arm

Signoff Criteria for Verification by Thinking Ahead

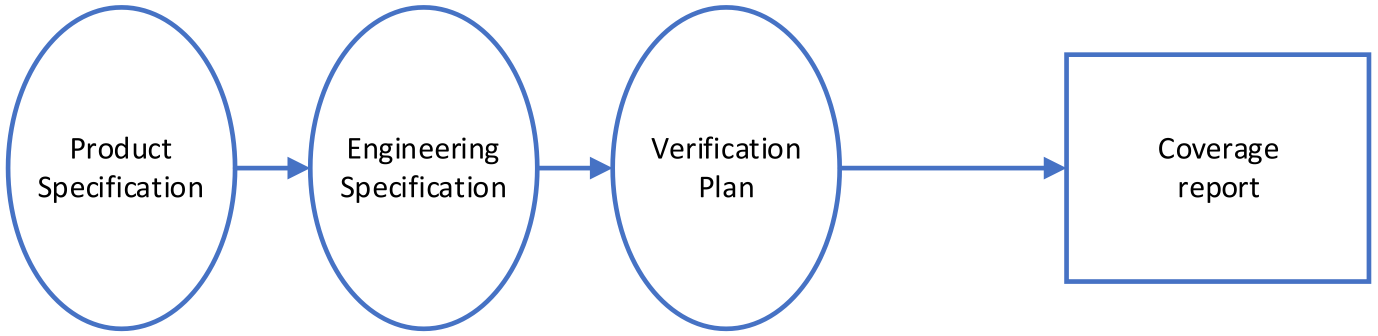
Nihit Chattar
24/11/2020

Agenda

- Requirements tracking
- Coverage and regression targets
- Top level targets
- Analysis and improvements
- Metrics and soak testing

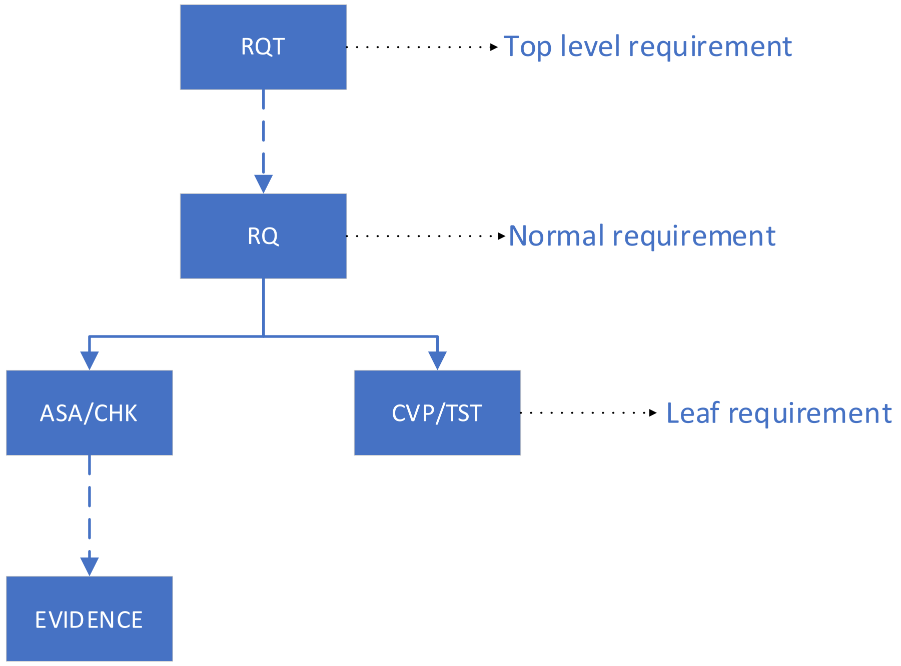
Requirements Tracking

- Requirements are mandatory tasks



- Examples of Verification requirements:
  - Coverpoints
  - Assertions
  - Checks
  - Tests

