Software Reliability Engineering

Karama Kanoun

Karama.Kanoun@laas.fr

Research Group on
Dependable Computing and Fault Tolerance
Toulouse, France

OUTLINE

- Motivations
- Methods for software reliability engineering
  - Data collection and analysis
    - Data collection and validation
    - Descriptive statistics
    - Trend analysis
  - Software dependability evaluation
    - Reliability growth models
    - Models in stable reliability
    - Controlled experimentation
- The maturity Process
- Case studies
- References

Why Software Reliability Engineering?

- Increasing role of software in real life systems
- System dependability is more and more synonymous of software reliability
- Difficulties in mastering the software development process and in reducing design faults for complex systems
- Increasing cost of system non-dependability
- Real needs for improving software reliability to improve system dependability and reduce maintenance cost
- Dependability requirements are part of system requirements (as important as functional requirements)
- Quantification is essential

Objectives of software reliability engineering

- Short term
  - Manage and improve the reliability of the software
  - Check the efficiency of development activities
  - Estimate the software reliability at the end of validation activities and in operation
  - Estimate the maintenance effort to “correct” faults activated during development and residual faults in operation
- Long term
  - Capitalize experience
  - Improve software reliability of successive generations

- ***Needs for experimental & analytical methods and techniques to reach these objectives***

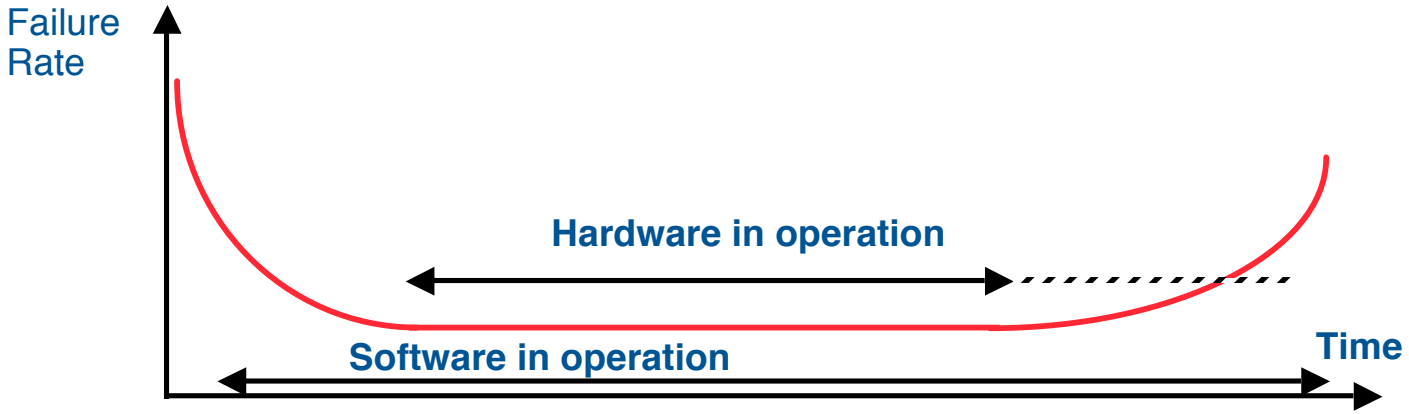
Software vs hardware reliability

Hardware

- Physical faults
- Operational life
- Stable reliability (constant failure rate)
- White-box approach
- Markov models
- Database for components failures

Software

- Only design faults
- Development and operation
- Reliability growth (↓ failure rate)
- Usually black-box approach
- Specific models
- Based on data collection



