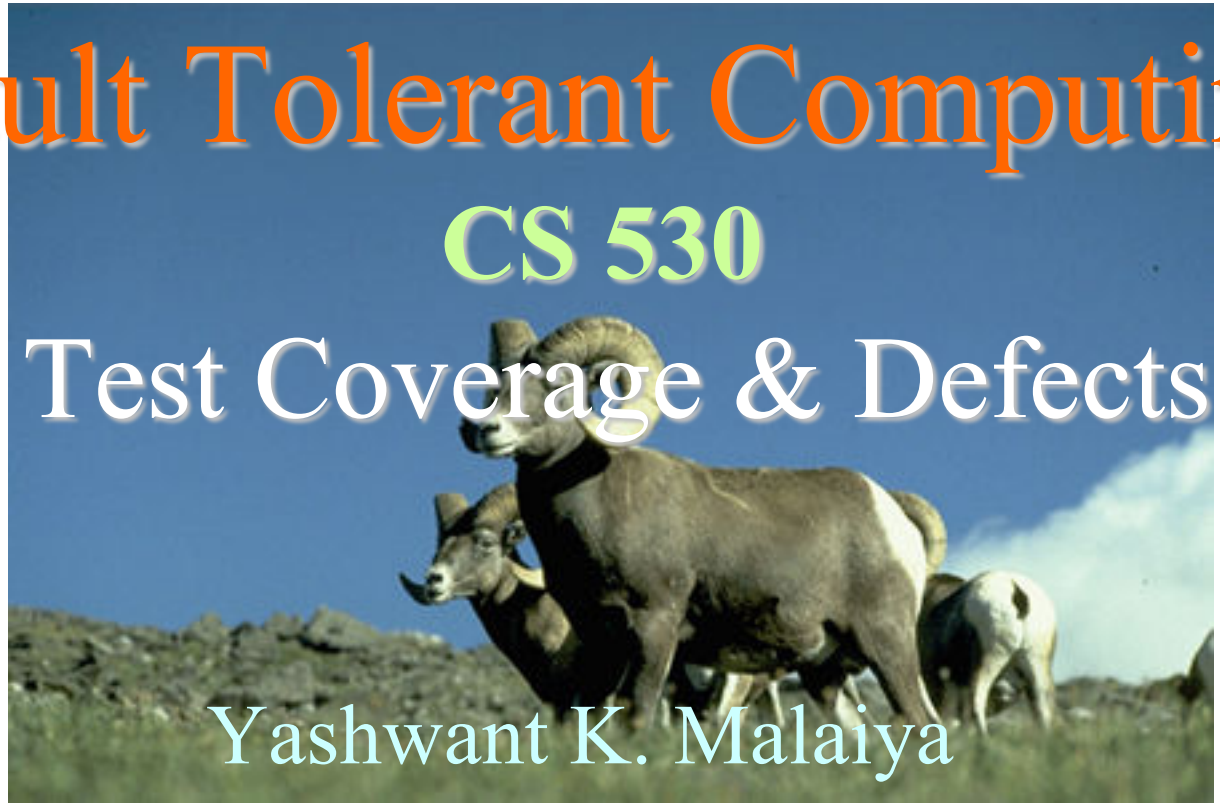
Fault Tolerant Computing

CS 530

Test Coverage & Defects



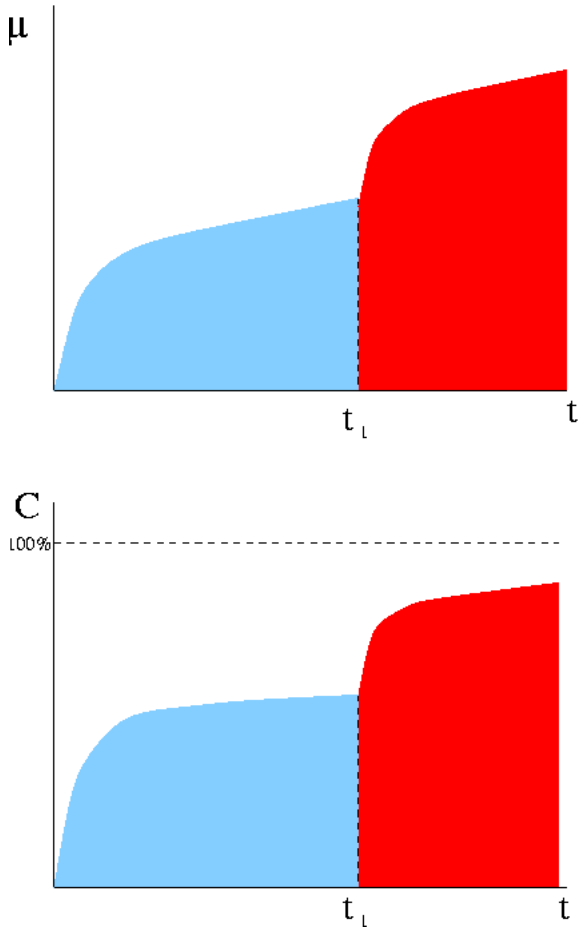
Yashwant K. Malaiya

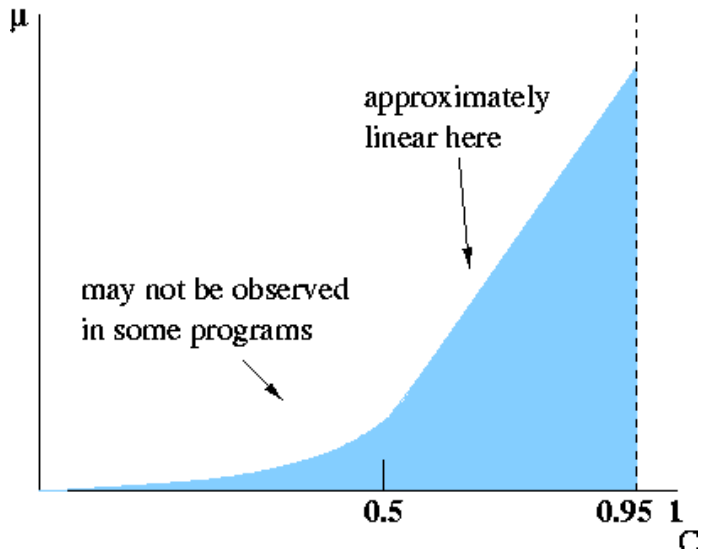
Colorado State University

Test Coverage Measures

- Statement or Block coverage
- Branch or decision coverage
