Fault Tolerant Computing

CS 530

Final Review

Yashwant K. Malaiya
Colorado State University

Also see

- Midterm Review Slides

Exponential Reliability Growth Model

- Most common and easiest to explain model. From 1970s
- Notation:
  - **Total expected faults** detected by time t: $\mu(t)$
  - **Failure intensity: fault detection rate** $\lambda(t)$
  - **Undetected defects present at time t:** N(t)
- **By definition,** $\lambda(t)$ is derivative of $\mu(t)$. Hence

$$\lambda(t) = \frac{d}{dt}\mu(t)$$

$$= -\frac{d}{dt}N(t)$$

Since faults found are no longer undetected

Exponential SRGM Derivation Pt 1

- Notation
  - T_s: average single execution time
  - k_s: expected fraction of faults found during T_s
  - T_L: time to execute each program instruction once

$$-\frac{dN(t)}{dt}T_s = k_s N(t)$$

Key assumption

$$-\frac{dN(t)}{dt} = \frac{K}{T_L}N(t) = \beta_1 N(t)$$

Notation: Here we replace K_s and T_s by more convenient K and $T_{L.}$

where $K = k_s \dfrac{T_L}{T_s}$ is fault exposure ratio

Exponential SRGM Derivation Pt 2

- **We get**

$$N(t) = N(0)\, e^{-\beta_1 t}$$

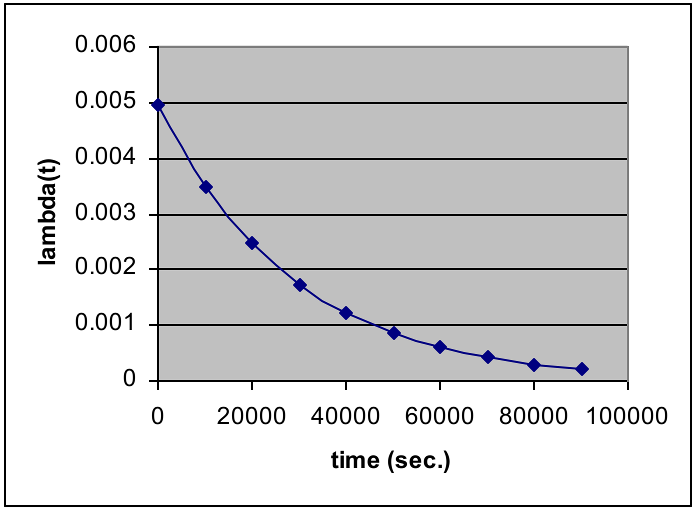
$$\mu(t) = \beta_o (1 - e^{-\beta_1 t})$$

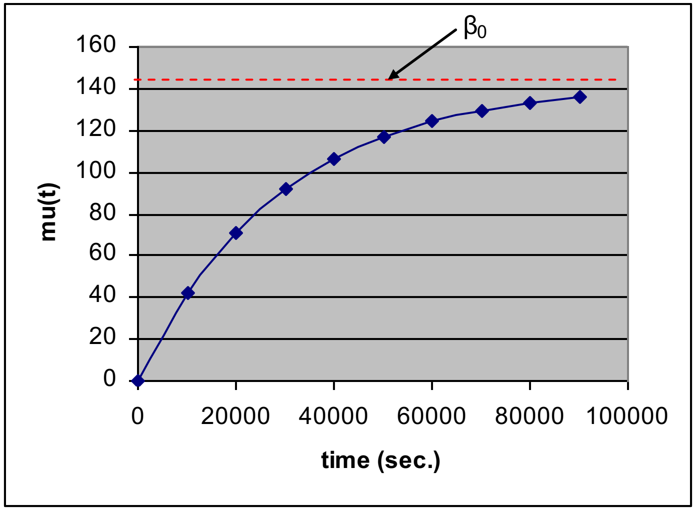
$$\lambda(t) = \beta_o \beta_1 e^{-\beta_1 t}$$

The 2 equations contain the same information.

- For $t \to \infty$, total $\beta_o = N(0)$ faults would be eventually detected. A "*finite-faults-model*".
- Assumes no new defects are generated during debugging.
- Proposed by Jelinski-Muranda '71, Shooman '71, Goel-Okumoto '79 and Musa '75-' 80. also called Basic.

Exponential SRGM





The plots show λ(t) and μ(t) for β0=142 and β1=3.5×10^{-5}. Note that μ(t) asymptotically approaches 142.

A Basic SRGM (cont.)

- **Note that parameter β_1 is given by:**

$$\beta_1 = \frac{K}{T_L} = \frac{K}{(S.Q.\frac{1}{r})}$$

- S: source instructions,
- Q: number of object instructions per source instruction typically between 2.5 to 6 (see page 7-13 of Software rteliability Handbook, sec 7)
- r: object instruction execution rate of the computer
- K: *fault-exposure ratio*, range 1×10^{-7} to 10×10^{-7}, (t is in CPU seconds). Assumed constant here*.
- Q, r and K should be relatively easy to estimate.

*Y. K. Malaiya, A. von Mayrhauser and P. K. Srimani, "An examination of fault exposure ratio," in IEEE Transactions on Software Engineering, vol. 19, no. 11, pp. 1087-1094, Nov 1993

Example: SRGM with Test Data (cont.)

- Fitting we get

  $\beta_o = 101.47$ and $\beta_1 = 5.22 \times 10^{-5}$

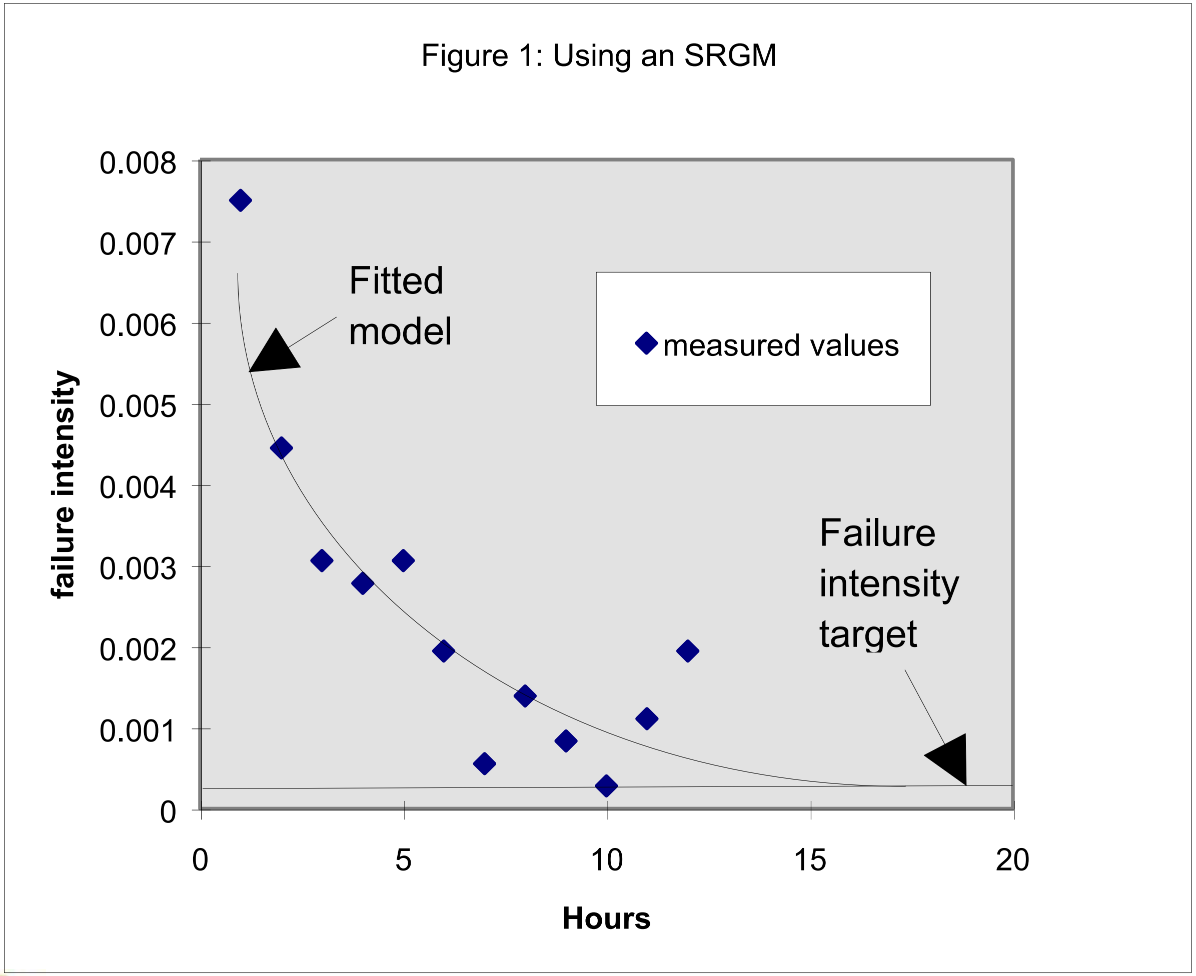
- stopping time t_f is then given by:

$$2.78 \times 10^{-4} = 101.47 \times 5.22 \times 10^{-5} e^{-5.22 \times 10^{-5} \times t_f}$$

- yielding t_f = 56, 473 sec., i.e. 15.69 hours

Note: The exact values of the parameter values estimated depend on the numerical methods used.

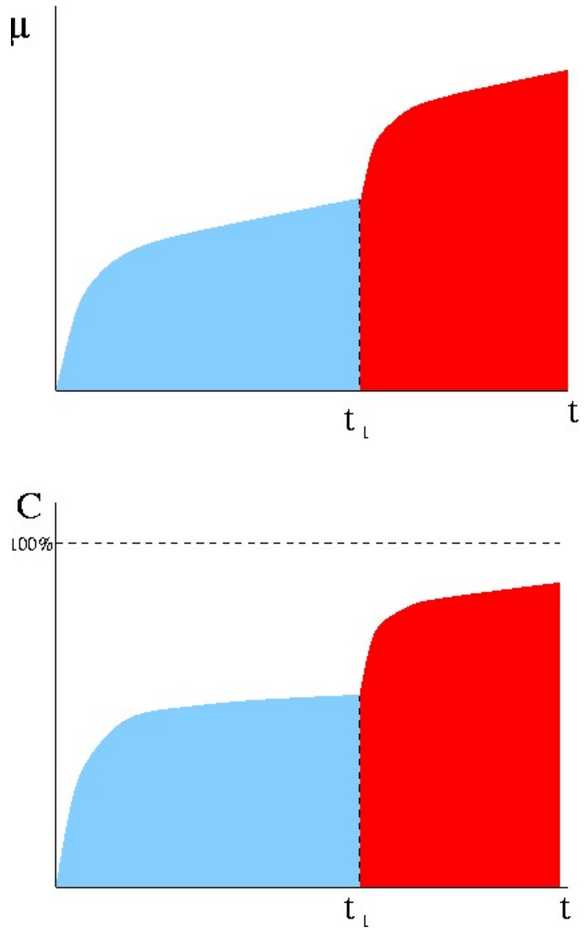
Example: SRGM with Test Data (cont.)

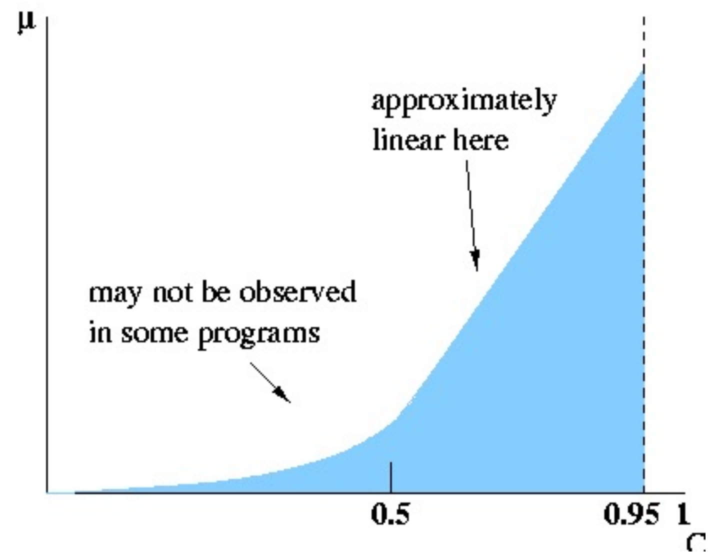


On-Line course Survey

- Log into Canvas
- Click Menu item Course Survey
- Take 15 minutes

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,

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

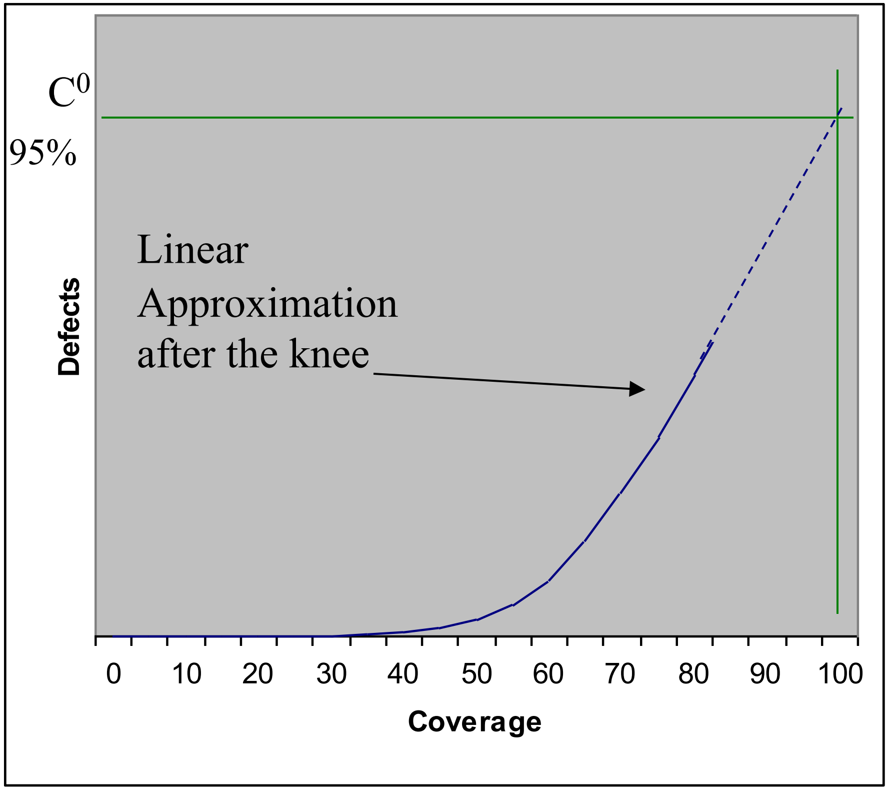
where C^0 : defect coverage, C^i : test coverage

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

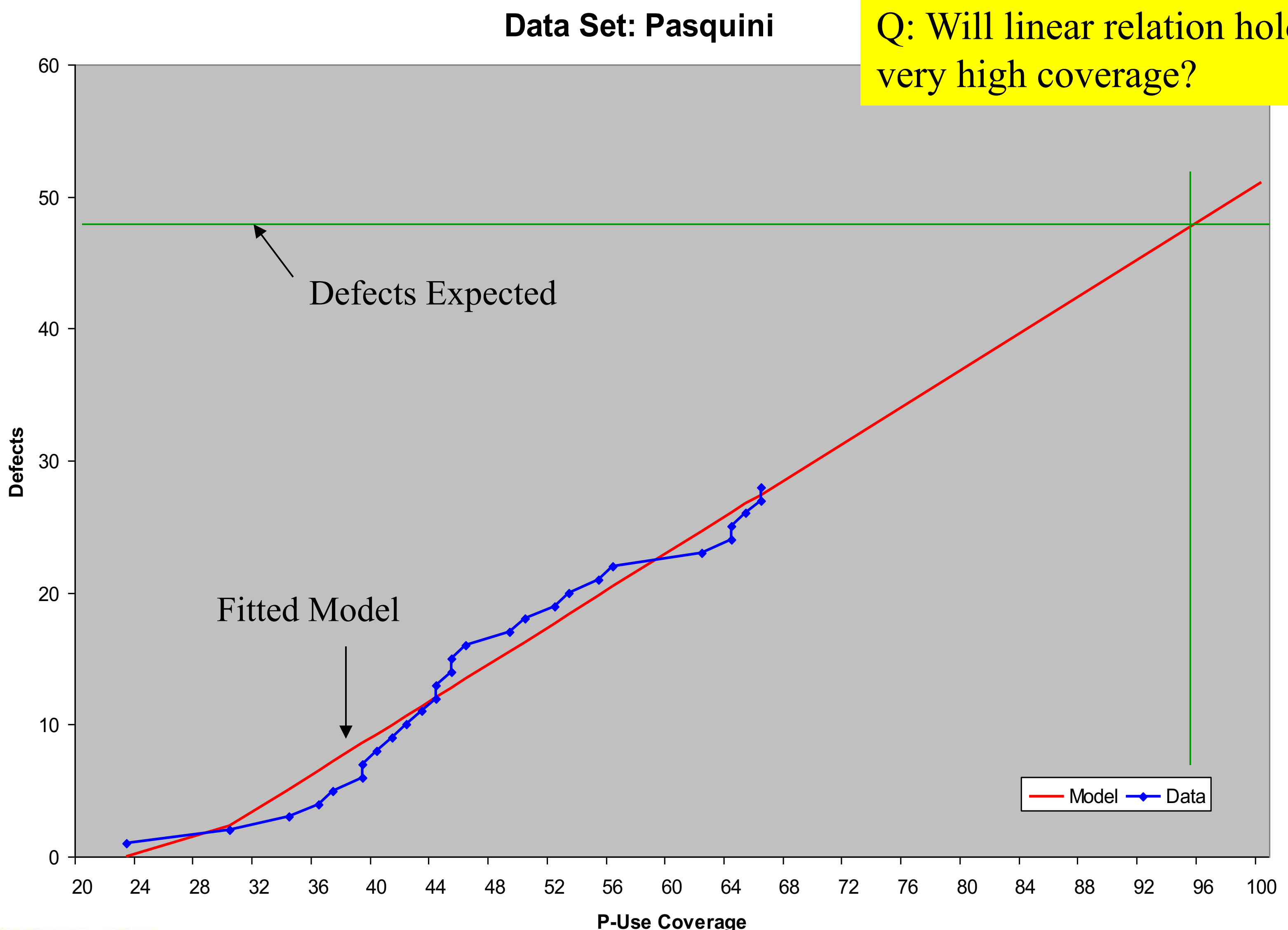


$$C^0 = -A^i + B^i C^i, \quad C^i > C^i_{knee}$$

- Only applicable after the knee
- Assumptions : Stable Software

Defects vs. P-Use Coverage

Q: Will linear relation hold at very high 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

Sequential execution

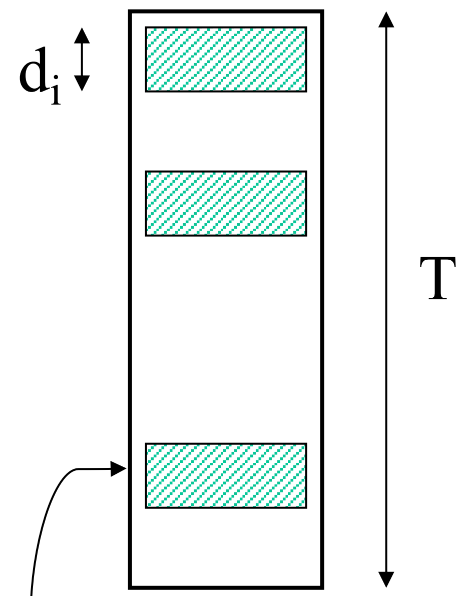
- Assume one module executed at a time.
- f_i: fraction of time module i under execution; λ_i its failure rate
- Mean system failure rate:

$$\lambda_{sys} = \sum_{i=1}^{n} f_i \lambda_i$$

Sequential Execution (cont.)

- T: mean duration of a single transaction
- module i is called e_i times during T, each time executed for duration d_i

$$f_i = \frac{e_i \ d_i}{T}$$



i called 3rd time

Sequential Execution (cont.)

- System reliability $R_{sys} = \exp(-\lambda_{sys}\, T)$

$$R_{sys} = \exp(-\sum_{i=1}^{n} e_i\, d_i\, \lambda_i)$$

$$\lambda_{sys} = \sum_{i=1}^{n} f_i\, \lambda_i$$

- Since $\exp(-d_i\lambda_i)$ is R_i,

$$R_{sys} = \prod_{i=1}^{n} (R_i)^{e_i}$$

$$f_i = \frac{e_i\, d_i}{T}$$

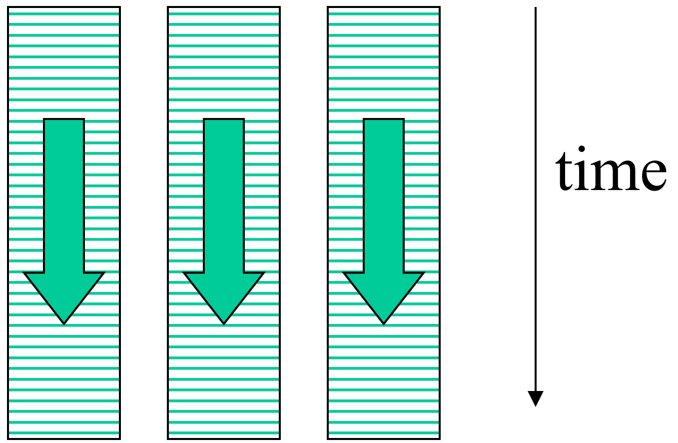
Sequential Execution Risk

- System Risk = Σ Risk due to failure type i

	Called times	Av duration	Fraction of T	Failure rate	Av cost/ failure	Fail prob/T	Risksi
Module i	ei	di	fi = di/T	λi	Ci	=1-exp(ei.di. λi)	potential loss per trans
a	1	3	12%	0.01	20	0.030	0.59
b	2	4	32%	0.03	100	0.213	21.34
c	7	2	56%	0.001	200	0.014	2.78
Total time	T	25	100%			**Total risk**	**24.71**

Concurrent execution

- Concurrently executing modules: all run without failures for system to run
- j concurrently executing modules



$$\lambda_{sys} = \sum_{j=1}^{m} \lambda_j$$

N-version systems: Correlation

- 3-version system
- q_3: probability of all three versions failing for the same input.
- q_2: probability that any two versions will fail together.
- Probability P_{sys} of the system *failing* for a transaction

$$P_{sys} = q_3 + 3q_2$$

N-version systems: Correlation

- Example: *data collected by Knight-Leveson; computations by Hatton*
- *3-version system, probability of a version failing for a transaction 0.0004*
- *in the absence of any correlated failures*

$$P_{sys} = (0.0004)^3 + 3(1 - 0.0004)(0.0004)^2$$
$$= 4.8 \times 10^{-7}$$

- Uncorrelated improvement factor of 0.0004/4.8 x 10^{-7} = 833.3

N-version systems: Correlation

$$P_{sys} = q_3 + 3q_2$$

- Uncorrelated improvement factor of $0.0004/4.8 \times 10^{-7} = 833.3$
- Correlated: $q_3 = 2.5\times10^{-7}$ and $q_2 = 2.5\times10^{-6}$
- $P_{sys} = 2.5\times10^{-7} + 3\times2.5\times10^{-6} = 7.75\times10^{-6}$
- improvement factor: $0.0004/7.75\times10^{-6}=$ **51.6**
- state-of-the-art techniques can reduce defect density only by a factor of **10!**
- Thus 3-version system may be worth considering in some cases.

Reliability Allocation for Software Systems

- a block i is under execution for a fraction x_i of the time where $\Sigma x_i = 1$
- Reliability allocation problem

$$\text{Minimize} \quad C = \sum_{i=1}^{n} \frac{1}{\beta_i} \ln\left(\frac{\lambda_{0i}}{\lambda_i}\right)$$

$$\text{subject to } \lambda_{ST} \geq \sum_{i=1}^{n} x_i \lambda_i$$

Solution using Lagrange multiplier

- solutions for the optimal failure rates

$$\lambda_1 = \frac{\dfrac{\lambda_{ST}}{x_1}}{\displaystyle\sum_{i=1}^{n} \frac{\beta_1}{\beta_i}} \qquad \lambda_2 = \frac{\beta_1 x_1}{\beta_2 x_2} \lambda_1 \qquad \cdots \qquad \lambda_n = \frac{\beta_1 x_1}{\beta_n x_n} \lambda_1$$

- optimal values of test times d_1 and d_i, $i \neq 1$

$$d_1 = \frac{1}{\beta_1} \ln \left(\frac{\lambda_{10} x_1 \displaystyle\sum_{i=1}^{n} \frac{\beta_1}{\beta_i}}{\lambda_{ST}} \right) \qquad d_i = \frac{1}{\beta_i} \ln \left(\frac{\lambda_{i0} \beta_i x_i}{\lambda_1 \beta_1 x_1} \right)$$

Colorado State University
Knowledge to Go Places

Ex: Optimal: Software with 5 blocks

$\lambda_{ST} \leq 0.04$

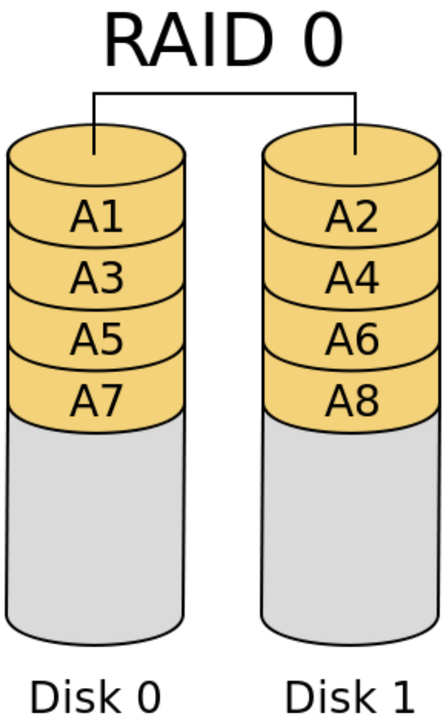
Block	B_1	B_2	B_3	B_4	B_5
Size KSLOC	1	2	3	10	20
Ini Defect density	10	10	10	15	20
β_i	4.59×10^{-3}	2.30×10^{-3}	1.53×10^{-3}	4.59×10^{-4}	2.30×10^{-4}
λ_{i0}	0.046	0.046	0.046	0.069	0.092
x_i	0.028	0.056	0.083	0.278	0.556
Optimal λ_i	0.04	0.04	0.04	0.04	0.04
Optimal d_i	30.1	60.1	90.2	1184	3620

- Optimal when all modules have the same failure rate!

Standard RAID levels

- RAID 0: striping
- RAID 1: mirroring
- RAID 2: bit-level striping, Hamming code for error correction (not used anymore)
- RAID 3: byte-level striping, parity (rare)
- RAID 4: block-level striping, parity
- RAID 5: block-level striping, distributed parity
- RAID 6: block-level striping, distributed double parity

RAID 0



- Data striped across n disks
- Read/write in parallel
- No redundancy.

$$R_{sys} = \prod_{i=1}^{n} R_i$$

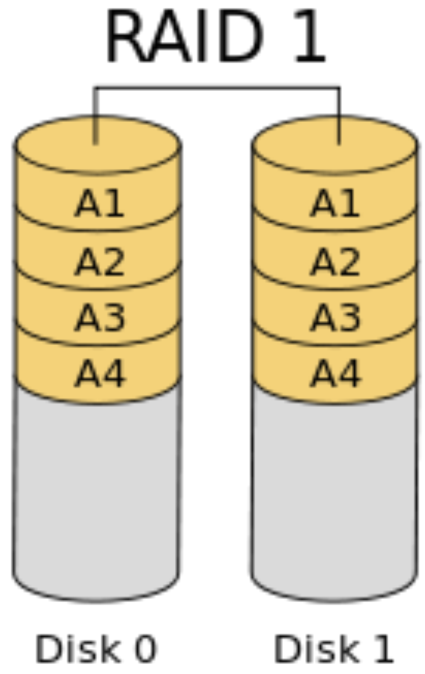
- Ex: 3 year disk reliability = 0.9 for 100% duty cycle. n = 14
- $R_{sys} = (0.9)^{14} = 0.23$

RAID 1

- Disk 1 mirrors Disk 0
- Read/write in parallel
- One of them may be used as backup.

$$R_{sys} = \prod_{i=1}^{n} [1-(1-R_i)^2]$$

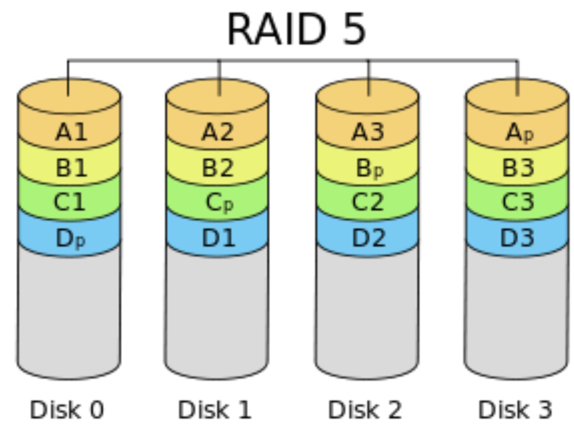
- Ex: 3 year disk reliability = 0.9 for 100% duty cycle. n = 7 pairs
- $R_{sys} = (2x0.9-(0.9)^2)^7 = 0.93$



Failed disk identified using internal CRC

RAID 5

- Distributed parity
- If one disk fails, its data can be reconstructed using a spare



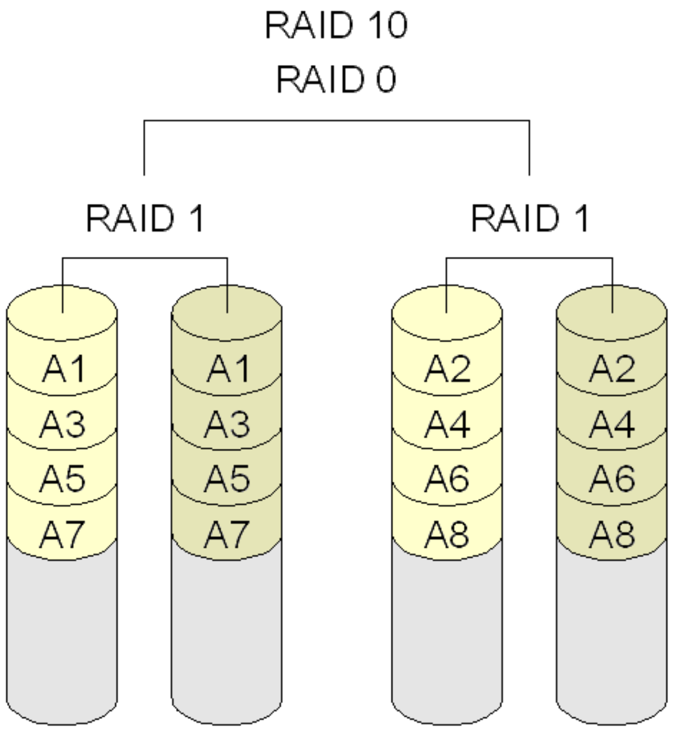
$$R_{sys} = \sum_{j=n-1}^{n} \binom{n}{j} R_j^{\ j} (1 - R_i)^{n-j}$$

- Ex: 3 year disk reliability = 0.9 for 100% duty cycle. n = 13, j = 12, 13
- $R_{sys} = 0.62$

RAID 10

- Stripe of mirrors: each disk in RAID0 is duplicated.

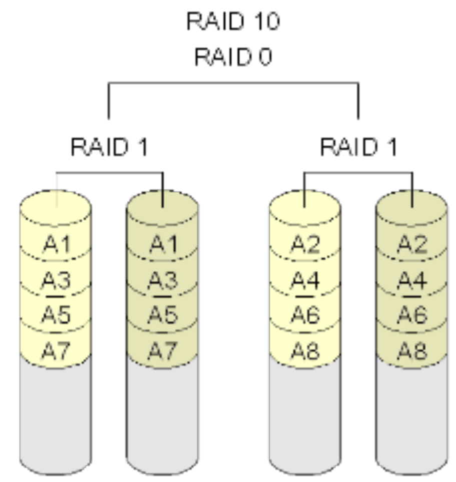
$$R_{sys} = \prod_{i=1}^{ns} [1-(1-R_i)^2]$$



- Ex: 3 year disk reliability = 0.9 for 100% duty cycle. ns = 6 pairs,
- R_{sys} = 0.94

RAID 10: redundancy at lower level

RAID 10: Example



- Consider 10 disks where 5 disks are of type A each having a reliability of 0.5 for 100% duty cycle, and the other 5 disks are of type B each having a reliability of 0.75 for 100% duty cycle. What is the system reliability if the disks are arranged in a RAID 10 structure where each disk of type A is paired with a disk of type B holding the same data?

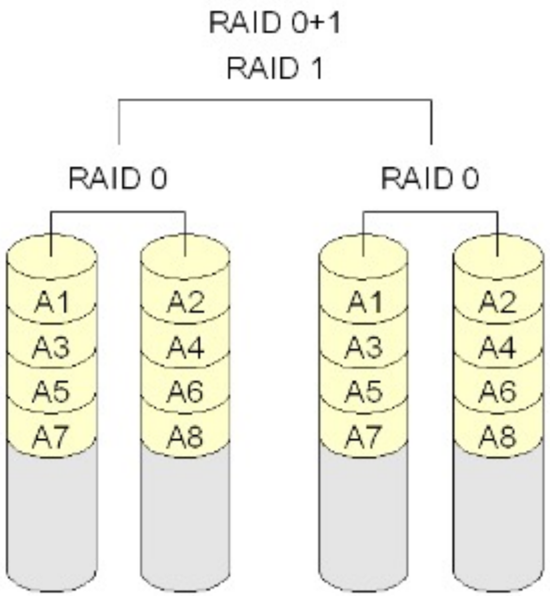
- $R_{sys} = \prod_{i=1}^{5}[1 - (1 - R_A)(1 - R_B)]$

- R_{sys} = [1-(1-0.5)*(1-0.75)]^5 = 0.5129

Pairing two types of disks makes a good question to test understanding. In practice ….

RAID 01

- Mirror of stripes: Complete RAID0 is duplicated.

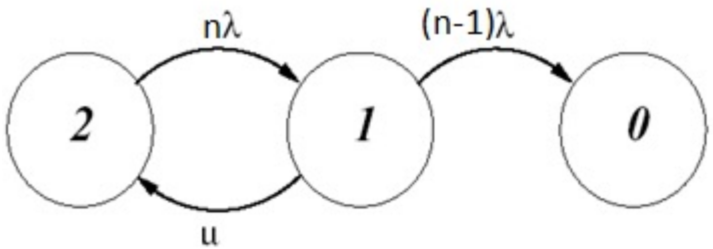


$$R_{sys} = [1 - (1 - \prod_{i=1}^{ns} R_i)^2]$$

- Ex: 3 year disk reliability = 0.9 for 100% duty cycle. ns = 6 for each of the two sets,
- $R_{sys} = 0.78$

RAID 01: redundancy at higher level

RAID4 - MTTDL Calculation



- RAID 4/5: data is lost if the second disk fails before the first failed (any one of n) could be rebuilt.

$$MTTDL = \frac{(2n-1)\lambda + \mu}{n(n-1)\lambda^2} \approx \frac{\mu}{n(n-1)\lambda^2}$$

- Detailed MTTDL calculators are available on the web.

Terminology

- Check-pointing: **saving part of the *process state***
  - Registers affected
  - Context
  - Part of the state (registers, memory) affected by next process segment
  - Entire data base etc.
- Rollback: **reestablishing a state of the process**
- Audit Trail: **chronological record of all transactions**
- Retry: **reexecution after rollback (inc. audit-trail reprocessing)**

Analysis of Overhead

- Assumptions:
  - ▹ Fault arrival rate: λ, interchkpt time: T
  - ▹ Additional retry time $\propto$ duration from last chkpt to error
  - ▹ No inputs/errors during chkpt/rollback
- Overhead per T:
  - ▹ $O(T) = F + V(T)$

    where F: fixed time to save/load chkpt info

    $V(T)$: Average retry time
  - ▹ Average retry time:

    $V(T) = P\{\text{error during } T\}.\text{avg error overhead}$

    $= \lambda T(F + k\frac{T}{2})$

    where k is utilization factor. Note overhead includes time lost due to error and time to rollback.

Justification?

Why T/2?

Analysis of Overhead (2)

- Hence fractional overhead $\rho(T)$:

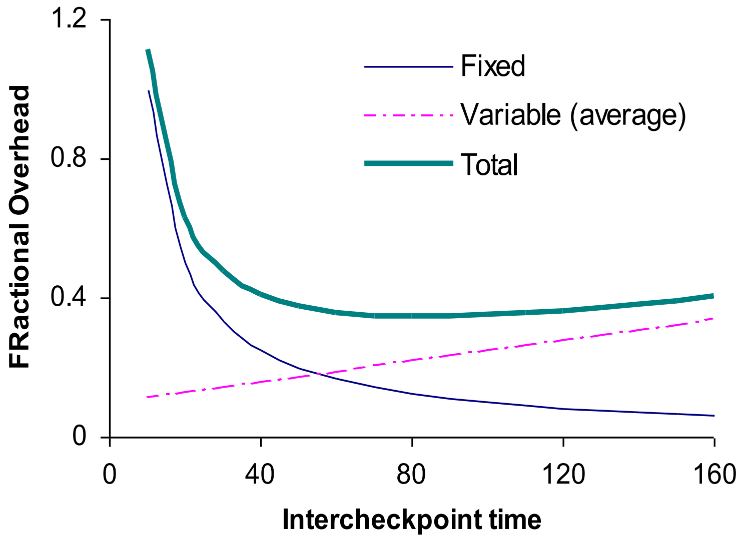
$$\rho(T) = \frac{O(T)}{T} = \frac{F}{T} + \lambda F + \frac{\lambda k}{2} T$$

Minimum occurs at

$$\frac{d\rho}{dT} = -\frac{F}{T^2} + \frac{\lambda k}{2} = 0$$

$$\therefore T_{opt} = \sqrt{\frac{2F}{\lambda k}}$$

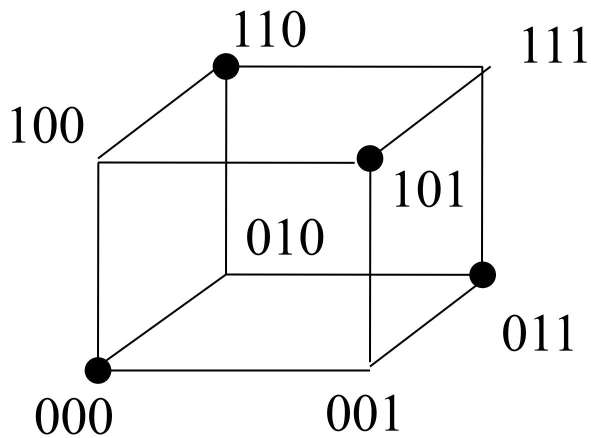
$$\text{Note} : k = \frac{\text{transaction arrival rate}}{\text{transaction processing rate}}$$



Ex: λ =0.01, k=0.3, F=10

yields T_{opt}=81.6 (above)

Detection Capability



Ex 1: consider code:

000
011
101
110

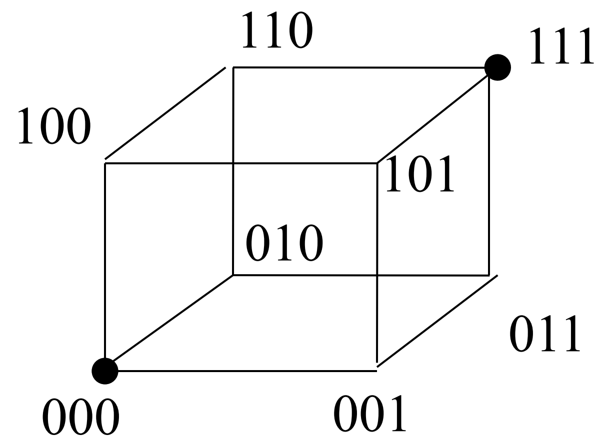
- All single bit errors result in non-code words. Thus all single-bit errors are detectable.
- **Error detection capability**: min Hamming dist d_{min}, p: number of errors that can be detected

$$p+1 \leq d_{min} \quad \text{or} \quad p_{max} = d_{min} - 1$$

Errors Correction Capability

Ex 2: Consider a code

000
111



- Assume single-bit errors are more likely than 2-bit errors.
- In Ex 2 all single bit errors can be corrected. All 2 bit errors can be detected.
- **Error correction capability**: t: number of errors that can be corrected:

$$2t+1 \leq d_{min} \qquad \text{or} \qquad t_{max}=\lfloor (d_{min}-1)/2 \rfloor$$

Proof?

Parity Check Matrix: Ex

$$v = (1 \quad 0 \quad 1) \left[\begin{array}{ccc:cc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{array}\right] \qquad v = (1 \quad 0 \quad 1 \quad 0 \quad 1)$$

$$H = \left[\begin{array}{ccc:cc} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

$v.H^t$ is

$$(1 \quad 0 \quad 1 \quad \underset{\Uparrow\, 0}{0} \quad 1) \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = (0 \quad \underset{\Uparrow\, 1}{0})$$

Systematic Cyclic Codes

- Ex: $G(x)=x^4+x^3+x^2+1$ n-k=4, n=7

message	$x^4M(x)$	C(x)	codeword
000	0(00 000)	0(0000)	000 0000
110	x^6+x^5 (1100000)	X^3+1(1001)	110 1001
111	$x^6+x^5+x^4$(1110000)	x^2(0100)	111 0100

- An error-free codeword divided by generator polynomial will give remainder 0.

Risk as a composite measure

Formal definition:

- Risk due to an adverse event e_i

  $Risk_i$ = $Likelihood_i$ x $Impact_i$

- Sometimes likelihood is split in two factors

  $Likelihood_i$ = P{$hole_i$ present}.

  P{exploitation|$hole_i$ present}

- A specific time-frame, perhaps a year, is presumed for the likelihood.

