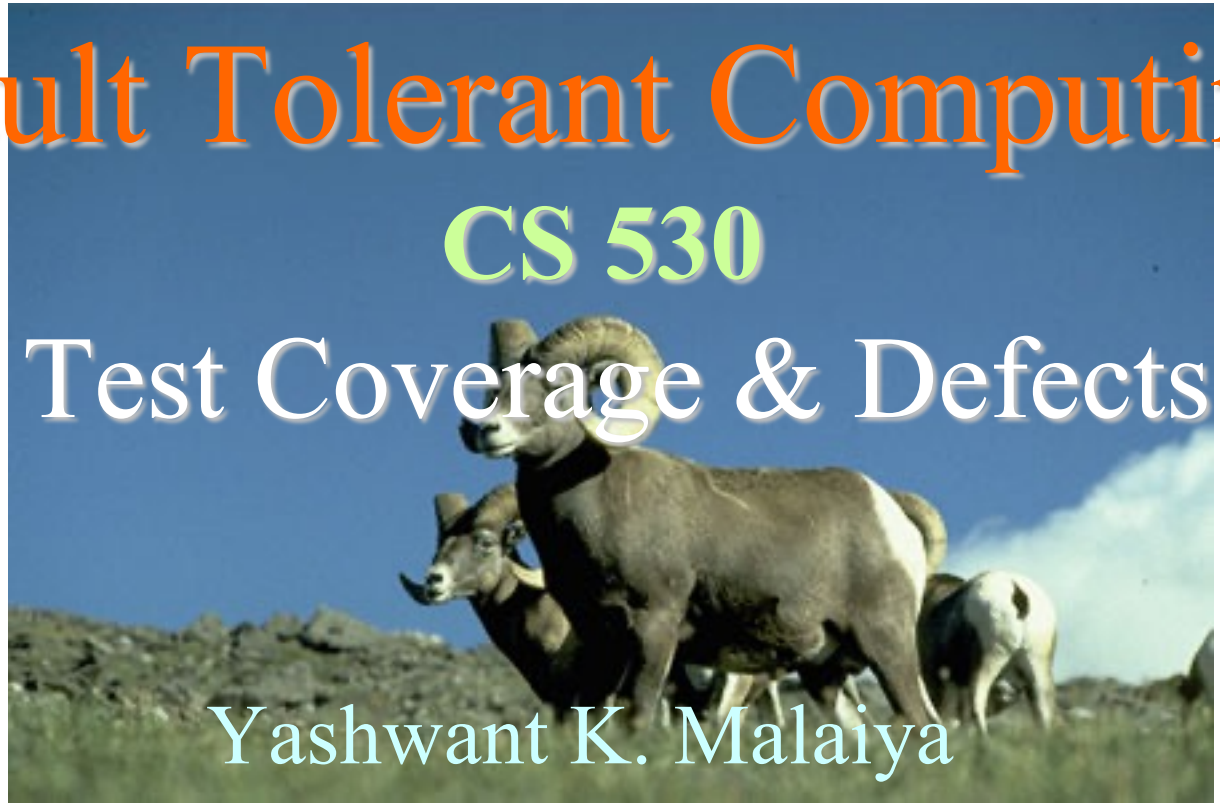
Fault Tolerant Computing

CS 530

Test Coverage & Defects



Yashwant K. Malaiya

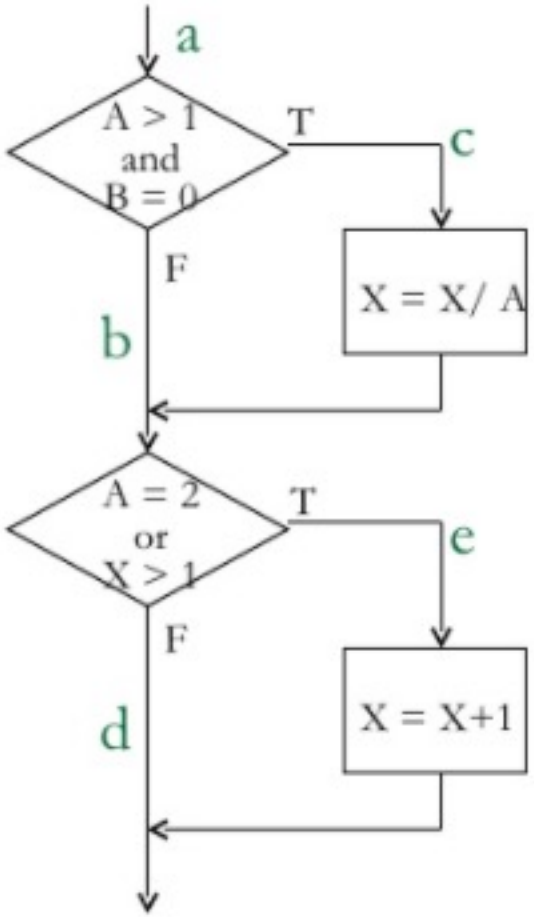
Colorado State University

Test Coverage Measures

- Structural:
  - Statement or Block coverage
  - Branch or decision coverage
- Data-flow:
  - P-use coverage: p-use pair: variable defined/modified - use as predicate
  - C-use coverage: similar -use for computation
- Subsumption hierarchy:
  - Covering *all branches* cover *all statements*
  - Covering *all p-uses* cover *all branches*

Test Coverage Measures

- Test case A =2, B=0, X=4
  - Covers branches a, c, e
  - Covers all the statements
- Test case A =1, B=1, X=1
  - Covers branches a, b, d
- Two test cases for 100% branch coverage.



