



# Reliability of Multi-component Systems

Yashwant K. Malaiya

# Transition Note

## Virtual interaction

- I will ask some questions during the video lecture only.
  - You will note down the question number and your response on paper and save it.
  - You will need to use your response in on-line quizzes, when the answer will be evaluated.

# Software Reliability: Review

We have discussed

- Defect density
- Testing
  - Exponential and other SRGMs
  - Failure intensity, fault exposure ratio

# Reliability of Multi-component Systems

- Software system: number of modules.
- Individual modules developed and tested differently: different defect densities and failure rates.
  - **Sequential execution**
  - **Concurrent execution**
  - **N-version systems**

# Reliability of Multi-component Systems

- Definition of reliability? Possible Perspectives:
  - System is reliable if there is no defect in any of the components.
    - Deterministic, not really useful.
  - Probability of a transaction running without a failure.
    - Depends on how often the failure is encountered.
  - If the system fails, what is the impact of the failure?
    - Risk due to a failure = its probability x its impact

Sometimes a transaction may be rerun after a failure. That is time redundancy.

# Sequential execution

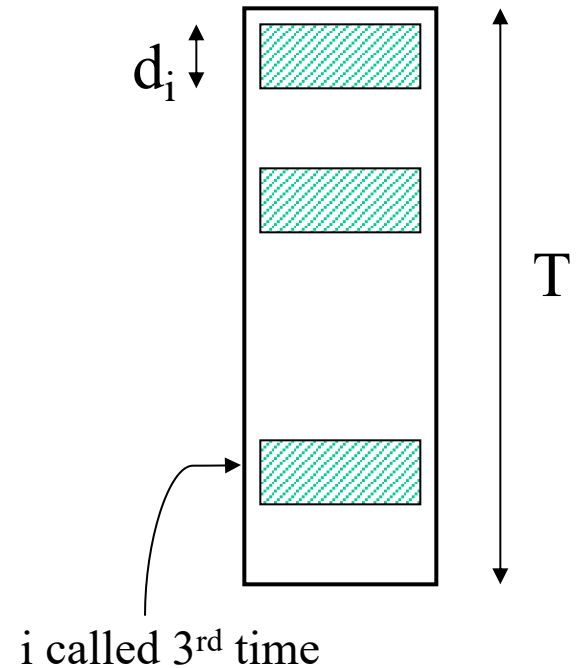
- Traditional sequential programs:
  - Assume one module executed at a time.
  - $f_i$ : fraction of time module  $i$  under execution
  - $\lambda_i$  its failure rate
- Mean system failure rate:

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

# Sequential Execution (cont.)

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

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



# Sequential Execution (cont.)

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

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

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

- 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

Reliability  $e^{-\lambda T}$      $\Pr(\text{failure in } T) = 1 - e^{-\lambda T}$

	Called times	Av duration	Fraction of T	Failure rate	Av failure rate/unit time	$R_{\text{sys}}$
Module i	$e_i$	$d_i$	$f_i$	$\lambda_i$	$\lambda_{\text{sys}}$	
a	1	3	12%	0.01		
b	2	4	32%	0.03		
c	7	2	56%	0.001		
	Total time T	25	100%		0.0114	<b>0.7527</b>

# Sequential Execution Risk

- System Risk =  $\Sigma$  Risk due to failure type I

For a specific time frame, assuming

- $Risk_i = \Pr(\text{failure type } i) \times \text{penalty for } i$ 
  - $\Pr(\text{failure type } i) = 1 - e^{-\lambda_i f_i T}$
  - $Risk_i = (1 - e^{-\lambda_i f_i T}) \times P_i$
  - System Risk =  $\sum_{i=1}^n (1 - e^{-\lambda_i f_i T}) \times P_i$

# Sequential Execution Risk

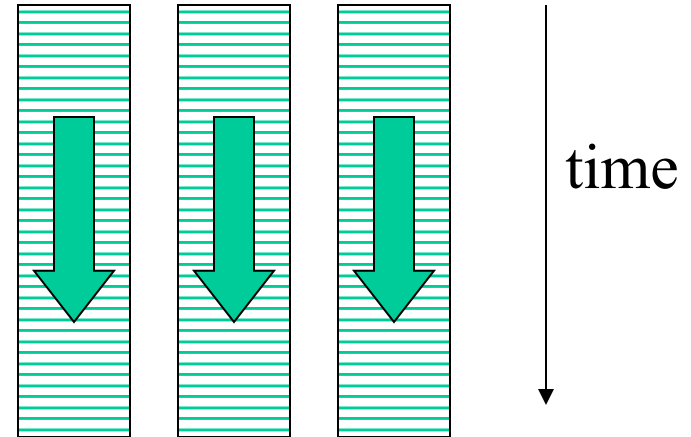
- System Risk =  $\Sigma$  Risk due to failure type  $i$

	Called times	Av duration	Fraction of T	Failure rate	Av cost/failure	Fail prob/T	Risks <sub>i</sub>
Module $i$	$e_i$	$d_i$	$f_i = d_i/T$	$\lambda_i$	$C_i$	$=1-\exp(-e_i \cdot d_i \cdot \lambda_i)$	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%			<b>Total risk</b>	<b>24.71</b>

# Concurrent execution

Concurrently executing modules, within a system or in a distributed system.

