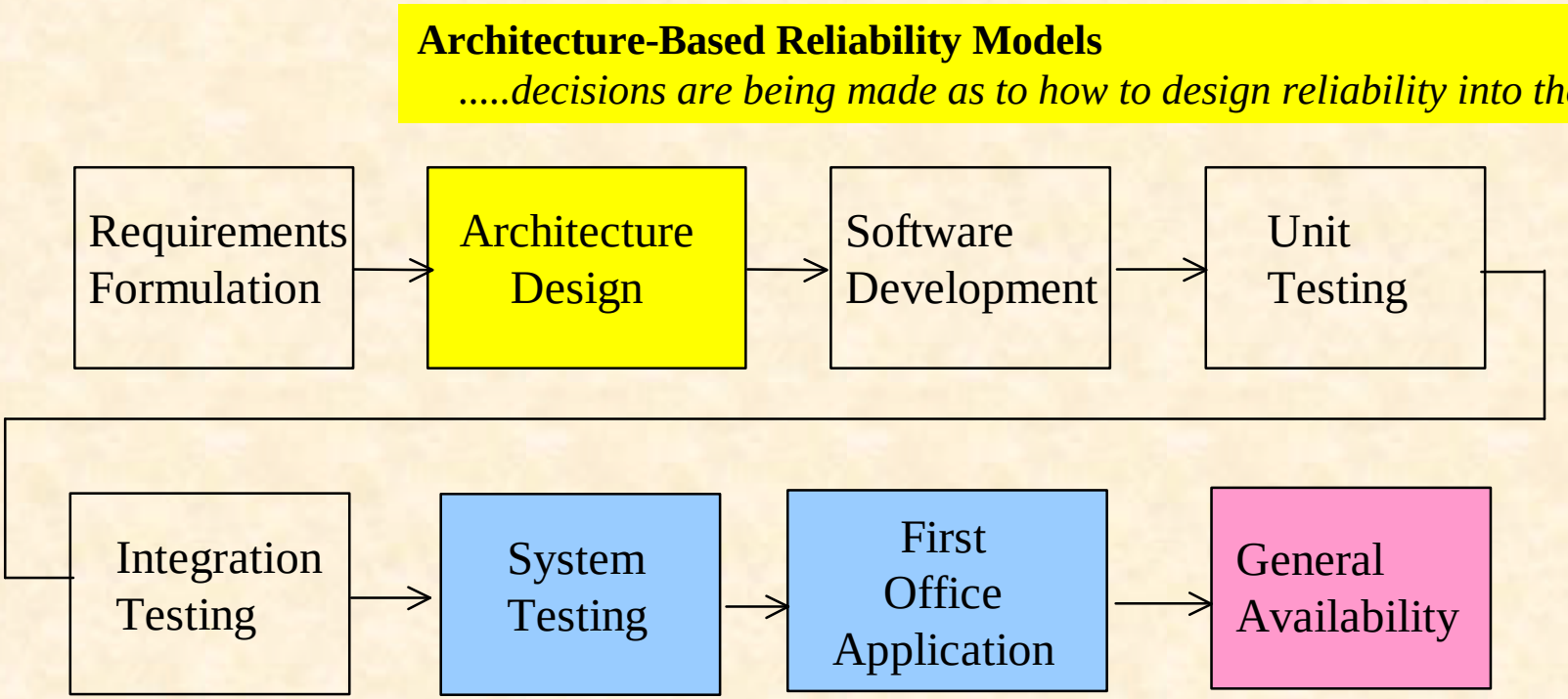
Some Successful Approaches to Software Reliability Modeling in Industry

Daniel R. Jeske and Xuemei Zhang
Bell Laboratories, Lucent Technologies
Holmdel, NJ

1. *Introduction and Context*
2. *Software Reliability Growth Models*
3. *Architecture-Based Software Reliability Models*
4. *Case Studies*
5. *Summary*

ASA Conference on Quality and Productivity
May 21, 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

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

Software Reliability Growth Models (SRGMs)

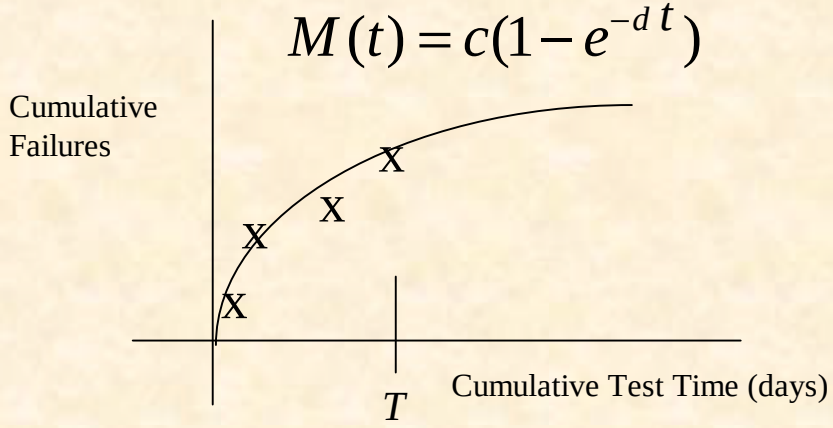
SRGMs: Questions to Answer

- 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

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



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

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

Motivation for Goel-Okumoto Model

The expected number of faults detected in a time interval $(t, t+\Delta t)$ is proportional to the number of faults remaining in the software at time t.

Thus,

$$M(t+\Delta t) - M(t) = d[c - M(t)]\Delta t$$

or equivalently,

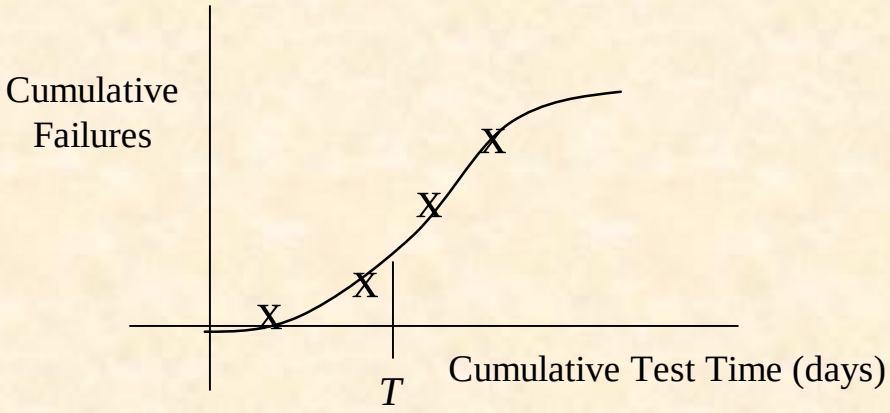
$$M'(t) + dM(t) = cd$$

implying,

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

Goel and Okumoto further assume that the number of faults observed in $(0, t)$ is Poisson$\{ M(t) \}$.