Coverage Tools

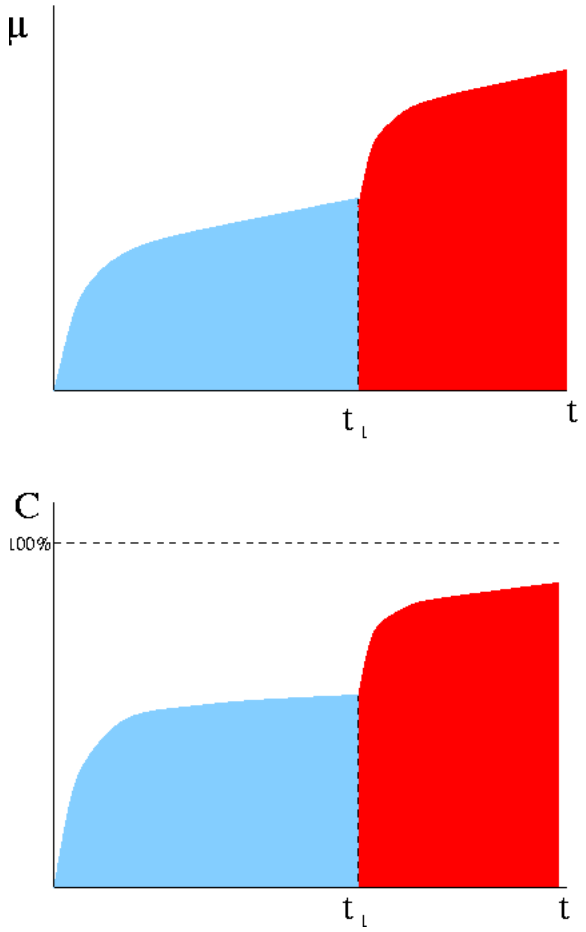
- There are several code coverage tools: Jcov, Gcov etc. for Java, C/C++ etc.
  - Compilation using the tool, instruments the compiled code to collect metrics covered.
  - Coverage metrics:
    - Statements/Blocks
    - Branches/Edges
    - Paths
    - Methods/Functions
    - Data-flow coverage metrics
  - Subsumption hierarchy
    - Complete Path coverage => 100% Branch coverage
    - Complete Branch coverage => 100% Statement coverage

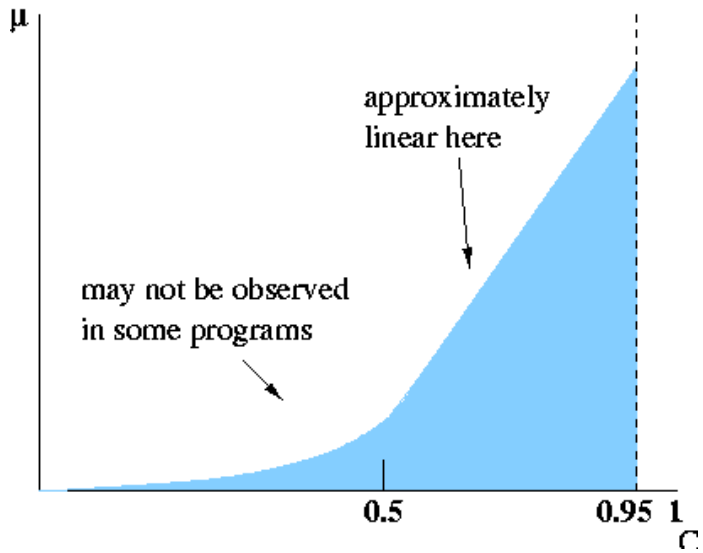
Assumptions:
- A fault is associated with one or more elements.
- Exercising the element may trigger the fault to create an error
- Complete coverage does not guarantee finding all the faults.

Colorado State University

S. C. Ntafos, "A comparison of some structural testing strategies," in *IEEE Trans. on Software Engineering*, vol. 14, no. 6, pp. 868-874, June 1988, doi: 10.1109/32.6165.

Modeling : Defects, Time, & Coverage





Malaiya, Li, Bieman, Karcich, Skibbe, 1994

Li, Malaiya, Denton, 1998

Coverage Based Defect Estimation

- Coverage is an objective measure of testing
  - Directly related to test effectiveness
  - Independent of processor speed and testing efficiency
- Lower defect density requires higher coverage to find more faults
- Once we start finding faults, expect coverage vs. defect growth to be linear

Y. K. Malaiya, Naixin Li, J. Bieman, R. Karcich and B. Skibbe, "The relationship between test coverage and reliability," *Proc IEEE Int Symp on Software Reliability Eng*, 1994, pp. 186-195.
Y. K. Malaiya and J. Denton, "Estimating the number of residual defects," *Proc Third IEEE Int High-Assurance Systems Eng Symp* 1998, pp. 98-105.

Logarithmic-Exponential Coverage Model

- Hypothesis 1: defect coverage growth follows logarithmic model

$$C^0(t) = \frac{\beta_0^0}{N^0}\ln(1+\beta_1^0 t), \quad C^0(t) \le 1$$

- Hypothesis 2: test coverage growth follows logarithmic model

$$C^i(t) = \frac{\beta_0^i}{N^i}\ln(1+\beta_1^i t), \quad C^i(t) \le 1$$

Log-Expo Coverage Model (2)

- Eliminating t and rearranging,

$$C^0 = a_0^i \ln[1 + a_1^i(\exp(a_2^i C^i) - 1)], \quad C^0 \le 1$$

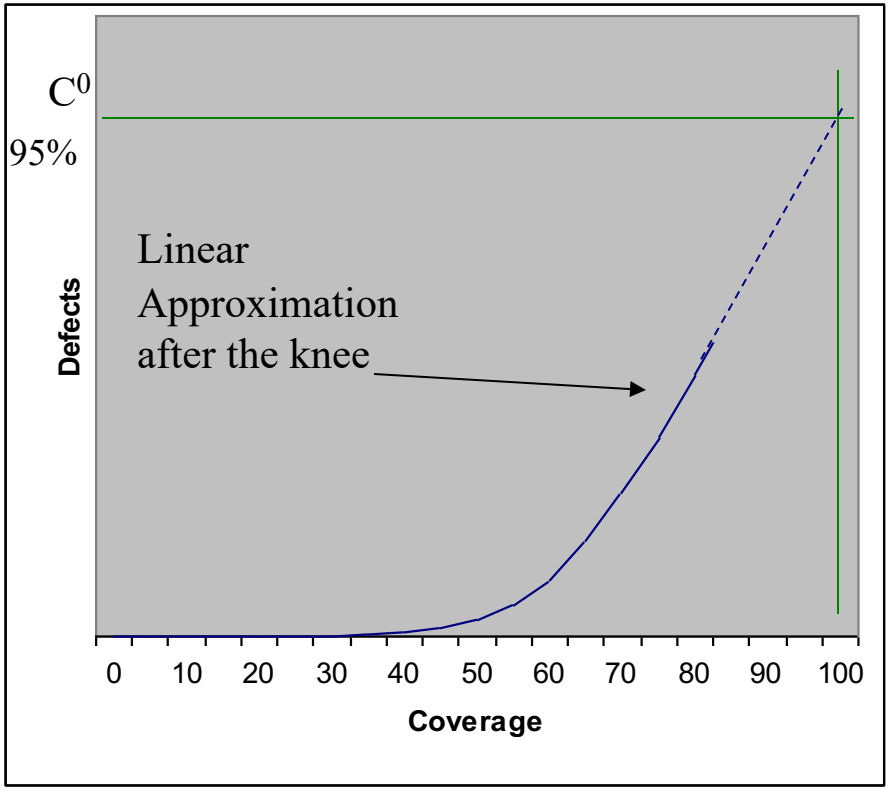
where C^0 : defect coverage, C^i : test coverage

a_0^i, a_1^i, a_2^i : parameters; i : branch cov, p-use cov etc.

- For "large" Ci, we can approximate

$$C^0 = -A^i + B^i C^i$$

Coverage Model, Estimated Defects



$$C^0 = -A^i + B^i C^i, \quad C^i > C^i_{knee}$$

- Only applicable after the knee
- Assumptions : Stable Software

Dead Code

- Some code, perhaps 3 to 16%, may be dead.
  - Code that can not be reached in the current release during any execution, or code that does not influence the output.
  - Software may have been modified leaving behind some code that is not longer a part of the functionality.
  - It may be possible to identify some dead code during compilation.
    - Undesirable things can happen because of dead code during maintenance.
  - Thus 100% coverage may be impossible to achieve in many cases.

S. Romano, C. Vendome, G. Scanniello and D. Poshyvanyk, "A Multi-Study Investigation into Dead Code," in *IEEE Transactions on Software Engineering*, vol. 46, no. 1, pp. 71-99, 1 Jan. 2020, doi: 10.1109/TSE.2018.2842781.

