Fault Tolerant Computing
CS 530
Software Reliability Growth

Yashwant K. Malaiya
Colorado State University
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
Software Reliability Growth Models

• This field is the classical part of “Software Reliability Engineering” (SRE).
• During testing and debugging, the number of bugs remaining reduces, and the bug finding rate tends to drop.
• **When to Stop Testing Problem**: Given a history of bug finding rate, when will it drop below an acceptable limit, so that the software can be released.
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** the input domain
  • **Random/Antirandom/Deterministic**

• **Usual assumption**: the test method does not change during testing.
  • In practice testing approach does change, which causes some statistical fluctuations.
Input mix: Test Profile

• The inputs to a system can represent different types of operations. The input mix called “Profile” can impact effectiveness of testing.

• For example a Search program can be tested for text data, numerical data, data already sorted etc. If most testing a done using numerical data, more bugs related to text data may remain unfound.
Input Mix: Testing “Profile”

- The ideal Profile (input mix) will depend on the objective
  - A. Find bugs fast? or
  - B. Estimate operational failure intensity?

A. Best mix for efficient bug finding (Li & Malaiya’94)
  - Quick & limited testing: *Use operational profile (next slide)*
  - High reliability: *Probe input space evenly*
    - Operational profile will not execute rare and special cases, the main cause of failures in highly reliable systems.
  - In general: Use combination

B. For acceptance testing: Need Operational profile

---


H. Hecht, P. Crane, Rare conditions and their effect on software failures, Proc. Annual Reliability and Maintainability Symposium, 1994, pp. 334-337
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

- Assume PhoneFollower software that handles incoming calls to a PABX unit.
- Incoming call types & other operations (total 7 types) are monitored to estimate get their probabilities (next slide).
- 74% of the calls were *voice calls*. In order to achieve better resolution, they were further divided into 5 type (next slide).
- The resulting Operational profile would have $5 + 6 = 11$ types of operations, with probabilities ranging from 0.18 (18%) to 0.000001.

Note that the code needed for Failure recovery is executed only rarely.
Operational Profile Example

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>Failure recovery</td>
<td>0.000001</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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

- Most common and easiest to explain model.
- Notation:
  - Total expected faults detected by time $t$: $\mu(t)$
  - Failure intensity: fault detection rate $\lambda(t)$
  - Undetected defects present at time $t$: $N(t)$
- By definition, $\lambda(t)$ is derivative of $\mu(t)$.

\[
\lambda(t) = \frac{d}{dt} \mu(t)
\]

\[
= -\frac{d}{dt} N(t)
\]

Since faults found are no longer undetected
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

\[
\begin{align*}
- \frac{dN(t)}{dt} T_s &= k_s N(t) \\
- \frac{dN(t)}{dt} &= \frac{K}{T_L} N(t) = \beta_1 N(t)
\end{align*}
\]

Here we replace $K_s$ and $T_s$ by more convenient $K$ and $T_L$.

where $K = k_s \frac{T_L}{T_s}$ is fault exposure ratio
Exponential SRGM (cont.)

• We get

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

\[ \mu(t) = \beta_0 (1 - e^{-\beta_1 t}) \quad \lambda(t) = \beta_0 \beta_1 e^{-\beta_1 t} \]

• For \( t \to \infty \), total \( \beta_0 = N(0) \) 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.

The 2 equations contain the same information.
The plots show $\lambda(t)$ and $\mu(t)$ for $\beta_0=142$ and $\beta_1=3.5 \times 10^{-5}$. Note that $\mu(t)$ asymptotically approaches 142.
A Basic SRGM (cont.)

• Note that parameter $\beta_1$ is given by:

$$\beta_1 = \frac{K}{T_L} = \frac{K}{(S \cdot Q \cdot \frac{1}{r})}$$

• S: source instructions,
• Q: number of object instructions per source instruction typically between 2.5 to 6 (see page 7-13 of Software Reliability Handbook, sec 7)
• r: object instruction execution rate of the computer
• K: fault-exposure ratio, range 1 $10^{-7}$ to 10 $10^{-7}$, (t is in CPU seconds). Assumed constant here.
• Q, r and K should be relatively easy to estimate.
SRGM : “Logarithmic Poisson”

• Many SRGMs have been proposed.

• Another model **Logarithmic Poisson** model, by Musa-Okumoto, has been found to have a good predictive capability

\[
\mu(t) = \beta_0 \ln(1 + \beta_1 t) \quad \quad \lambda(t) = \frac{\beta_0 \beta_1}{1 + \beta_1 t}
\]

• Applicable as long as \( \mu(t) \leq N(0) \). Practically always satisfied. Term *infinite-faults-model* misleading.

• Parameters \( \beta_0 \) and \( \beta_1 \) don’t have a simple interpretation. An interpretation has been given by Malaiya and Denton ([What Do the Software Reliability Growth Model Parameters Represent?](https://www.cst.tum.de/~mala/)).
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 \(i\): \(D_i\)

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

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

• We used many datasets from diverse projects for comparing different models.
Comparing models

• Next slide shows the result of a comparison using test data from a number of diverse sources.
• The Logarithmic Poisson model is most accurate, the Exponential model is moderately accurate.
• Both the Logarithmic Poisson and the Exponential models tend to underestimate the number of defects that will eventually be found.
• Inverse Polynomial, Power and S-shaped models are not discussed here, you can find them in the literature.
Bias in SRGMs

- Malaiya, Karunanithi, Verma ('90)
Using an SRGM

• An SRGM can be used in two ways
  ▪ For preliminary planning, even before testing begins (provided you can estimate the parameters)
  ▪ During testing: You can fit the available test data to make projections.

• We’ll see examples of both next.
SRGM: Use for 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 $4 \times 10^{-7}$
  ▪ Task: Estimate the testing time needed for defect density 2.5/KLOC

• Procedure:
  ▪ Find $\beta_0$, $\beta_1$
  ▪ Find testing time $t_1$
SRGM: Preliminary Planning (cont.)

• From exponential model

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

\[ \beta_1 = \frac{K}{(S \cdot Q \cdot \frac{l}{r})} = \frac{4.0 \times 10^{-7}}{10,000 \times 2.5 \times \frac{1}{70 \times 10^6}} \]

\[ = 11.2 \times 10^{-4} \text{ per sec} \]
SRGM: Preliminary Planning (cont.)

• Reliability at release depends on

\[
\frac{N(t_1)}{N(O)} = \frac{2.5 \times 10}{25 \times 10} = \exp(-11.2 \times 10^{-4} \cdot t_1)
\]

\[
t_1 = \frac{-\ln(0.1)}{11.2 \times 10^{-4}} = 2056 \text{ sec. (CPU time)}
\]

\[
\lambda(t_1) = 250 \times 11.2 \times 10^{-4} e^{-11.2 \times 10^{-4} t_1}
\]

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

Note \(N(t_1)\) is the total number of defects at the end of testing, which is defect density size = 2.5/KLOC 10 KLOC
SRGM: Preliminary Planning (cont.)

• For the same environment, product $\beta_1 S$ is constant, since $\beta_1$ is inversely proportional to $S$. For example,
  - If for a prior 5 KLOC project $\beta_1$ was $2 \times 10^{-3}$ per sec.
  - Then for a 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 \times 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:
  \[ 2.78 \times 10^{-4} = 101.47 \times 5.22 \times 10^{-5} \times e^{-5.22 \times 10^{-5} \times t_f} \]
- Yielding \( t_f = 56473 \) sec., i.e. 15.69 hours
Example: SRGM with Test Data (cont.)

Figure 1: Using an SRGM

Failure intensity
target
measured values
Fitted
model
Failure intensity
Hours
0 5 10 15 20
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0
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
For further reading

• Software Reliability Assurance Handbook by Lakey and Neufelder