Requirements Tracking



A requirement can be covered by multiple downstream requirements

✓ Complete implementation of requirements

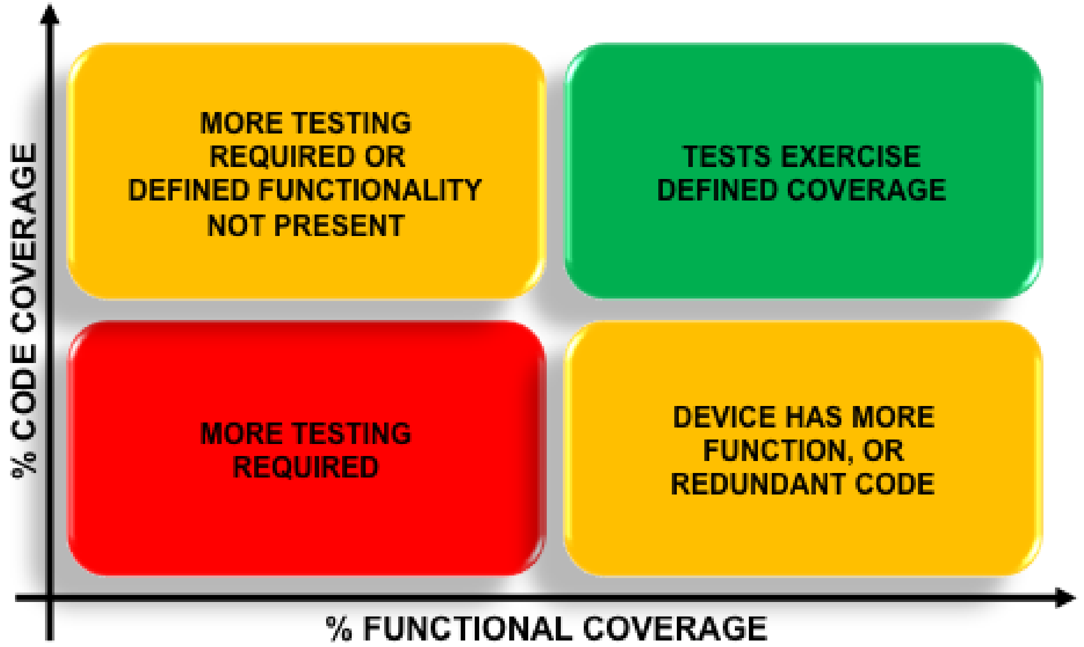
✓ Requirements tracking - traceability flow for the completeness

Coverage and Regression Targets

- ✓ Stimulus – Implemented with extension for coverage and bugs analysis
- ✓ Checkers – Fully enabled
- ✓ Smoke suite – Fully passing, coverage optimised
- ✓ Full regression –
  - Includes stress testing
  - Bugs analysis extension
  - Coverage analysis extension
  - 100% passing
- ✓ Assertions – No errors, warnings fully analysed
- ✓ Protocol checkers/VIPs for standard interfaces – No fails

Coverage and Regression Targets

- Code coverage
  - ✓ Line/branch – 95% cover, 100% explained
  - ✓ Fsm/Condition – High 90s, 100% explained
  - ✓ Toggle – 95% cover, 100% explained
- Functional coverage
  - ✓ Tier 1 – Fully covered
  - ✓ Tier 2 – High 90s, 100% explained
  - ✓ Bugs analysis extension


MORE TESTING REQUIRED OR DEFINED FUNCTIONALITY NOT PRESENT
TESTS EXERCISE DEFINED COVERAGE
MORE TESTING REQUIRED
DEVICE HAS MORE FUNCTION, OR REDUNDANT CODE
% CODE COVERAGE
% FUNCTIONAL COVERAGE

Top Level Targets

- Additional Top-level regression targets:
  - ✓ Architectural checkers enabled and passing
  - ✓ Architecture Verification suite 100% passing
  - ✓ Device Verification suite 100% passing
  - ✓ Configuration testing (including extended configs)
- Additional System-level regression targets:
  - ✓ OS compatibility and stress testing
  - ✓ System level profiling fully analysed

✓ Power aware DVS regression – 100% passing (including extended configs)

✓ Power aware specific coverage – 100%

Analysis and Improvements

Motivation :

- Random simulations are an effective way to find "hard to think of" bugs. However they tend to be very inefficient.
- Number of test runs required to hit coverage – Trial and error technique which can be inefficient.
- Associated costs – time and compute resources

✓ Regular analysis of bugs to identify test types most likely to hit issues.