Location of the knee

$$C_{knee} = 1 - \left(\frac{E_{min}}{D_{min} E_0} \right) D_0$$

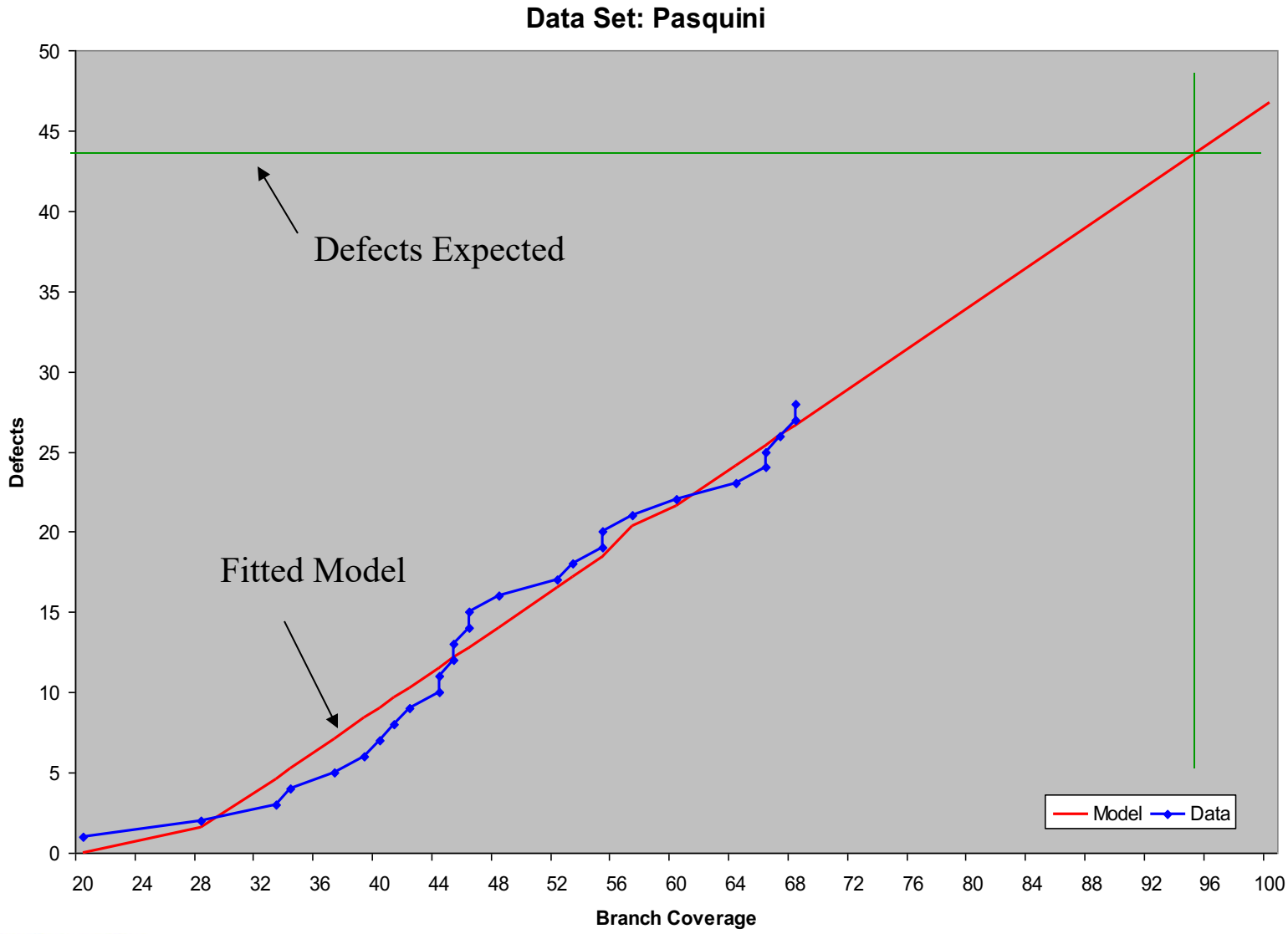
- Location of knee based on initial defect density
  - Lower defect densities cause knee to occur at higher coverage, since high detectability defects have already been removed.
- Based on interpretation through logarithmic model parameter significance: Malaiya and Denton

•Y.K. Malaiya and J. Denton, " Estimating the Number of Residual Defects" Proc. IEEE High Assurance Systems Engineering Symposium (HASE '98), November 1998, pp. 98-105.

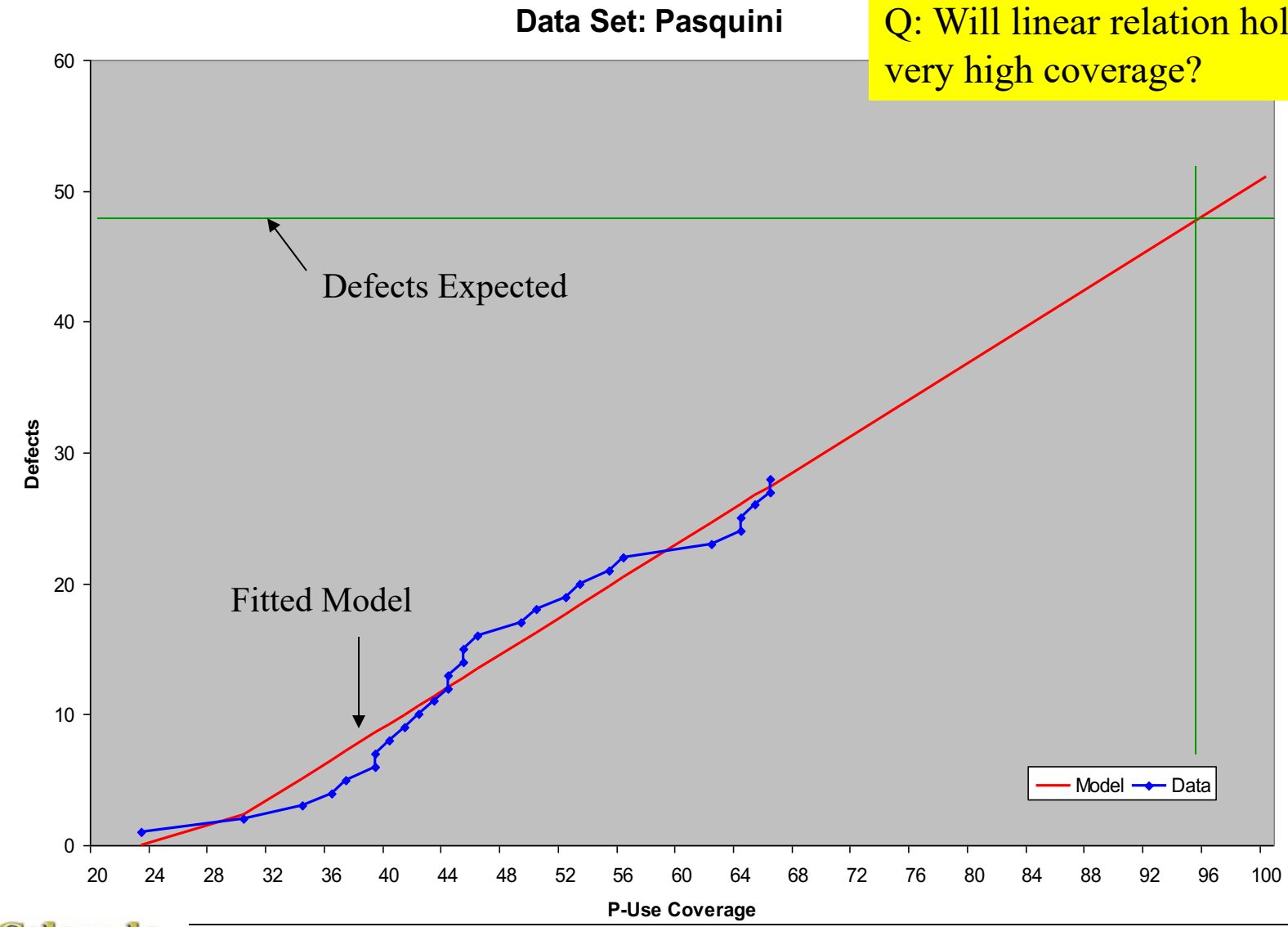
Data Sets Used
Vouk and Pasquini

- Vouk data
  - from N version programming project to create a flight controller
  - Three data sets, 6 to 9 errors each
- Pasquini data
  - Data from European Space Agency
  - C Program with 100,000 source lines
  - 29 of 33 known faults uncovered

Defects vs. Branch Coverage



Defects vs. P-Use Coverage



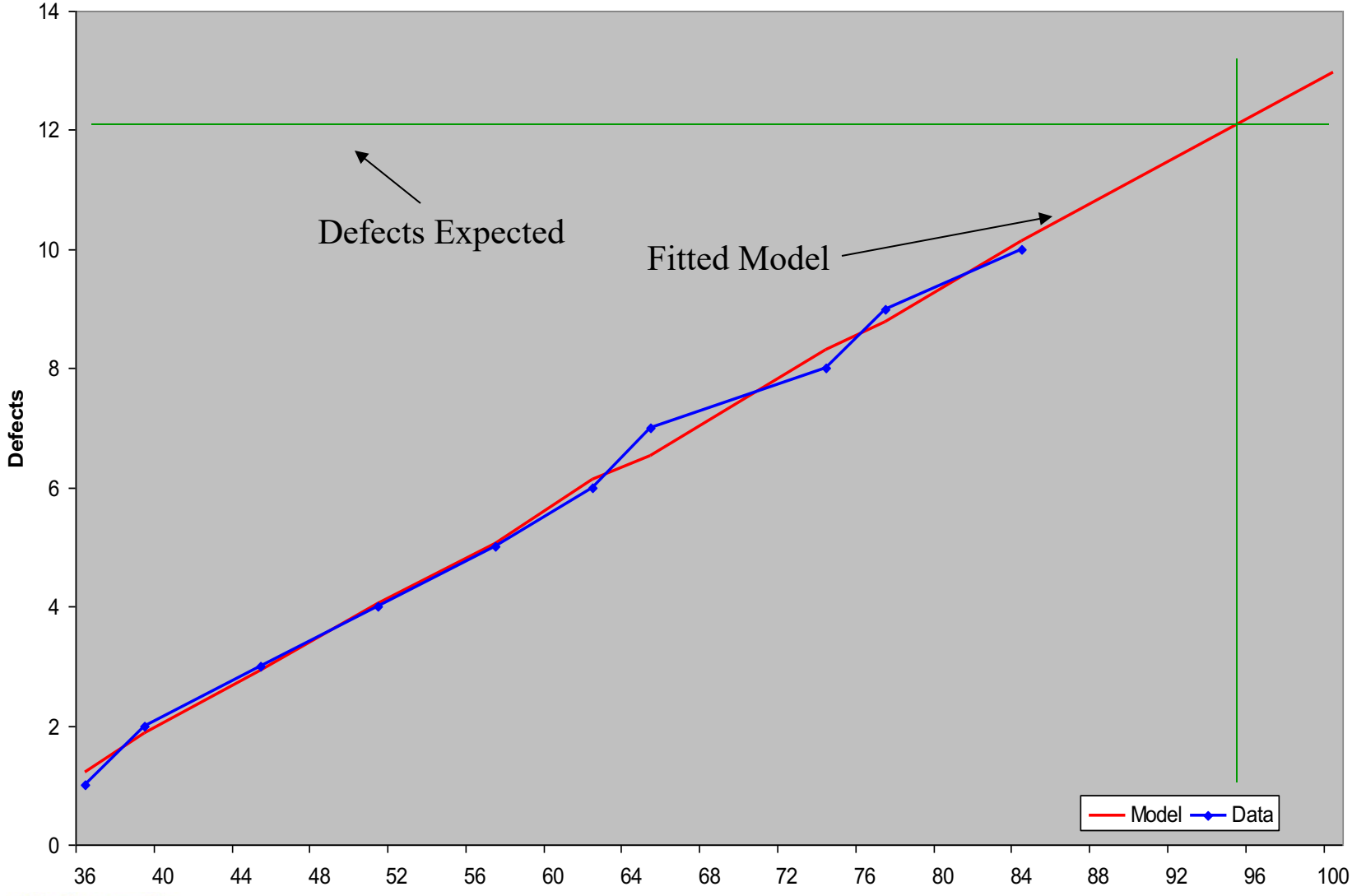
Estimation of Defect Density

- Estimated defects at 95% coverage, for Pasquini data (assume 5% *dead code*)
- 28 faults found, and 33 known to exist

