Ph.D Thesis Defense

Coverage-Based Testing Strategies and Reliability Modeling for Fault-Tolerant Software Systems

Presented by: CAI Xia

Supervisor: Prof. Michael R. Lyu

August 24, 2006

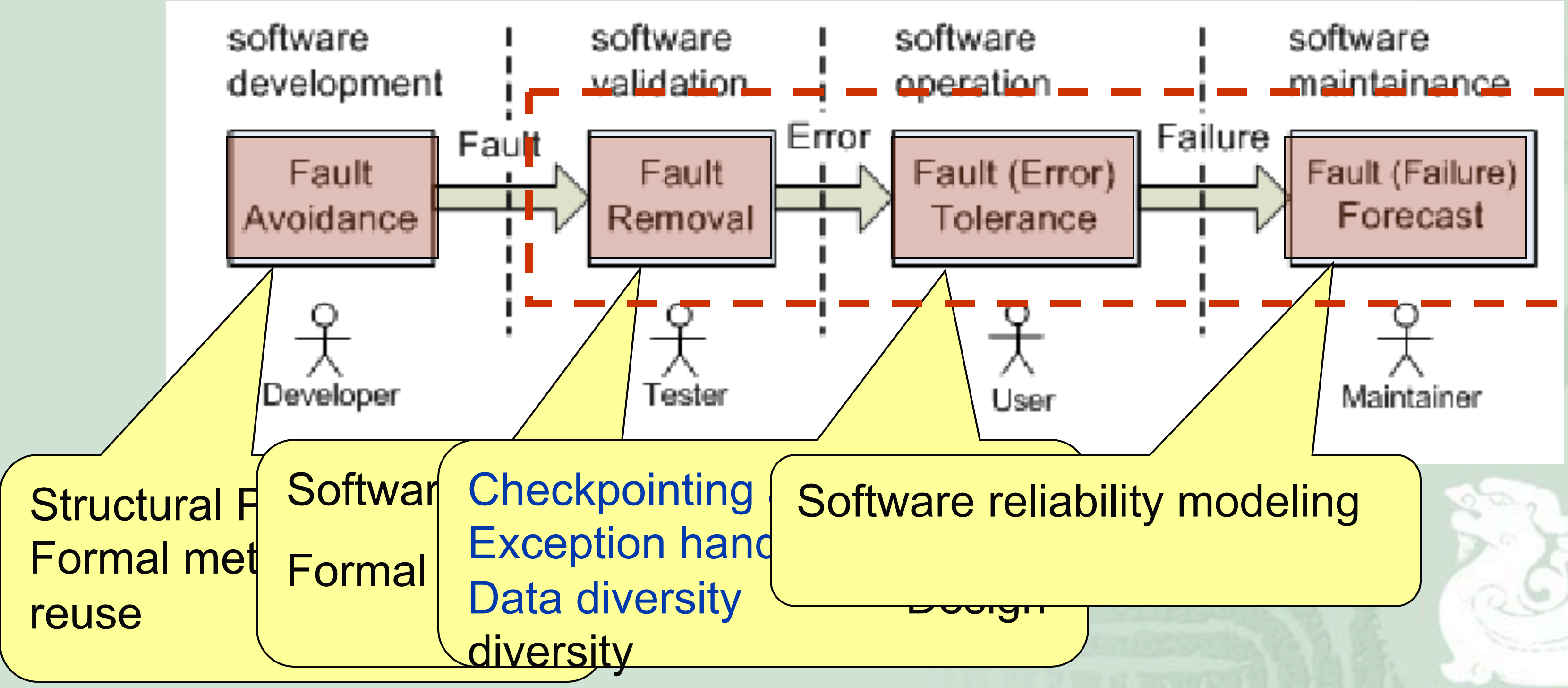
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Outline

- Background and related work
- Research methodology
- Experimental setup
- Evaluations on design diversity
- Coverage-based testing strategies
- Reliability modeling
- Conclusion and future work

Background

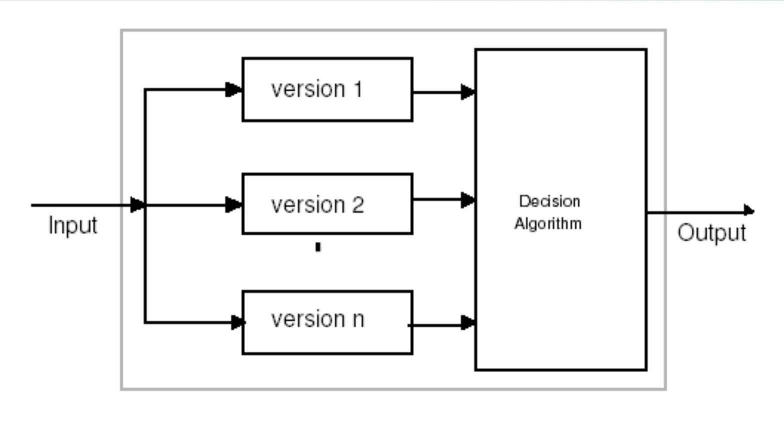
- Four technical methods to achieve reliable software systems



Fault-tolerant software

- Single-version technique
  - Checkpointing and recovery
  - Exception handling
- Multi-version technique (design diversity)
  - Recovery block (RB)
  - N-version programming (NVP)
  - N self-checking programming (NSCP)

NVP model



Design diversity

- Requirement
  - Same specification;
  - The multiple versions developed differently by independent teams;
  - No communications allowed between teams;
- Expectation
  - Programs built differently should fail differently
- Challenges
  - Cost consuming;
  - Correlated faults?

Experiments and evaluations

- Empirical and theoretical investigations have been conducted based on experiments, modeling, and evaluations
  - Knight and Leveson (1986), Kelly et al (1988), Eckhardt et al (1991), Lyu and He (1993)
  - Eckhardt and Lee (1985), Littlewood and Miller (1989), Popov et al. (2003)
  - Belli and Jedrzejowicz (1990), Littlewood. et al (2001), Teng and Pham (2002)
- No conclusive estimation can be made because of the size, population, complexity and comparability of these experiments

Software testing strategies

- Key issue
  - test case selection and evaluation
- Classifications
  - Functional testing (black-box testing)
    - Specification-based testing
  - Structural testing (white-box testing)
    - Branch testing
    - Data-flow coverage testing
  - Mutation testing
  - Random testing
- Comparison of different testing strategies:
  - Simulations
  - Formal analysis

Subdomain-based testing

Code coverage: measurement of testing completeness?

Code coverage

- Definition
  - measured as the fraction of program codes that are executed at least once during the test.

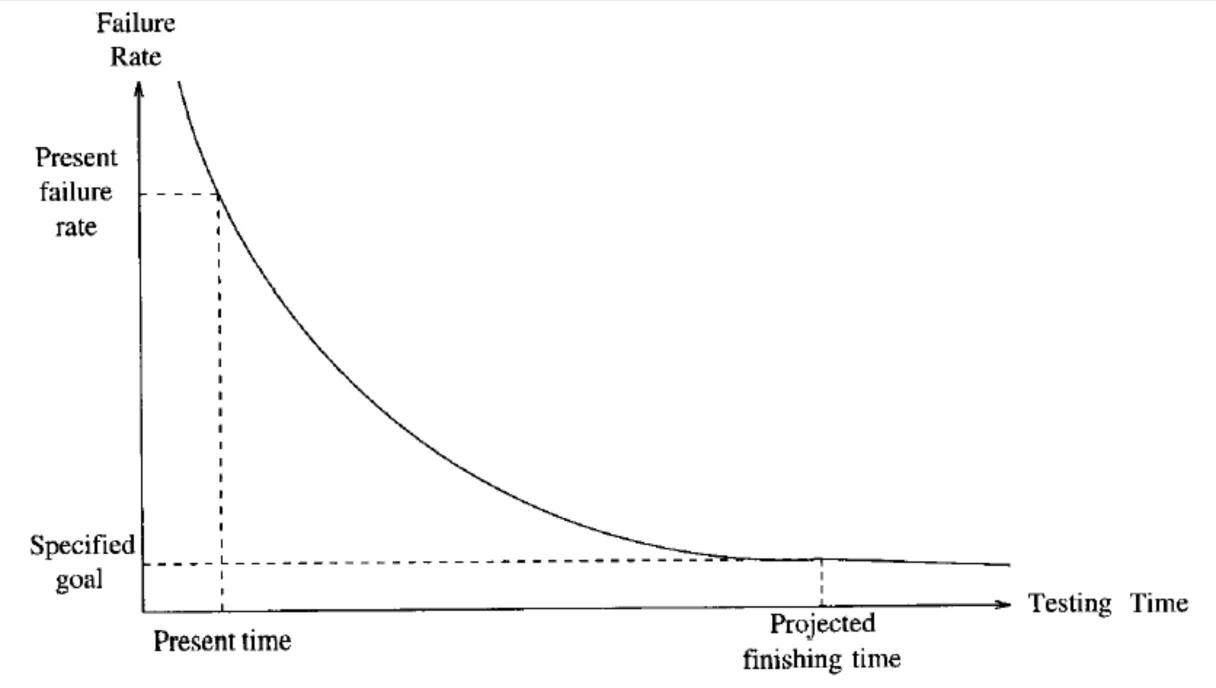
- Classification
  - *Block coverage*: the portion of basic blocks executed.
  - *Decision coverage*: the portion of decisions executed
  - *C-Use coverage*: computational uses of a variable.
  - *P-Use coverage*: predicate uses of a variable

Code coverage: an indicator of testing effectiveness?

