Useful Software Reliability Modeling Practices in Industry Environments

Daniel R. Jeske
University of California
Riverside, CA
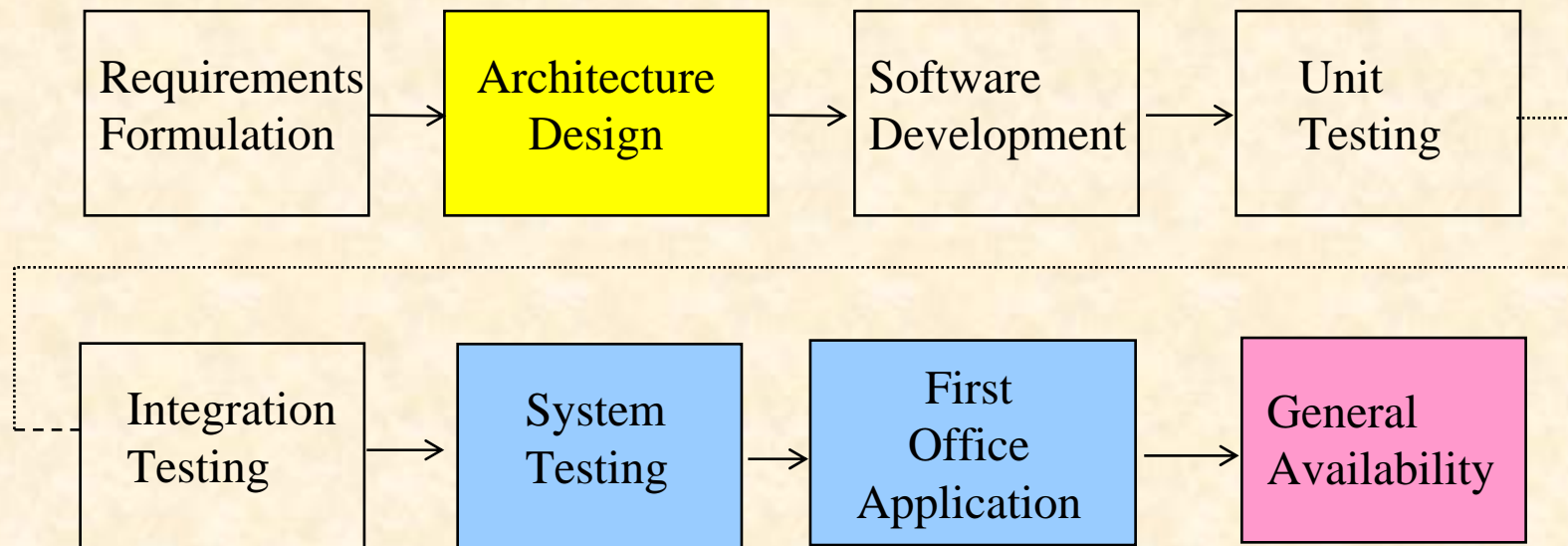
1. *Introduction and Context*
2. *Architecture-Based Software Reliability Models*
3. *Software Reliability Growth Model*
4. *Case Studies*
5. *Summary*

U.S. Army Conference on Applied Statistics
October 29-31, 2003

When are Software Reliability Models Typically Applied?

Architecture-Based Reliability Models
.....decisions are being made as to how to design reliability into the system

Requirements Formulation	→	Architecture Design	→	Software Development	→	Unit Testing

Integration Testing	→	System Testing	→	First Office Application	→	General Availability

Software Reliability Growth Models
......reliability is quantified and influences the release decision

Software Reliability Growth Models
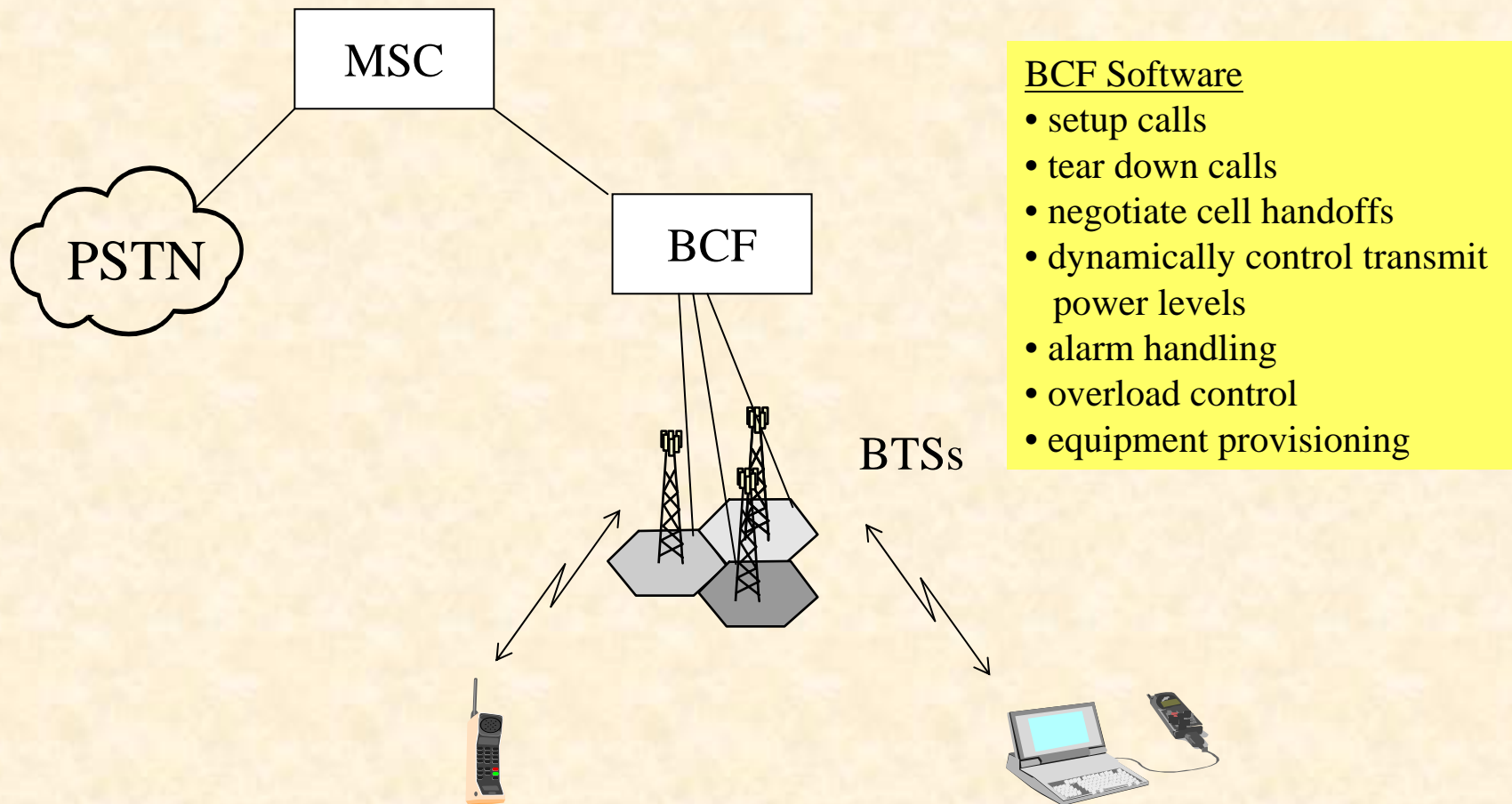*......reliability predictions are verified*
......parameters useful for modeling reliability of next release are estimated

2

Architecture-Based Software Reliability Models
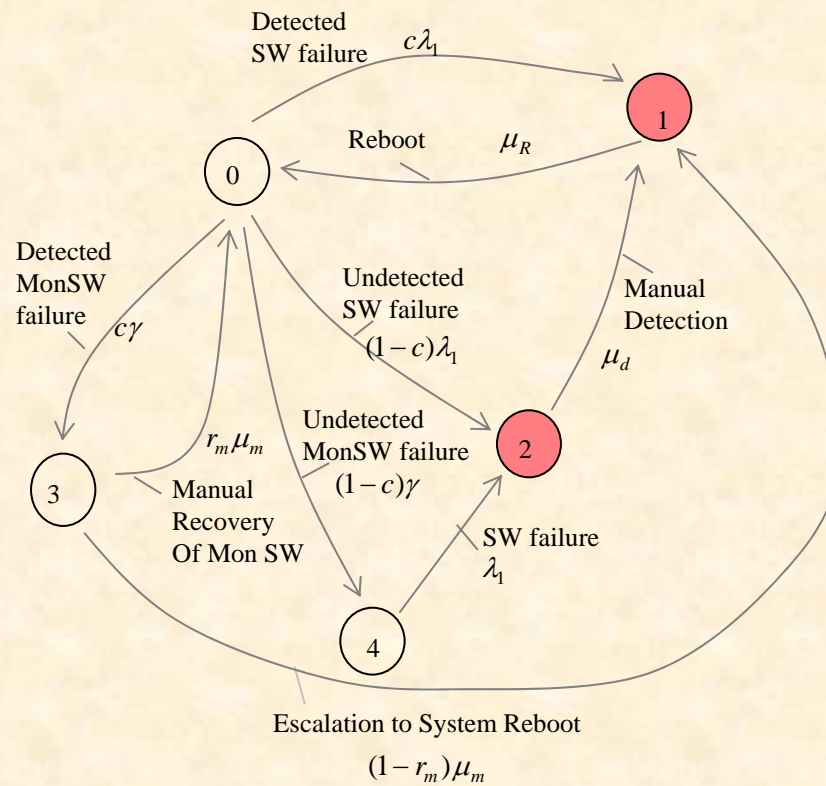
Questions to Answer

- What type of software redundancy, if any, is needed in order to achieve the reliability target?

- How fast do fault recovery times need to be?

- Should software processes try to restart before a failover is attempted?

- How thorough do system fault diagnostics need to be?

- What is the required software failure rate?