S-shaped Model



$$d(t) = \frac{d}{1 + b e^{-dt}}$$

$$M(t) = \frac{1}{(1 + be^{-dt})} \times c(1 - e^{-dt})$$

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 field environments is not instantaneous
- Quality of software reliability data

Non-Instantaneous Fault Removal Times In Field Environments

G-O Model

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

$$l(t) = cde^{-dt}$$

G-O Model with Imperfect Debugging

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

$$l(t) = cde^{-dpt}$$

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 m denotes the mean time to remove a fault and there are n systems in the field

E(# of occurrences of each fault)= $1+ nmd$

$$\frac{1}{p} = 1 + nmd$$

Non-Instantaneous Fault Removal Times In Field Environments

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

$$l(t) = cde^{-\frac{d}{1+nmd} \times t}$$

where:
- c initial number of faults at time of field deployment
- d average per fault failure rate in the field environment
- m average time to remove a fault ($m = 0$ 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

$$l(t) = (ce^{-dT})de^{-\frac{d}{1+nmd}\times t}$$

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

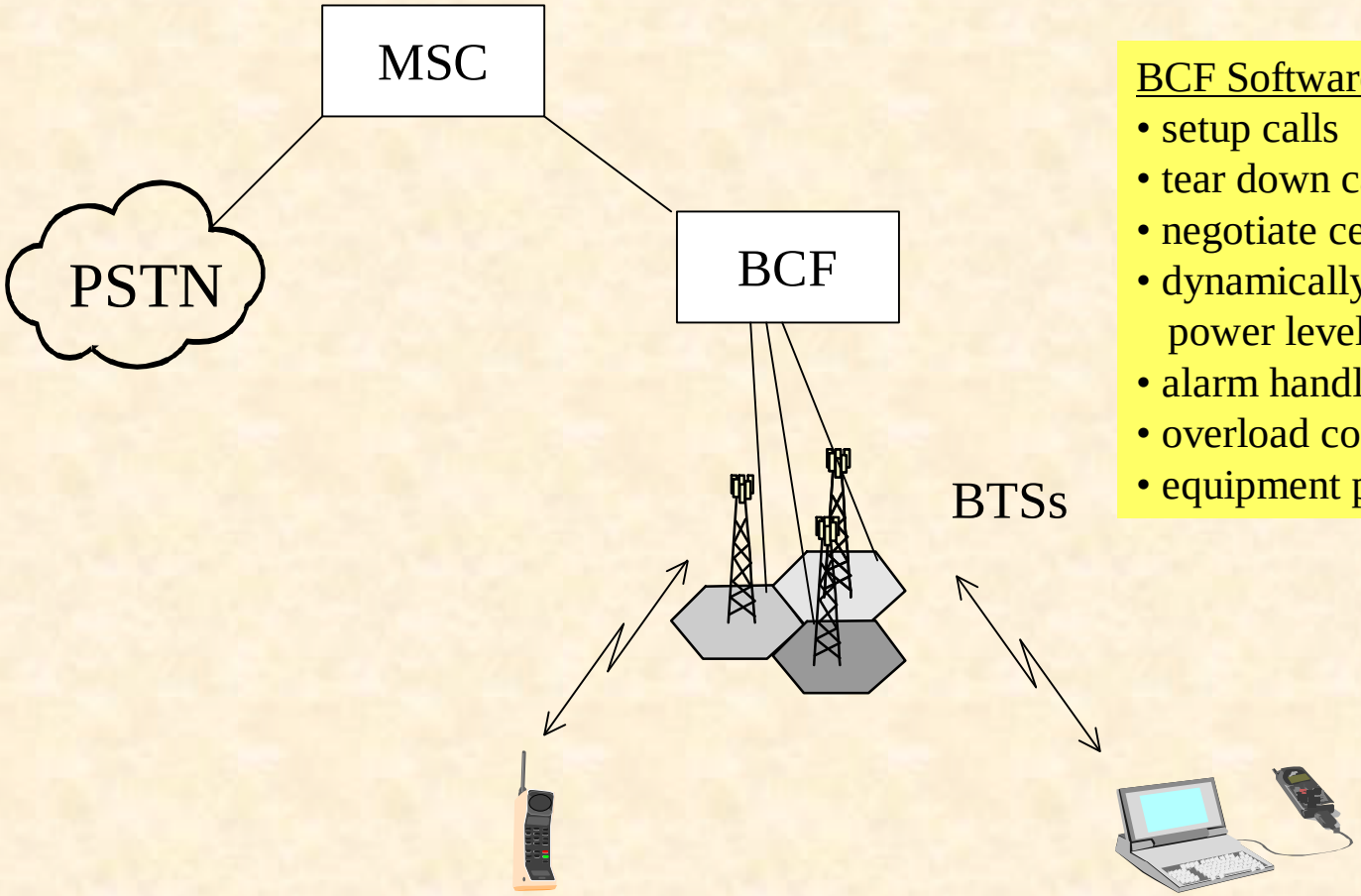
2) Define $K = d / d^*$, where d^* is the average per fault failure rate in the field environment