- Positive evidence
  - *high code coverage brings high software reliability and low fault rate*
  - *both code coverage and fault detected in programs grow over time, as testing progresses.*
- Negative evidence
  - *Can this be attributed to causal dependency between code coverage and defect coverage?*
- Controversial, not conclusive

Software reliability growth modeling (SRGM)

- To model past failure data to predict future behavior



SRGM: some examples

- Nonhomogeneous Poisson Process (NHPP) model

$$\mu(t) = N(1 - e^{-bt})$$

- S-shaped reliability growth model

$$\mu(t) = \alpha[1 - (1 + \beta t)e^{-\beta t}]$$

- Musa-Okumoto Logarithmic Poisson model

$$\mu(t) = \beta_0(ln(\beta_1 t + 1))$$

μ(t) is the mean value of cumulative number of failure by time *t*

Reliability models for design diversity

- Echhardt and Lee (1985)
  - Variation of difficulty on demand space
  - Positive correlations between version failures
- Littlewood and Miller (1989)
  - Forced design diversity
  - Possibility of negative correlations
- Dugan and Lyu (1995)
  - Markov reward model
- Tomek and Trivedi (1995)
  - Stochastic reward net
- Popov, Strigini et al (2003)
  - Subdomains on demand space
  - Upper bounds and “likely” lower bounds for reliability

Our contributions

- For Fault Tolerance:
  - Assess the effectiveness of design diversity
- For Fault Removal:
  - Establish the relationship between fault coverage and code coverage under various testing strategies
- For Fault Forecast:
  - Propose a new reliability model which incorporates code coverage and testing time together

Outline

- Background and related work
- Research methodology
- Experimental setup
- Evaluations on design diversity
- Coverage-based testing strategies
- Reliability modeling
- Conclusion and future work

Motivation

- Fault-tolerant software
  - A necessity
  - Yet controversial
- Lack of
  - Conclusive assessment
  - creditable reliability model
  - effective testing strategy
  - Real-world project data on testing and fault tolerance techniques together

Research procedure and methodology

- A comprehensive and systematic approach
  - Modeling
  - Experimentation
  - Evaluation
  - Economics
- Modeling
  - Formulate the relationship between testing and reliability achievement
  - Propose our own reliability models with the key attributes

Research procedure and methodology

- Experimentation
  - Obtain new real-world fault-tolerant empirical data with coverage testing and mutation testing
- Evaluation
  - Collect statistical data for the effectiveness of design diversity
  - Evaluate existing reliability models for design diversity;
  - Investigate the effect of code coverage;
- Economics
  - Perform a tradeoff study on testing and fault tolerance

Outline

- Background and related work
- Research methodology
- Experimental setup
- Evaluations on design diversity
- Coverage-based testing strategies
- Reliability modeling
- Conclusion and future work

Project features

- Complicated and real-world application
- Large population of program versions
- Controlled development process
- Mutation testing with real faults injection
- Well-defined acceptance test set

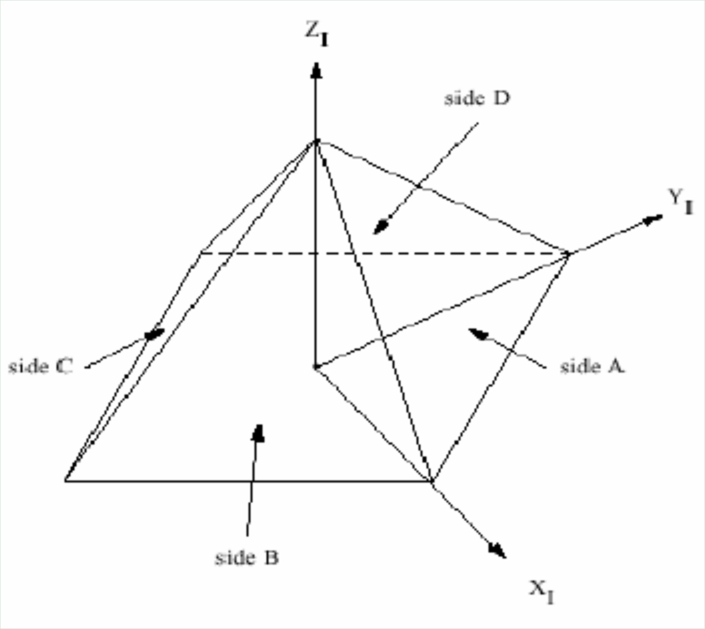
Experimental setup

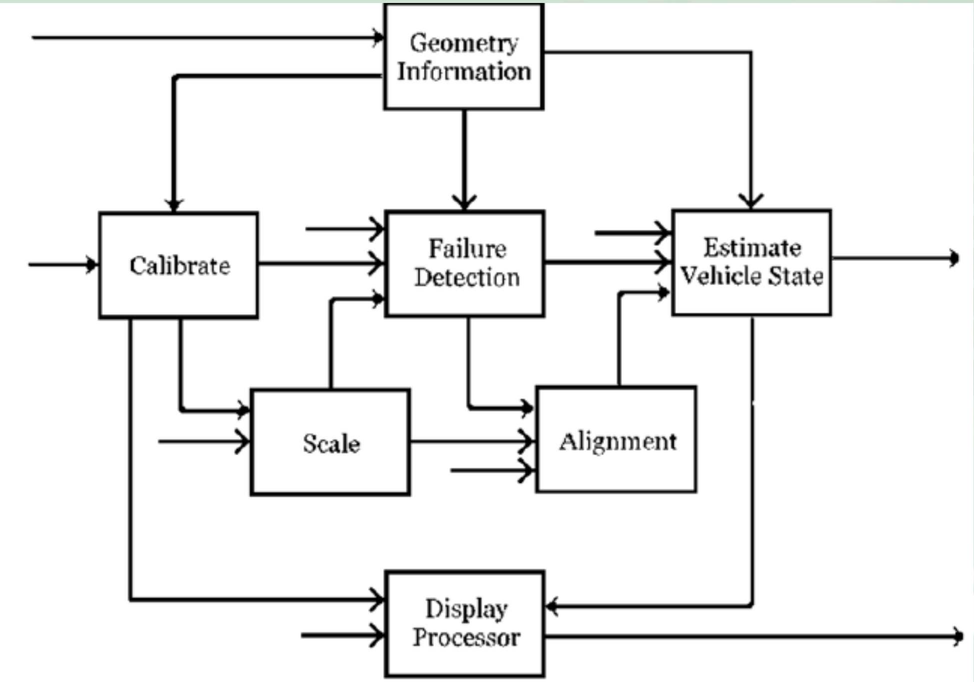
- *Time*: spring of 2002
- *Population*: 34 teams of four members
- *Application*: a critical avionics application
- *Duration*: a 12-week long project
- *Developers*: senior-level undergraduate students with computer science major
- *Place*: CUHK

Experimental project description

Redundant Strapped-Down Inertial Measurement Unit (RSDIMU)

- Geometry
- Data flow diagram





Software development procedure

1. Initial design document (3 weeks)
2. Final design document (3 weeks)
3. Initial code (1.5 weeks)
4. Code passing unit test (2 weeks)
5. Code passing integration test (1 weeks)
6. Code passing acceptance test (1.5 weeks)

Mutant creation

- Revision control applied and code changes analyzed
- Mutants created by injecting real faults identified during each development stage
- Each mutant containing one design or programming fault
- 426 mutants created for 21 program versions

Program metrics

Id	Lines	Modules	Functions	Blocks	Decisions	C-Use	P-Use	Mutants
01	1628	9	70	1327	606	1012	1384	25
02	2361	11	37	1592	809	2022	1714	21
03	2331	8	51	1081	548	899	1070	17
04	1749	7	39	1183	647	646	1339	24
05	2623	7	40	2460	960	2434	1853	26
07	2918	11	35	2686	917	2815	1792	19
08	2154	9	57	1429	585	1470	1293	17
09	2161	9	56	1663	666	2022	1979	20
12	2559	8	46	1308	551	1204	1201	31
15	1849	8	47	1736	732	1645	1448	29
17	1768	9	58	1310	655	1014	1328	17
18	2177	6	69	1635	686	1138	1251	10
20	1807	9	60	1531	782	1512	1735	18
22	3253	7	68	2403	1076	2907	2335	23
24	2131	8	90	1890	706	1586	1805	9
26	4512	20	45	2144	1238	2404	4461	22
27	1455	9	21	1327	622	1114	1364	15
29	1627	8	43	1710	506	1539	833	24
31	1914	12	24	1601	827	1075	1617	23
32	1919	8	41	1807	974	1649	2132	20
33	2022	7	27	1880	1009	2574	2887	16
Average	2234.2	9.0	48.8	1700.1	766.8	1651.5	1753.4	Total: 426

Setup of evaluation test

- ATAC tool employed to analyze the compare testing coverage
- 1200 test cases exercised as acceptance test
- All failures analyzed, code coverage measured, and cross-mutant failure results compared
- 60 Sun machines running Solaris involved with 30 hours one cycle and a total of 1.6 million files around 20GB generated
- 1M test cases in operational test

Outline

- Background and related work
- Research methodology
- Experimental setup
- Evaluations on design diversity
- Coverage-based testing strategies
- Reliability modeling
- Conclusion and future work

Static analysis result (1)

Fault types	Number	Percentage
Assign/Init:	136	31%
Function/Class/Object:	144	33%
Algorithm/Method:	81	19%
Checking:	60	14%
Interface/OO Messages	5	1%

Fault Type Distribution

Qualifier	Number	Percentage
Incorrect:	267	63%
Missing:	141	33%
Extraneous:	18	4%

Qualifier Distribution

Static analysis result (2)

Stage	Number	Percentage
Init Code	237	55.6%
Unit Test	120	28.2%
Integration Test	31	7.3%
Acceptance Test	38	8.9%

Development Stage Distribution

Lines	Number	Percentage
1 line:	116	27.23%
2-5 lines:	130	30.52%
6-10 lines:	61	14.32%
11-20 lines:	43	10.09%
21-50 lines:	53	12.44%
>51 lines:	23	5.40%
Average	11.39	