Case Study: GSM Wireless System

MSC

PSTN

BCF

BTSs

BCF Software
- setup calls
- tear down calls
- negotiate cell handoffs
- dynamically control transmit
  power levels
- alarm handling
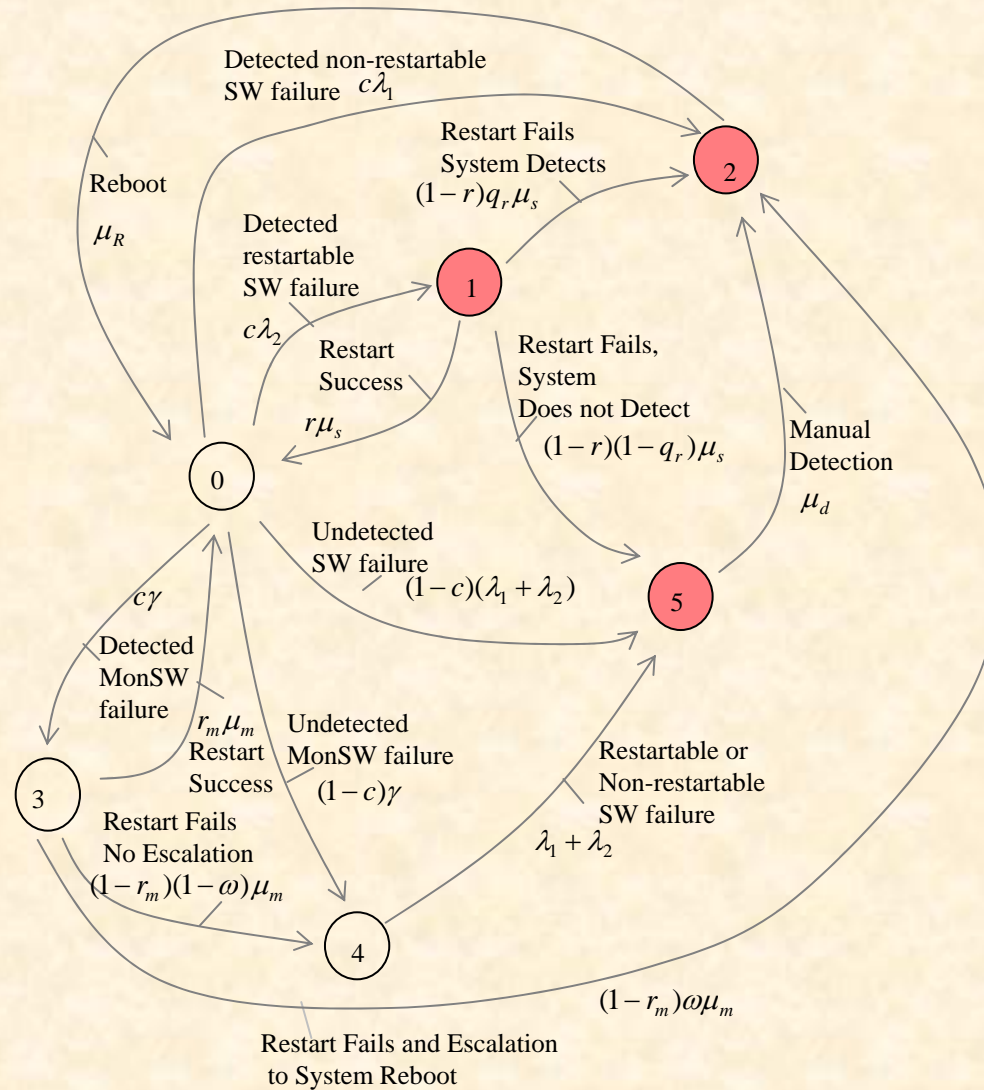- overload control
- equipment provisioning

BCF   - Base Station Controller Frame
BTS   - Base Transceiver Station
MSC   - Mobile Switching Center
PSTN   - Public Switched Telephone Network

Simplex Design Without Restart Feature

Detected SW failure $c\lambda_1$

Reboot μ_R

Detected MonSW failure $c\gamma$

Undetected SW failure $(1-c)\lambda_1$

Manual Detection μ_d

$r_m\mu_m$

Undetected MonSW failure $(1-c)\gamma$

Manual Recovery Of Mon SW

SW failure λ_1

Escalation to System Reboot $(1-r_m)\mu_m$

Detected non-restartable
SW failure $c\lambda_1$

Restart Fails
System Detects
$(1-r)q_r\mu_s$

2

Reboot

μ_R

Detected
restartable
SW failure

1

$c\lambda_2$

Restart
Success

$r\mu_s$

Restart Fails,
System
Does not Detect
$(1-r)(1-q_r)\mu_s$

Manual
Detection

μ_d

0

Undetected
SW failure
$(1-c)(\lambda_1+\lambda_2)$

5

$c\gamma$

Detected
MonSW
failure

$r_m\mu_m$
Restart
Success

Undetected
MonSW failure
$(1-c)\gamma$

Restartable or
Non-restartable
SW failure

$\lambda_1+\lambda_2$

3

Restart Fails
No Escalation
$(1-r_m)(1-\omega)\mu_m$

4

$(1-r_m)\omega\mu_m$

Restart Fails and Escalation
to System Reboot

Active-Standby Software Design

Restart Fails $(1-r_m)\mu_m$

Non-restartable SW failure on Active side λ_2

Restartable failure on Active side λ_1

Restart Success $r_m\mu_m$

λ_3 MonSW failure on Active side

Detected non-restartable SW failure $c\lambda_2$

Restart Fails, System Does not Detect $(1-r)(1-q_r)\mu_s$

Manual Detection $\mu_d + \lambda_3$

Restart Fails $(1-r)\mu_s$

MonSW failure on Active side

Restart Success $r\mu_s$

Restartable or Non-restartable Stdby SW failure $\lambda_{1s} + \lambda_{2s}$

Undetected Stdby MonSW failure $(1-c)\lambda_{3s}$

Restart Fails System Detects $(1-r)q_r\mu_s$

Detected restartable SW failure $c\lambda_1$

AutoFailover Fails, System Detects $(1-p)q_f\mu_f$

AutoFailover Fails, System Does not Detect $(1-p)(1-q_f)\mu_f$

Auto Failover Succeeds $p\mu_f$

Restart Success $r_m\mu_m$

Restartable Failure on Active side λ_1

Restart Success $r\mu_s$

Manual Detection μ_d

Manual Detection μ_d

Restart Success $r\mu_s$

Restart Fails $(1-r_m)\mu_m$

Restart Fails $(1-r)\mu_s$

Manual Detection $\mu_d + \lambda_{3s}$

λ_2 Non-restartable SW failure on Active side

Undetected Restartable or Non-restartable Stdby SW failure $(1-c)(\lambda_{1s} + \lambda_{2s})$

Reboot μ_R

Restart Success

Parallel Reboot μ_R

Reboot μ_R

Manual Failover Success $p_m\mu_{mf}$

Undetected SW failure $(1-c)(\lambda_1 + \lambda_2)$

μ_R Parallel Reboot

Manual Failover Fails $(1-p_m)\mu_{mf}$

Manual Detection $\mu_d + \lambda_3$

Detected MonSW failure $c\lambda_3$

$r_m\mu_m$ Restart Success

Undetected MonSW failure $(1-c)\lambda_3$

Manual Detection $\mu_d + \lambda_3$

λ_2 Non-restartable SW failure on Active side

MonSW failure on Active side λ_3

Restartable failure on Active side λ_1

Restart Success $r\mu_s$

$r_m\mu_m$ Restart Success

Restart Fails No Escalation $(1-r_m)(1-w)\mu_m$

Restartable or Non-restartable SW failure $\lambda_1 + \lambda_2$

AutoFailover Fails μ_f

Restart Fails $(1-r)\mu_s$

$(1-r_m)\mu_m$

Parallel Reboot μ_R

Restart Fails and Escalation to AutoFailover $(1-r_m)w\mu_m$

Feasibility Regions for Critical Parameters

Level Curves for 99.999% Availability
(*all other parameters are fixed*)

Software Failure Rate

- No Restart
- Restart
- Active-Standby

Software Fault Coverage Factor

Identifying Influential Parameters

Challenges

- Availability formula is straight-forward to obtain, but is complicated in that it is highly non-linear and it depends on numerous (23) parameters

- Analysis of derivatives, while also straight-forward is not particularly insightful, since they themselves depend on the same set of parameters.

- "Tornado Graphs" depends too much on the fixed values for the other parameters.

Approach

- Identify likely ranges for each parameter to define the domain of the availability function

- Uniformly sample N times from the domain to obtain $A_1, A_2, \ldots, A_N$

- Use an efficient second-order polynomial to approximate the availability formula

- Use the t-statistics as the associated measure of influence

Case Study – Continued

(Likely Ranges for Parameter Values in Active-Standby Architecture)

λ_1 Fails/yr	[0.1 , 1.0]	μ_{mf} Failovers/yr	26,280
λ_2 Fails/yr	[0.1 , 1.0]	μ_m Recoveries/yr	17,520
λ_3 Fails/yr	[0.1 , 0.5]	r	[0.9 , 0.99]
λ_{1s} Fails/yr	0	r_m	[0.9 , 0.99]
λ_{2s} Fails/yr	[0.1 , 0.5]	q_r	[0.9 , 0.99]
λ_{3s} Fails/yr	[0.1 , 0.5]	q_f	[0.9 , 0.99]
μ_s Restarts/yr	31,536,000	p	[0.9 , 0.99]
μ_f Failovers/yr	3,153,600	p_m	[0.9 , 0.99]
μ_r Reboots/yr	26,280	c	[0.9 , 0.99]
μ_d Detections/yr	[1095 , 2190]	w	[0.9 , 0.99]

Case Study – Continued
(*Efficient Second-Order Regression Model*)

$$D(\text{min/yr}) = 497 + 379\lambda_1 + 378\lambda_2 - 358\lambda_1 c - 356\lambda_2 c + 284(\mu_d/1000)c - 268(\mu_d/1000) - 461c$$
$$- 12.2\lambda_1(\mu_d/1000) + 319\lambda_3 - 12\lambda_2(\mu_d/1000) - 298\lambda_3 c$$
$$- 22p - 11.6\lambda_3(\mu_d/1000) - 11.7r - 9.6q_f - 9q_r - 5.8r_m - 5w - 2.1p_m$$

Adequacy of Fit
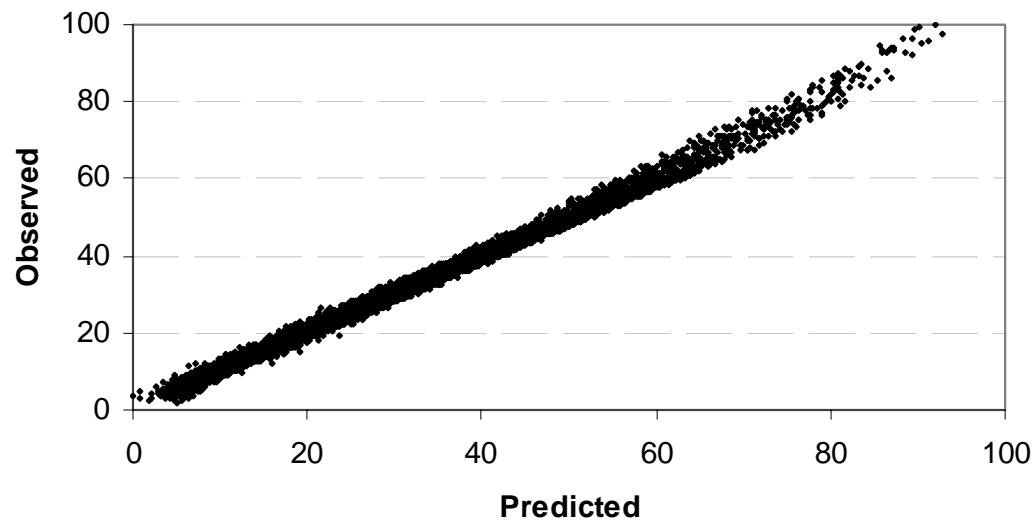
Interpretation

Influential Parameters
- Software failure rates on active side
- Silent failure detection time
- Coverage factor
- Cross-product of coverage factor with software failure rates and silent failure detection time

Weakly Influential Parameters
- All of the success probabilities

12

Software Reliability Growth Models (SRGMs)

SRGMs: Questions to Answer

- What would the user-perceived failure rate of the software be if
  if the software was to be released now?

- How much more testing is needed to achieve the reliability targets?

- How many faults exist in the code at the end of system test?

A Well-Known SRGM

Goel-Okumoto Model

$$M(t) = c(1 - e^{-dt})$$

Cumulative
Failures

T Cumulative Test Time (days)
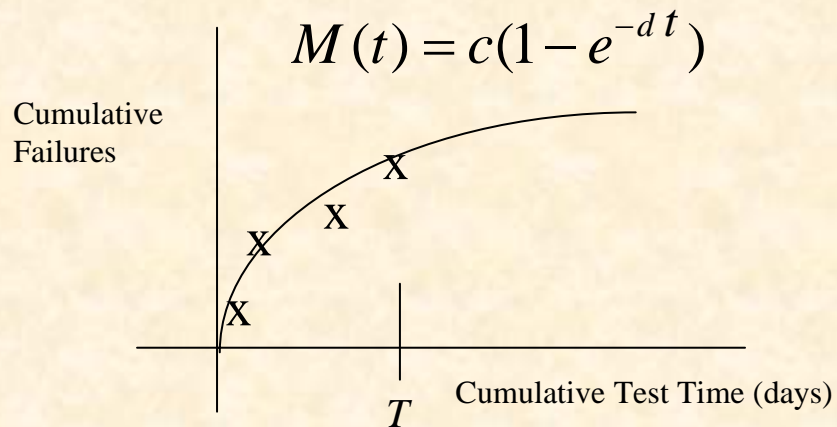
c = initial number of faults

d = average per-fault failure rate

t = cumulative test time, aggregated
   across all installations

T = current value of t

Failure Rate Estimate

$$\lambda(T) = M'(T)$$
$$= cde^{-dT}$$

Challenges With Applications

Traditional SRGMs are applied to system test data with the hope of obtaining an estimate of software failure rate that will be observed in the field. The following issues have to be taken into consideration:

- The usage profile in the field is typically very different than the testing profile

- Fault removal in <u>field</u> environments is not instantaneous

- Quality of software reliability data

Non-Instantaneous Fault Removal Times
In Field Environments

G-O Model

$$M(t) = c[1 - e^{-dt}]$$

$$\lambda(t) = cde^{-dt}$$

G-O Model with Imperfect Debugging

$$M(t) = \frac{c}{p}[1 - e^{-dpt}]$$

$$\lambda(t) = cde^{-dpt}$$

where:
c initial number of faults at time of field deployment
d average per fault failure rate in a field environment
p probability that a detected fault is successfully removed

Non-Instantaneous Fault Removal Times
In Field Environments

• Perfect debugging assumption is an acceptable assumption
  (based on thorough regression tests)

• We relate non-instantaneous fault removal to imperfect debugging

Under the imperfect debugging model:

E(# of occurrences of each fault)= $1/p$

Under the situation of non-instantaneous
fault removals, if μ denotes the mean time
to remove a fault and there are n systems
in the field

$$\frac{1}{p} = 1 + n\mu d$$

E(# of occurrences of each fault)= $1 + n\mu d$

Non-Instantaneous Fault Removal Times
In Field Environments

$$M(t) = c(1 + n\mu d)[1 - e^{-\frac{d}{1+n\mu d} \times t}]$$

$$\lambda(t) = cde^{-\frac{d}{1+n\mu d} \times t}$$

where: c initial number of faults at time of field deployment
          d average per fault failure rate in the field environment
          μ average time to remove a fault ($\mu = 0$ gives back the G-O model)
          n number of systems in the field
          t cumulative exposure time, aggregated across all installations

Field Failure Rate Prediction **if** Testing Profile Matches the Field Usage Profile

The number of initial faults at time of field deployment time is the same as the number of residual faults after testing has completed

The average per fault failure rate in the field environment is identical to the average per fault failure rate in the test environment

For the field failure rate model, we can use the G-O model replacing c with ce^{-dT} and adjusting for non-instantaneous removal times

$$\lambda(t) = (ce^{-dT})de^{-\frac{d}{1+n\mu d} \times t}$$

Field Failure Rate Prediction When Testing Profile Does Not Match the Field Usage Profile