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

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

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

Case Study: GSM Wireless System



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

Objective

Goals

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

Data Available

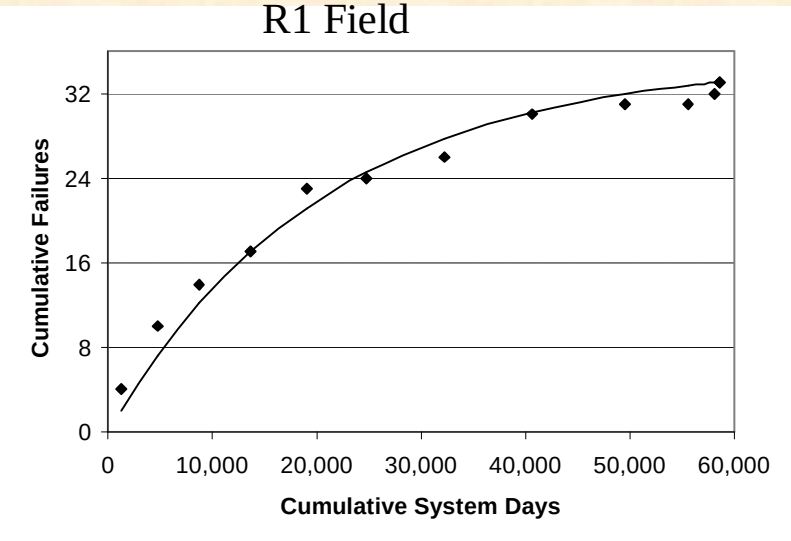
1) R3 Test Data in a non-operational profile environment

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



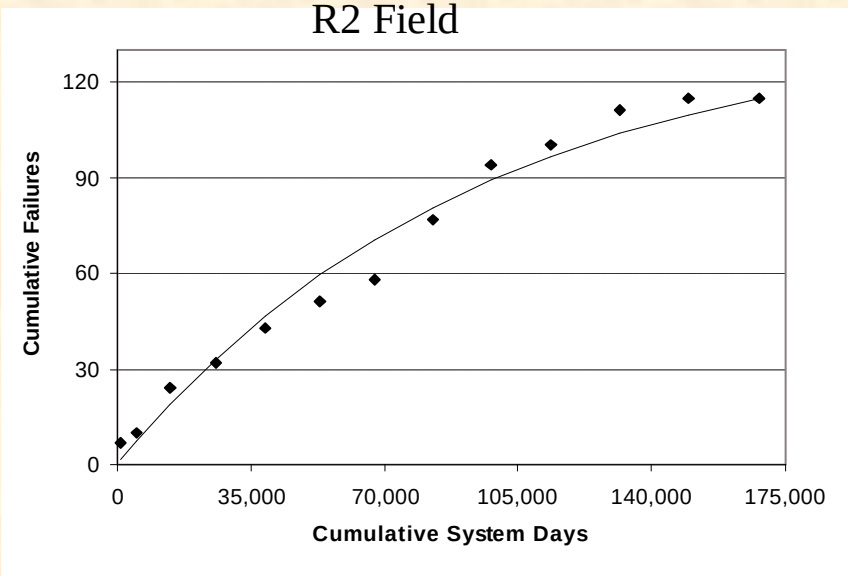
$T_1 = 58{,}633$ system-days (13 months , 167 systems)

$m = 45$ days

$\hat{c}_1 = 22.3$ faults

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

(StdError: 3.07×10^{-5})



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

$m = 45$ days

$\hat{c}_2 = 114.01$ faults

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

(StdError: 3.07×10^{-6})

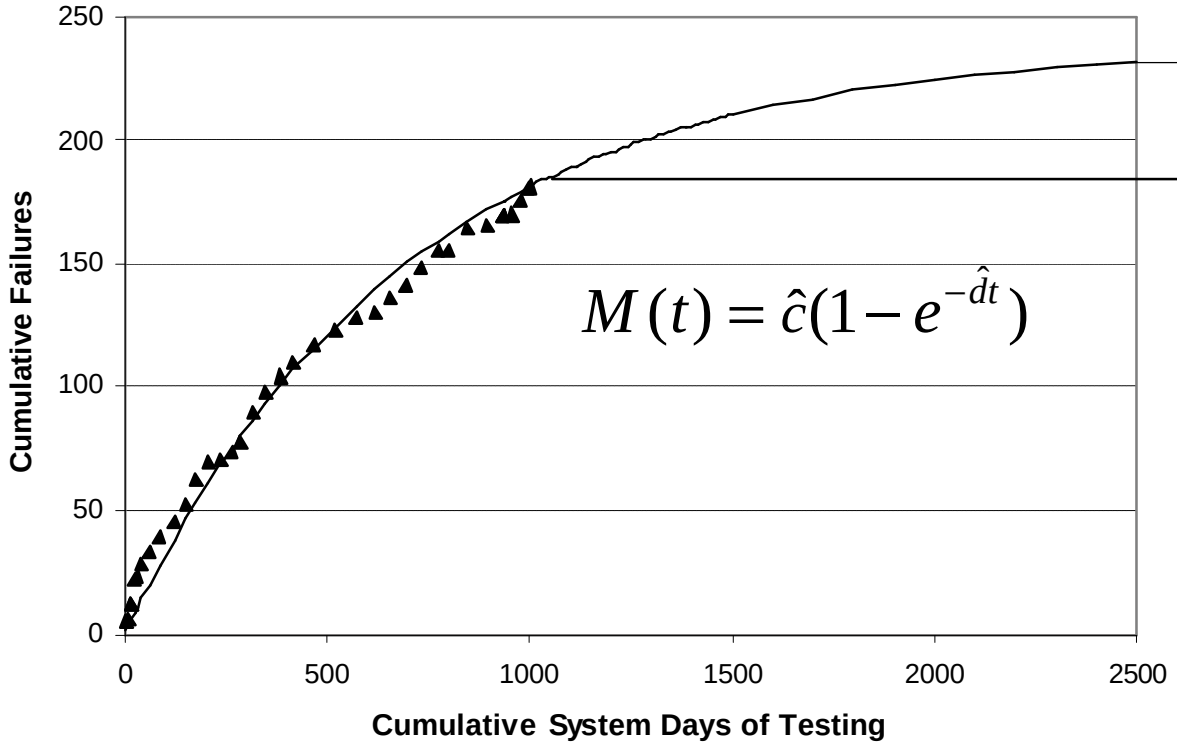
R3 Test Data

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
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.

G-O Model Fit to the R3 Test Data

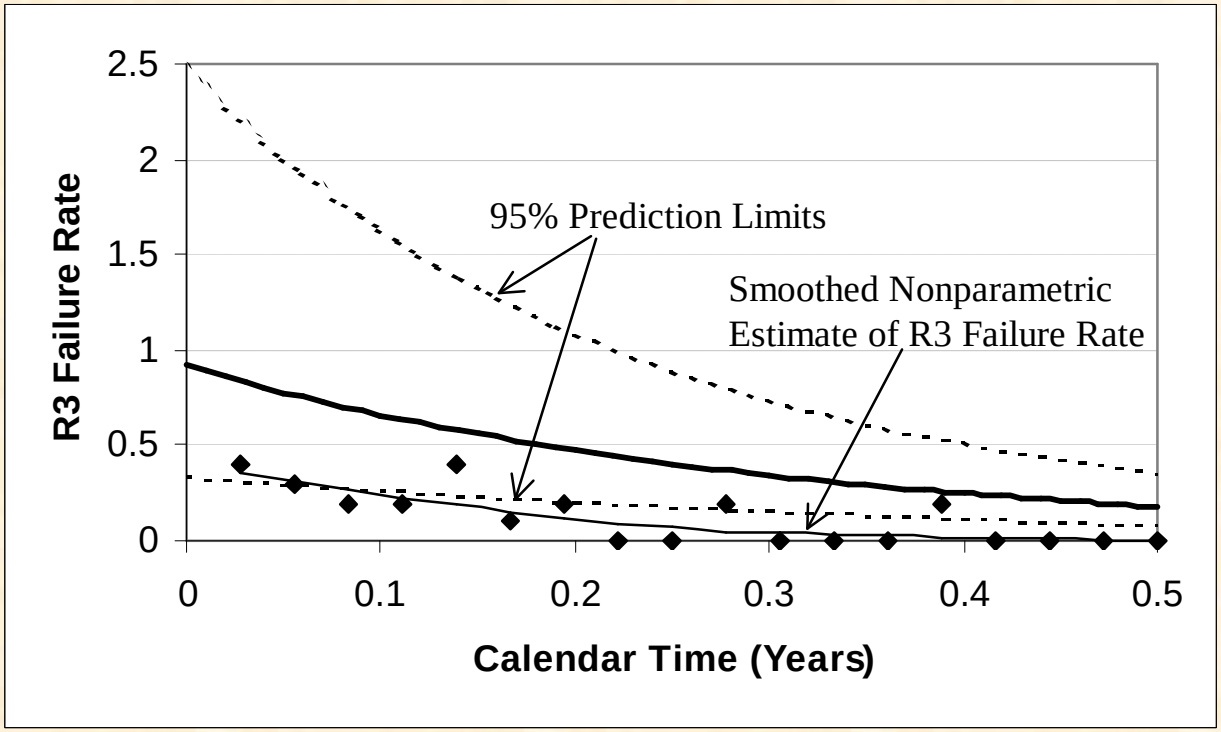


$$\begin{pmatrix} \hat{c} \\ \hat{d} \end{pmatrix} = \begin{pmatrix} 238.3 \text{ faults} \\ 0.00142 \text{ fails/day/fault} \end{pmatrix}$$

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

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

R3 Field Failure Rate Prediction



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

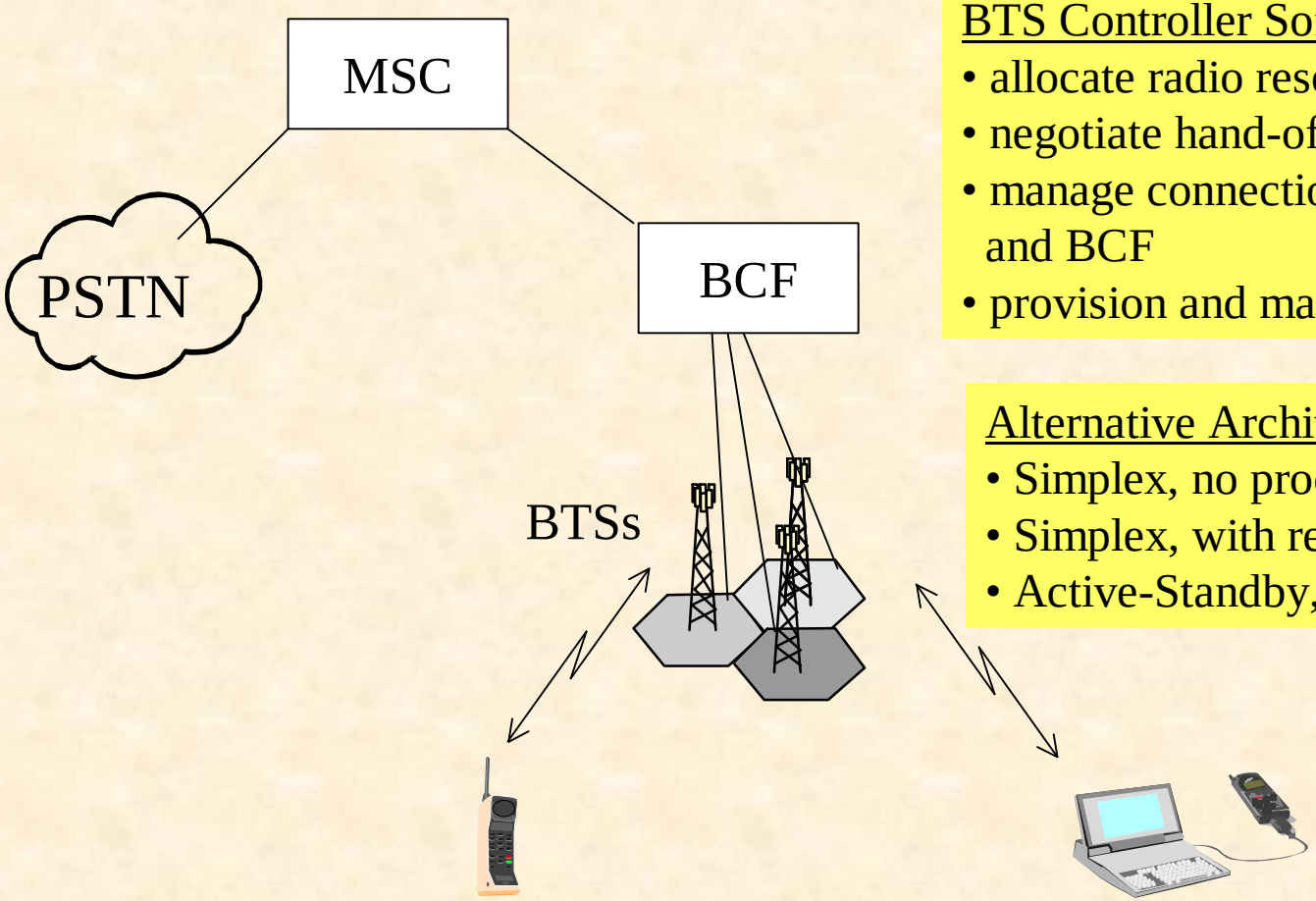
(fails/year, after t calendar-years of field exposure)