Fault Effect Code Lines

Mutants relationship

- Related mutants:
  - same success/failure 1200-bit binary string
- Similar mutants:
  - same binary string with the same erroneous output variables
- Exact mutants:
  - same binary string with same values of erroneous output variables

Relationship	Number of pairs	Percentage
Related mutants	1067	1.18%
Similar mutants	38	0.042%
Exact mutants	13	0.014%

Total pairs: 90525

Cross project comparison

Features	NASA 4-University project	Our experiment
Commonality		
1.same specification	initial version (faults involved)	mature version
2.similar development duration	10 weeks	12 weeks
3.similar development process	training, design, coding, testing, preliminary acceptance test	initial design, final design, initial code, unit test, integration test, acceptance test
4.same testing process	acceptance test, certification test, operational test	unit test, integration test, acceptance test, operational test
5.same operational test environment (i.e., determined by the same generator)	1196 test cases for certification test	1200 test cases for acceptance test
Difference		
1.Time (17 year apart)	1985	2002
2.Programming Team	2-person	4-person
3.Programmer experience	graduate students	undergraduate students
4.Programmer background	U.S.	Hong Kong
5.Language	Pascal	C

Cross project comparison

- NASA 4-university project: 7 out of 20 versions passed the operational testing
- Coincident failures were found among 2 to 8 versions
- 5 of the 7 related faults were not observed in our project

Item	Our project	NASA 4-University project
no. of test cases	100,000	920,746
failure probability	0.00139	0.06881
number of faults	6	7
fault density	1 per 10,000 lines	1.8 per 10,000 lines
2-version coincident failures	57	21173
3 or more version coincident failures	0	372
3-version improvement	900 to 20,000 times	80 to 330 times

Major contributions or findings on fault tolerance

- Real-world mutation data for design diversity

- A major empirical study in this field with substantial coverage and fault data

- Supportive evidence for design diversity
  - Remarkable reliability improvement (10^2 to 10^4)
  - Low probability of fault correlation

Outline

- Background and related work
- Research methodology
- Experimental setup
- Evaluations on design diversity
- Coverage-based testing strategies
- Reliability modeling
- Conclusion and future work

Research questions

- Is code coverage a positive indicator for fault detection capability?
- Does such effect vary under different testing strategies and profiles?
- Does any such effect vary with different code coverage metrics?

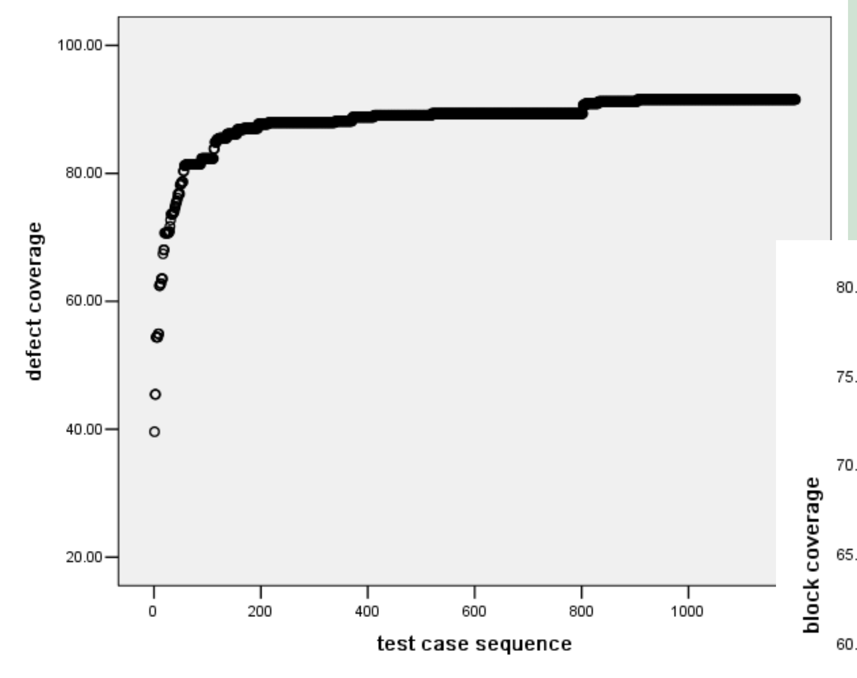
Fault detection related to changes of test coverage

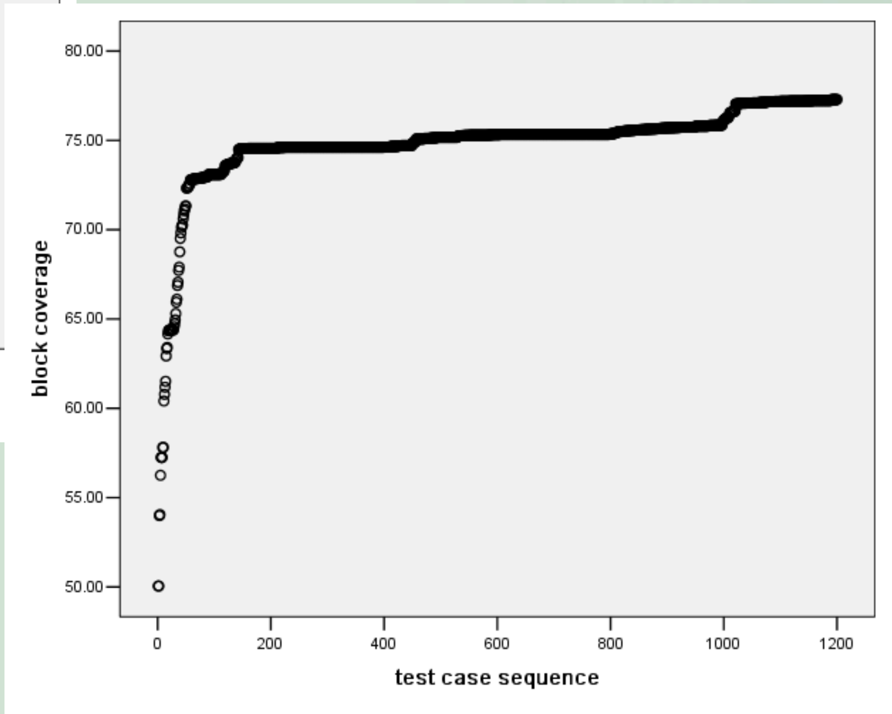
Version ID	Blocks	Decisions	C-Use	P-Use	Any
1	6/8	6/8	6/8	7/8	7/8 (87.5%)
2	9/14	9/14	9/14	10/14	10/14 (71.4%)
3	4/7	4/7	3/7	4/7	4/7 (57.1%)
4	7/11	8/11	8/11	8/11	8/11 (72.5%)
5	7/10	7/10	5/10	7/10	7/10 (70%)
7	5/10	5/10	5/10	5/10	5/10 (50%)
8	1/5	2/5	2/5	2/5	2/5 (40%)
9	7/9	[illegible]	[illegible]	7/9	7/9 (77.8%)
12	[illegible]	[illegible]	[illegible]	17/20	18/20 (90%)
15	[illegible]	[illegible]	[illegible]	[illegible]	6/11 (54.5%)
17	[illegible]	[illegible]	[illegible]	[illegible]	5/7 (71.4%)
18	[illegible]	[illegible]	[illegible]	5/6	5/6 (83.3%)
20	9/11	[illegible]	8/11	10/11	10/11 (90.9%)
22	12/13	12/13	12/13	12/13	12/13 (92.3%)
24	5/7	5/7	5/7	5/7	5/7 (71.4%)
26	2/12	[illegible]	[illegible]	4/12	4/12 (33.3%)
27	4/7	[illegible]	[illegible]	5/7	5/7 (71.4%)
29	10/18	[illegible]	[illegible]	10/18	12/18 (66.7%)
31	7/11	7/11	[illegible]	7/11	8/11 (72.7%)
32	3/7	4/7	5/7	5/7	5/7 (71.4%)
33	7/13	7/13	9/13	10/13	10/13 (76.9%)
Overall	131/217 (60.4%)	145/217 (66.8%)	137/217 (63.1%)	152/217 (70%)	155/217 (71.4%)

Coverage increase => more faults detected!

