelerates our own QA)
- Future: (v1.7): integrate LAVA
  - Members of the project use LAVA already
  - Have LAVA maintainers willing to work on our integration issues
  - Currently working through our 1.7 plan
QA Achievements – Automation Impact

- FP reduced from 14.5 man days to 9.75 man days