Preparing for Test

Bojan Cukic

Department of Computer Science and Electrical Engineering
West Virginia University
So far

SRE Ideas

- Focus resources on the most critical/used functions
- Process used throughout the life-cycle
- User oriented, rather than system oriented
- Necessary reliability defined early in the life-cycle
- Make testing realistically represent field conditions
  - How will users realistically deploy this product
  - Develop operational modes
  - Define operations
  - Develop operational profile
For the given requirements specification, define:

- **Necessary reliability (Workshop 1)**
  - Define failure of the product. Your answer should allow testers to derive classification of failures.
  - Choose the natural or execution time unit for the system
  - Set failure intensity objective.

- **Develop operational profile**
  - What factors yield different operational modes; List operational modes.
  - Identify operations and operation initiators (user types, external syst.)
  - Decide on tabular vs. graphical representation of the OP
  - List operational modes and operations in the chosen representation
  - Determine occurrence rates and probabilities? Make estimates.
The big picture

List Associated Systems
Define Necessary Reliability
Develop Operational Profiles

Prepare for Test

Execute Tests & Apply Failure Data to Guide Decisions

Requirements and architecture
Design and Implementation
Test & Validation
Preparing for Test

- Preparation of test cases
- Preparation of test procedures
- SRE provides guidance for
  - Feature testing
    - execution of each system feature, interaction minimized
  - Load testing
    - Mimics field usage, concurrent executions, interactions
    - Test each mode separately, maximizing the realism of interactions
  - Regression tests
    - Feature tests, conducted after the system build changes
Time intervals for regression tests: Week-Month.
- Depends on system size, volatility of requirements, response to changed market conditions.

What to include in regression tests?
- What to exclude (ripple effects)?
  - Typically, includes all system features
  - If resources limited, select features/operations according to OP

Load testing is ideal for measurement of performance characteristics in addition to pure reliability issues.
- Opportunity to define performance related faults and failure modes.
Concepts

- Testing performed by executing a set of RUNS.
  - RUN: A specific instance of program operation
  - Characterized by its *input state* (set of input and “state” variables) and their values.
- Input variables
  - Variables external to the program that affect its execution.
- Input space is a complete set (union) of input states.
Definitions

- **Input variables can be:**
  - *Direct*: arguments, menu selections, entered data, which, in turn, controls the operation execution.
  - *Indirect*: traffic load, soak time, interactivity, database degradation, OS degradation.

- The impact of indirect (environmental) influence on program execution frequently too difficult to explain.
  - Resource conflicts, overflows in look-up tables, accesses to shared memory.
Different runs characterize each operation

A run involves the execution of a test case

- **TEST CASE + indirect variables = RUN**
- **Same test case executed in different modes generates different runs.**
- **Test cases are partial specifications of test runs!**

<table>
<thead>
<tr>
<th>Direct</th>
<th>Indirect</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Test case)</td>
<td>Operational mode = prime hours</td>
</tr>
<tr>
<td>originator = 304-293-0405</td>
<td>Database state: time dependent</td>
</tr>
<tr>
<td>forwardee = 908-124-7465</td>
<td>Resource state: time dependent</td>
</tr>
<tr>
<td>Billing type = per call</td>
<td></td>
</tr>
<tr>
<td>Dialing type = tone</td>
<td></td>
</tr>
<tr>
<td>Screening = yes</td>
<td></td>
</tr>
</tbody>
</table>
Test Runs

- Restrict the influence of indirect variables in feature (and possibly regression) tests.
  - Frequently achieved by re-initialization.
- Maximize the influence of indirect variables during the load tests.
  - Load tests divided for operational modes.
  - This eases the debugging process (fault identification).
- TEST PROCEDURE
  - Sets environmental conditions and invokes test cases.
Test procedures

- Characterized by invocation of randomly chosen test cases at random times.
  - Requires the ability to generate random tests!!!
- Selection based on the OP for an operational mode.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Occ. Rate (per hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone no. entry</td>
<td>30,000</td>
</tr>
<tr>
<td>Process voice call, no pager, answer</td>
<td>110,000</td>
</tr>
<tr>
<td>Process voice call, no pager, no answer</td>
<td>90,000</td>
</tr>
<tr>
<td>Process voice call, pager, answer</td>
<td>90,000</td>
</tr>
<tr>
<td>Process voice call, pager, answer on page</td>
<td>60,000</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Procedure

- Procedure involves:
  - Preparing test cases.
  - Preparing test procedures.
  - Recording time for failure acknowledgement purposes.
Preparing Test Cases

- Recording real inputs in the field.
- Might mirror “old” system usage, old system loads.
- Should not miss rarely occurring but critical runs.
- Steps.
  1. Estimating the number of new test cases needed to test the current release.
  2. Allocating tests among systems to be tested.
  3. Allocating NEW test cases.
  4. Specifying new test cases.
  5. Merging old and new test cases (if multiple releases).
Estimating # new test cases

- The state of practice indicates:
  - Take a **minimum** of
    - How much time you have available
      (The time between the stabilization of requirements and the date to start testing).
    - How much is the development of new tests going to cost.

- Test case development usually takes 5% - 10% of the overall budget!
  - Grows with required reliability level.
Available time = 600[h], staff = 3.5[person], average time to develop test = 3 hours
   2100[staff hours] / 3 = 700 tests

Available budget = $2m, 10% devoted to test case development, cost of preparing a test = $250.
   $200,000 / $250 = 800 tests

Plan to develop 700 tests!

Run a SANITY CHECK
   # New tests >> # new operations (50 to 100 more, if frequently occurring operations).
   Experience.
Improvements

- Develop more tests in the given time.
  - Experience.