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 - Continued

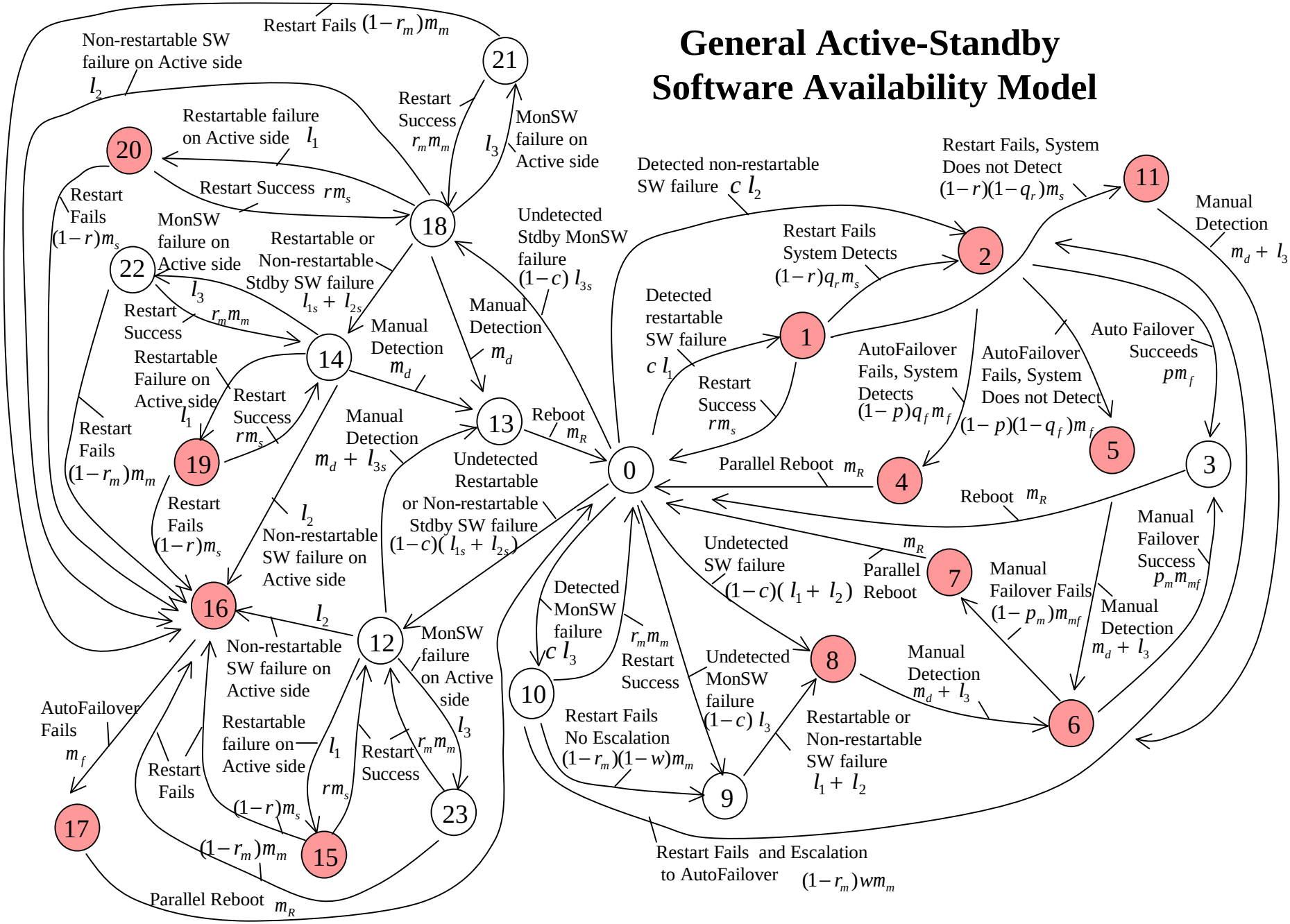


BTS Controller Software
- allocate radio resources
- negotiate hand-offs
- manage connections between BTS and BCF
- provision and maintenance

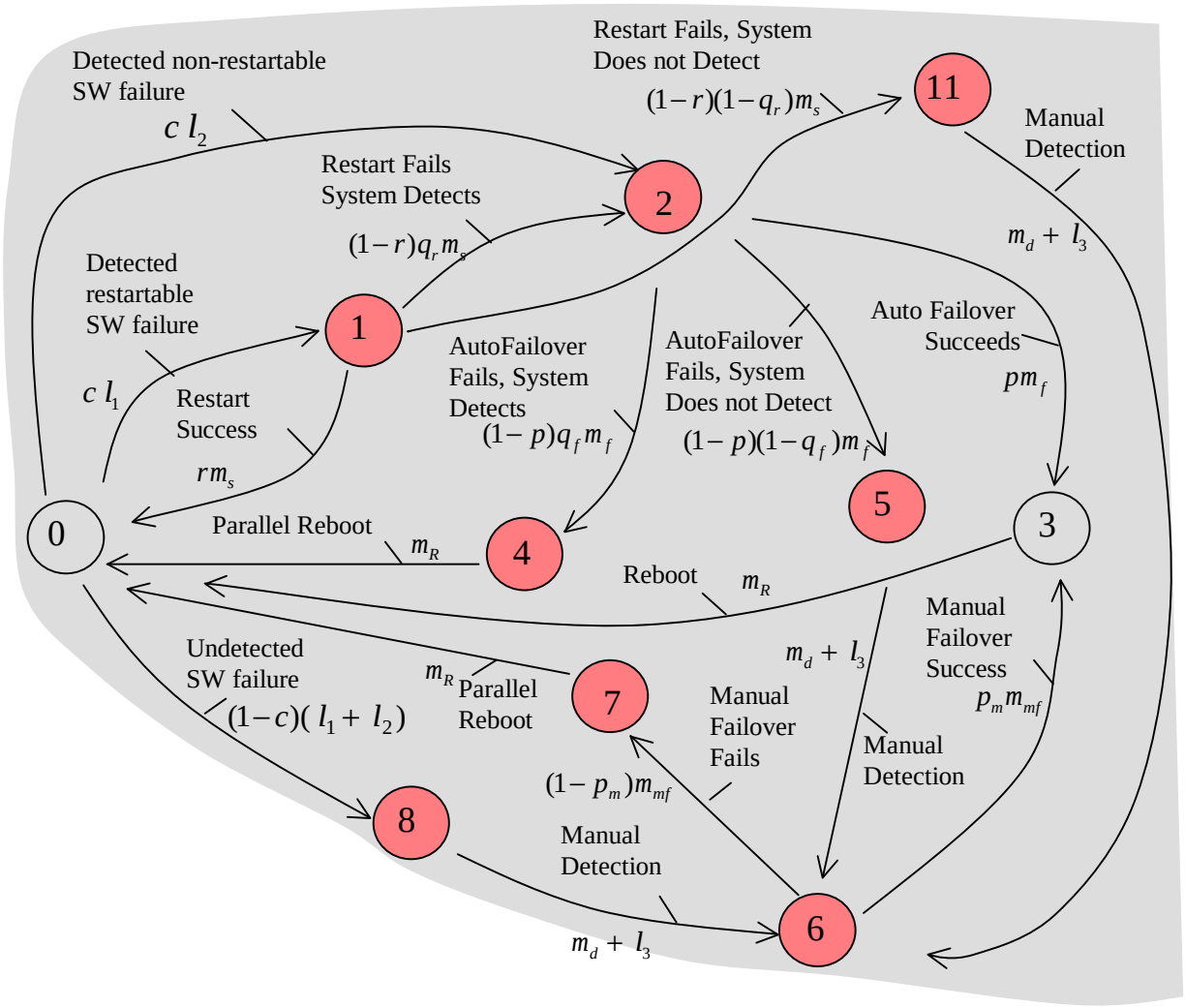
Alternative Architectures
- Simplex, no process restart capability
- Simplex, with restart capability
- Active-Standby, with fail-over capability

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

General Active-Standby Software Availability Model



Special Case: Cold Standby



Simplex Systems as Special Cases

Simplex System With No Restart Capability

$l_1 = 0$ (no restartable software)

$l_{1s} = l_{2s} = l_{3s} = 0$ (no standby side)

$r = 0$ and $m_s = \infty$ (no restart attempts)

$p = 0$ and $m_f = \infty$ (no automatic failover attempts)

$p_m = 0$ and $m_{mf} = \infty$ (no manual failover attempts)

$q_m = q_f = 1$ (no restart or failover attempts)

$r_m = 1$ and $m_m =$ manual recovery duration of

Simplex System With Restart Capability

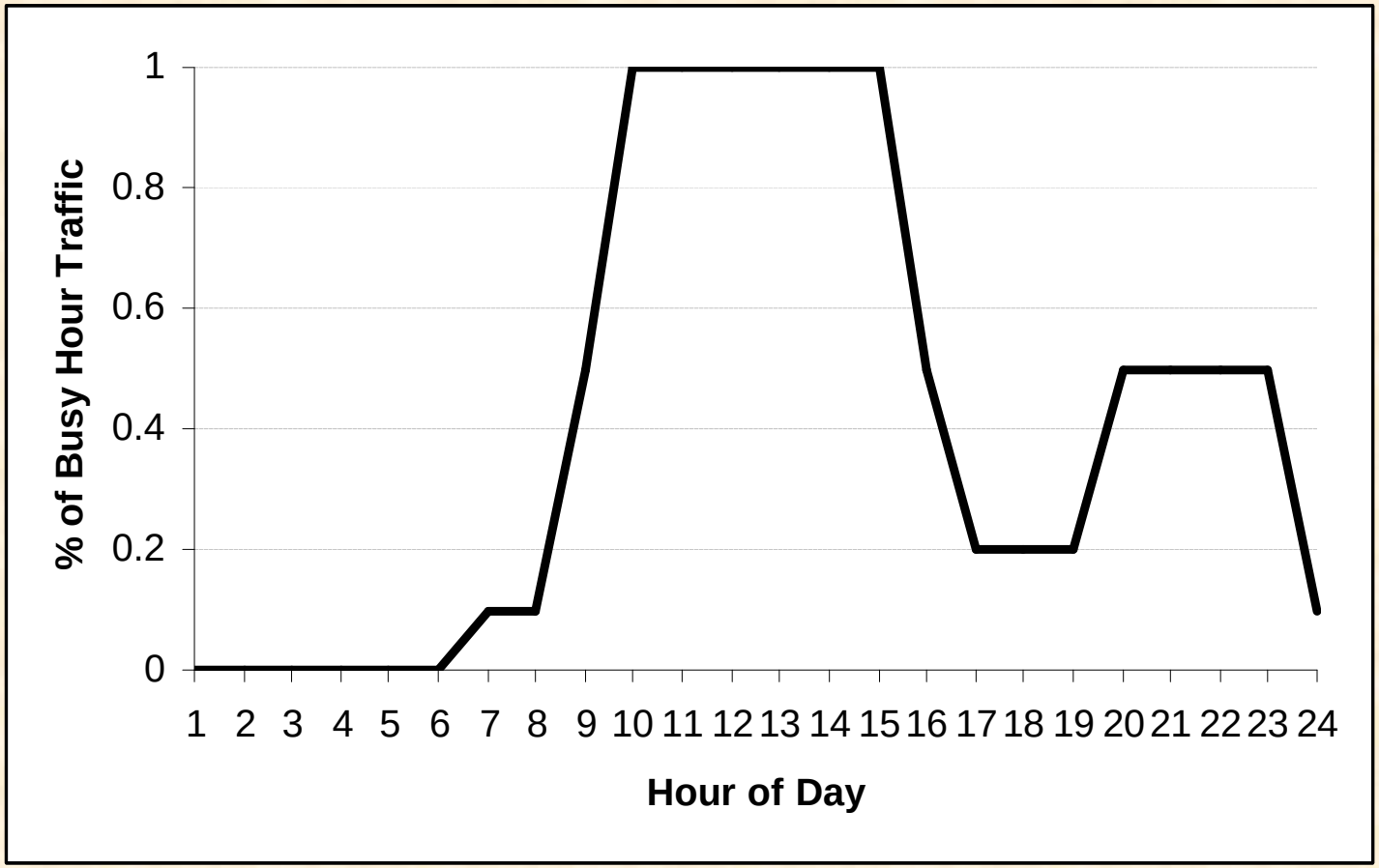
$l_{1s} = l_{2s} = l_{3s} = 0$ (no standby)

$p = 0$ and $m_f = \infty$ (no automatic failover attempts)

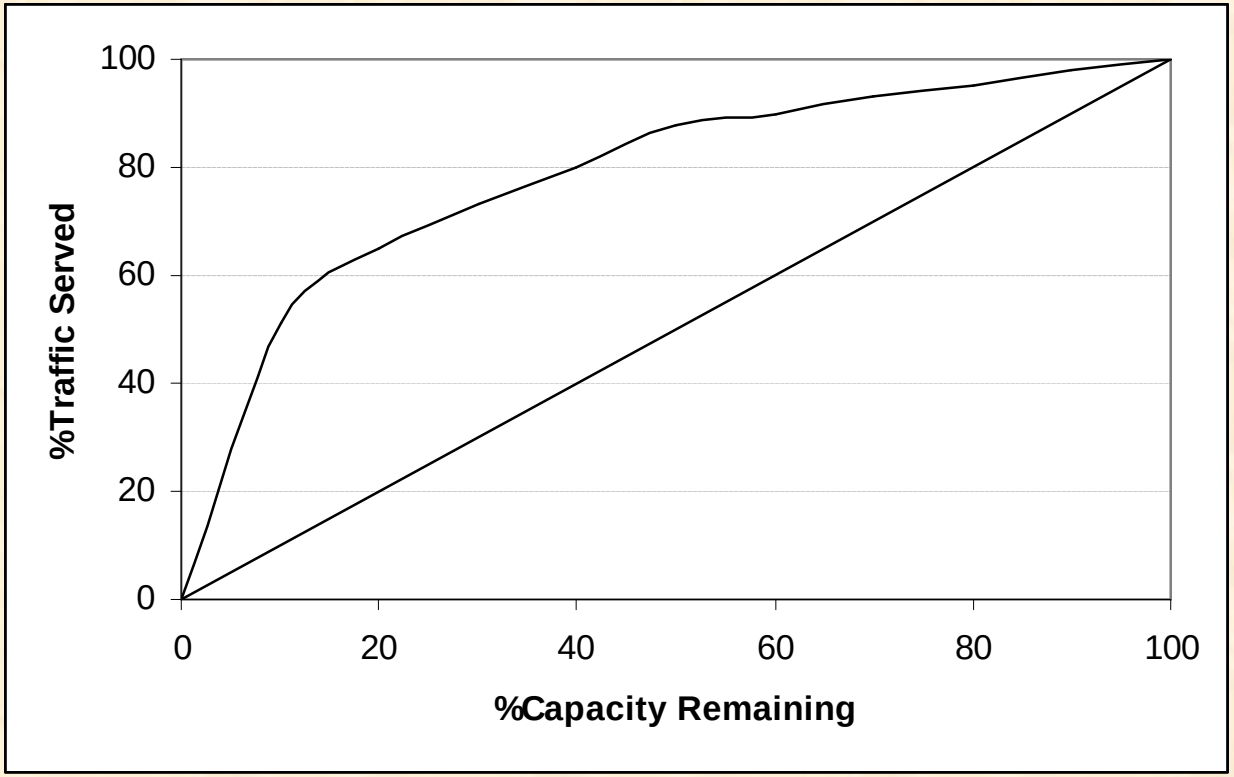
$p_m = 0$ and $m_{mf} = \infty$ (no manual failover attempts)

$q_f = 1$ (no automatic failover attempts)

