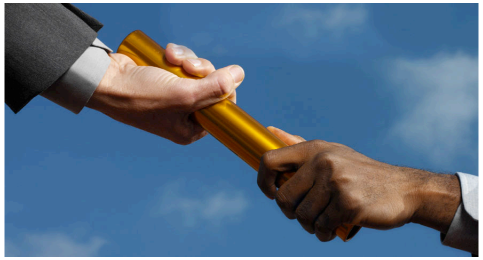
Verification and Validation (PA2405)

Lecture 3: Software Reliability Engineering

What is software reliability?



- Software reliability:
  - The probability of failure-free software operation for a specified period of time in a specified environment (ANSI91)
- Specified environment:
  - Conditions of operation have to be specified
  - e.g. usage profile, environment, and so forth
- Reliability principles adapted from other domains
  - e.g. mechanics/hardware

Some definitions

- Mean time to failure (MTTF) – Expected time that the next failure will occur (also: mean time between failures - MTBF)
- Mean time to repair (MTTR) – Expected time to repair after the failure has occurred
- Availability = MTTF / (MTTF + MTTR); or Exp[Uptime]/(Exp[Uptime]+Exp[Downtime])
- Failure intensity – Mean rate of failure per time unit (also called ROCOF)
- Operational profile – Probability of the occurrence of input classes or operations

Some hints of how to develop reliable software

Be good in defect prevention (Avoid defects slipping into development)

Be good in catching bugs (testing practices, early defect detection, static V&V)

Evaluation (measure and analyze your failure and defect data as well as your processes)

Removal (Correct the mistakes made to avoid future failures)

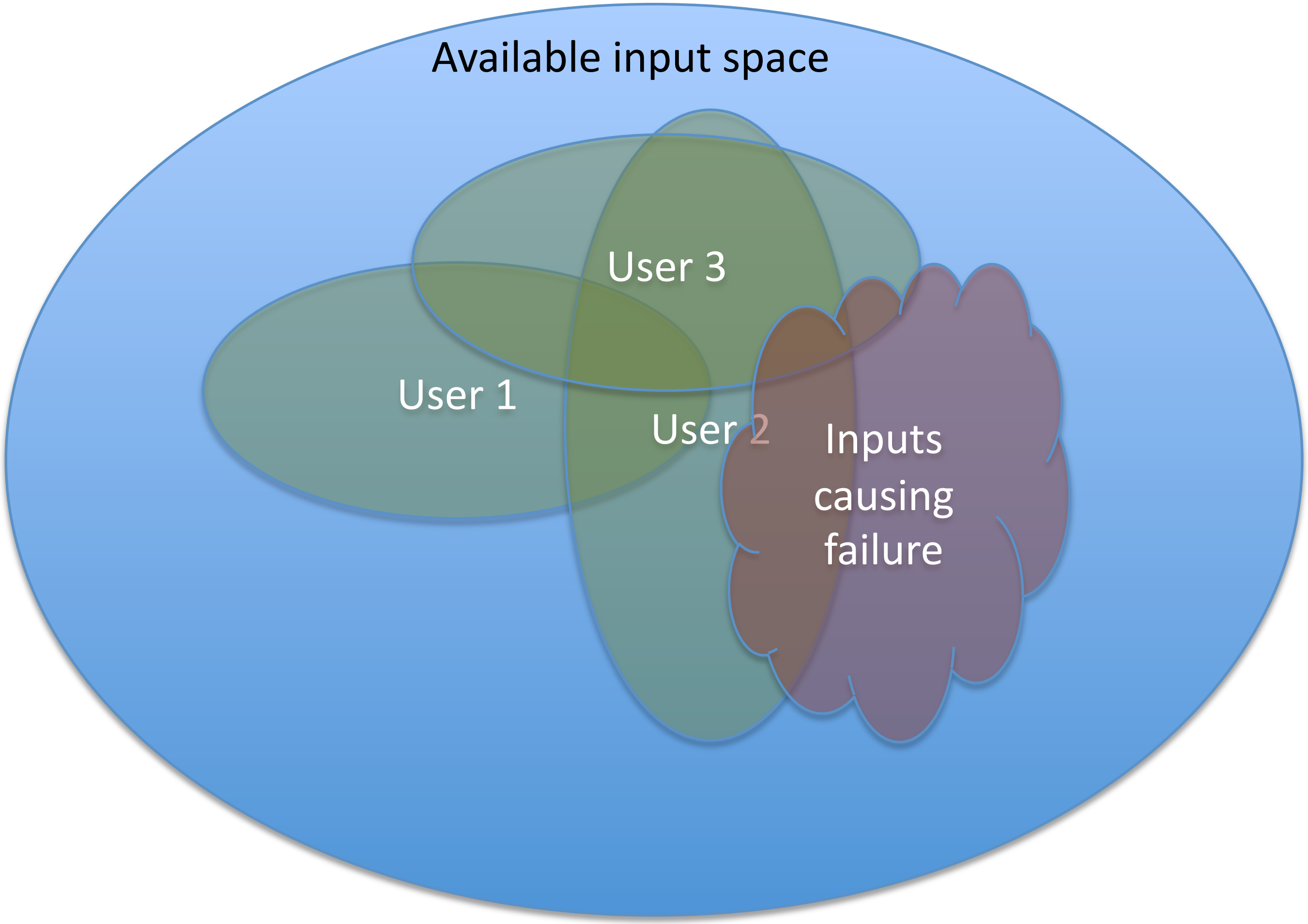
Build fault tolerant systems (if the system fails it can recover)

