Software Reliability Growth: Outline

• Testing approaches
• Operational Profile
• Software Reliability Growth Models
  • Exponential
  • Logarithmic
• Model evaluation: error, bias
• Model usage
  • Static estimation before testing
  • Making projections using test data
Test methodologies

• Static (review, inspection) vs. dynamic (execution)

• Test views
  • Black-box (functional): input/output description
  • White box (structural): implementation used
  • Combination: white after black

• Test generation
  • Partitioning
  • Random/Antirandom/Deterministic

• Input mix

Input Mix: Operational Profile

• Need to do
  • find bugs fast?
  • estimate operational failure intensity?
  • Best mix for efficient bug finding (Li & Malaiya)
    • Quick & limited testing: Use operational profile
    • High reliability: Probe input space evenly
      • Operational profile will not execute rare and special cases
  • In general: Use combination
• For acceptance testing: Need Operational profile
Operational Profile

- **Profile**: set of disjoint actions, operations that a program may perform, and their probabilities of occurrence.
- **Operational profile**: probabilities that occur in actual operation
  - Begin-to-end operations & their probabilities
  - Markov: states & transition probabilities
- There may be multiple operational profiles.
- Accurate operational profile determination may not be needed.

### Operational Profile Example

- **“Phone follower” call types (Musa)**

<table>
<thead>
<tr>
<th></th>
<th>Operation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Voice call</td>
<td>0.74</td>
</tr>
<tr>
<td>B</td>
<td>FAX call</td>
<td>0.15</td>
</tr>
<tr>
<td>C</td>
<td>New number entry</td>
<td>0.10</td>
</tr>
<tr>
<td>D</td>
<td>Data base audit</td>
<td>0.009</td>
</tr>
<tr>
<td>E</td>
<td>Add subscriber</td>
<td>0.0005</td>
</tr>
<tr>
<td>F</td>
<td>Delete subscriber</td>
<td>0.000499</td>
</tr>
<tr>
<td>G</td>
<td>Hardware failure recovery</td>
<td>0.000001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Operation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Voice call, no pager, answer</td>
<td>0.18</td>
</tr>
<tr>
<td>A2</td>
<td>Voice call, no pager, no answer</td>
<td>0.17</td>
</tr>
<tr>
<td>A3</td>
<td>Voice call, pager, voice answer</td>
<td>0.17</td>
</tr>
<tr>
<td>A4</td>
<td>Voice call, pager, answer on page</td>
<td>0.12</td>
</tr>
<tr>
<td>A5</td>
<td>Voice call, pager, no answer on page</td>
<td>0.10</td>
</tr>
</tbody>
</table>
Modeling Reliability Growth

- Testing cost can be 60% or more
- Careful planning to release by target date
- Decision making using a software reliability growth model (SRGM). Obtained using
  - Analytically using assumptions, or and
  - Based on experimental observation
- A model describes a real process approximately
- Ideally should have good predictive capability and a reasonable interpretation

Exponential Reliability Growth Model

- We need a model to describe
  - **Total expected faults** detected by time $t$: $\mu(t)$
  - **Failure intensity: fault detection rate** $\lambda(t)$
  - **Undetected defects present at time $t$**: $N(t)$
- **Note that**
  \[
  \lambda(t) = \frac{d}{dt} \mu(t) = -\frac{d}{dt} N(t)
  \]
Exponential SRGM (cont)

- \( T_s \): average single execution time
- \( k_s \): expected fraction of faults found during \( T_s \)
- \( T_L \): time to execute each program instruction once

\[
- \frac{dN(t)}{dt} = k_s N(t) \\
- \frac{dN(t)}{dt} = \frac{K}{T_L} N(t) = \beta_1 N(t)
\]

where \( K = k_s \frac{T_L}{T_s} \) is fault exposure ratio

Exponential SRGM (Cont.)

- We get

\[
N(t) = N(0) e^{-\beta_1 t}
\]

\[
\mu(t) = \beta_o (1 - e^{-\beta_1 t}) \quad \lambda(t) = \beta_o \beta_1 e^{-\beta_1 t}
\]

- For \( t \to \infty \), total \( \beta_o = N(0) \) faults would be eventually detected. A “finite-faults-model”.
- Assumes no new defects are generated during debugging.
- Proposed by Jelinski-Muranda ’71, Shooman ‘71, Goel-Okumoto ’79 and Musa ’75-’80. also called Basic.
A Basic SRGM (Cont.)

- **Parameter** $\beta_1$ **is given by**:

$$\beta_1 = \frac{K}{T_k} = \frac{K}{(S, Q, r)}$$

- $S$: source instructions,
- $Q$: number of object instructions per source instruction,
- $r$: object instruction execution rate of the computer
- $K$: *fault-exposure ratio*, range $1 \times 10^{-7}$ to $10 \times 10^{-7}$, (t is in CPU seconds). Assumed constant here.