- 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*

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

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,

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

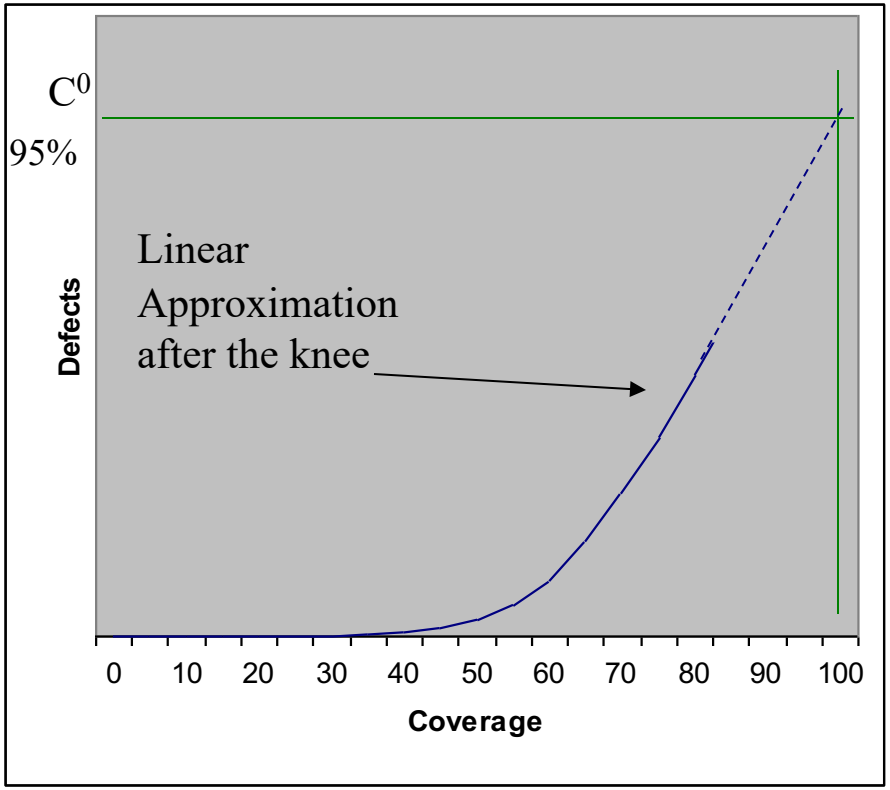
where C^0 : defect coverage, C^i : test coverage

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

Location of the knee

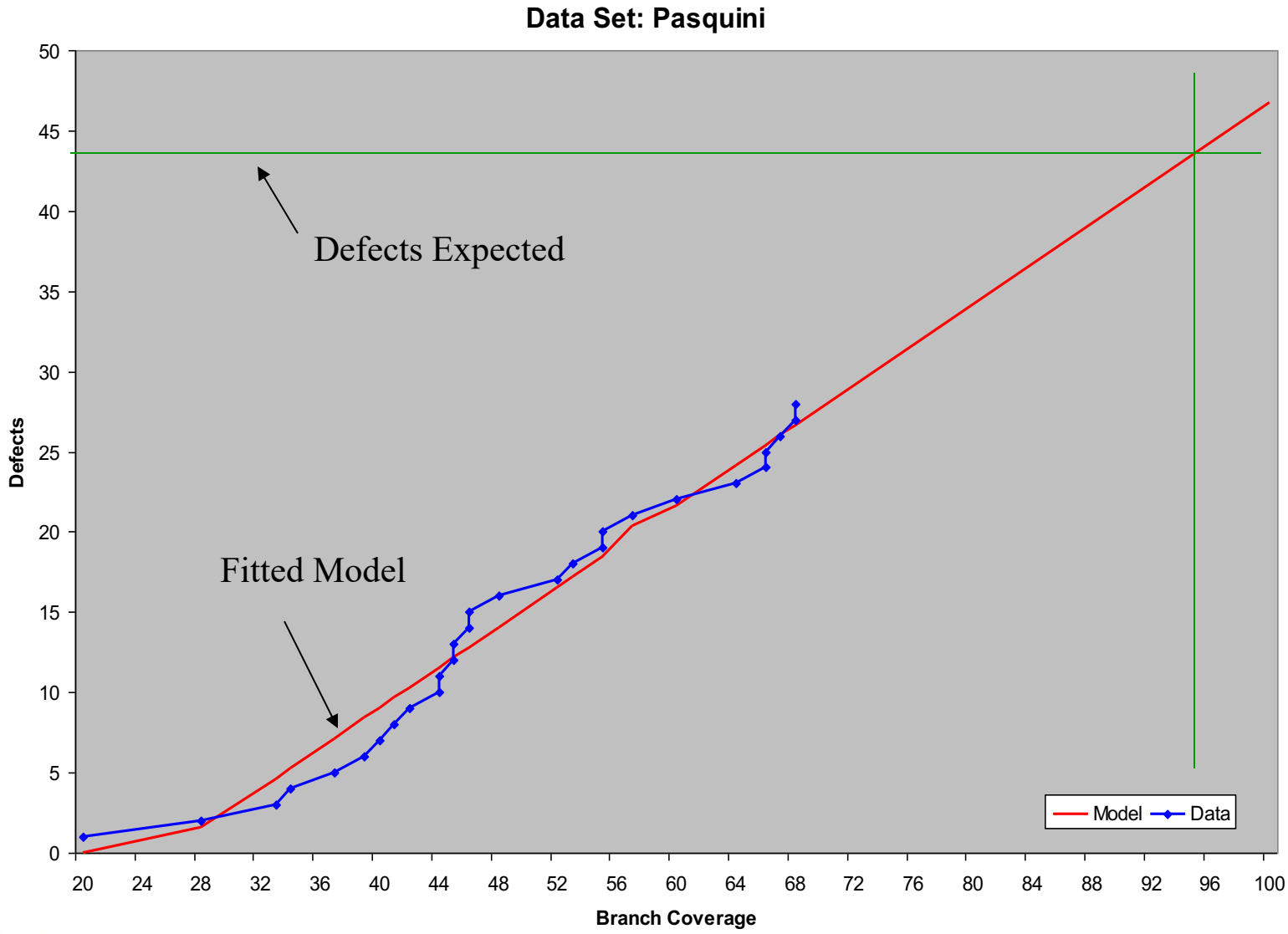
$$C_{knee} \quad 1 - \left(\frac{E_{min}}{D_{min} E_0} \right) \quad D_0$$

- Based on interpretation through logarithmic model
- Location of knee based on initial defect density
- Lower defect densities cause knee to occur at higher coverage
- Parameter estimation : Malaiya and Denton (HASE ‘98)

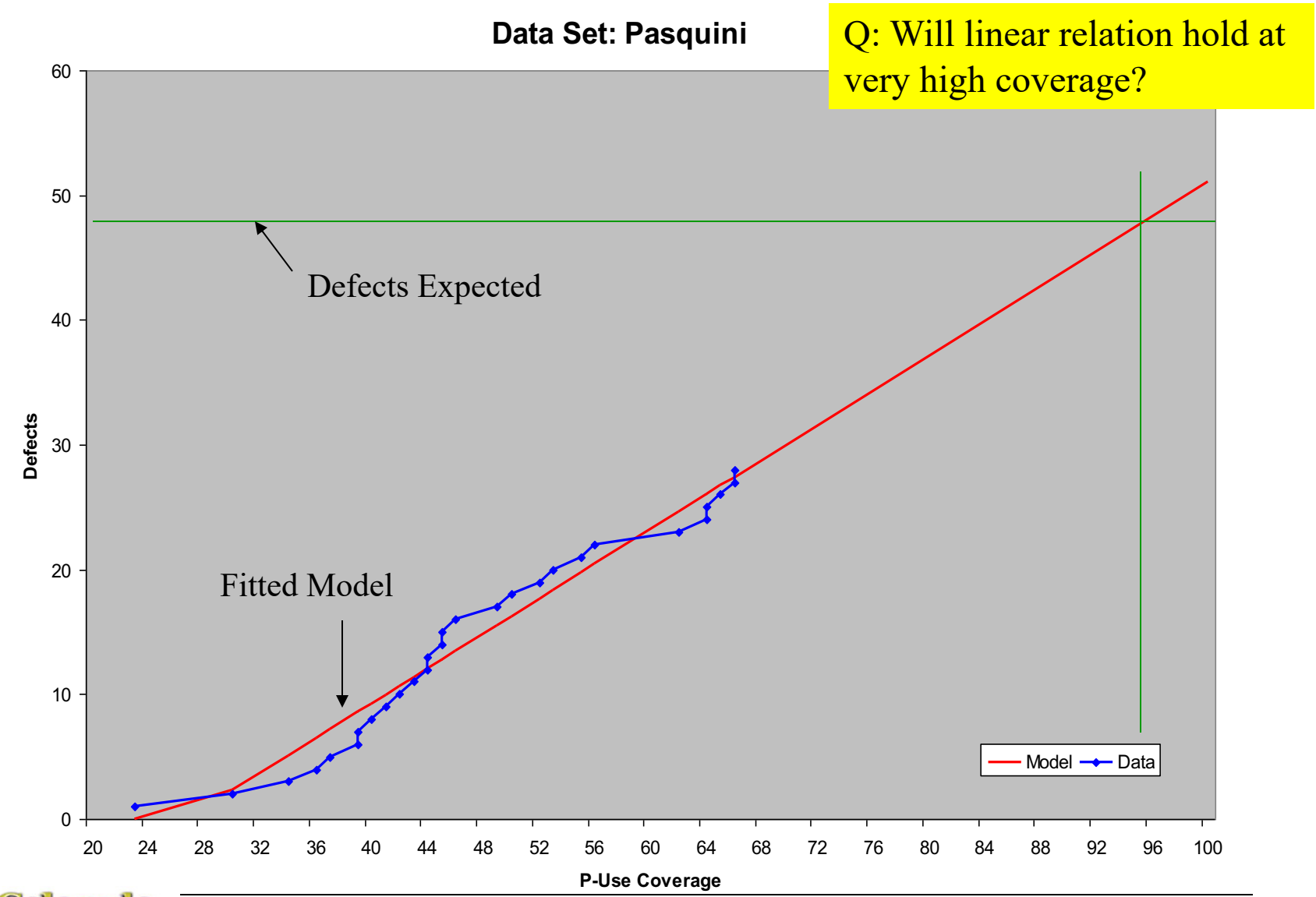
Data Sets Used
Vouk and Pasquini

- 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



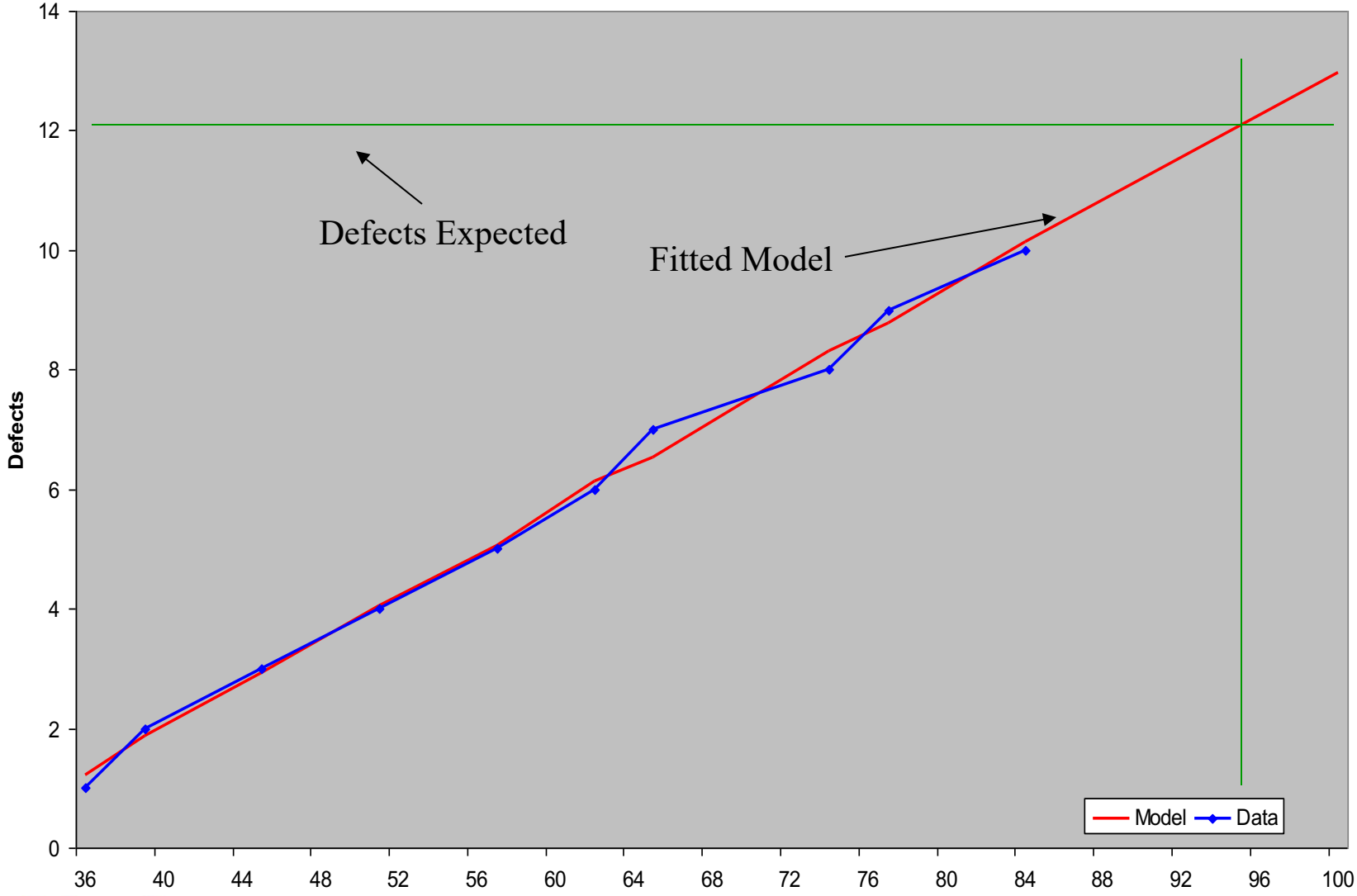
Estimation of Defect Density

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

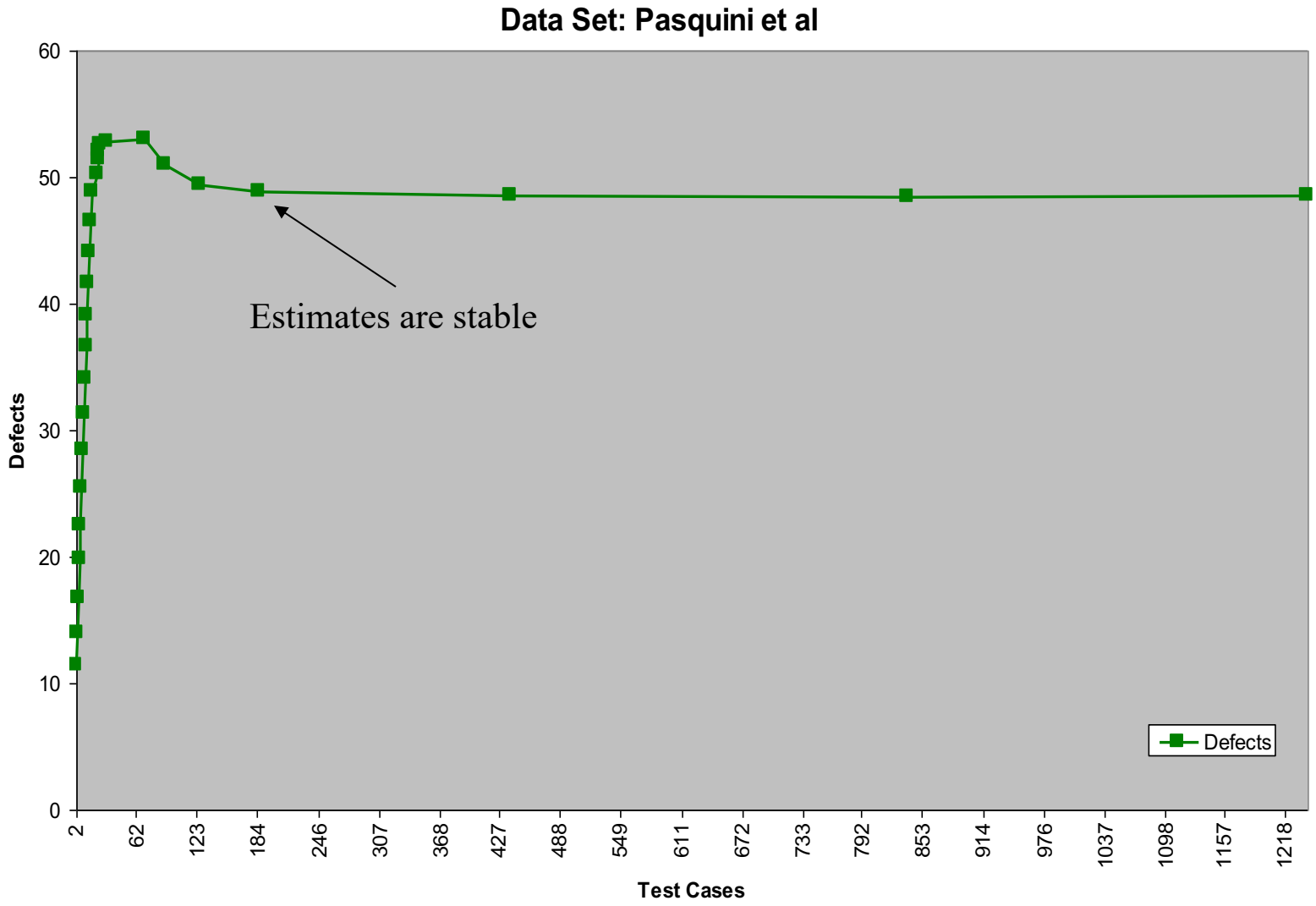
Measure	Coverage Achieved	Expected Defects
Block	82%	36
Branch	70%	44
P-uses	67%	48