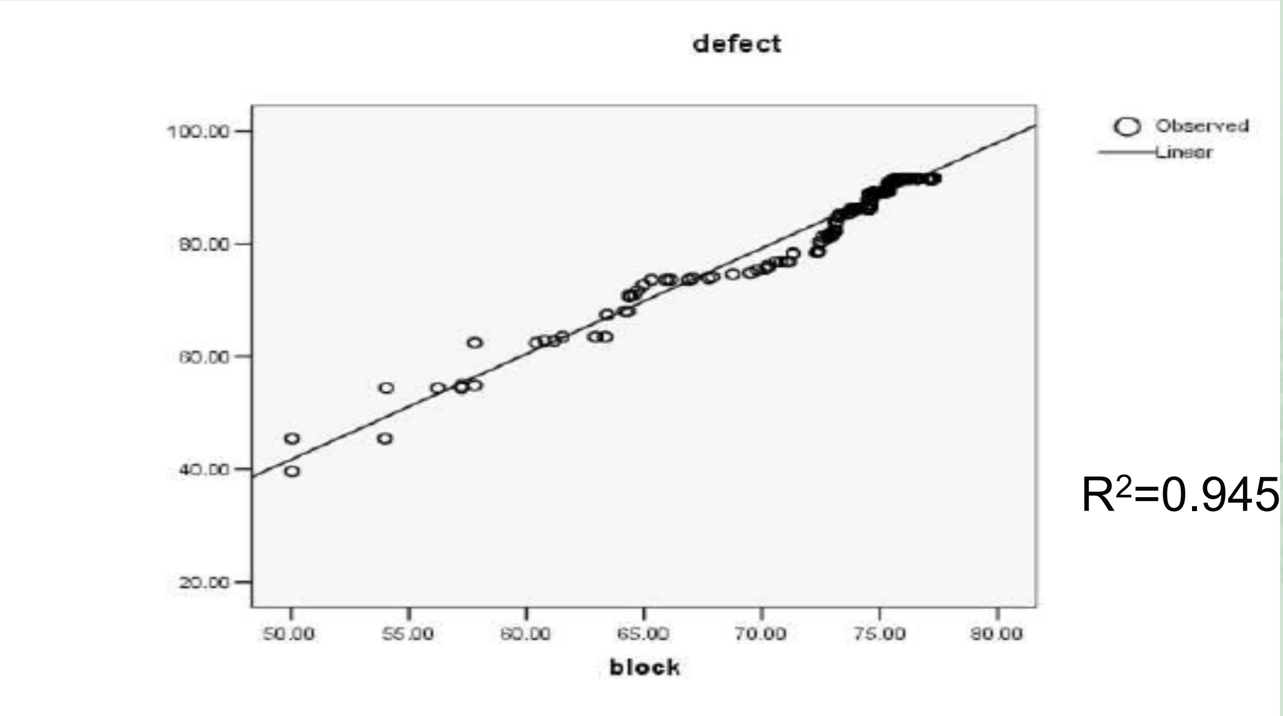
426
-174
- 35
= 217

Cumulated defect/block coverage





Cumulated defect coverage versus block coverage



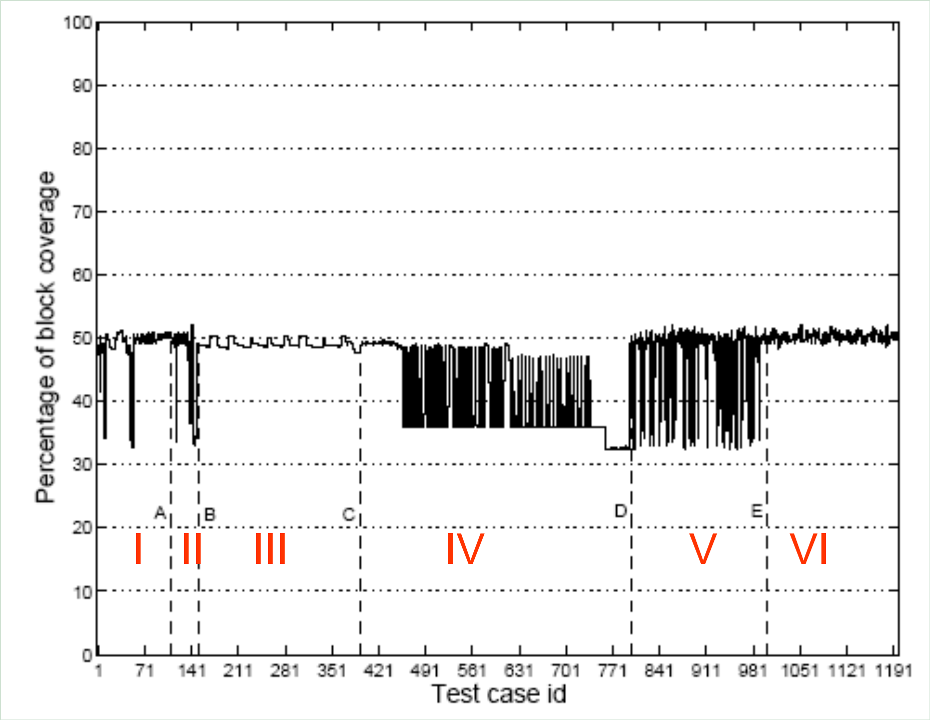
R^2=0.945

Test cases description

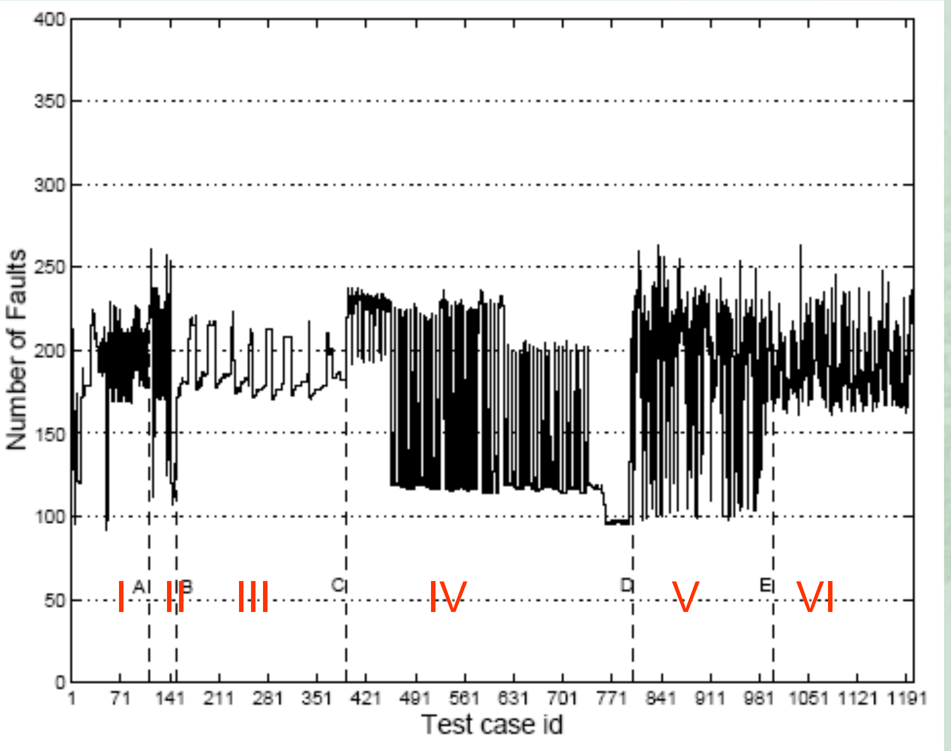
I	1	A fundamental test case to test basic functions.
I	2-7	Test cases checking vote control in different order.
I	8	General test case based on test case 1 with different display mode.
I	9-19	Test varying valid and boundary display mode.
I	20-27	Test cases for lower order bits.
I	28-52	Test cases for display and sensor failure.
I	53-85	Test random display mode and noise in calibration.
I	87-110	Test correct use of variable and sensitivity of the calibration procedure.
II	86, 111-149	Test on input, noise and edge vector failures.
II	150-151	Test various and large angle value.
III	152-392	Test cases checking for the minimal sensor noise levels for failure declaration.
IV	393-800	Test cases with various combinations of sensors failed on input and up to one additional sensor failed in the edge vector test.
V	801-1000	Random test cases. Initial random seed for 1st 100 cases is: 777, for 2nd 100 cases is: 1234567890
VI	1001-1200	Random test cases. Initial random seed is: 987654321 for 200 cases.

Block coverage vs. fault coverage

- **Test case contribution on block coverage**



- **Test case contribution on fault coverage**

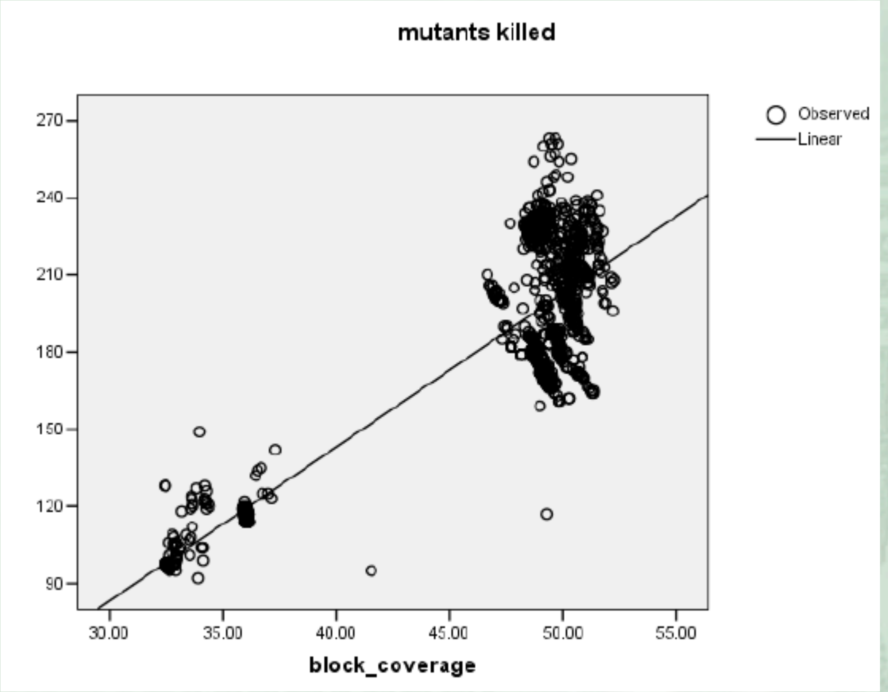


Correlation between block coverage and fault coverage

- Linear modeling fitness in various test case regions

Test case region	R-square
Overall (1-1200)	0.781
Region I (1-111)	0.634
Region II (112-151)	0.724
Region III (152-392)	0.672
Region IV (393-800)	0.981
Region V (801-1000)	0.778
Region VI (1001-1200)	0.189