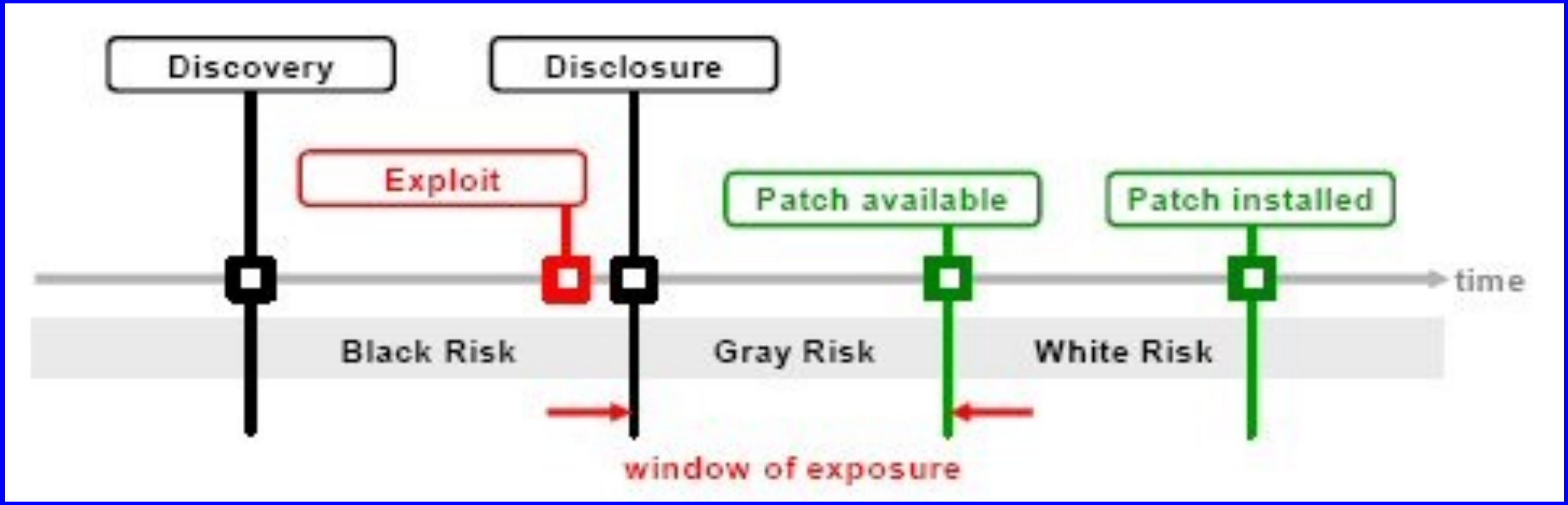
In classical risk literature, the internal component of Likelihood is termed “Vulnerability” and external “Threat”. Both are probabilities. There the term “vulnerability” does not mean a security bug, as in computer security.

Likelihood & Impact scales

- Quantitative or descriptive levels
  - Number of levels may depend on resolution achievable
- Scale: Logarithmic, Linear or combined
- Risk = Likelihood x Impact
  - Log(Risk) = Log(Likelihood) + Log(Impact)
- If “Score” is proportional to Log value
  - Risk score = Likelihood score + Impact score
  - Adding scores valid if scores represent logarithmic values.

Vulnerability Lifecycle

Vulnerabilities: “defect which enables an attacker to bypass security measures” [Schultz et al]



Exploit code (“exploit”) : usually available after disclosure

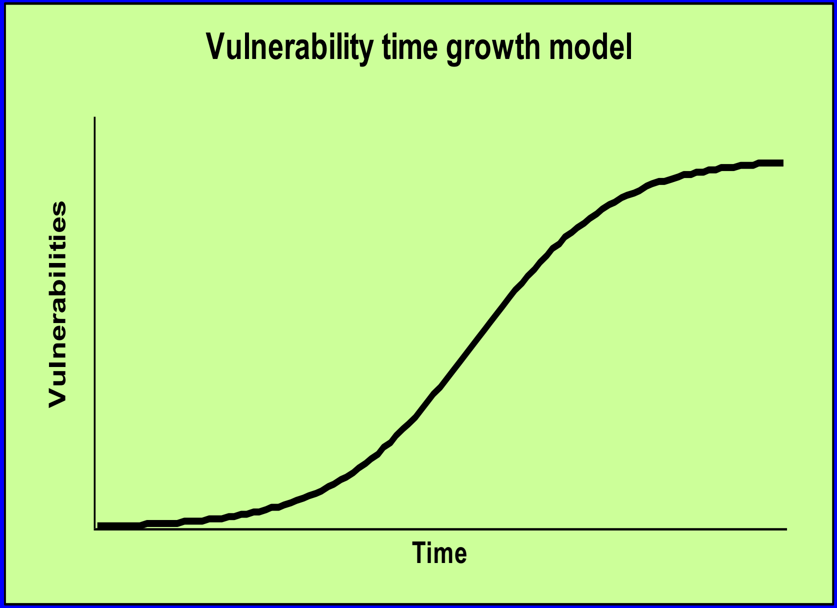
Time–vulnerability Discovery model

3 phase model S-shaped model.

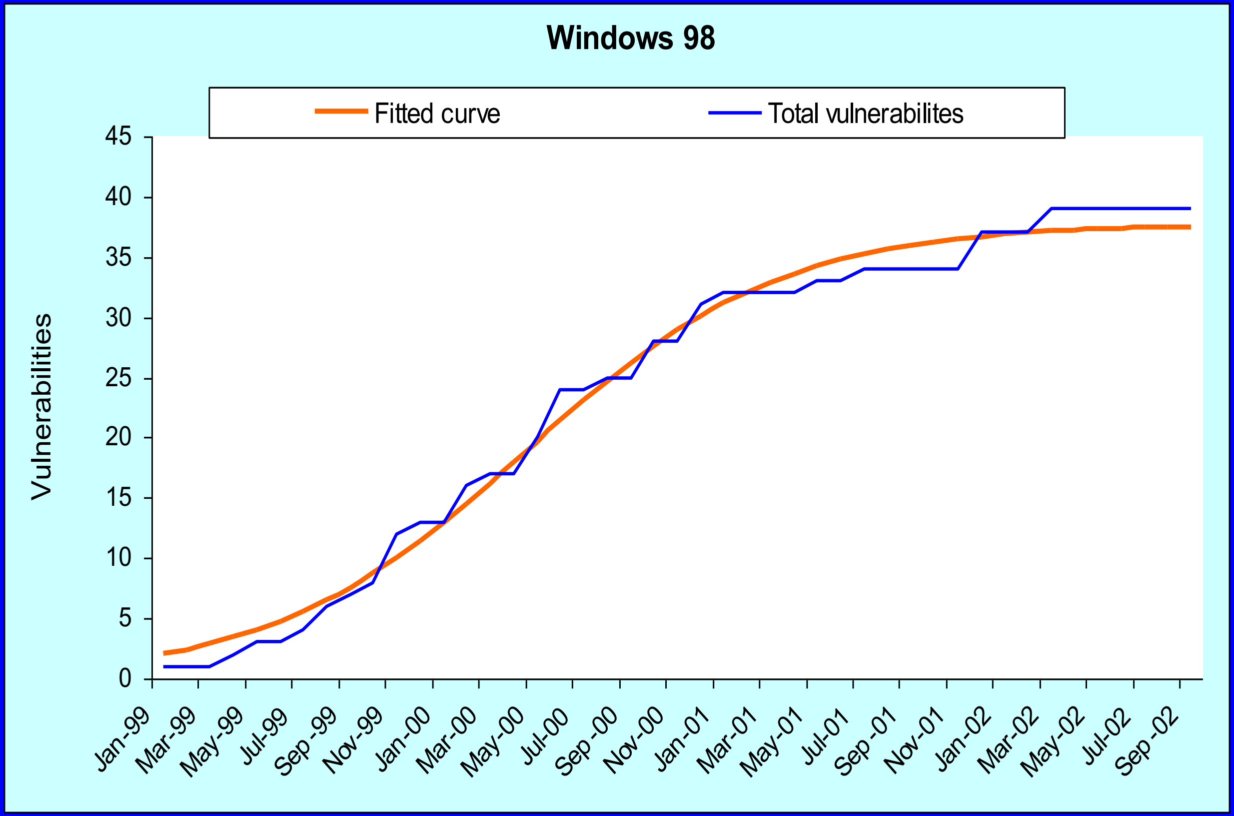
- Phase 1:
  - Installed base –low.
- Phase 2:
  - Installed base–higher and growing/stable.
- Phase 3:
  - Installed base–dropping.