✓ Use Machine learning tools to analyse and improve test suites for coverage

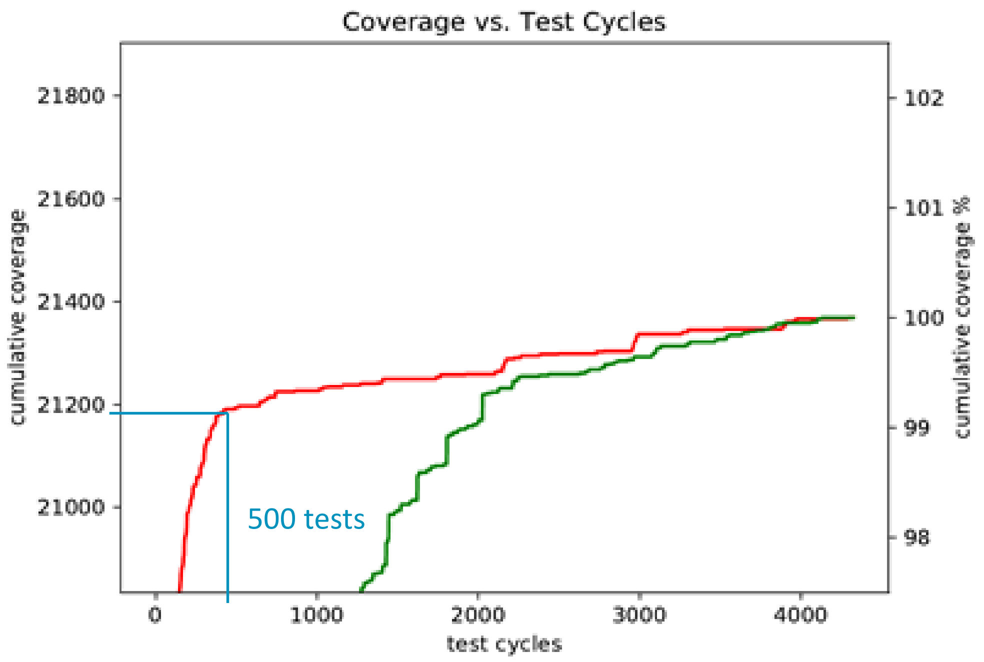
Analysis and Improvements

Example of ML tool –

- Takes test coverage database files (.ucdb files) as training data
- Measures probability of each functional coverage bin being hit by each test type
- Sorts and selects the test types most likely to hit multiple bins
- Finds new allocation of runs (seeds) between test types to hit same bins in least number of tests

Green: Training data

Red: Tool Forecast



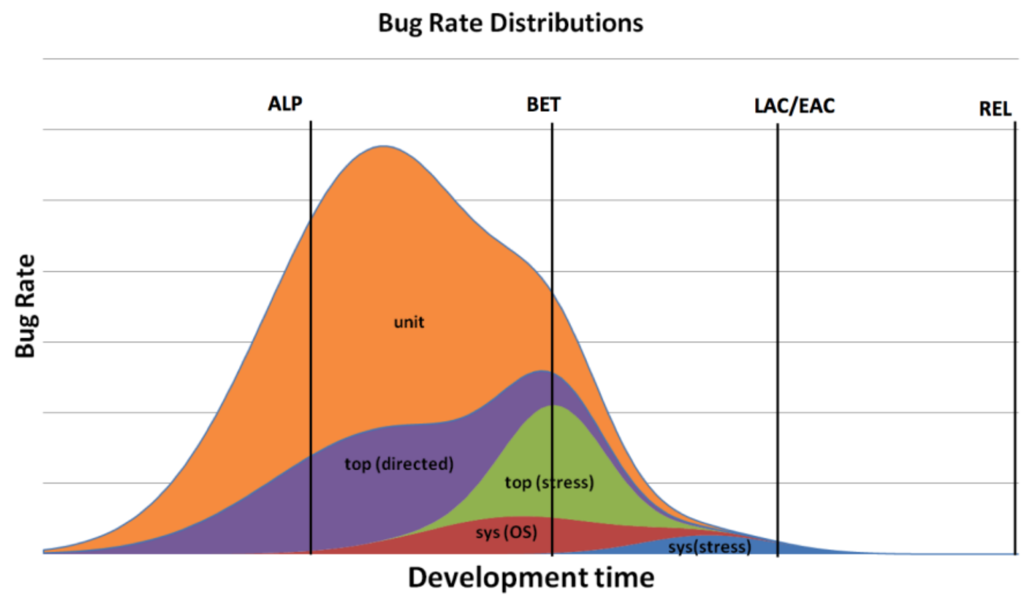
✓ Have reached slightly higher coverage than training data using only **<u>44%</u>** of the tests

Regression	Average func. coverage	Number of tests
Using original weightings (training data)	77.68	5976
Using tool optimised weightings	77.73	2650

Metrics and Soak Testing

Report metrics for:

- ✓ Volume of Unit and Top-Level soak testing cycles
- ✓ Number and severity of bugs



- ✓ Soak testing has met cycle targets with no failures
- ✓ Full bug tracking of all bugs (RTL + Verification)
- ✓ High severity RTL bugs are closed
- ✓ All confirmed bugs post LAC are presented for review and publication approval.

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה