CSC 405 Lecture Notes Week 9

OCU/Manet Testing Details

I. Revisiting SQLite as a "best practice" example.

A. Go over each point in the exec summary.

B. Consider if/how these points will be concretely realized in the ocu/manet system.

II. Executive Summary

- Three independently developed test harnesses
- 100% branch test coverage
- Millions and millions of test cases

Executive Summary, cont'd

- Out-of-memory tests
- I/O error tests
- Crash, power loss tests
- Fuzz tests
- Boundary value tests
- Disabled opt'n tests
- Regression tests
- Malformed DB tests
- Assert run-time checks
- Valgrind analysis

III. SQLite test harnesses

A. Tcl/Tk unit tests.

B. Deployed system tests.

C. User-level SQL Logic tests.

Proposal for OCU/Manet Test Harnesses

A. ***Harness 1:*** OCU Unplugged.

1. Test with "conventional" CppUnit or Google Test.
2. Alternatively, use larger-grain unit testing framework, akin to Tcl/Tk testing,
3. Driven by loop that programmtically supplies inputs to ocu via comm model.

OCU/Manet Test Harnesses, cont'd

B. ***Harness 2:*** Manet Unplugged.

   1. ***Harness 2s:*** 80211s Unplugged.

   2. ***Harness 2b:*** Batman Unplugged.

   3. Both driven by loops that programmtically supply different network configurations.

OCU/Manet Test Harnesses, cont'd

C. ***Harness 3:*** OCU + Manet integrated.

1. ***Harness 3s:*** OCU + 80211s.
2. ***Harness 3b:*** OCU + Batman.
3. Driven by manet-unplugged driver.

OCU/Manet Test Harnesses, cont'd

D. ***Harness 4:*** OCU + Manet + Laptops in the Football Field.

E. ***Harness 5:*** OCU + Manet + Surrogates and Robots in "Live" Environment.

F. ***Harness 6:*** OCU + Manet + Simulated Robots.

IV. 100% branch test coverage

A. Use `gcov` and/or `lcov`.

B. Critically important to ensure coverage of black-box tests.

C. SQLite testing handles coverage of defensive code in a novel way

V. Millions and millions of test cases

A. As a practical matter, these are programmatically generated.

B. SQLite has some interesting, potentially reusable strategies.

VI. Out-of-memory tests

A. Particularly important for C++ code, to test for memory leakage.

B. Important in general for all forms of malloc errors.

C. Use test-configured versions of malloc.

VII. I/O error tests

A. In SQLite terms "the system responds sanely to filed I/O operations".

B. Can be done with simulated I/O errors.

VIII. **Crash and power loss tests**

A. Test that state of OCU is non-corrupted if Manet or its OS crashes.

B. Test other deployed-configuration power-loss scenarios.

IX. **Fuzz tests**

A. May want to test for mutation-inducing failures of damaged robots.

B. E.g., "fuzzy" behavior that occurs when robot gets partially blown up.

X. **Boundary value and range tests**

A. The main driver of test case generation loops.

B. Data range parameters currently identified:

1. throughput
2. latency
3. signal strength
4. number of nodes
5. network target node
6. path configuration

XI. Disabled optimization tests

A. For SQLite testing, this refers to specific forms of query processing.

B. For OCU/Manet testing, it can mean that tests need to be run on both `-g` and `-O` version of the compiled code.

XII. **Regression tests**

A. Of course.

XIII. **Malformed data tests**

A. For SQLite, these are tests on various database malformations.

B. For OCU/Manet, comparable tests are for various network malformations.

C. I’m not entirely clear what external causes there may be of network malformations.

XIV. Extensive use of assert() and run-time checks

A. In SQLite, the production build disables asserts, for performance.

B. I think the same should be true in OCU/Manet.

XV. Valgrind analysis

A. `Valgrind` is a Linux simulator that analyzes for a variety of runtime errors.

B. If we have a simulated test harness, it might be interesting to run it under `valgrind`.

XVI. Re-visit testing repository structure.

XVII. **Some practical implementation details.**

A. Do a sample loop that shows concretely what programmatic driving of ocu/manet could look like.

B. Ask Batman and 80211s teams what such a loop would look like for their side of things.

XVIII. **Action Items for this Week**

A. Agreed testing framework for OCU teams.

B. Agreed testing framework for 80211s teams.

C. Committed testing for 80211s and OCU teams.

D. Project-wide regression test makefile.

E. Agreement, as appro, on preceding practices.

XIX. Suggested repository updates.

A. Populate `testing` subirs for all 4 subprojects.

B. Move `batmobile/implementation/.../*Tests*` to `batmobile/testing/implementation/.../*Tests*`.

C. Code tests and install for `80211s`, `kareem-nassar`, `ocunited`.

Repository additions and modifications, cont'd

D. Add `manet-ocu/testing` dir, with `Make-file` for project-wide test build and execute.

E. Install bug-tracking supporting infrastructure (if not already there).

F. Add `requirements` dir and put SRS there.

G. Add `administration` dir and put project-wide admin docs there.