1) Testing and field usage profiles drive the average per fault failure rate. Usually the failure rate of faults is smaller in field environments than in test environments

2) Define $K = d/d^*$, where d^* is the average per fault failure rate in the field environment

3) K is the "per fault failure rate" calibration factor

4) Estimate K from previous releases of the software, or from related projects

$$\lambda_{adj}(t) = (ce^{-dT})(d/K)e^{-\frac{d/K}{1+n\mu d/K} \times t}$$

Case Study - Continued

MSC

PSTN

BCF

BTSs
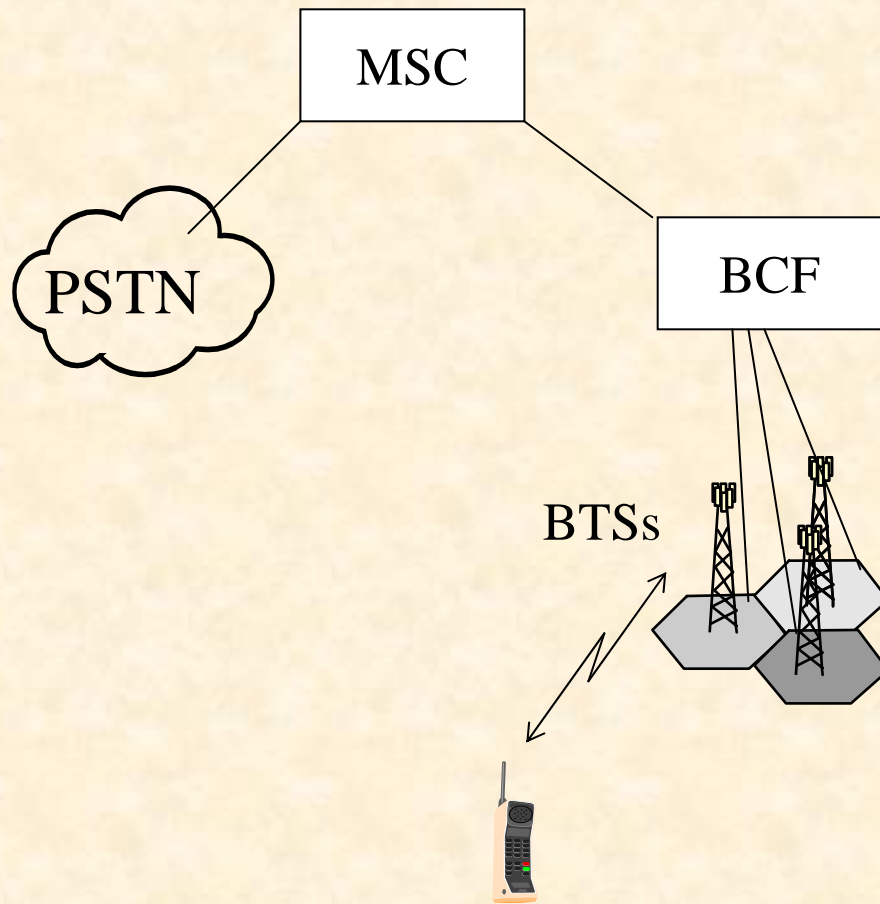
BTS Controller Software
- allocate radio resources
- negotiate hand-offs
- manage connections between BTS and BCF
- provision and maintenance

Alternative Architectures
- Simplex, no process restart capability
- Simplex, with restart capability
- Active-Standby, with fail-over capability

BCF   - Base Station Controller Frame
BTS   - Base Transceiver Station
MSC   - Mobile Switching Center
PSTN - Public Switched Telephone Network

Objective

Goals

Estimate the field failure rate of R3 BCF software which is currently in system test

Data Available

1) R3 Test Data in a non-operational profile environment

2) Field failure data for R1 and R2

Field Failure Data for R1

	System Days		Failures	
Month	Days	Cumulative	Month	Cumulative
1	1,249	1,249	4	4
2	3,472	4,721	6	10
3	4,065	8,786	4	14
4	4,883	13,669	3	17
5	5,425	19,094	6	23
6	5,656	24,750	1	24
7	7,549	32,299	2	26
8	8,295	40,594	4	30
9	8,882	49,476	1	31
10	6,120	55,596	0	31
11	2,465	58,061	1	32
12	527	58,588	1	33
13	45	58,633	0	33