- all concurrent modules must run without failures for system to function.
- $j$  concurrently executing modules

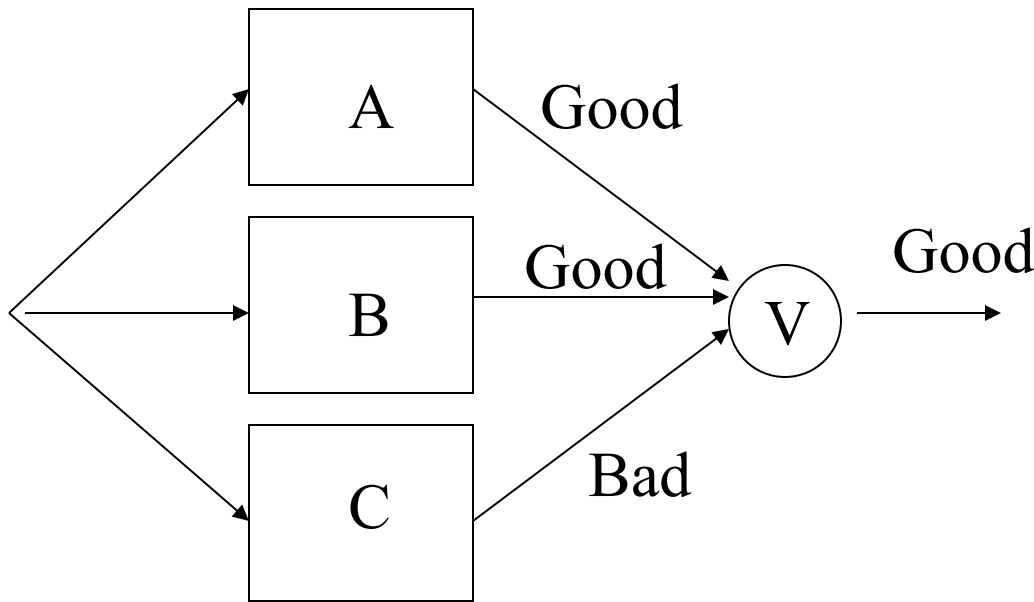


$$\lambda_{sys} = \sum_{j=1}^m \lambda_j$$

# N-version systems

- Critical applications, like defense or avionics
- Each version is implemented and tested independently
- Common implementation uses triplication and voting on the result

# N-version Systems (Cont.)



$$R_{\text{sys}} = 1 - (1-R)^3 - 3R(1-R)^2$$

$$R=0.9 \Rightarrow R_{\text{sys}}=.972$$

$$\bar{R}=0.1 \Rightarrow \bar{R}_{\text{sys}}=.028$$

# N-version systems: Correlation

- Correlation significantly degrades fault tolerance
- Significant correlation common in N-version (Knight-Leveson)
- Is it cost effective?

# 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  $P_{\text{sys}}$  of the system *failing* for a transaction

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

# 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*

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

- Uncorrelated improvement factor of  $0.0004 / 4.8 \times 10^{-7} = 833.3$

Les Hatton. 1997. N-Version Design Versus One Good Version. IEEE Software, 14, 6 (November 1997), 71-76.

J. C. Knight, N. G. Leveson and L. D. S. Jean, "A Large Scale Experiment in N-Version Programming", in 15th Int. Symp. on Fault Tolerant Computing (FTCS-15), pp.135-139, IEEE Computer Society Press, 1985

# N-version systems: Correlation

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

- Uncorrelated improvement factor of  $0.0004/4.8 \times 10^{-7} = 833.3$
- **Correlated:**  $q_3 = 2.5 \times 10^{-7}$  and  $q_2 = 2.5 \times 10^{-6}$
- $P_{sys} = 2.5 \times 10^{-7} + 3 \times 2.5 \times 10^{-6} = 7.75 \times 10^{-6}$
- improvement factor:  $0.0004/7.75 \times 10^{-6} = \mathbf{51.6}$
- 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.