Surrogate Oracles, Generalized Dependency and Simpler Models

Larry Wilson
Old Dominion University
Supported by NAG-1-750

Abstract

Software reliability models require the sequence of interfailure times from the debugging process as input. We have previously illustrated that using data from replicated debugging could greatly improve reliability predictions. However, inexpensive replication of the debugging process requires the existence of a cheap, fast error detector. We can design laboratory experiments around a gold version which is used as an oracle or around an n-version error detector. Unfortunately, we can not expect software developers to have an oracle or to bear the expense of n-versions. We are investigating a generic technique for approximating replicated data by using the partially debugged software as a difference detector.

We believe that the failure rate of each fault has significant dependence on the presence or absence of other faults. Thus, in order to discuss a failure rate for a known fault, we need to specify the presence or absence of each of the other known faults.

Also, we are interested in simpler models which use shorter input sequences without sacrificing accuracy. In fact we, conjecture a possible gain in performance.

To investigate these propositions, we are using NASA computers running LIC (RTI) versions to generate data. This data will be used to label the debugging graph associated with each version. These labeled graphs will be used to test the utility of a surrogate oracle, to analyze the dependent nature of fault failure rates and to explore the feasibility of reliability models which use the data of only the most recent failures.
Normal Debugging

\[ P \quad P_1 \quad P_2 \quad \ldots \quad P_n \]

0 \quad T_1 \quad T_2 \quad \ldots \quad T_n \quad \text{time}

\[ t_i = T_i - T_{i-1} \quad \text{interfailure times} \]

Models take as input \((t_1, t_2, \ldots, t_n)\) and produce estimates of \(\lambda\)

\[ \text{INPUT DISTRIBUTION / RANDOM INPUTS} \]
\[ \text{ERROR DETECTOR} \]
• Conclusion 1: These SR models should never be used to predict software reliability if we only use normal debugging process.

• Conclusion 2: The models are stable after the randomness is removed by replicated debugging. With replication, conceivable to use models.

• Future:

1. GCS
2. Front end for repliability models.
3. Estimate and control the cost of replication.
4. Analyze debug graph to get better models.
Partial Debug Graph

V

V1

V12 V13 V1n

V2

V23 V2n

Vn

Vn-1,n

V12...n
Weighted Partial Debug Graph

\[ V \]

\[ W_1 \quad W_2 \quad \ldots \quad W_n \]

\[ V_1 \quad V_2 \quad \ldots \quad V_{n-1, n} \]

\[ V_{12} \quad V_{13} \quad V_{1n} \quad V_{23} \quad V_{2n} \]

\[ V_{12, \ldots, n} \]
\( \Delta \text{RPDG1} \)

\[ V \quad 42.344\% \]

\[ V_1 \quad 94.27 \quad V_2 \quad 43.80 \quad V_3 \quad 42.50 \quad V_4 \quad 42.90 \quad V_5 \quad 42.88 \quad V_6 \quad 42.72 \quad V_7 \quad 42.70 \quad V_8 \quad 42.40 \quad V_9 \quad 42.40 \quad V_{10} \quad 42.4 \]
RPDG 9

V_1..10
99.999%
\( \Delta \text{RPDG9} \)

\[
\begin{align*}
45.729 & \quad 97.254 & \quad 99.797 & \quad 98.618 & \quad 98.984 & \quad 99.847 & \quad 99.967 & \quad 99.998 & \quad 99.998 \\
V_{-1} & \quad V-2 & \quad V-3 & \quad V-4 & \quad V-5 & \quad V-6 & \quad V-7 & \quad V-8 & \quad V-9 & \quad V-1
\end{align*}
\]

\[
\begin{align*}
54\% & \quad 2.7\% & \quad 0.2\% & \quad 1.4\% & \quad 0.15\% & \quad 0.03\% & \quad 0.002\% & \quad 0.002\% & \quad 0.001\%
\end{align*}
\]

V1..10
99.999%
$\Delta$PDG9

$\Delta$SPDG

V-1 V-2 V-3 V-4 V-5 V-6 V-7 V-8 V-9 V-10

54.44\% 2.85\% 0.21\% 1.02\% ?? 0.99\% ?? 0.99\%
1. ANALYZE BUG DEPENDENCY
   FILL IN $\Delta$RPDG

2. NEW MODELS
   FILL IN ROWS 8 AND 9 $\Delta$SRPDG
   ANALYZE

3. REPEAT FOR LIC 3.C

4. CONTINUE WITH GCS DATA