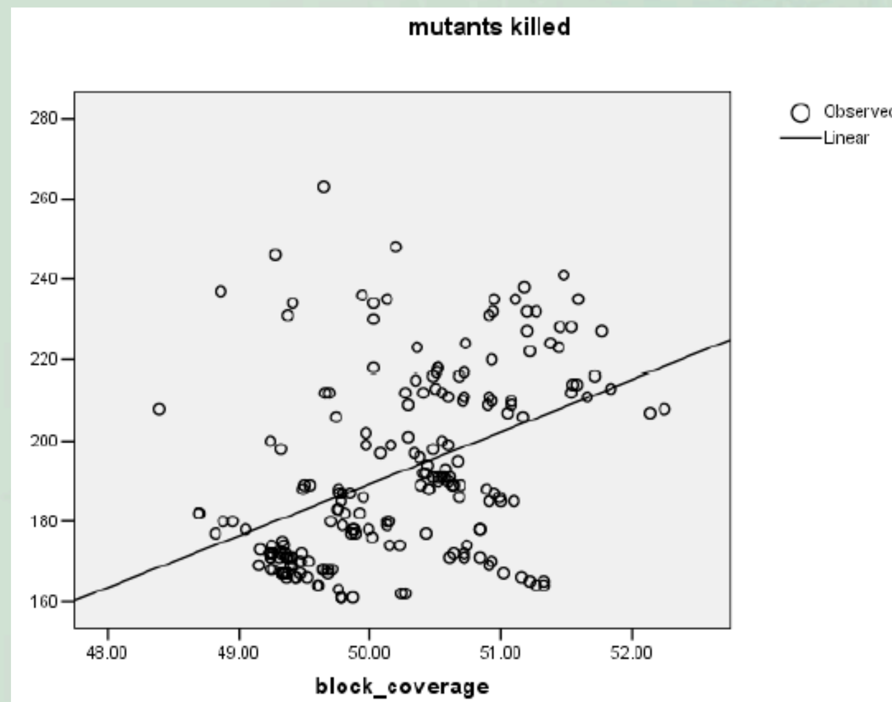
- Linear regression relationship between block coverage and defect coverage in the whole test set



The correlation at various test regions

- Linear regression relationship between block coverage and defect coverage in Region IV

- Linear regression relationship between block coverage and defect coverage in Region VI



Under various testing strategies

Testing profile (size)	R-squared
Whole test set(1200)	0.781
Functional test(800)	0.837
Random test(400)	0.558
Normal test(827)	0.045
Exceptional test(373)	0.944

Testing Combination	R-Squared
random & normal	0.045
random & exceptional	0.949
functional & normal	0.076
functional & exceptional	0.950

- Functional test: 1-800
- Random test: 801-1200
- Normal test: the system is operational according to the spec
- Exceptional test: the system is under severe stress conditions.

With different coverage metrics

Testing profile(size)	block coverage	decision coverage	C-use	P-use
Whole test set(1200)	0.781	0.832	0.774	0.834
Functional test(800)	0.837	0.880	0.830	0.881
Random test(400)	0.558	0.646	0.547	0.648
Normal test(827)	0.045	0.368	0.019	0.398
Exceptional test(373)	0.944	0.952	0.954	0.954

- The correlations under decision, C-use and P-use are similar with that of block coverage

Answers to the research questions

- Is code coverage a positive indicator for fault detection capability?
  - Yes.
- Does such effect vary under different testing strategies and profiles?
  - Yes. The effect is highest with exceptional test cases, while lowest with normal test cases.
- Does any such effect vary with different code coverage metrics?
  - Not obvious with our experimental data.

Major contributions or findings on software testing

- High correlation between fault coverage and code coverage in exceptional test cases
  - Give guidelines for design of exceptional test cases
- This is the first time that such correlation has been investigated under various testing strategies

Outline

- Background and related work
- Research methodology
- Experimental setup
- Evaluations on design diversity
- Coverage-based testing strategies
- Reliability modeling
- Conclusion and future work

Work on reliability modeling

- Evaluate current probability reliability models for design diversity with our experimental data
- Propose a new reliability model which incorporates test coverage measurement into traditional software growth model

Results of PS Model with our project data

- Popov, Strigini et al (2003)

Table 5. Upper bounds on the joint pfds under Demand Profiles

Pair		$P_{117\_90\%}$	$P_{305\_90\%}$	$\min(P_{117\_90\%}, P_{305\_90\%})$	$P_{117,305_{upper90\%}}$
(117, 305)	DP1	0.0146	0.0332	0.0146	0.0146
	DP2	0.0400	0.0626	0.0400	0.0386
	DP3	0.0715	0.0796	0.0715	0.0644
	DP4	0.0483	0.0562	0.0483	0.0379
		$P_{215\_90\%}$	$P_{382\_90\%}$	$\min(P_{215\_90\%}, P_{382\_90\%})$	$P_{215,382_{upper90\%}}$
(215, 382)	DP1	0.0146	0.0149	0.0146	0.0146
	DP2	0.0429	0.0672	0.0429	0.0429
	DP3	0.0709	0.1091	0.0709	0.0709
	DP4	0.0415	0.0656	0.0415	0.0415
		$P_{382\_90\%}$	$P_{403\_90\%}$	$\min(P_{382\_90\%}, P_{403\_90\%})$	$P_{382,403_{upper90\%}}$
(382, 403)	DP1	0.0149	0.0146	0.0146	0.0146
	DP2	0.0672	0.0400	0.0400	0.0391
	DP3	0.1091	0.0715	0.0715	0.0670
	DP4	0.0656	0.0483	0.0483	0.0417

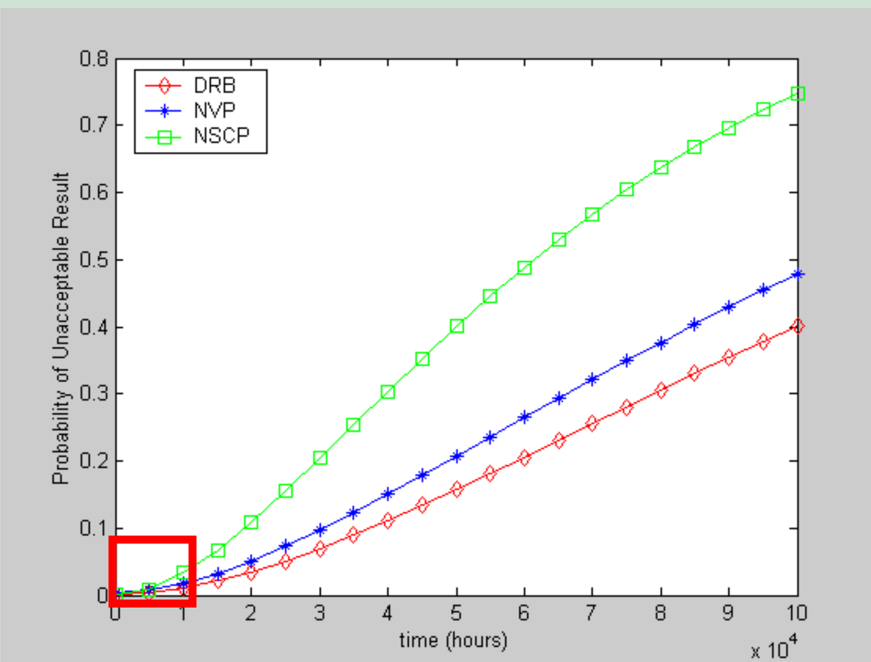
Results of PS Model with our project data

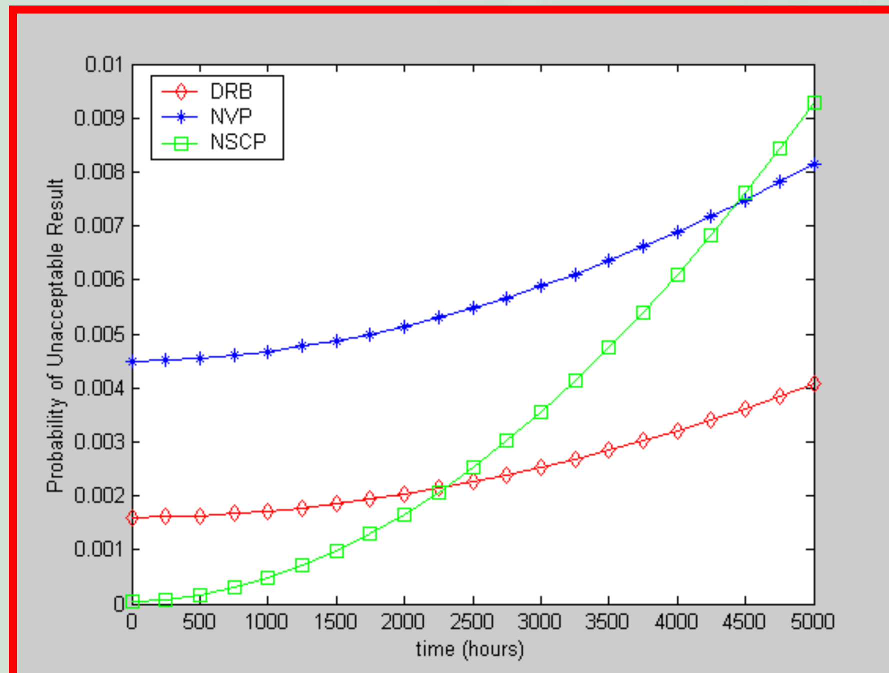
Table 7. Lower bounds on the joint pfds under Demand Profiles