Defects vs. P-Use Coverage

Data Set: Vouk 3



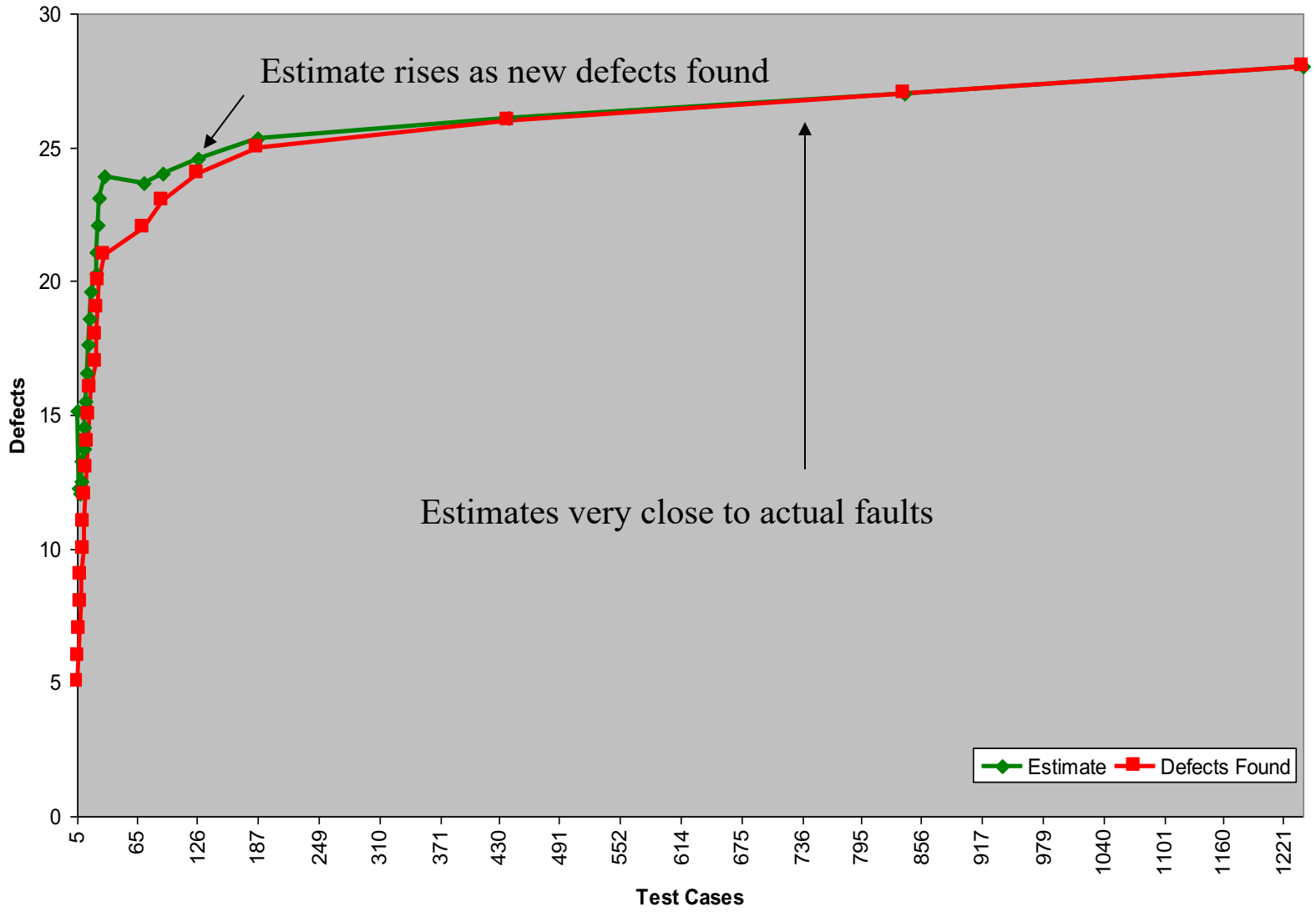
Coverage Based Estimation



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



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.

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!
- Reason: Different implementations may result in different testability.

Research Ideas

Some research that I would like someone to do

- Compare alternative models using data
- Model software evolution
- Connect detectability profile to our model