$$\frac{dy}{dt} = Ay(B - y)$$

$$y = \frac{B}{BCe^{-ABt} + 1}$$



Time–based model: Windows 98



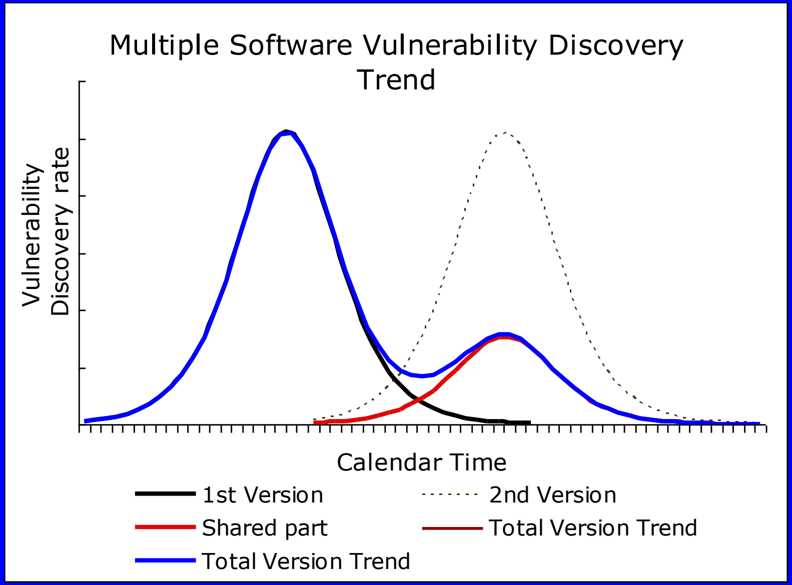
	Windows 98
A	0.004873
B	37.7328
C	0.5543
X^2	7.365
$X^2_{critial}$	60.481
P-value	$1- 7.6 \times 10^{-11}$

Vulnerability density and defect density

- **Vulnerability densities**: 95/98: 0.003-0.004 NT/2000/XP: 0.01-0.02
- V_{KD}/D_{KD}: 0.68-1.62% about 1%

System	MSLOC	Known Defects (1000s)	D_{KD} (/Kloc)	Known Vulner - abilies	V_{KD} (/Kloc)	Ratio V_{KD} /D_{KD}
Win 95	15	5	0.33	46	0.0031	0.92%
NT 4.0	16	10	0.625	162	0.0101	1.62%
Win 98	18	10	0.556	84	0.0047	0.84%
Win2000	35	63	1.8	508	0.0145	0.81%
Win XP	40	106.5*	2.66*	728	0.0182	0.68%*

Multi-version Vulnerability Discovery Model



$$\Omega(t) = \frac{B}{BCe^{-ABt}+1} + \alpha\frac{B'}{B'C'e^{-A'B'(t-\varepsilon)}+1}$$

	Previous Version	Next Version	Shared Code Ratio α
Apache	1.3.24 (3-21-2002)	2.0.35 (4-6-2002)	20.16%
Mysql	4.1.1 (12-1-2003)	5.0.0 (12-22-2003)	83.52%

Seasonal Index

Seasonal Index Values

	WinNT	IIS	IE
Jan	1.95	1.36	0.41
Feb	0.93	0.91	0.86
Mar	0.56	0.81	0.59
Apr	0.60	1.00	0.78
May	0.84	1.09	1.11
Jun	1.12	1.55	1.22
Jul	0.84	1.00	1.43
Aug	0.79	0.64	1.14
Sep	0.51	0.55	0.70
Oct	0.65	0.55	0.54
Nov	0.84	0.64	0.70
Dec	2.37	2.55	2.51
χ_c^2	19.68	19.68	19.68
χ_s^2	78.37	46	130.43
p-value	3.04e-12	3.23e-6	1.42e-6

- Seasonal index: measures how much the average for a particular period tends to be above (or below) the expected value
- H_0: no seasonality is present. We will evaluate it using the monthly seasonal index values given by [4]:

$$s_i = \frac{d_i}{d}$$

where, s_i is the seasonal index for i^{th} month, d_i is the mean value of i^{th} month, d is a grand average

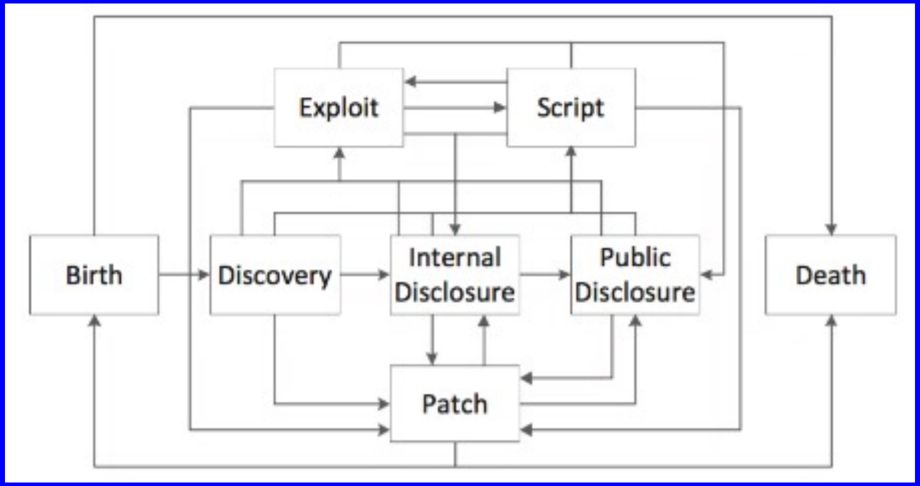
[4] Hossein Arsham. Time-Critical Decision Making for Business Administration. Available: http://home.ubalt. edu/ntsbarsh/Business-stat/stat-data/Forecast.htm#rseasonindx

CVSS Base metric: Observation

- *Exploitability sub-score* - measure of Likelihood of exploitation of the vulnerability.
- *Impact sub-score* - a measure of Impact.
- *CVSS Base Score* is a form of a risk measure. They could have computed *CVSS Base Score* by simply multiplying the *Exploitability* and the *Impact sub-scores*. It would result in a similar distribution of score with somewhat better resolution.
- *CVSS Base Score* for prioritizing vulnerabilities. Base score 7.0-10.0 **critical**, 4.0-6.9 **major**, 0-3.9 **minor**.
- The CVSS Base Score formula was determined by a committee and not formally derived or explained.

Likelihood of Individual Vulnerabilities Discovery

- **Ease of discovery**
  - Human factor (skills, time, effort, etc.), Discovery technique, Time
- Time:



- Apache HTTP server
- CVE-2012-0031, **(01/18/2012)**
- V. 1.3.0→1998-06-06

Time to Discovery = Discovery Time Date – First Effected version Release Date

Types of Vulnerability Markets