Pair		$P_{117\_10\%}$	$P_{305\_10\%}$	$cov(S_{117\_10\%}, S_{305\_10\%})$	$P_{117\_10\%}P_{305\_10\%}$	$P_{117,305_{sub\_ind10\%}}$
	DP1	$6.73 \cdot 10^{-4}$	$6.91 \cdot 10^{-3}$	$3.86 \cdot 10^{-8}$	$4.65 \cdot 10^{-6}$	$\boxed{4.69 \cdot 10^{-6}}$
(117,	DP2	$2.37 \cdot 10^{-3}$	$9.62 \cdot 10^{-3}$	$-4.26 \cdot 10^{-6}$	$2.28 \cdot 10^{-5}$	$1.86 \cdot 10^{-5}$
305)	DP3	$5.87 \cdot 10^{-3}$	$8.02 \cdot 10^{-3}$	$-1.80 \cdot 10^{-5}$	$4.71 \cdot 10^{-5}$	$2.91 \cdot 10^{-5}$
	DP4	$5.87 \cdot 10^{-3}$	$7.79 \cdot 10^{-3}$	$-2.47 \cdot 10^{-5}$	$4.57 \cdot 10^{-5}$	$2.09 \cdot 10^{-5}$
		$P_{215\_10\%}$	$P_{382\_10\%}$	$cov(S_{215\_10\%}, S_{382\_10\%})$	$P_{215\_10\%}P_{382\_10\%}$	$P_{215,382_{sub\_ind10\%}}$
	DP1	$6.80 \cdot 10^{-4}$	$8.27 \cdot 10^{-4}$	$2.39 \cdot 10^{-6}$	$5.26 \cdot 10^{-7}$	$\boxed{2.95 \cdot 10^{-6}}$
(215,	DP2	$3.05 \cdot 10^{-3}$	$1.78 \cdot 10^{-2}$	$1.98 \cdot 10^{-4}$	$5.43 \cdot 10^{-5}$	$\boxed{2.52 \cdot 10^{-4}}$
382)	DP3	$4.95 \cdot 10^{-3}$	$2.76 \cdot 10^{-2}$	$2.50 \cdot 10^{-4}$	$1.37 \cdot 10^{-4}$	$\boxed{3.86 \cdot 10^{-4}}$
	DP4	$2.83 \cdot 10^{-3}$	$1.63 \cdot 10^{-2}$	$1.70 \cdot 10^{-4}$	$4.62 \cdot 10^{-5}$	$\boxed{2.16 \cdot 10^{-4}}$
		$P_{382\_10\%}$	$P_{403\_10\%}$	$cov(S_{382\_10\%}, S_{403\_10\%})$	$P_{215\_10\%}P_{382\_10\%}$	$P_{382,403_{sub\_ind10\%}}$
	DP1	$8.27 \cdot 10^{-4}$	$6.73 \cdot 10^{-4}$	$4.62 \cdot 10^{-7}$	$5.57 \cdot 10^{-7}$	$\boxed{1.02 \cdot 10^{-6}}$
(382,	DP2	$1.78 \cdot 10^{-2}$	$2.37 \cdot 10^{-3}$	$1.61 \cdot 10^{-5}$	$4.23 \cdot 10^{-5}$	$\boxed{5.84 \cdot 10^{-5}}$
403)	DP3	$2.76 \cdot 10^{-2}$	$5.87 \cdot 10^{-3}$	$-4.86 \cdot 10^{-5}$	$1.62 \cdot 10^{-4}$	$1.13 \cdot 10^{-4}$
	DP4	$1.63 \cdot 10^{-2}$	$5.86 \cdot 10^{-3}$	$-1.16 \cdot 10^{-5}$	$9.56 \cdot 10^{-5}$	$8.40 \cdot 10^{-5}$

Results of DL model with our project data

- Dugan and Lyu (1995)
- Predicted reliability by different configurations
- The result is consistent with previous study





Introducing coverage into software reliability modeling

- Most traditional software reliability models are based on time domain
- However, time may not be the only factor that affects the failure behavior of software
- Test completeness may be another indicator for software reliability

A new reliability model

- Assumptions:

1. The number of failures revealed in testing is related to not only the execution time, but also the code coverage achieved;
2. The failure rate with respect to time and test coverage together is a parameterized summation of those with respect to time or coverage alone;
3. The probabilities of failure with respect to time and coverage are not independent, they affect each other by an exponential rate.

Model form

$$\lambda(t,c) \;=\; \alpha_1\gamma_1 e^{-\gamma_1 c}\lambda_1(t) + \alpha_2\gamma_2 e^{-\gamma_2 t}\lambda_2(c)$$

joint failure
intensity func

failure intensity
function w

Dependency
factors

failure intensity function
coverage

- $\lambda_2(c)$: failure intensity func respect to coverage
- α_1,γ_1, α_2, γ_2: parameters with the constraint of

  $\alpha_1 + \alpha_2 = 1$

Estimation methods

- Method A:
  - Select a model for $\lambda_1(t)$ and $\lambda_2(c)$;
  - Estimate the parameters in$\lambda_1(t)$ and $\lambda_2(c)$ independently;
  - Optimize other four parameters afterwards.

Existing reliability models: NHPP, S-shaped, logarithmic, Weibull …

???

- M
  - nd $\lambda_2(c)$;
  - Optimize all parameters together.
- Least-squares estimation (LSE) employed

λ(c) : Modeling defect coverage and code coverage

- A Hyper-exponential model

$$F_c = \sum_{i=1}^{K} N_i(1 - e^{-\beta_i c})$$

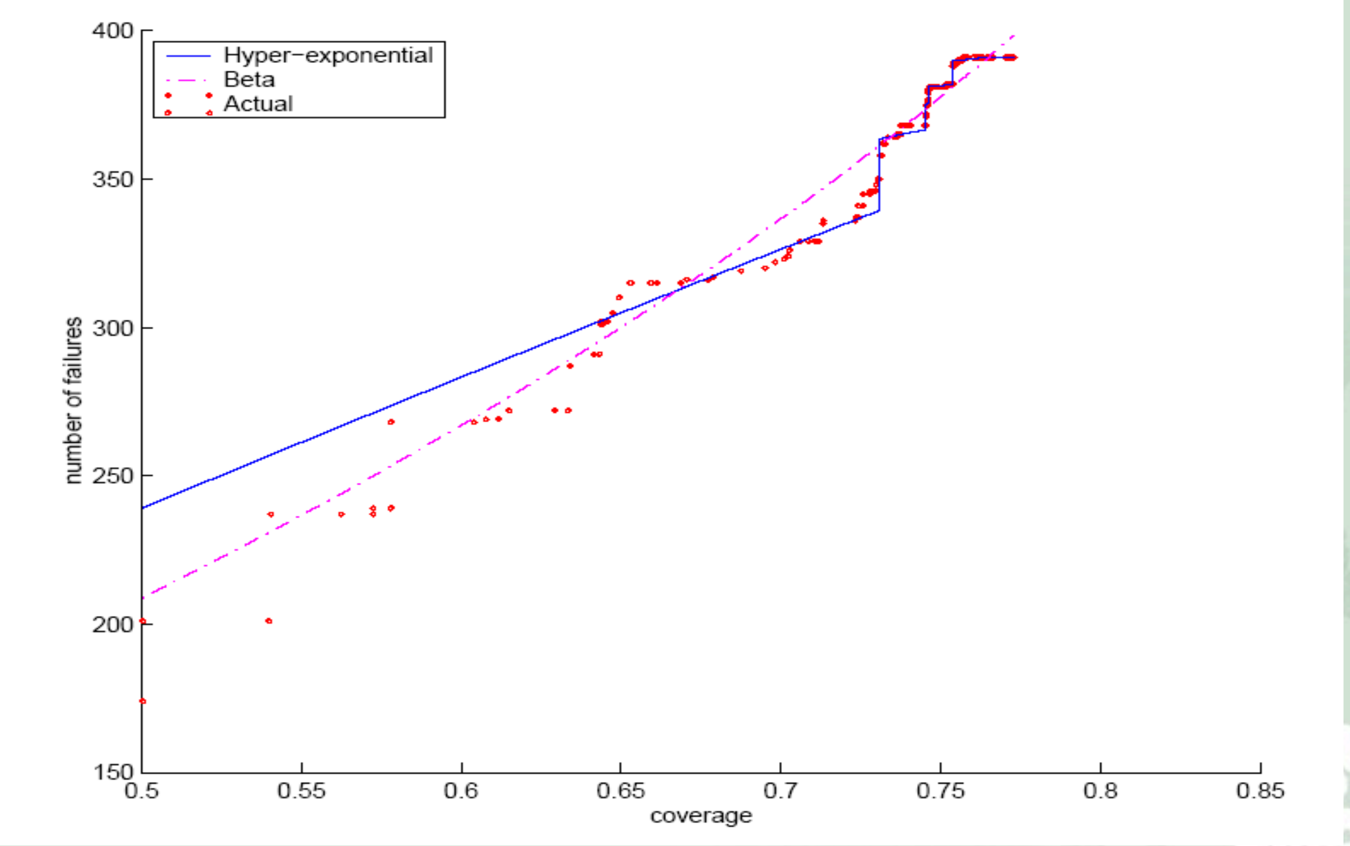
  - F_c: cumulated number of failures when coverage c is achieved
  - K: number of classes of testing strategies;
  - N_i: the expected number of faults detected eventually in each class

- A Beta model

$$F_c = N_1[1 - (1 - \frac{c}{N_2})^{\alpha}]$$

  - N_1: the expected number of faults detected eventually
  - N_2: the ultimate test coverage