Be good in defect prevention

- Follow good practices (this depends on your specific needs)
  - How do you specify the requirements?
  - Do you interact with your customer?
  - Are your developers well trained and motivated?
  - Do you follow design principles and patterns?
  - etc.

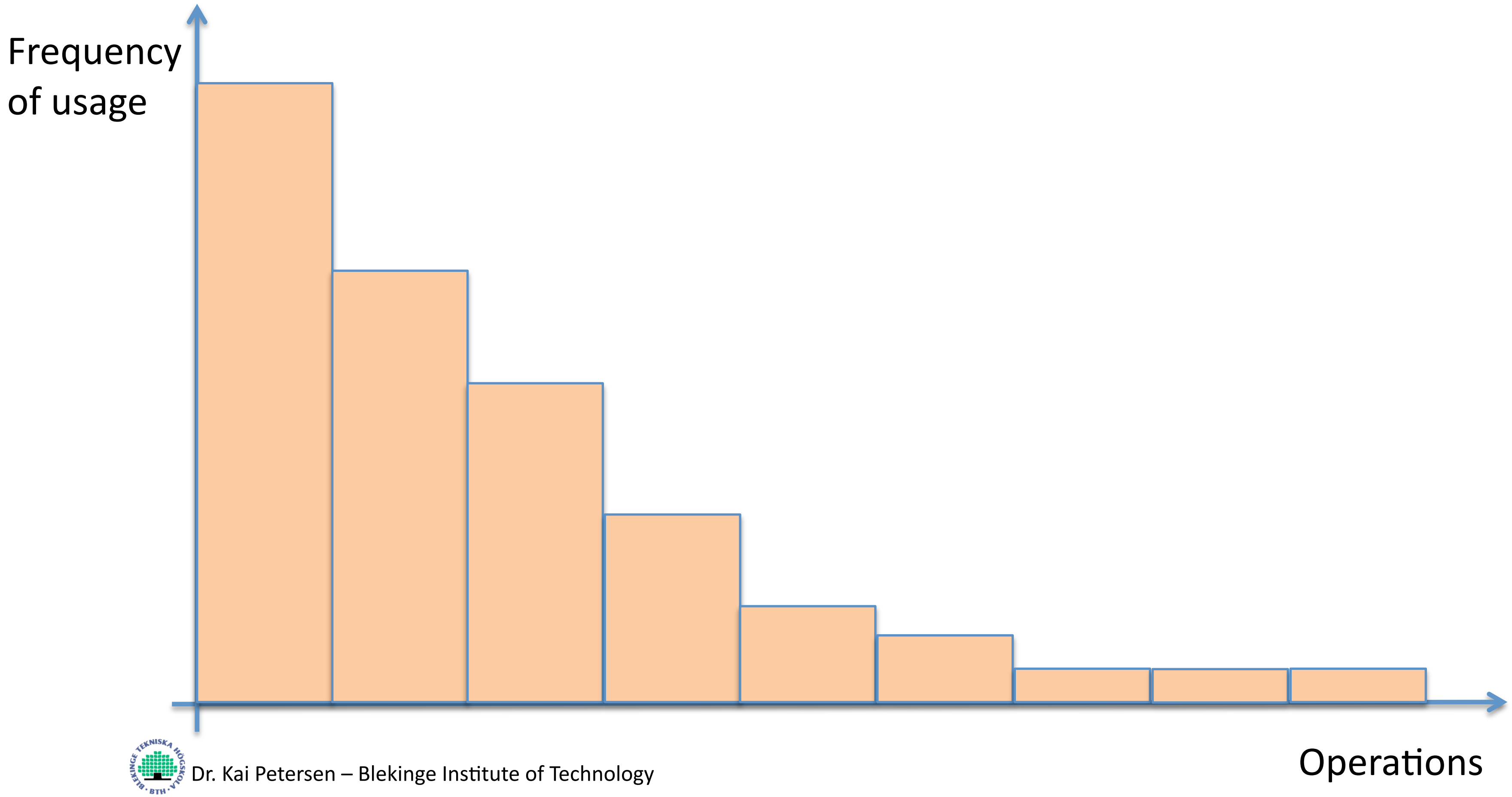
Be good in catching bugs

- … that matter!



Be good in catching bugs

- Establish an operational profile and test according to that profile
- Usage-based testing (also called statistical usage testing)



Be good in catching bugs

Probabilities of usage

Activities	Module 1	Module 2	Module 3
Coverage testing	1/3	1/3	1/3
Usage testing	0.001	0.01	0.989
Operation 1 (actual)	0.002	0.05	0.948
Operation 2 (actual)	0.10	0.15	0.75

MTBF values per module

Before test	Module 1	Module 2	Module 3
Baseline reliability	100	100	100
After coverage test	1000	1000	1000
After usage test	121	314	21230

Be good in catching bugs

Factoring in the probabilities of usage in the reliability of the overall system

Activities	Module 1	MTBF
ESTIMATION FROM TEST	(usage M1)* Rel. M1 + (usage M2) *Rel. M2 + (usage M3) * Rel. M3	
Coverage test	1/3*1000+1/3*1000+1/3*1000	1000
Usage test	0.001*121+0.01*314+0.989*21230	21000
PERCEIVED IN OPERATION		
After usage test (Operation 1)	0.002*121+0.05*314+0.948*21230	20142
After usage test (Operation 2)	0.1*121+0.15*314+0.75*21230	15982