- Execute many more runs than test cases.
  - Have runs “evenly spaced” across an operation, i.e., equidistant in covering the “inputs” of an operation.
  - This maximizes the likelihood of finding faults.

- Improve (automate) test case generation.
  - Tools + research.
  - Random test case generators.
    - For example, randomly generate indirect variables within a test case.
  - Path coverage generators.
  - *Test trajectory generators.*
Allocating test cases to (sub)systems

- Allocate bulk of tests to the SYSTEM (product).
  - Adjust to subsystems to reflect changes.
- Base the selection on the operational usage.
- Allocate test cases to acquired components in the first release.
  - Exception in cases of substitution, later releases may require additional tests.
- Target risky components and frequently used components.

1. Convert operational profile to tabular representation.
   ↗ Estimate execution probability.

2. Identify rare but critical operations and preassign the number of test cases to each.

3. Determine allocation probabilities for other new operations.

4. Preassign one new test to each new infrequent operation.

5. Assign remaining test cases in accordance with the occurrence probabilities.
500 preassigned test cases in the first release.

For release 2, depending on the amount of change and operational distribution, a different number would be assigned.
Specifying test cases

- Specify test cases in such a way that they “cover” the test space.
  - Long distance calls vs. local calls.
  - Within the same “level” (variable values likely to trigger same faults) assign values to variables randomly.

- Runs are defined by “ranges of values” assigned to variables.
  - Define the set of runs forming an “equivalence class”.
  - Try to cover all combinations of classes.

- Write test scripts.
  - Add new tests to the database and use for future releases.
## Equivalence classes

<table>
<thead>
<tr>
<th>Originator</th>
<th>Forwardee</th>
<th>Billing Type</th>
<th>Dialing Type</th>
<th>Originator Screaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>On screenlist</td>
<td>Local calling area</td>
<td>Flat rate</td>
<td>Standard</td>
<td>Yes</td>
</tr>
<tr>
<td>Not on screenlist</td>
<td>Within area code</td>
<td>Per call</td>
<td>Abbreviated</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Outside area code</td>
<td>Discount</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>International</td>
<td>Per minute</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>800 number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other carrier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wireless</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For each new test case, for each new operation:

- Randomly select equivalence class for each direct input variable.
  - Equal probability among possible choices.
  - Reject duplication of equivalence classes.
- Select value of direct input variable within each equivalence class.
  - Strongly failure prone (boundary value), and/or.
  - Random.
Procedure

- Procedure involves
  - Preparing test cases
  - Preparing test procedures
  - Recording time for failure acknowledgement purposes
Preparing Test Procedures

One test procedure for each operational mode must be developed.

Start with:

1. Operational mode’s operational profile for the first release.
2. Test operational profile (or usage profile) from previous releases, if they exist.
3. Account for the rare but critical events.
Unadjusted OP, peak hours operational mode

<table>
<thead>
<tr>
<th>Operation</th>
<th>Occurrence Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice call, no pager, answer</td>
<td>0.22</td>
</tr>
<tr>
<td>Voice call, no pager, no answer</td>
<td>0.18</td>
</tr>
<tr>
<td>Voice call, pager, answer</td>
<td>0.18</td>
</tr>
<tr>
<td>Fax call</td>
<td>0.14</td>
</tr>
<tr>
<td>Voice call, pager, answer on page</td>
<td>0.12</td>
</tr>
<tr>
<td>Voice call, pager, no answer</td>
<td>0.1</td>
</tr>
<tr>
<td>Phone number entry</td>
<td>0.06</td>
</tr>
<tr>
<td>Recover from HW failure</td>
<td>0.000001</td>
</tr>
<tr>
<td>Add subscriber</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
</tr>
</tbody>
</table>

- These occurrence probabilities are different from the system level (no admin tasks).
- By assigning 2 out of 500 tests to “recover from HW failure”, we “increased” its probability 4000 times! *Its new probability is 0.004.*
- This factor is called *acceleration factor.*
  - Accelerated testing the only way to test rare events.
- How much did we change operational distribution?
  - About 0.4%
  - Changes between 0% and 25% appear to be reasonable.
**Adjusted OP, peak hours operational mode**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Initial Operational Profile</th>
<th>Final operational Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice call, no pager, answer</td>
<td>0.22</td>
<td>0.219</td>
</tr>
<tr>
<td>Voice call, no pager, no answer</td>
<td>0.18</td>
<td>0.179</td>
</tr>
<tr>
<td>Voice call, pager, answer</td>
<td>0.18</td>
<td>0.179</td>
</tr>
<tr>
<td>Fax call</td>
<td>0.14</td>
<td>0.139</td>
</tr>
<tr>
<td>Voice call, pager, answer on page</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td>Voice call, pager, no answer</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Phone number entry</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>Recover from HW failure</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Add subscriber</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1.04</strong></td>
<td><strong>1</strong></td>
</tr>
</tbody>
</table>

Column “Final Operational Profile indicates the test operational profile
For testing of the first release of Fone Follower in the peak hours
Operational mode.
Interaction factor 0.2 assumed when creating initial operational profile for the 2nd release.

Final OP obtained by dividing initial OP with the sum (0.4).
Improvements

- Test automation
  - test management
  - test failure identification
  - automated test oracles, the field still in its infancy

- Improving test efficacy
  - \( n = 1/f_i \times df_i/dt \), where
    - \( n \) is the relative rate of reduction in failure intensity
    - \( f_i \) is the failure intensity
    - \( t \) is the execution time

- Define the set of runs forming an “equivalence class”