Random Testing: Outline

- RT: advantages and tradeoffs
- RT vs pseudorandom testing (PR)
- Coverage and detectability profile
- Hardware and software DPs
- C(L) for random and pseudorandom tests
- High and low testability faults during early & late testing
- Implications of a late asymmetric profile
Random Testing

- Extensively used for both hardware and software
- Ideally each input is selected randomly. PR (Pseudorandom) schemes approximate random.
- Generally quite effective for moderate coverage.
  - Coverage hard to determine a priori.
  - Ineffective for random-pattern-resistant faults.
  - Coverage tools: Random (functional) followed by Structural testing.

Random Testing: Advantage

- No test generation using structural information needed.
- Test set-up using comparison:
  - Unit under test
  - Random pattern generator
  - Gold Unit
  - comparator
  - Unit under test
  - Random pattern generator
  - Stored response
  - comparator

- Alternative: Is response reasonable? (software testing)
Pseudorandom (PR) Testing

- Unlike true random, reproducible.
- Will not repeat until all combinations applied.
- Generation: usually just-in-time (not stored).
  - Autonomous linear feedback shift register (ALFSR).
  - Cellular automata etc possible.
- Some randomness properties satisfied, but not all.

Coverage Achieved

- Coverage grows fast in the beginning, saturates near end.
- Is it described by
  - \( C(L) = 1 - e^{-aL} \) ?
  - No, doesn’t fit.
- It is controlled by distribution of detectability of faults.
- Detectability profile (Malaiya & Yang '84):
  - \( H = \{ h_1, h_2, \ldots, h_N \} \)
    - \( N \): total possible vectors
    - \( h_k \): number of faults detected by exactly \( k \) vectors.
  - Total faults \( M = \sum h_k \)
    - \( h_1 \): number of least testable faults

\[
\begin{align*}
0.975 & \quad 0.274 \\
0.95 & \quad 0.25 \\
0.9 & \quad 0.26 \\
0.8 & \quad 0.26 \\
0.75 & \quad 0.26 \\
0.5 & \quad 0.26 \\
0.25 & \quad 0.26 \\
0 & \quad 0.26
\end{align*}
\]
Detectability Profiles: Ex

- **CECL Full adder**
  - Inputs = 4 (N=16), M=90
  - H = (h₁, h₂, h₃, h₄, h₅, h₆, h₈) = (1, 11, 43, 21, 4, 8)

- **Schneider’s counterexample:**
  - Inputs = 4 (N=16), M=44
  - H = (h₁, h₂, h₃, h₁₄) = (23, 19, 1, 1)

Coverage with L random vectors

- hₖ out of M defects detectable by exactly k vectors: detection probability k/N
- P(a defect with dp k/N not detected by a vector) = \((1 - \frac{k}{N})\)
- P(a defect with dp k/N not detected by L vectors) = \((1 - \frac{k}{N})^L\)
- Of hₖ faults, expected number not covered is \((1 - \frac{k}{N})^L hₖ\)
- Expected test coverage with L vectors
  \[ C(L) = 1 - \sum_{k=1}^{N} \frac{(1 - \frac{k}{N})^L hₖ}{M} \]
Ex: C(L) and components for CECL Full Adder

**CECL full adder**

<table>
<thead>
<tr>
<th>k = 1</th>
<th>11</th>
<th>2</th>
<th>43</th>
<th>21</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.0625</td>
<td>0.1250</td>
<td>0.1875</td>
<td>0.2500</td>
<td>0.3125</td>
<td>0.3750</td>
<td>0.5000</td>
</tr>
<tr>
<td>0.2758</td>
<td>0.4871</td>
<td>0.6459</td>
<td>0.7627</td>
<td>0.8464</td>
<td>0.9046</td>
<td>0.9688</td>
</tr>
<tr>
<td>0.4755</td>
<td>0.7389</td>
<td>0.8746</td>
<td>0.9437</td>
<td>0.9794</td>
<td>0.9909</td>
<td>0.9990</td>
</tr>
<tr>
<td>0.6292</td>
<td>0.8651</td>
<td>0.9059</td>
<td>0.9646</td>
<td>0.9994</td>
<td>0.9993</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.7249</td>
<td>0.9008</td>
<td>0.9433</td>
<td>0.9968</td>
<td>0.9994</td>
<td>0.9999</td>
<td>0.9999</td>
</tr>
</tbody>
</table>

**Coverage of partitions**

- **covered**
  - 0.72
  - 10.24
  - 1.97
  - 42.86
  - 20.99
  - 4.00
  - 8.00

- **remaining**
  - 0.28
  - 0.76
  - 0.03
  - 0.14
  - 0.01
  - 0.00
  - 0.00
Shift in profile with progress in testing

Coverage Obtained by L Vectors

- For PR tests (McClusky 87)
  \[ C(L) = 1 - \frac{\sum_{k=1}^{N-L} \frac{N-k}{C_k} \frac{h_k}{M}}{M} \]
  \[ = 1 - \sum_{k=1}^{N} \left( 1 - \frac{k}{N} \right) \frac{h_k}{M} \quad \text{(for Random)} \]
- For large L, terms with only low k (i.e. faults that are hard to test) have an impact. Thus only lower elements of H need to be estimated.
- For CECL Full Adder,
  \[ C(15) = 1 - [4.2 + 16.4 + 0.9 + 6.3 + 0.84 + 0.03 + 0 + ...] \cdot 10^{-3} \]
Detectability Profile: software

- Regardless of initial profile, after some initial testing, the profile will become asymmetric.
- Dunham’s data based on NASA experiments for 16 faults.

![Graph showing detectability profile for software](image1)

Detectability Profile: software

- Adam’s Data

![Graph showing Adam’s data for Product 1](image2)
Detectability Profile: Software

- Software detectability profile is exponential (Adam’s data, IBM).
- Justification: Early testing will find & remove easy-to-test faults.
- Testing methods need to focus on hard-to-find faults.

Implications: Fault Seeding

- A program has x defects. We want to estimate x.
- Seed j new faults.
- Do some testing. Let faults found be j₁ seeded faults and x₁ original faults.
- Assuming j₁/j = x₁/x we get \[ x = x₁ \frac{j}{j₁} \]
- However, in reality the x faults include harder faults to test,
  \[ \frac{j₁}{j} > \frac{x₁}{x} \text{ hence } x > \frac{x₁j}{j₁} \]
Implications: Estimation by Inspection Sampling

- Software with $x$ bugs is inspected by two separate teams that finds $x_1$ and $x_2$ bugs respectively, of which $x_3$ are shared.
- Assuming $x_1/x = x_2/x_3$ we get
  \[ x = \frac{x_1 x_2}{x_3} \]
- However actually since $x$ includes more harder to test faults,
  \[ \frac{x_3}{x_2} > \frac{x_1}{x} \text{ hence } x > \frac{x_1 x_2}{x_3} \]

Implications: fault exposure ratio

Let $N(t)$ be the number of bugs at time $t$ during testing, then if $a$ is a parameter,

\[ \frac{dN(t)}{dt} = -aN(t) \]

If $a$ is constant, then $N(t) = N(0)e^{-at}$ [expo SRGM]

However in random testing $a$ should decline as faults get harder to find.

If testing is intelligent, then $a$ can rise, which can give rise to Logarithmic SRGM.
References


• J R Dunham, ”Experiments in software reliability: Life-critical applications,” IEEE Tran. SE, January 1986, pp. 110 - 123