Field Failure Data Analysis

R1 Field
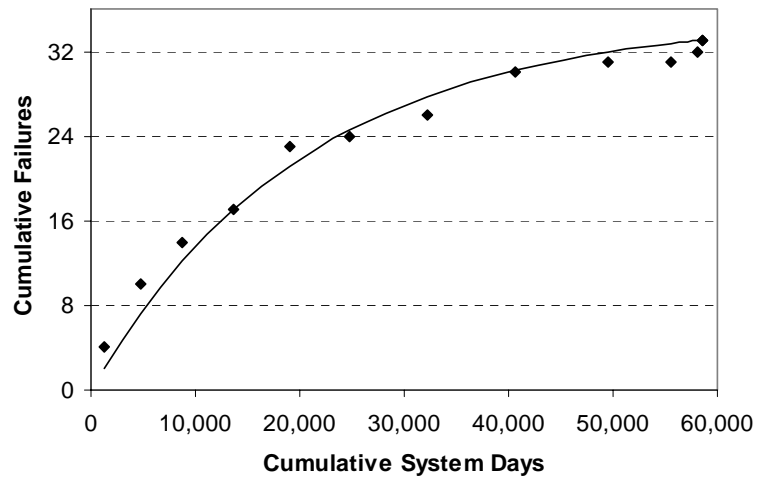
$T_1 = 58,633$ system-days (13 months , 167 systems)
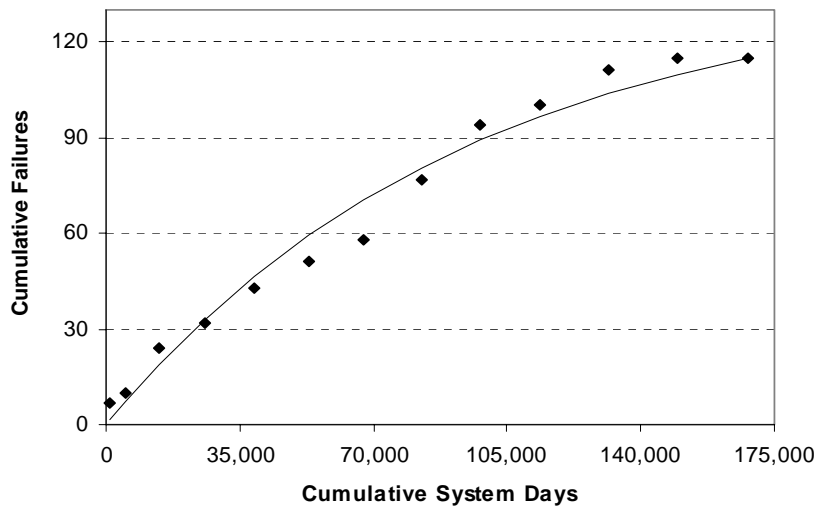
$\mu = 45$ days

$\hat{c}_1 = 22.3$ faults

$\hat{d}_1 = 0.0000746$ failures/day/fault

   (StdError: 3.07×10^{-5})

R2 Field

$T_2 = 167,900$ system-days (13 months , 370 systems)

$\mu = 45$ days

$\hat{c}_2 = 114.01$ faults

$\hat{d}_2 = 0.0000128$ failures/day/fault

   (StdError: 3.07×10^{-6})

R3 Test Data

Calendar Week	Number of Days of Each Software Installation											Cumulative System-Days of Testing	Cumulative Failures
	1	2	3	4	5	6	7	8	9	10	11		
1				5								5	5
2				4								9	6
3				4								13	13
4				5								18	13
5		5		5								28	22
6				5								33	24
7		5		5								43	29
8	5	5		5					5			63	34
9	5	5		5			5			5		88	40
10	5	5		5			5	5	5	5		123	46

↕ ↕

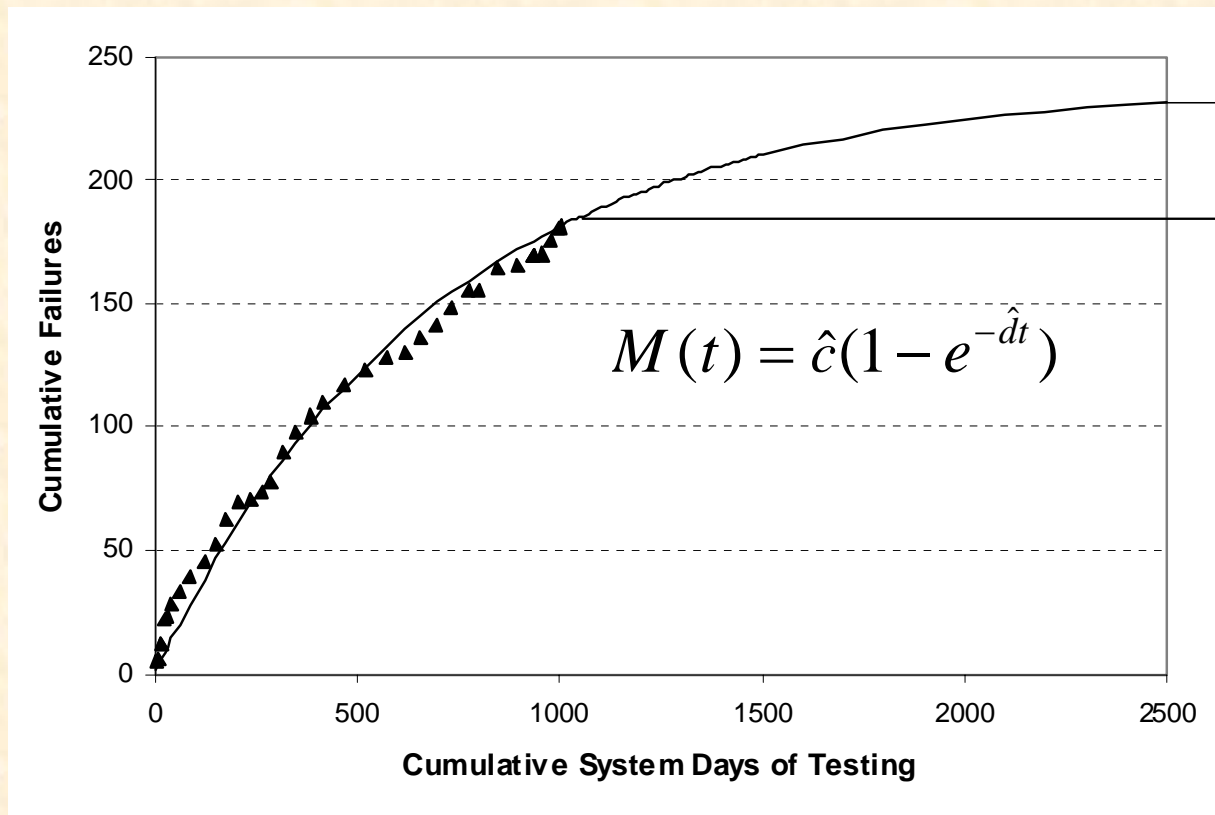| Calendar Week	Number of Days of Each Software Installation											Cumulative System-Days of Testing	Cumulative Failures
33				5		5	5				6	955	170
34				5		5	6				6	977	176
35				5		5	6				6	999	180
36							2					1001	181