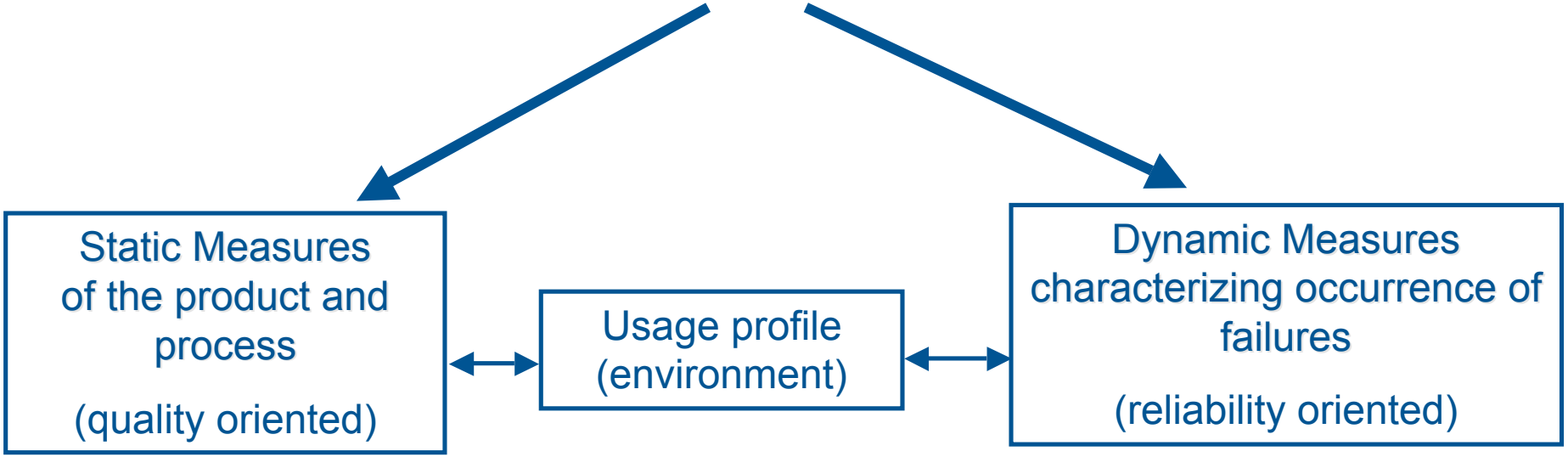
Objectives of Software Reliability Engineering

- Supplier point of view
  - During development:
    - development follow up
      (failure intensity, fault density)
    - evaluation of software reliability before operation
      (MTTF, pre-operational failure rate)
  - During operation
    - product reliability follow up
      (residual failure rate, MTTF)
    - maintenance planning
      (cumulative number of failures)
- Users / customers, operational life
    - be confident in the reliability level of the product
      (residual failure rate, MTTF)

Difficulties

- Non-repetitive process
- No relationship between failures and corrections
- Continuous evolution of usage profile
  - According to the development phase
  - Within each phase
- Overselling of reliability growth models
- Judgement on quality of the software developers
- What is software reliability?
  - Residual number of faults, fault density, complexity measures?
  - MTTF, failure intensity, failure rate?

Measures



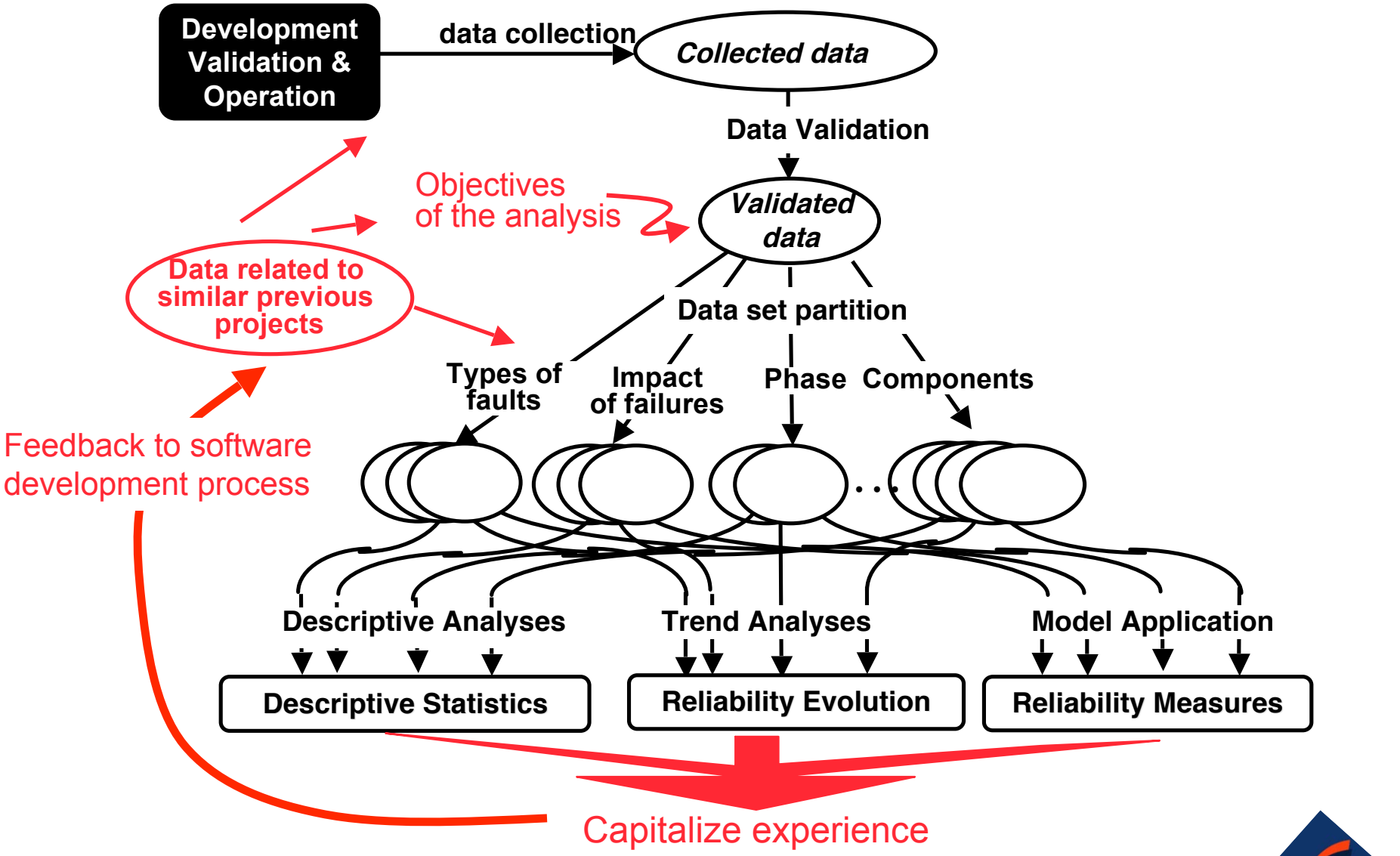
Number of faults
Fault density
Complexity measures
…