λ(c) : Experimental evaluation



λ(c) : Parameters estimation results

- Hyper-exponential model
- Beta model

$$F_c = 1101 \times [1 - (1 - c^{0.303}]$$

SSE=38365

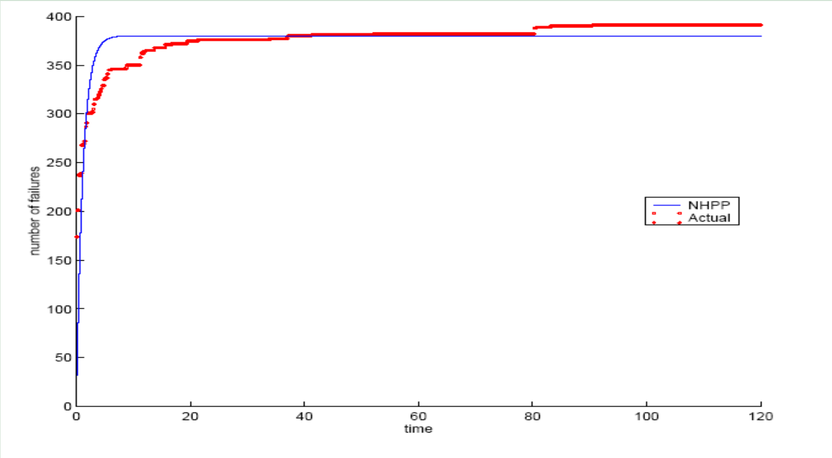
Modeling	N	β	SSE
NHPP (1)	1475	0.39	146110
NHPP (2)	5467	0.096	118200
Hyper-exponential	4087	–	23928
Region I	1989	0.256	22195
Region II	476	1.97	133
Region III	411	3.29	1315
Region IV	406	3.75	66
Region V	414	3.77	219
Region VI	391	21.3	1.01e-009

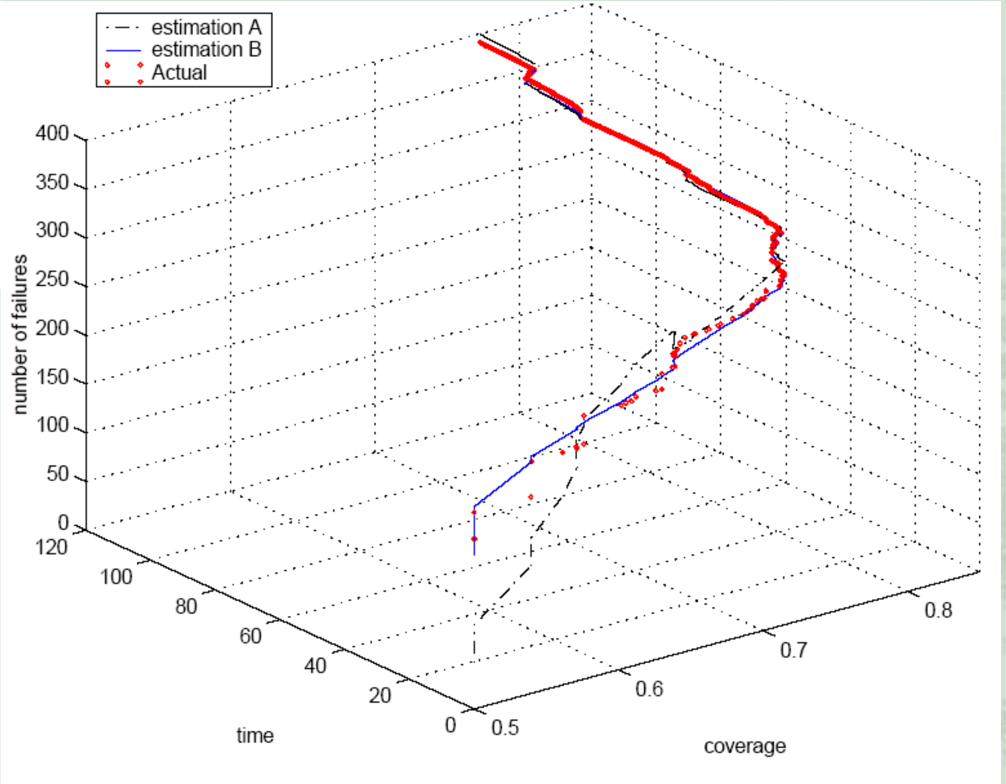
Parameter estimation (1)

Method	α_1	γ_1	N_1	β_1	γ_2	N_2	β_2	SSE
A	-1.3844	3.0819	380	0.87	1.5110	1475	0.39	93849
B	1.7713	0.824	380	11.716	0.121	1475	-0.082	14130
NHPP Model	-	-	380	0.87	-	-	-	279230

- $\lambda_1(t)$, $\lambda_2(c)$: exponential (NHPP)
- NHPP model: original SRGM

Prediction accuracy (1)



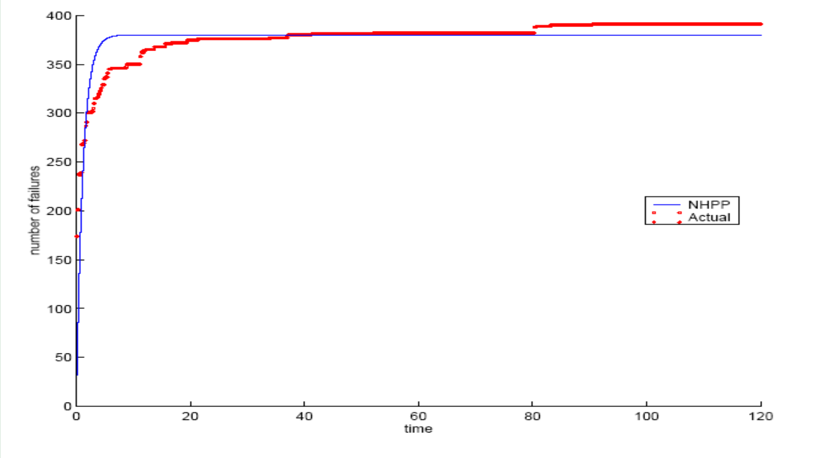


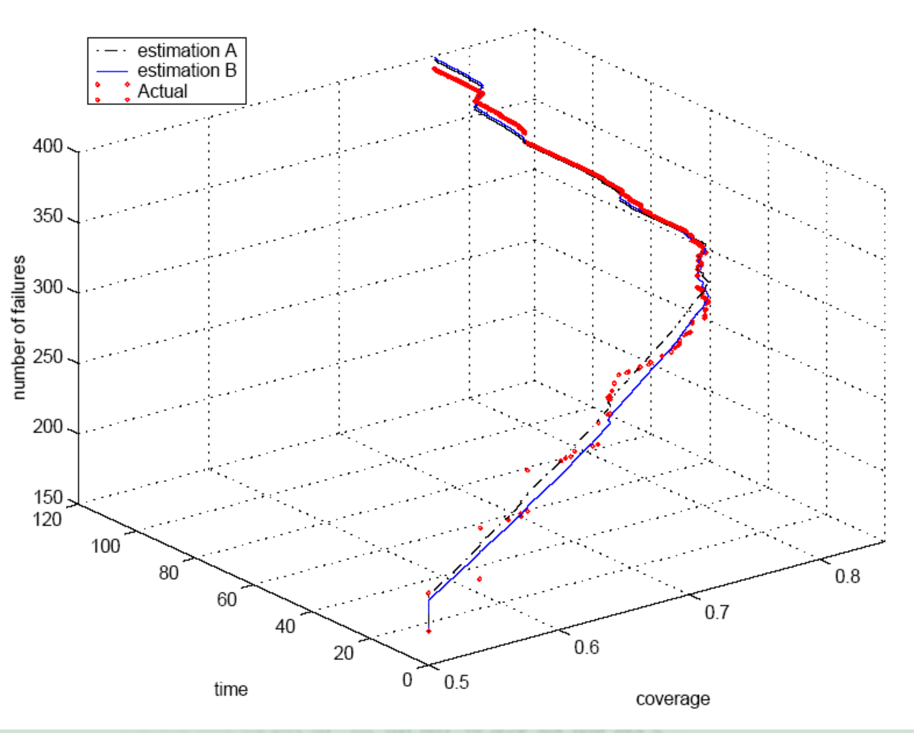
Parameter estimation (2)

Method	α_1	γ_1	N_1	β_1	γ_2	N_2	β_2	SSE
A	0.0407	16.097	380	0.87	19.516	1101	0.303	36825
B	0.0565	20.182	380	0.098	21.138	1101	0.305	25712
NHPP Model	-	-	380	0.87	-	-	-	279230

- $\lambda_1(t)$: NHPP
- $\lambda_2(c)$: Beta model

Estimation accuracy (2)





Department of Computer Science and Engineering
The Chinese University of Hong Kong

Major contributions or findings on software reliability modeling

- The first reliability model which combines the effect of testing time and code coverage together
- The new reliability model outperforms traditional NHPP model in terms of estimation accuracy

Outline

- Background and related work
- Research methodology
- Experimental setup
- Evaluations on design diversity
- Coverage-based testing strategies
- Reliability modeling
- Conclusion and future work

Conclusion

- Propose a new software reliability modeling
  - Incorporate code coverage into traditional software reliability growth models
  - Achieve better accuracy than the traditional NHPP model

The first reliability model combining the effect of testing time and code coverage together

Conclusion

- Assess multi-version fault-tolerant software with supportive evidence by a large-scale experiment
  - High reliability improvement
  - Low fault correlation
  - Stable performance

A major empirical study in this field with substantial fault and coverage data

Conclusion

- Evaluate the effectiveness of coverage-based testing strategies:
  - Code coverage is a reasonably positive indicator for fault detection capability
  - The effect is remarkable under exceptional testing profile

The first evaluation looking into different categories of testing strategies

Future work

- Further evaluate the current reliability model using comparisons with existing reliability models other than NHPP
- Consider other formulations about the relationship between fault coverage and test coverage
- Further study on the economical tradeoff between software testing and fault tolerance

Publication list

- Journal papers and book chapters
  - **Xia Cai**, Michael R. Lyu and Kam-Fai Wong, *A Generic Environment for COTS Testing and Quality Prediction*, Testing Commercial-off-the-shelf Components and Systems, Sami Beydeda and Volker Gruhn (eds.), Springer-Verlag, Berlin, 2005, pp.315-347.
  - Michael R. Lyu and **Xia Cai**, *Fault-tolerant Software*, To appear in Encyclopedia on Computer Science and Engineering, Benjamin Wah (ed.), Wiley. .
  - **Xia Cai**, Michael R. Lyu, *An Experimental Evaluation of the Effect of Code Coverage on Fault Detection*, Submitted to IEEE Transactions on Software Engineering, June 2006.
  - **Xia Cai**, Michael R. Lyu, Mladen A. Vouk, *Reliability Features for Design Diversity :Cross Project Evaluations and Comparisons*, in preparation.
  - **Xia Cai**, Michael R. Lyu, *Predicting Software Reliability with Test Coverage*, in preparation.