As many as 11 frames were being used in parallel during system test.

'Failure' is defined as a severity 1, 2 or 3 MR.

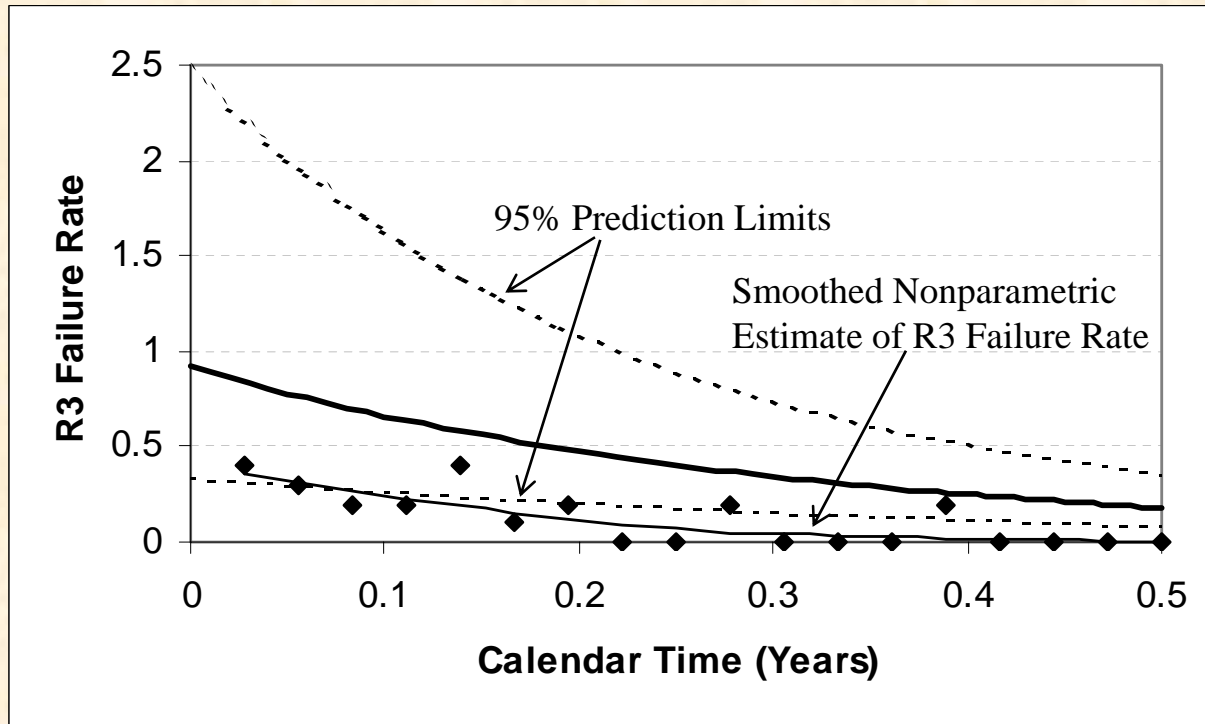$$M(t) = \hat{c}(1 - e^{-\hat{d}t})$$

$$\begin{pmatrix} \hat{c} \\ \hat{d} \end{pmatrix} = \begin{pmatrix} 238.3 \text{ faults} \\ 0.00142 \text{ fails/day/fault} \end{pmatrix}$$

$$K = \frac{\hat{d}}{(\hat{d}_1 + \hat{d}_2)/2} = 31$$

$$V\hat{a}r\begin{pmatrix} \hat{c} \\ \hat{d} \end{pmatrix} = \begin{bmatrix} 27.02^2 & -0.00552 \\ -0.00552 & 0.00027^2 \end{bmatrix}$$

27

R3 Field Failure Rate Prediction

Possible Sources for
Conservative Predictions
• Field Failures often under-reported
• Not all Severity 1/2/3 are failures
• Need to calibrate fault content

$$\hat{\lambda}_{adj}(t) \ = \ 58 \times (365 \times 0.00142/31) \times e^{-\frac{0.00142/31}{1+345\times45\times0.00142/31}\times365\times345\,t}$$

(fails/year, after t calendar-years of field exposure)

Needed Research

- SRGM fitting tools

- Small sample inference procedures

- Failure rate template

- Improved methods for linking SRGMs
  to architecture-based models