

New Quality Estimations in Random Testing

S. Mankefors, R. Torkar and A. Boklund
ISSRE '03

10/1/04

1

Outline

- Introduction
- The Basics
- Implementation
- Analysis
- Results
- Conclusions

10/1/04

2

Testing Software

- NASA space shuttle avionics software
 - Fault density: 0.0001 defects per line of code
 - \$1000 per line of code
- Galileo spacecraft
 - Fault density: 0.0089 defects/LOC

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

10/1/04

3

Testing Software – Problem

- Quality of a test
- No “success theory”
- Is random testing an option?

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

10/1/04

4

Random Testing

- Using test cases with randomized input and comparing the output to a well-known correct answer

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

10/1/04

5

Why use random testing?

- Only samples a small fraction of all possible input
- A lot of important ranges of input could be missed completely

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

10/1/04

6

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Random Testing: Pros

- Easy and straightforward to implement
 - Oracle assumption
- Provides a mathematical basis for analysis
- Fast generation of test cases
 - Pseudo-random number generators
 - Large amount of easily created data

10/1/04 7

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Random Testing: Cons

- Actual input vs. random input
 - May not be a uniform distribution of input
 - All random input may not be applicable to the program
- Does not encompass illegal or extreme values
- Oracle assumption
 - Cannot test anything above unit-level

10/1/04 8

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Random Testing: Redemption?

- Offers testing where straightforward oracles are available
- In this paper: create quality estimations for random testing
 - Improved use and interpretation of a random test suite
- Strength of results ensured by using massive simulations encompassing hundreds of billions of tests

10/1/04 9

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

The Math

- Failure frequency for uniformly distributed random input:

$$\Theta_t = \frac{\text{number of failed test cases/}}{\text{total number of tests performed}}$$

10/1/04 10

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

The Math

- M failures, N tests
- Θ_g = guess of true failure frequency
- α = upper confidence bound

$$1 - \sum_{j=0}^M \binom{N}{j} \Theta_g^j (1 - \Theta_g)^{N-j} \geq \alpha$$

10/1/04 11

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

The Math

- A piece of software will either fail or succeed for a specific set of input, regardless of input type and software at hand
 - Non-random and repeatable process
 - Integrals

10/1/04 12

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

The Math

- Definition 1: A software S with regular input set X is defined to be all combinations of legal values with respect to the input variable types of S
- Definition 2: Each software S has a failure function F_s defined on the regular set X. $F_s(x) = 0$ for a given value x in X if no failures occur, and $F_s(x) = 1$ if a failure is detected when the software is executed

10/1/0413

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

The Math

Figure 3. The general problem of integration of a failure function F_s and the relation to testing.

10/1/0414

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

The Math

- How do we integrate an unknown function?
- Monte Carlo integration
- Central-limit theorem

10/1/0415

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

The Math

- If a software S with regular input set X and the associated failure function F_s exists and the number of test inputs is large enough, the total number of failures for S on X are given within k standard deviations by:

$$\int_x F_s dx \approx |X| (\langle F_s \rangle \pm k \sqrt{\frac{\langle F_s^2 \rangle - (\langle F_s \rangle)^2}{N}})$$

10/1/0416

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

The Math

- Reduction in complexity in evaluation of random testing
 - Simpler calculations
- Does not cover lower end of failure frequency
- Must extend estimations

10/1/0417

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

The Math

- If a software S with regular input set X and associated failure function F_s exists which has been probed by N test inputs, the total number of failures for S on X are given within (at least) k standard deviations by:

$$\int_x F_s dx \approx |X| \left(2k \sqrt{\frac{1}{N(N+1)} - \frac{1}{N(N+1)^2}} \right) \leq \frac{2k}{N}$$

10/1/0418

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Mutation Analysis

- Random testing is good on small “toy” programs, but what about software used in real life?
- Mutation analysis - seed errors in software that is known to be defect-free and well known in the software community in order to establish a common baseline for comparison

10/1/04 19

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Mutation Analysis

- Mathematical software
 - Widely used, trusted
 - Extensively tested and corrected
 - Modulus error program

10/1/04 20

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Basic Steps

- Seed artificial defects
- Calculate true (theoretical) failure rate
- Perform tests to find actual failure rate in “mutated” program
- Compare theoretical and actual results
- Repeat steps 2-5 with a new error frequency

10/1/04 21

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Modulus Error Program

```

i = abs(Mod(testinteger, ErrorFrequency))
i = i + ErrorFrequency/2.0

k1 = 0
If (i > ErrorFrequency) then
  k1 = 1
endif
_____oracle_____

k2 = 0
If (i >= ErrorFrequency) then
  k2 = 1
endif
  
```

10/1/04 22

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Testing – Modulus Error Program

- 10, 100, 1000 tests per suite
- Error frequency varied from 1/10 to 1/1500 in 1/10 increments
- 10 million trial runs
- 166.5 billion test executions

10/1/04 23

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Testing Larger Programs

- CISI – 70 lines of code, input: floats
- SVDCMP – 240 lines of code, input: 4x4 matrix of floats
- 2 kinds of defects
 - Parameter offsets
 - Mutated if statements
- 5 total defects introduced into each program

10/1/04 24

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Modulus Program Results

- Theoretical and actual failure rates correspond
- As number of observed failures approaches zero the upper quality bound becomes more likely to contain the true failure frequency

10/1/0425

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

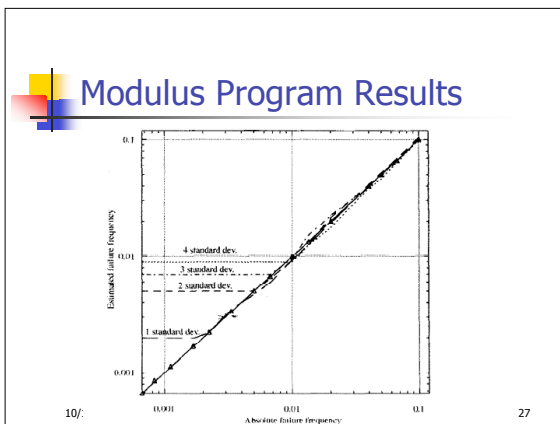
Modulus Program Results

- Even if the test suite is an usually “bad” sampling of the code it should still result in a quality estimation that stays within the upper quality boundary

$$\int_x F_s dx \approx |X| \left(2k \sqrt{\frac{1}{N(N+1)} - \frac{1}{N(N+1)^2}} \right) \leq \frac{2k}{N}$$

- k = 2, within 5%
- k = 3, within 0.3%
- k = 4, within 0.01%

10/1/0426



Introduction
The Basics
Implementation
Analysis
Results
Conclusions

CISI and SVDCMP Results

- Theoretical results agree with evaluated results in both cases
- Accuracy of estimation formula is independent of the complexity of the input, if the statistics are sufficient for integral evaluation

10/1/0428

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Conclusions

- Random testing offers a set of mathematical tools that are lacking in other testing approaches
- It is possible to transform the issue of random testing into an equivalent problem using integrals, Monte Carlo integration and the central limit theorem

10/1/0429

Introduction
The Basics
Implementation
Analysis
Results
Conclusions

Conclusions

- An upper bound on the quality of a test suite returning zero failures can be established, “success theory”
- There is still room for random testing in today’s testing practices

10/1/0430