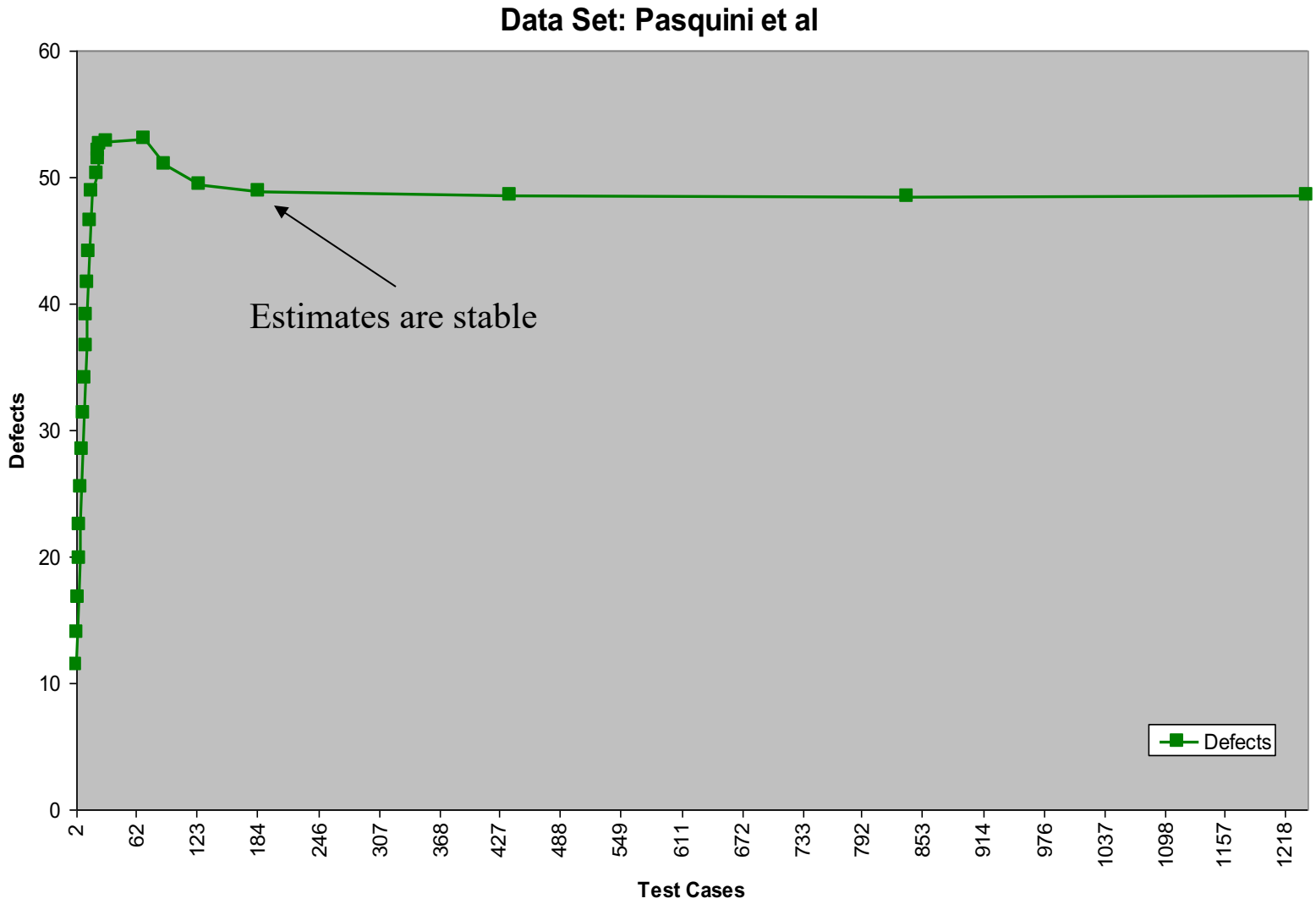
Measure	Coverage Achieved	Expected Defects
Block	82%	36
Branch	70%	44
P-uses	67%	48