Publication list

- Conference papers
  - Michael R. Lyu, Zubin Huang, Sam K. S. Sze and **Xia Cai**, “An Empirical Study on Testing and Fault Tolerance for Software Reliability Engineering,” Proceedings of the 14th IEEE International Symposium on Software Reliability Engineering (ISSRE'2003), Denver, Colorado, Nov. 2003, pp.119-130. This paper received the ISSRE'2003 Best Paper Award.
  - **Xia Cai** and Michael R. Lyu, “An Empirical Study on Reliability and Fault Correlation Models for Diverse Software Systems,” ISSRE’2004, Saint-Malo, France, Nov. 2004, pp.125-136.
  - **Xia Cai** and Michael R. Lyu, “The Effect of Code Coverage on Fault Detection under Different Testing Profiles,” ICSE 2005 Workshop on Advances in Model-Based Software Testing (A-MOST), St. Louis, Missouri, May 2005.
  - **Xia Cai**, Michael R. Lyu and Mladen A. Vouk, “An Experimental Evaluation on Reliability Features of N-Version Programming,” ISSRE’2005, Chicago, Illinois, Nov. 8-11, 2005, pp. 161-170.
  - **Xia Cai** and Michael R. Lyu, “Predicting Software Reliability with Testing Coverage Information,” In preparation to International Conference on Software Engineering (ICSE’2007).

Q & A

Thanks!

Previous work on modeling reliability with coverage information

- Vouk (1992)
  - Rayleigh model

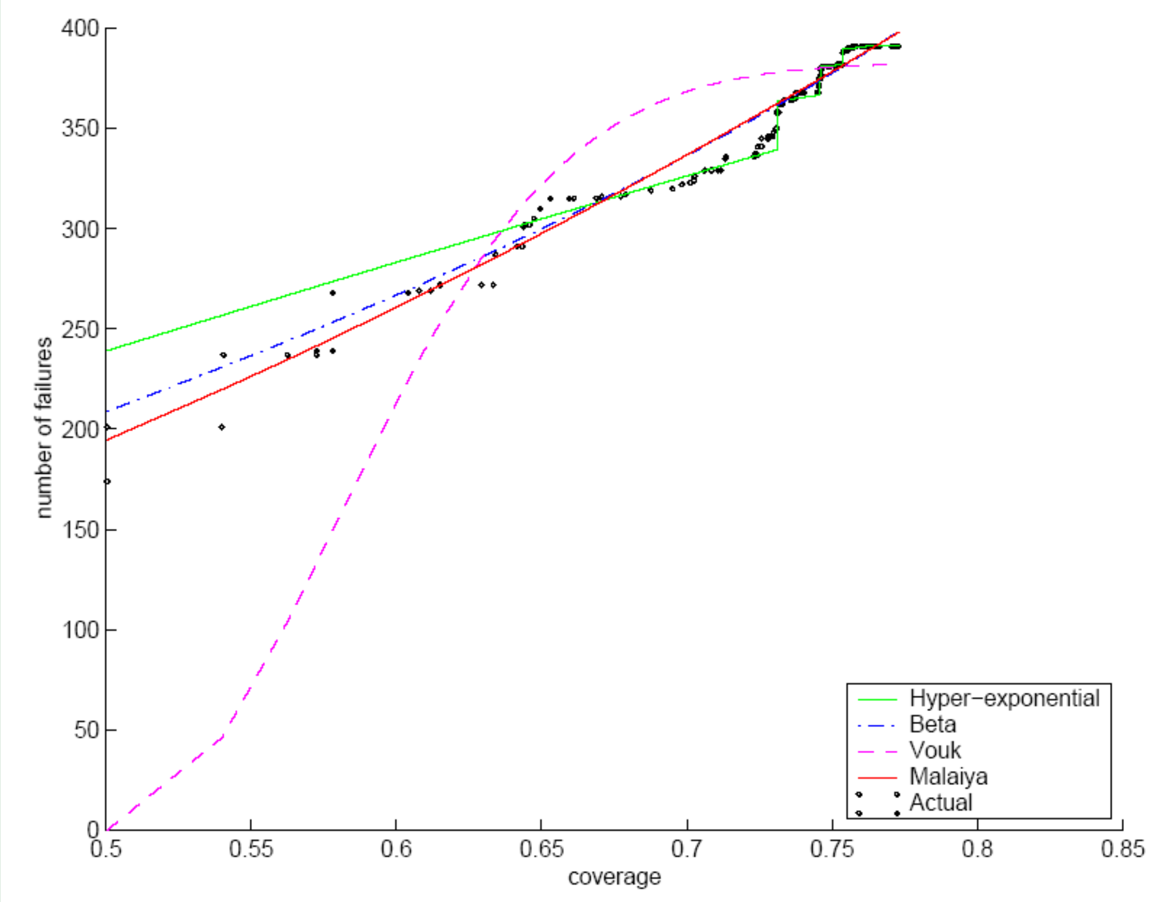
$$F_c = N[1 - e^{-\beta(c - c_{min})^2}]$$

- Malaiya et al.(2002)
  - Logarithmic-exponential model

$$F_c = \alpha_0 \cdot log[1 + \alpha_1 \cdot (e^{\alpha_2 \cdot c} - 1)]$$

- Chen et al. (2001)
  - Using code coverage as a factor to reduce the execution time in reliability models

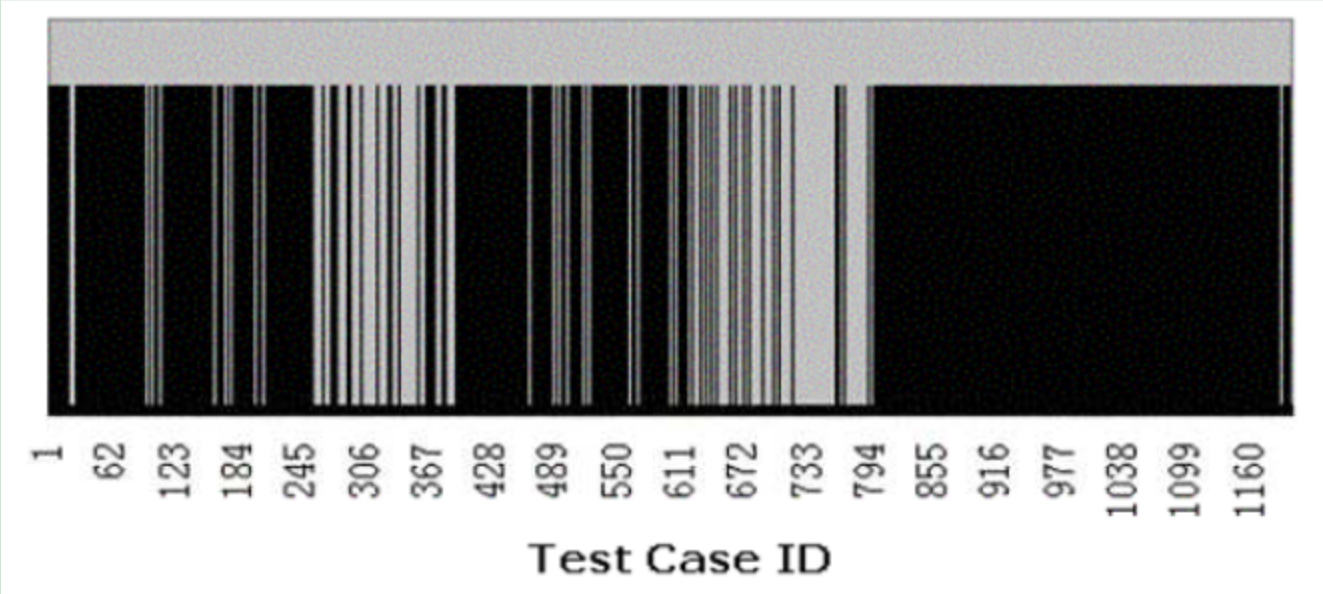
Comparisons with previous estimations



- The number of mutants failing in different testing

Test case type	Mutants killed	Mean failure number	Std. deviation
Functional testing	20/382	4.50	3.606
Random testing	9/371	3.67	2.236
Normal testing	36/371	120.00	221.309
Exceptional testing	20/355	55.05	99.518

Non-redundant set of test cases



Test set reduction with normal testing

Criteria	test set size	mutants killed	mutants killed by random set
original	827	371	—
block inc.>0.01%	87	351	353 (100.6%)
block inc.>0.05%	59	346	348 (100.6%)
block inc.> 0.25%	28	341	334 (97.9%)
block inc.>1%	11	308	304 (98.7%)
block inc.>2%	8	303	292 (96.4%)

Test set reduction with exceptional testing

Criteria	test set size	mutants killed	mutants killed by random set
original	373	355	—
block inc.>0.01%	29	327	298 (91.1%)
block inc.>0.05%	19	316	277 (87.7%)
block inc.>0.25%	12	270	243 (90.0%)
block inc.>1%	7	238	216 (90.8%)
block inc.>2%	3	228	180 (78.9%)