**Random-Walk Simulation of Reliability Testing**

**Students**

TBD graduate student.

**Goal**

Goal: Provide student experience performing reliability testing.

Approach: Simulate the failure discovery events of software undergoing reliability testing as a *random walk*. Faults are seeded at specific coordinates in the walk. When a fault is reached, a failure is reported. The tester fixes the fault, and resumes reliability testing. Based on the time (count) of failure, a time-based or count-based reliability growth model can be constructed and used to determine when to stop testing.

Prototype the design for a test arcade, where players compete solving testing problems. Players are ranked by time-to-complete and correctness.

**Importance**

Reliability testing requires considerable time and expense. Simulation overcomes this limitation when giving students experiences defining reliability growth models.

**Status**

START-UP.

**Technical Issues**

Major activities:

The architecture of the RANDREL is shown in the following pages.

**Resources**

*Tools:*

*Data:* TestLab.

*Hardware:* Sun Unix system.

**References**

See companion paper,

(1) ACM Southeast Conference
(2) International Conference on Software Testing
(3) Journal of Computing in Small Colleges.
RANDOM-WALK RELIABILITY SIMULATOR

Each fault has a failure signature (wrong result, wrong result with loss of state (loss of current position within the walk), program termination. When a fault point is reached, the failure is reported in the Failures Report and recorded in the Fix List file. Next, the the impact of the fault on the system is simulated, resulting in program termination or continued execution.

The tester uses the Failures Report to determine the failure times. These are used in the reliability model. The tester then removes the faults by executing the Remove_Faults program. Remove_Faults may model perfect fixes or imperfect fixes (additional faults are introduced). After Remove_Faults has been executed, reliability testing resumes.

Reliability testing continues until the reliability growth criteria have been satisfied.

![Diagram](image)

**Generality:** Several. This simulation models state space exploration during testing. Several parameters can be adjusted to inject realism into the simulation.

- **State transition times.** The time required to transition between states can be modeled as deterministic or stochastic, with a time reflective of the time scale used during reliability testing. Generally, the MTBF increases as total testing time increases.

- **Fault discovery.** Fault masking can be modeled when the simulation loses its current state. Suppose faults exist at successive positions \( p \) and \( q \), where \( q = \text{successor}(p) \). Only \( p \) will be discovered.

- **Failure discovery.** The Failures Report can be modified to output time-stamped failures or time-stamped walk positions. With the latter output the tester is responsible for detecting failures and identifying the location of the fault.

- **Fault removal.** The tester identifies the fault location, then invokes the Remove_Fault program to remove the fault. Remove_Fault has a fault-injection probability argument reflecting imperfect fault removal. For elementary studies, the probability is zero. There is also an
associated probability of injecting multiple faults. Finally, there is a mean time to repair that is applied when at the time of fault removal.

**Simulation**

Several styles of simulation are permitted:

1. Manual Simulation – the simulation is interrupted each time a severe/fatal failure is encountered. A failures report is generated at the time of simulation interruption. The tester manually diagnoses the failure, and locates the fault, then runs the Remove_Fault program.

2. Continuous Simulation – the simulation is run to completion (criteria include #failures, elapsed time, fault rate/intensity). Each failure is diagnosed by the simulator, which attempts to remove the corresponding fault.

For both styles, fault-based or time-based analysis may be selected. Time based simulation requires a cumulative execution model, and a MTTR model.

**Utilization**

TBD.