Defects vs. P-Use Coverage

Data Set: Vouk 3



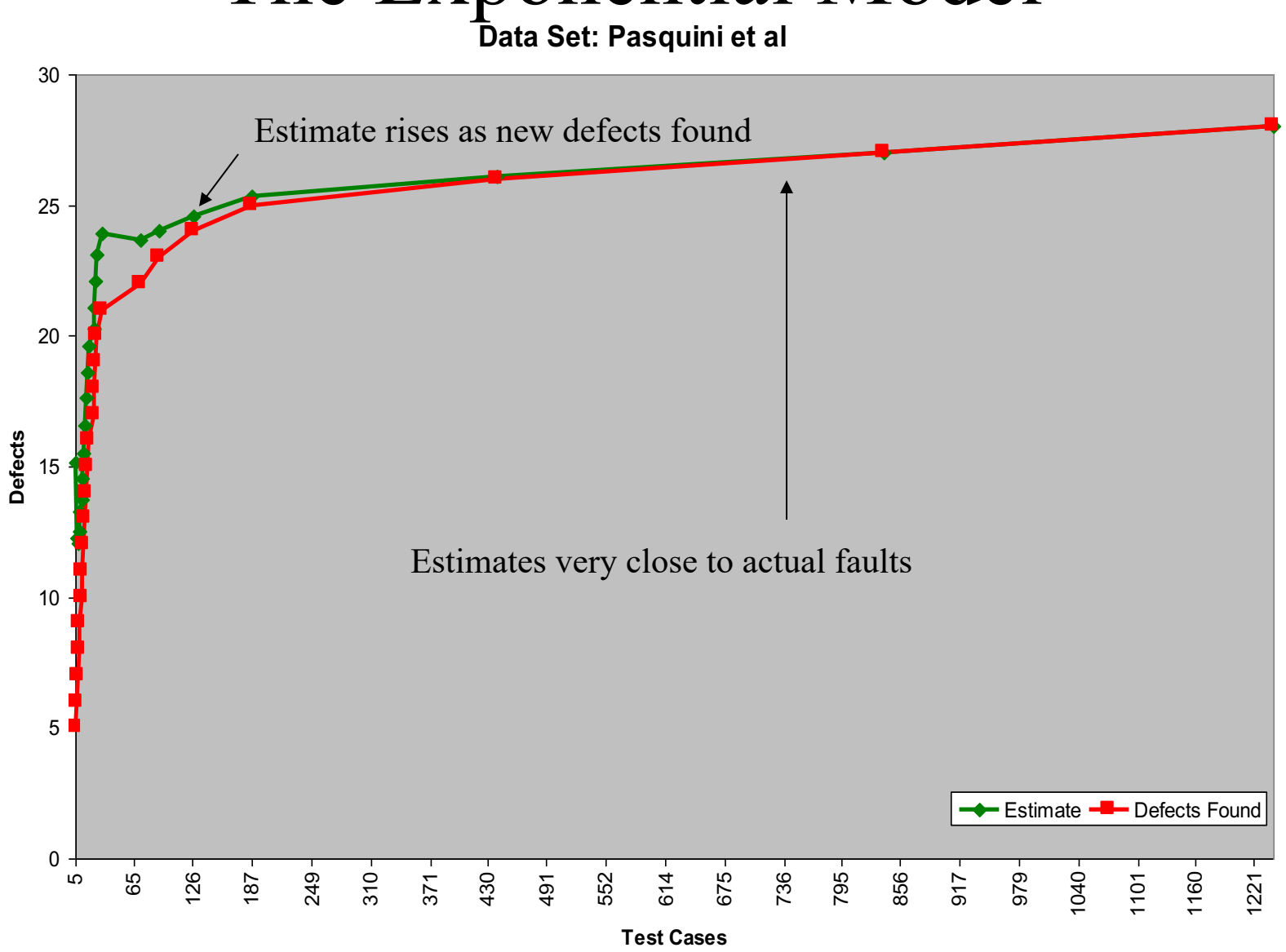
Coverage Based Estimation



Current Methods

- Development process based models allow for *a priori* estimates
  - Not as accurate as methods based on test data
- Sampling methods often assume faults found as easy to find as faults not found
  - Underestimates faults
- Exponential model
  - Assume applicability of exponential model
  - We present results of a comparison

The Exponential Model



Use of coverage tools

- Identify statements/branches not covered. Generate tests specifically to cover them.
  - Directed structural testing
- To reduce test-cases, remove inputs that do not increase coverage.
- Use coverage to guide fuzzing for finding vulnerabilities.

Test Case Reduction: Beyond Bugs by Alastair Donaldson and David MacIver, May 25, 2021

Jeff Offutt, Jie Pan and Jeff Voas, Procedures for Reducing the Size of Coverage-based Test Sets, Proc of the Twelfth Int Conf on Testing Computer Software, pages 111--123, June 1995.

Related articles

- Frankl & Iakouneno, Proc. SIGSOFT ‘98
  - 8 versons of European Space Agency program, 10K LOC, Single fault reinsertion
- Williams, Mercer, Mucha, Kapur, 2001
  - "Code coverage, what does it mean in terms of quality?,“
  - analysis from first principles
- Peter G Bishop, SAFECOMP 2002
  - A related model, unreachable code

Related articles

- Mockus, A.; Nagappan, N.; Dinh-Trong, T.T., "Test coverage and post-verification defects: A multiple case study," Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on , vol., no., pp.291,301, 15-16 Oct. 2009
- Avaya lab data
- "*The test effort increases exponentially with test coverage, but the reduction in field problems increases linearly with test coverage*."

Observations and Conclusions

- Estimates with new method are very stable
  - Visual confirmation of earlier projections
- Which coverage measure to use?
  - Stricter measure will yield closer estimate
- Some code may be dead or unreachable
  - Found with compile or link time tools
  - May need to be taken into account

Voak's Observation

He thought that a model is not possible because he collected data for programs

- That were functionally identical (for redundancy)
- but independent implemented
- Problem: defects found with the same coverage did not match!
- He gave up but gave us the data.
- Likely reason: Different implementations may result in different testability. Prior testing may have been different.

Voak's Observation

He thought that a model is not possible because he collected data for programs

- That were functionally identical (for redundancy) but independent implemented
- Problem: defects found with the same coverage did not match!