SRGM: “Logarithmic Poisson”

- Many SRGMs have been proposed.
- **Logarithmic** model, by Musa-Okumoto, found to have a good predictive capability

$$\mu(t) = \beta_o \ln(1 + \beta_1 t) \quad \lambda(t) = \frac{\beta_o \beta_1}{1 + \beta_1 t}$$

- Applicable as long as $\mu(t) \leq N(0)$. Practically always satisfied. Term infinite-faults-model misleading.
- Parameters $\beta_o$ and $\beta_1$ don’t have a simple interpretation. A useful interpretation by Malaiya and Denton.
Comparing Models

- Goodness of fit: may be misleading
- Predictive capability
  - Data points: \((\lambda_i, t_i), i=1 \text{ to } n\)
  - Total defects found: \(D\), estimated at \(i\): \(D_i\)

  \[
  \text{Average error} : \ AE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{D_i - D}{D} \right|
  \]

  \[
  \text{Average bias} : \ AB = \frac{1}{n} \sum_{i=1}^{n} \frac{D_i - D}{D}
  \]

- We used a many datasets from diverse projects for comparing different models.

Bias in SRGMs

- Malaiya, Karunanithi, Verma ('90)
SRGM: Preliminary Planning

- Example:
  - initial defect density estimated 25 defects/KLOC
  - 10,000 lines of C code
  - computer 70 million object instructions per second
  - fault exposure ratio $K$ estimated to be $4 \times 10^{-7}$
  - Estimate the testing time for defect density 2.5/KLOC

- Procedure:
  - Find $\beta_0$, $\beta_1$
  - Find testing time $t_1$

SRGM: Preliminary Planning (cont.)

- From exponential model

\[
\beta_0 = N(0) = 25 \times 10 = 250 \text{ defects,}
\]

\[
\beta_1 = \frac{K}{(S.Q. \cdot \frac{l}{r})} = \frac{4.0 \times 10^{-7}}{10,000 \times 2.5 \times \frac{1}{70 \times 10^6}} = 11.2 \times 10^4 \text{ per sec}
\]
SRGM: Preliminary Planning (cont.)

• Reliability at release depends on

\[
\frac{N(t)}{N(O)} = \frac{2.5 \times 10}{25 \times 10} = \exp(-11.2 \times 10^{-4} t)
\]

\[
t = \frac{-\ln(0.1)}{11.2 \times 10^{-4}} = 2056 \text{ sec. (CPU time)}
\]

\[
\lambda(t) = 250 \times 11.2 \times 10^{-4} e^{-11.2 \times 10^{-4} t}
\]

\[
= 0.028 \text{ failures/sec}
\]

SRGM: Preliminary Planning (cont.)

• For the same environment, \( \beta_1 \), S is constant.
  • Prior 5 KLOC project \( \beta_1 \) was \( 2 \times 10^{-3} \) per sec.
  • New 15 KLOC project, \( \beta_1 \) can be estimated as \( 2 \times 10^{-3}/3 = 0.66 \times 10^{-3} \) per sec.

• Value of fault exposure ratio (K) may depend on initial defect density and testing strategy (Li, Malaiya '93).
SRGM: During Testing

- Collect and pre-process data:
  - To extract the long-term trend, data needs to be smoothed
  - *Grouped* data: test duration intervals, average failure intensity in each interval.

- Select a model and determine parameters:
  - past experience with projects using same process
  - exponential and logarithmic models often good choices
  - model that fits early data well, may not have best predictive capability
  - parameters estimated using *least square* or *maximum likelihood*
  - parameter values used when *stable* and *reasonable*.

SRGM: During Testing (cont.)

- Compute how much more testing is needed:
  - fitted model to project additional testing needed
    - desired failure intensity
    - estimated defect density
  - recalibrating a model can improve projection accuracy
  - Interval estimates can be obtained using statistical methods.
Example: SRGM with Test Data

<table>
<thead>
<tr>
<th>CPU Hours</th>
<th>Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
</tr>
</tbody>
</table>

• Target failure intensity 1/hour (2.78 x 10^{-4} per sec.)

Example: SRGM with Test Data (cont.)

• Fitting we get
  \[ \beta_0 = 101.47 \quad \text{and} \quad \beta_1 = 5.22 \times 10^{-5} \]

• Stopping time \( t_f \) is then given by:
  \[ 2.78 \times 10^{-4} = 101.47 \times 5.22 \times 10^{-5} e^{-5.22 \times 10^{-5} t_f} \]

• Yielding \( t_f = 5 \, 647, \, 473 \) sec., i.e. 15.69 hours
Example: SRGM with Test Data (cont.)

- Accuracy of projection:
  - Experience with Exponential model suggests
  - estimated $\beta_0$ tends to be lower than the final value
  - estimated $\beta_1$ tends to be higher
  - true value of $t_f$ should be higher. Hence 15.69 hours should be used as a lower estimate.

- Problems:
  - test strategy changed: spike in failure intensity
    - smoothing
  - software under test evolving - continuing additions
    - Drop or adjust early data points