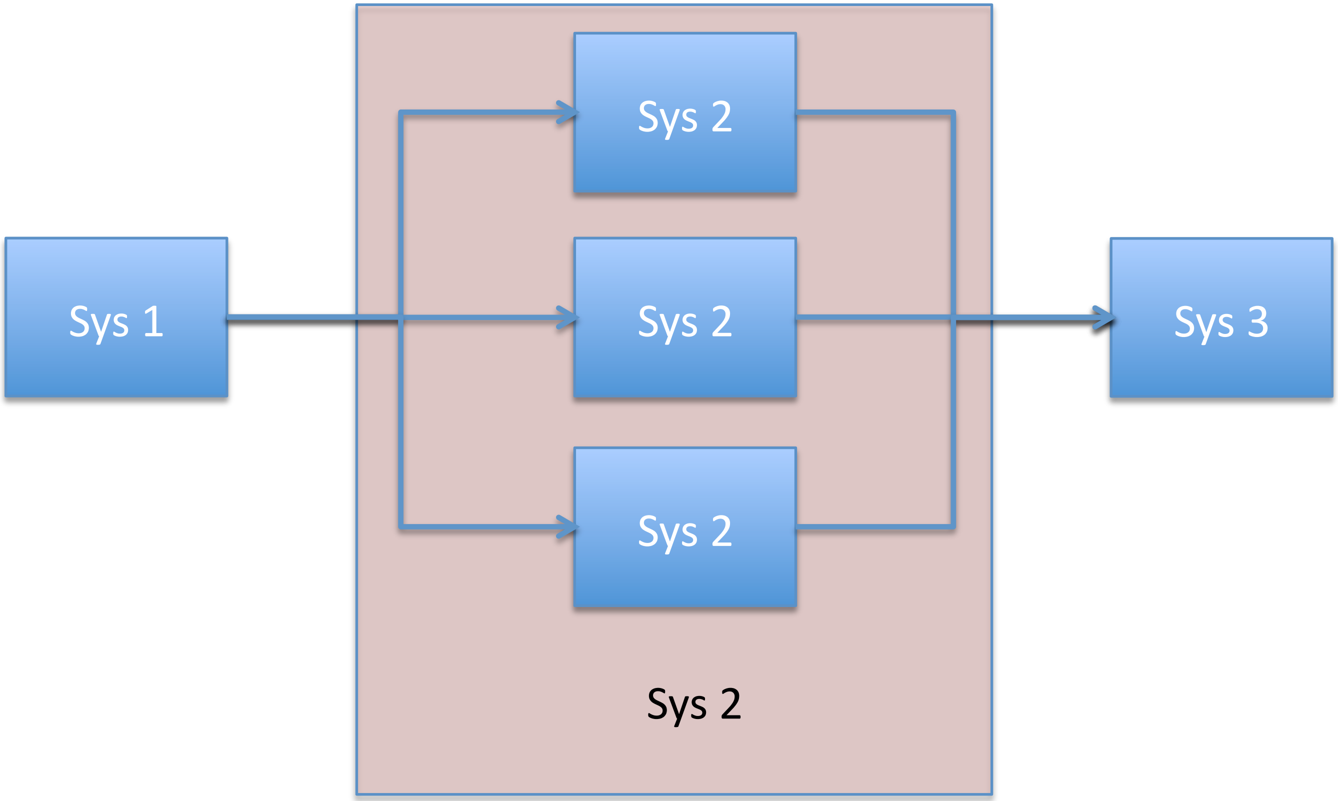
Observe: Usage-based testing improved system reliability by focusing test effort on frequently used parts of the system and increased the reliability level of these systems.

Removal (Identification and removal of cause of failure)

- After a failure root-cases have to be investigated
- Risks:
  - Failure is not easily re-producible and the root-case (bug) is not obvious to find
  - Fixing a bug can lead to new bugs
- If external failure: Pressure to quickly release a correction (e.g. need sufficient time for re-testing)

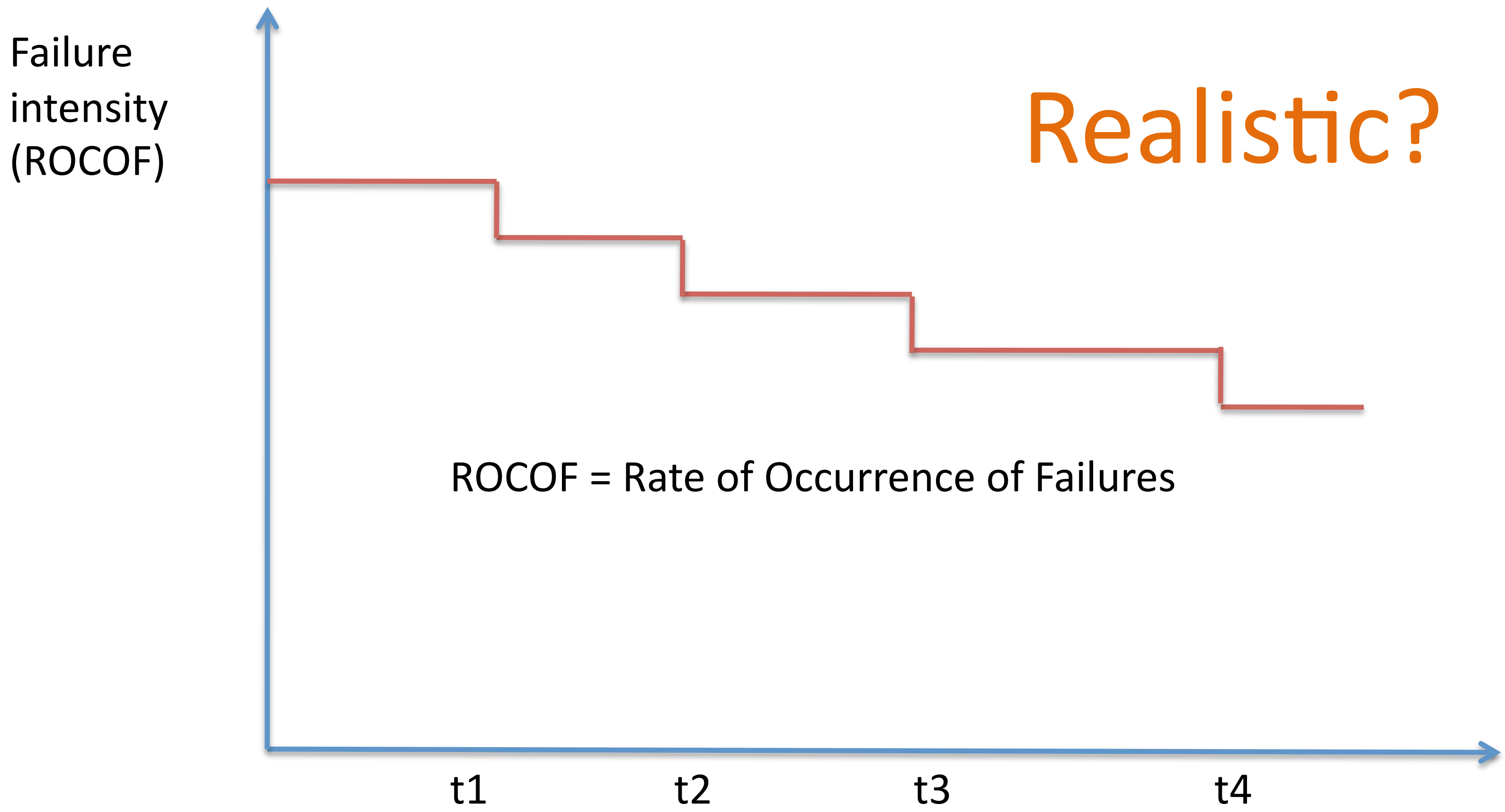
Build fault-tolerant systems

Example: Redundancy



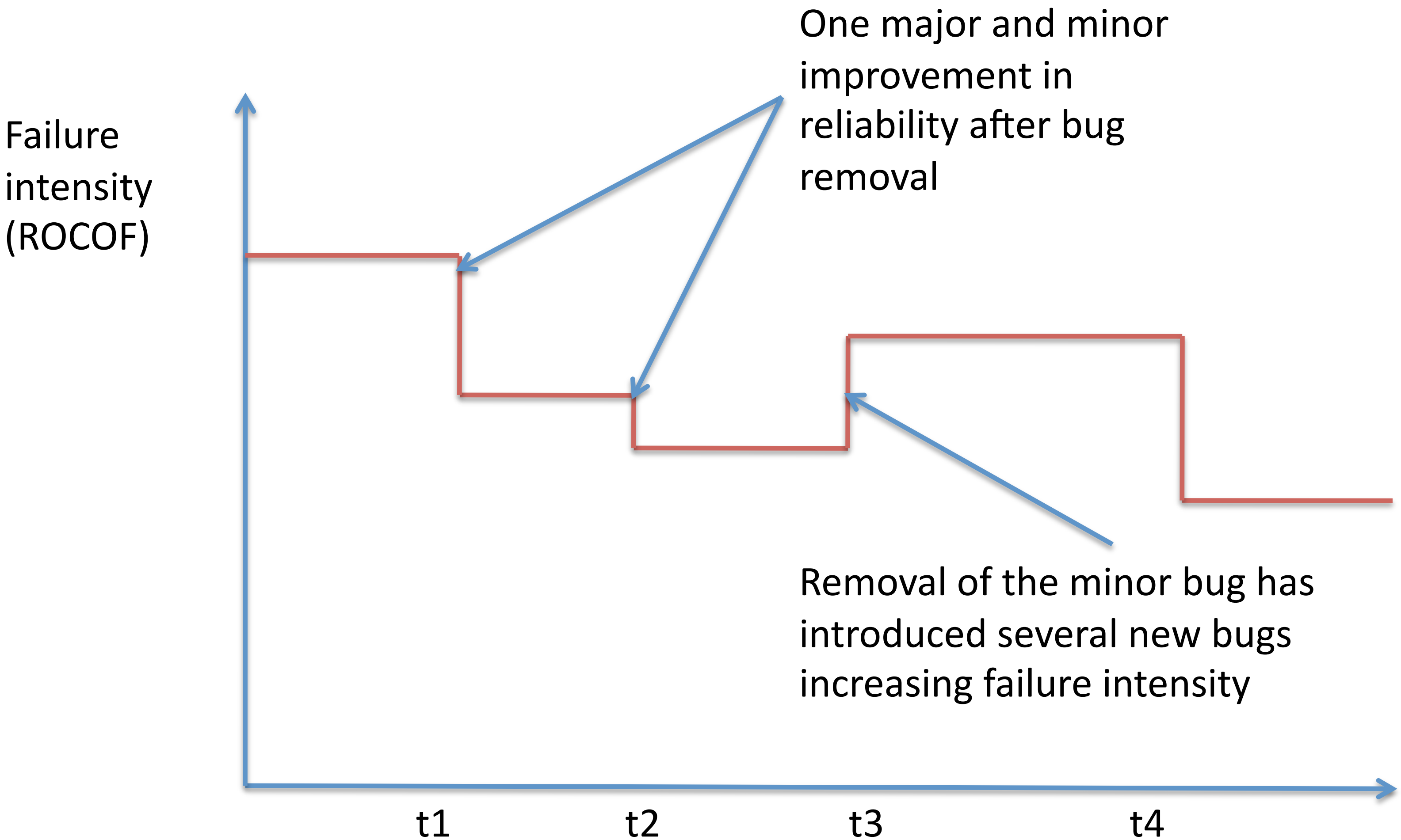
Evaluation (Analyze reliability growth based on given MTBF and fault data)

- Simple reliability growth model (equal step reliability growth)



Evaluation (Analyze reliability growth based on given MTBF and fault data)

More realistic model

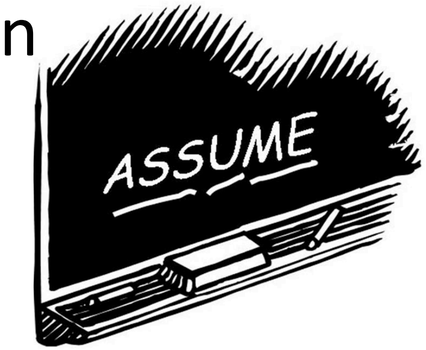


Evaluation (Reliability Growth Models)