Research Ideas

Some research that I would like someone to do

- Compare alternative models using data
  - Collect data and models
- Modeling software evolution
- Connect detectability profile to our model
- Compare mutation testing and fault coverage
  - How representative are mutations
- Using fault coverage for vulnerability detection
- Applicability of fault coverage for high or very low defect densities
- Using fault coverage with deterministic testing: limitations

Appendix
Software Testing Tools

Manual vs. automated testing

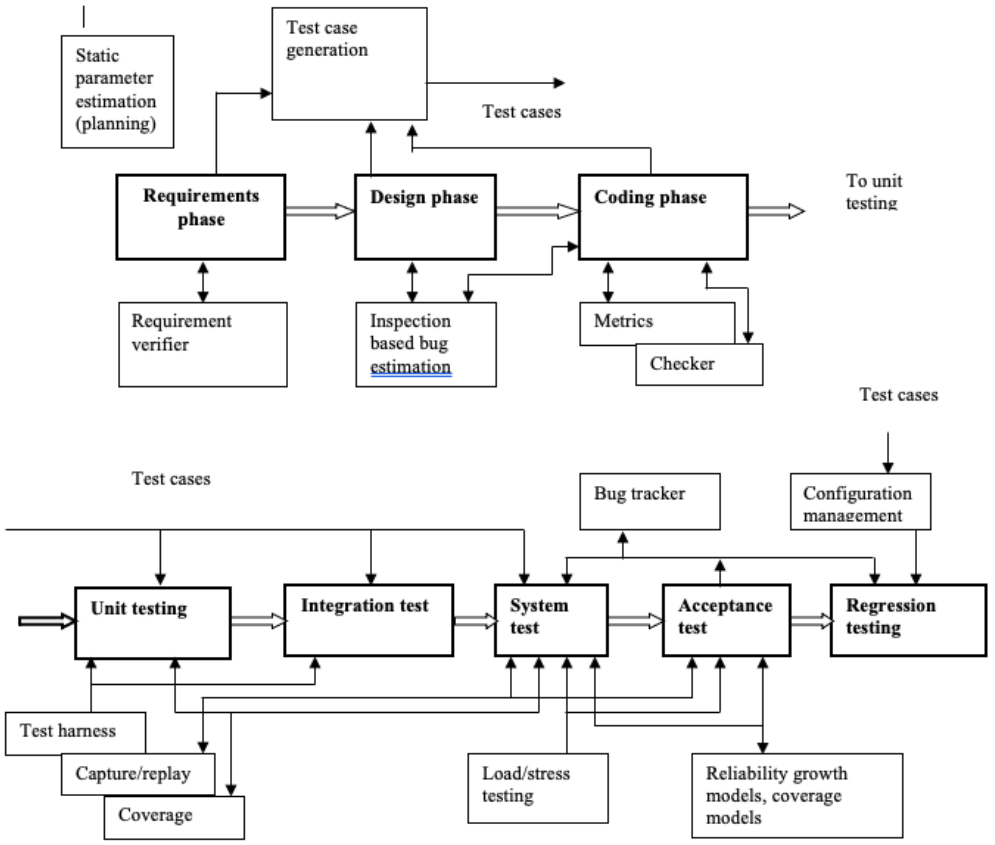
Possible Benefits

Test step	Manual testing	Automated testing	Percent Improvement
Test plan development	32	40	-25%
Test case development	262	117	55%
Test execution	466	23	95%
Test result analyses	117	58	50%
Defect tracking	117	23	80%
Report creation	96	16	83%
Total hours	1090	277	75%

QA Quest, The Newsletter of the Quality Assurance Institute, Nov. 1995.

Software Testing tools

Figure 1: Software development/test phases and test/reliability tools.



Y.K. Malaiya, "Automated Test Software", for Wiley Encyclopedia of Electrical and Electronics Engineering, Editor: J.G. Webster, Pub. John Wiley & Sons, Inc. 1999, pp. 135-141.

Comparison of modern software testing tools

Features	Katalon Studio	Selenium	UFT	TestComplete
Test development platform	Cross-platform	Cross-platform	Windows	Windows
Application under test	Windows desktop, Web, Mobile apps, API/Web services	Web apps	Windows desktop, Web, Mobile apps, API/Web services	Windows desktop, Web, Mobile apps, API/Web services
Scripting languages	Java/Groovy	Java, C#, Perl, Python, JavaScript, Ruby, PHP	VBScript	JavaScript, Python, VBScript, JScript, Delphi, C++ and C#
Programming skills	Not required. Recommended for advanced test scripts	Advanced skills needed to integrate various tools	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts
Object storage and maintenance	Built-in object repository, XPath, object re-identification	XPath, UI Maps	Built-in object repository, smart object detection and correction	Built-in object repository, detecting common objects
DevOps/ALM integrations	Many	No (require additional libraries)	Many	Many
Continuous integrations	Popular CI tools (e.g. Jenkins, Teamcity)	Various CI tools (e.g. Jenkins, Cruise Control)	Various CI tools (e.g. Jenkins, HP Quality Center)	Various CI tools (e.g. Jenkins, HP Quality Center)
Test Analytics	Katalon TestOps	No	No	No
Product support	Community, Business support service, Dedicated staff	Open-source community	Dedicated staff, Community	Dedicated staff, Community
License type	Proprietary	Open-source (Apache 2.0)	Proprietary	Proprietary
Cost	Freemium	Free	License and maintenance fees	License and maintenance fees

https://katalon.com/resources-center/blog/comparison-automated-testing-tools