Network Traffic Profile



Reward Model



C = % Capacity Remaining After the Failure

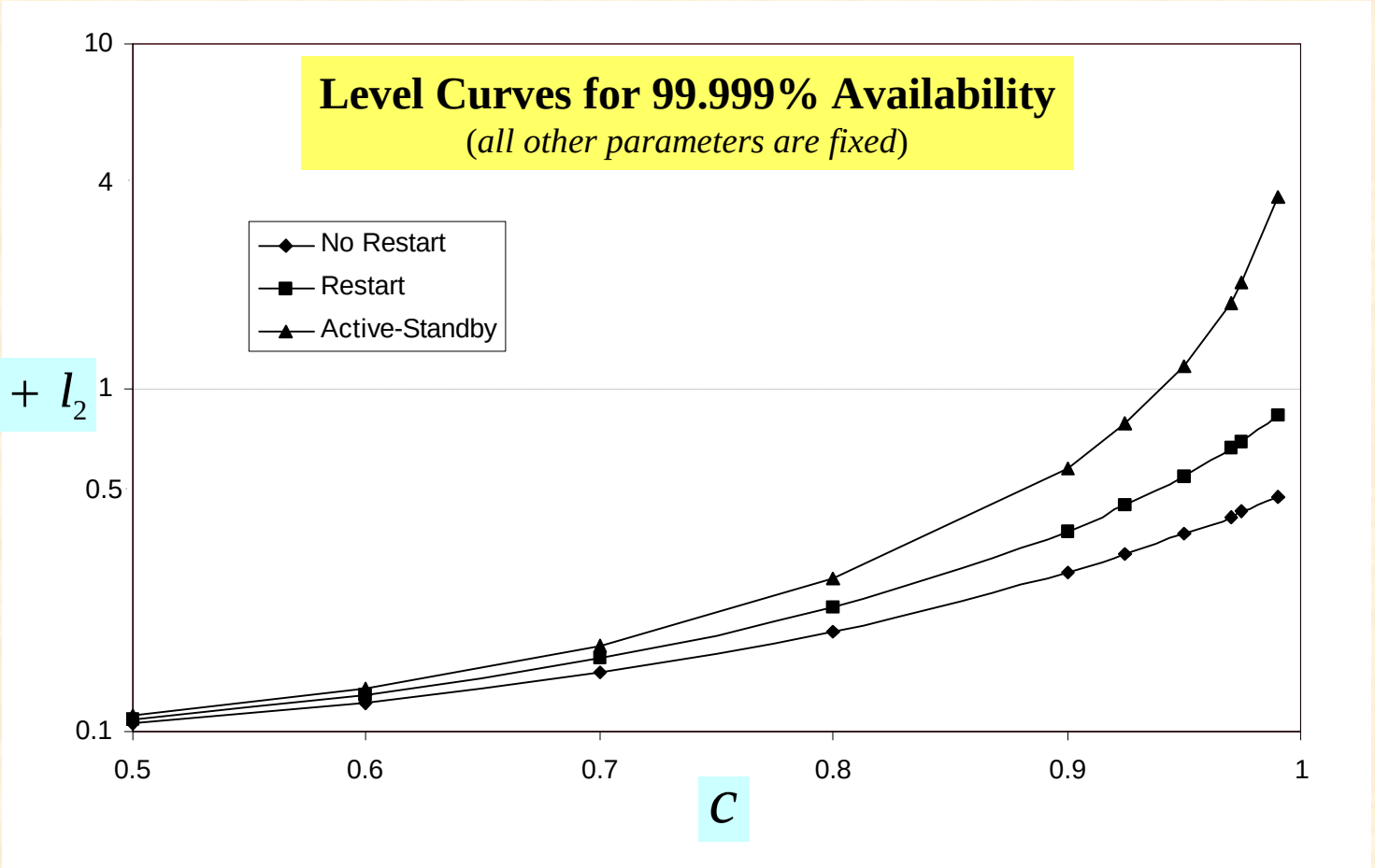
C_i = Capacity Required for Traffic Volume Offered in Hour-i

S_i = % Offered Traffic Served if Failure Occurs in Hour-i

= $\text{Min}(C_i , C) / C_i$

$\overline{S}$ = Expected %Traffic Served, Given a Failure That Leaves Capacity at C

Feasibility Regions for Critical Parameters



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

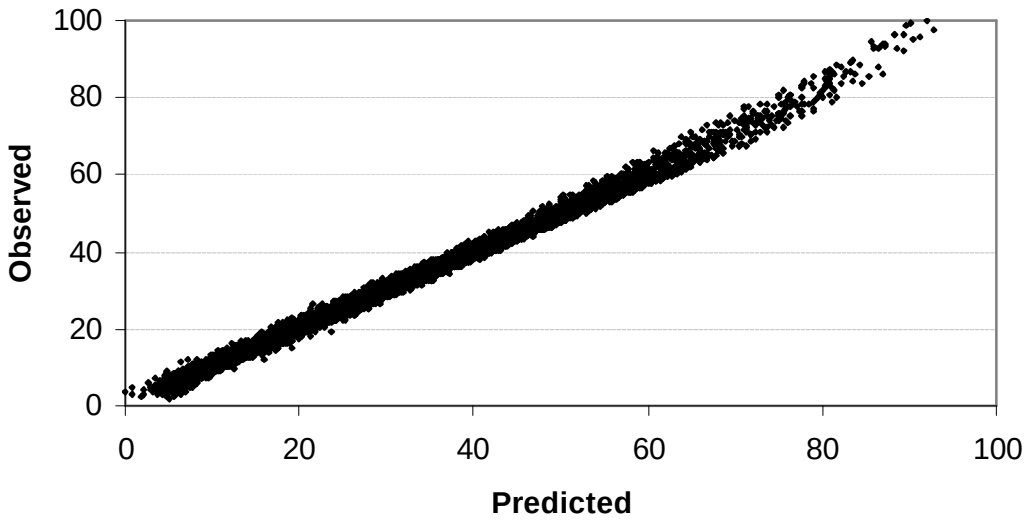
(*Likely Ranges for Parameter Values in Active-Standby Architecture*)

l_1 Fails/yr	[0.1 , 1.0]	m_{mf} Failovers/yr	26,280
l_2 Fails/yr	[0.1 , 1.0]	m_m Recoveries/yr	17,520
l_3 Fails/yr	[0.1 , 0.5]	r	[0.9 , 0.99]
l_{1s} Fails/yr	0	r_m	[0.9 , 0.99]
l_{2s} Fails/yr	[0.1 , 0.5]	q_r	[0.9 , 0.99]
l_{3s} Fails/yr	[0.1 , 0.5]	q_f	[0.9 , 0.99]
m_s Restarts/yr	31,536,000	p	[0.9 , 0.99]
m_f Failovers/yr	3,153,600	p_m	[0.9 , 0.99]
m_r Reboots/yr	26,280	c	[0.9 , 0.99]
m_d Detections/yr	[1095 , 2190]	w	[0.9 , 0.99]

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

$$D(\text{min/yr}) = 497 + 379\,l_1 + 378\,l_2 - 358\,l_1c - 356\,l_2c + 284(m_d/1000)c - 268(m_d/1000) - 461c$$
$$-12.2\,l_1(m_d/1000) + 319\,l_3 - 12\,l_2(m_d/1000) - 298\,l_3c$$
$$-22\,p - 11.6\,l_3(m_d/1000) - 11.7r - 9.6q_f - 9q_r - 5.8r_m - 5w - 2.1p_m$$

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

Needed Research

- SRGM fitting tools
- Small sample inference procedures
- Failure rate template
- Improved methods for linking SRGMs to architecture-based models