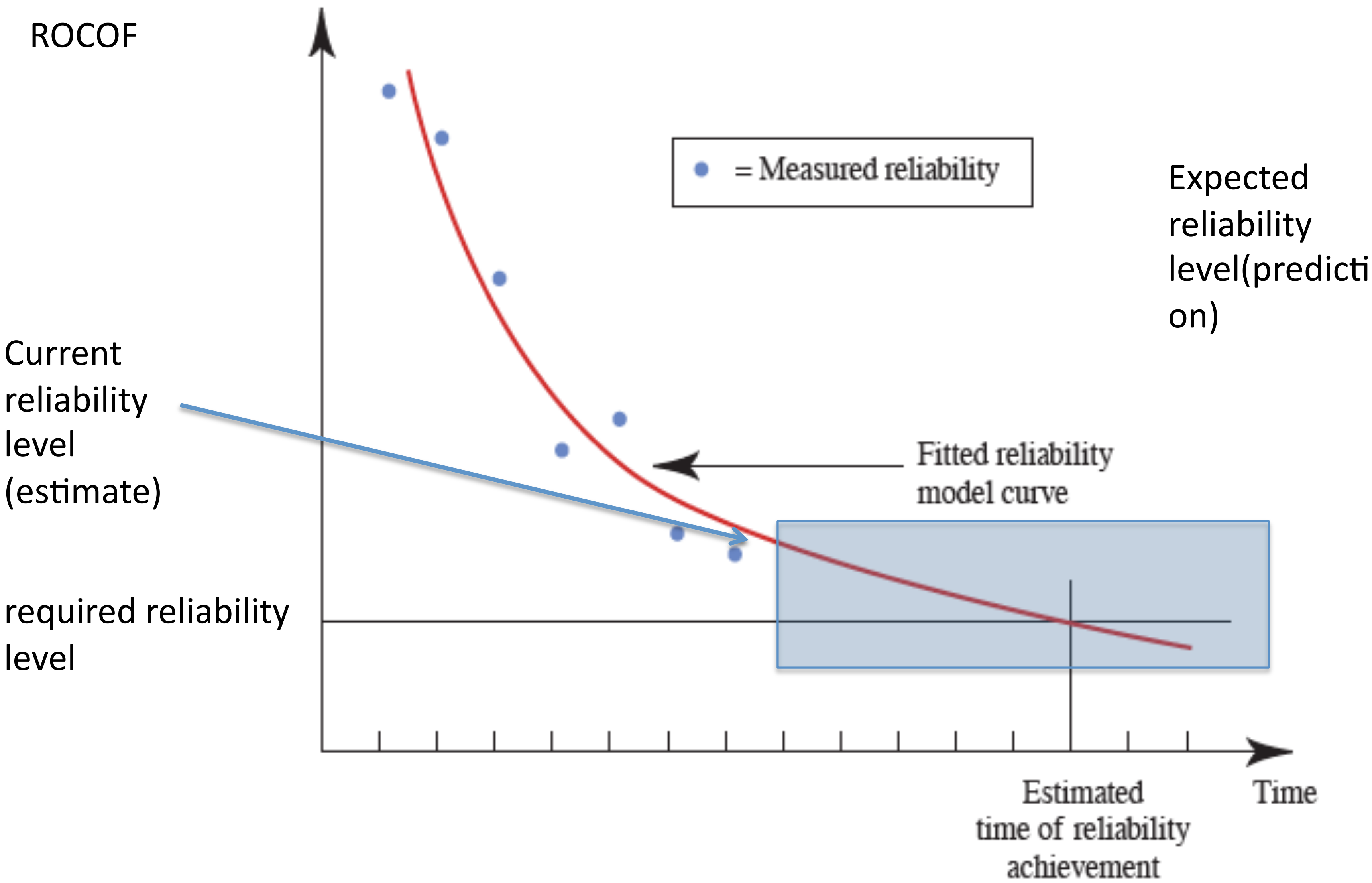
- Time between failure models
  - Jelinski and Moranda
  - Schick and Wolverton
  - etc. (see literature)
- Fault count data
  - Goel-Okumoto Nonhomogeneous Poission Process Model
  - Mosa Execution Time Model

Jelinski and Moranda

- Assumptions:
  - The number of initial software faults is an unknown, but fixed constant
  - A detected fault is removed immediately and no new fault is introduced
  - Time between failures are independent, exponentially distributed random quantities
  - All reaming software faults contribute the same amount to the software failure inten



Evaluation (Find a model that fits)



References:

- A.L. Goel, Software reliability models: assumptions, limitations, and applicability, IEEE Transactions on Software Engineering, 11(12), 1985
- C.A. Asad, M.I. Ullah, M.J. Rehman, An approach for software reliability model selection, Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), 2004, IEEE Computer Society
- Michael R. Lyu, Handbook of Software Reliability Engineering, McGraw-Hill, 1995 – you can download it at: http://www.cse.cuhk.edu.hk/~lyu/book/reliability/

Problem set

-> This time no problem set as you will have the assignment on this topic.