Failure intensity
Failure rate
MTTF
Reliability
…

Example:
Percentage of faults and corresponding MTTF (published by IBM)

MTTF (years) / Product	5000	1580	500	158	50	15.8	5	1.58
1	34,2	28,8	17,8	10,3	5,0	2,1	1,2	0,7
2	34,3	28,0	18,2	9,7	4,5	3,2	1,5	0,7
3	33,7	28,5	18,0	8,7	6,5	2,8	1,4	0,4
4	34,2	28,5	18,7	11,9	4,4	2,0	0,3	0,1
5	34,2	28,5	18,4	9,4	4,4	2,9	1,4	0,7
6	32,0	28,2	20,1	11,5	5,0	2,1	0,8	0,3
7	34,0	28,5	18,5	9,9	4,5	2,7	1,4	0,6
8	31,9	27,1	18,4	11,1	6,5	2,7	1,4	1,1
9	31,2	27,6	20,4	12,8	5,6	1,9	0,5	0,0

Overview of a global reliability analysis method



Setting up of a data collection process

- Some rules
  - Define clearly the objectives and the data to be collected
  - Motivate and imply people that will be involved
  - Simplify the collection process and reduce the number of data items to be collected
    - Support tools
    - Practical organization of people involved
  - Record and analyze data in real-time
  - Feedback
- Origin of collected data
  - Internal: recorded during development and validation
  - External: by the customers

Data to be collected

- Background information
  - Product itself: software size, language, functions, current version, workload
  - Usage environment: verification and validation methods, tools, etc.
- Data relative to failures and corrections
  - Date of occurrence, nature of failures, consequences
  - Type of faults, fault location
- Usually, recorded through
  - Failure Reports (FR)
  - Correction Reports (CR)
- Well defined headings, well structured, easy to fill in
- Short tick-off questions
- Manually or automatically

- Failure Report (FR)

  Required Information

  - Serial number (for identification)
  - Report editor
  - Product reference, version affected (or prototype)
  - Date and time of failure occurrence

  Desirable Information

  - Failure occurrence condition
  - Failure criticality or consequences
  - Affected function or task
  - Action proposed (if any)

- Correction Report (CR)

  Required information

  - Serial number (for identification)
  - Report editor
  - Date of correction
  - Correction nature
  - Product reference
  - Reference to the FR

  Desirable Information

  - Identification of the modified components
- Integration with already existing data collection programs
- Importance of training

Data Validation

- Objectives
  - check the validity and usability of the information recorded
  - Keep only genuine software faults in the database
- Elimination of:
  - Duplicated data (FR reporting of the same failure)
  - FR proposing a correction related to an already existing FR (COR)
  - False FR (signalling a false or non identified problem)
  - FR proposing an improvement (IMPROVE)
  - incomplete FRs or FRs containing inconsistent data (Unusable)
  - FR related to a hardware failure
  - …

Example 1: a telecommunications equipment

(analyzed at LAAS)

- 2 146 Failure Reports
- Validation ⇒ 1 172 kept in the database
- Discarded RFs:

Duplicated FRs	816	38.0%
COR	53	2.5%
False FR	29	1.4%
IMPROVE	21	1.0%
Unusable	20	0.9%
Hardware	35	1.6%
Total	974	45.4%

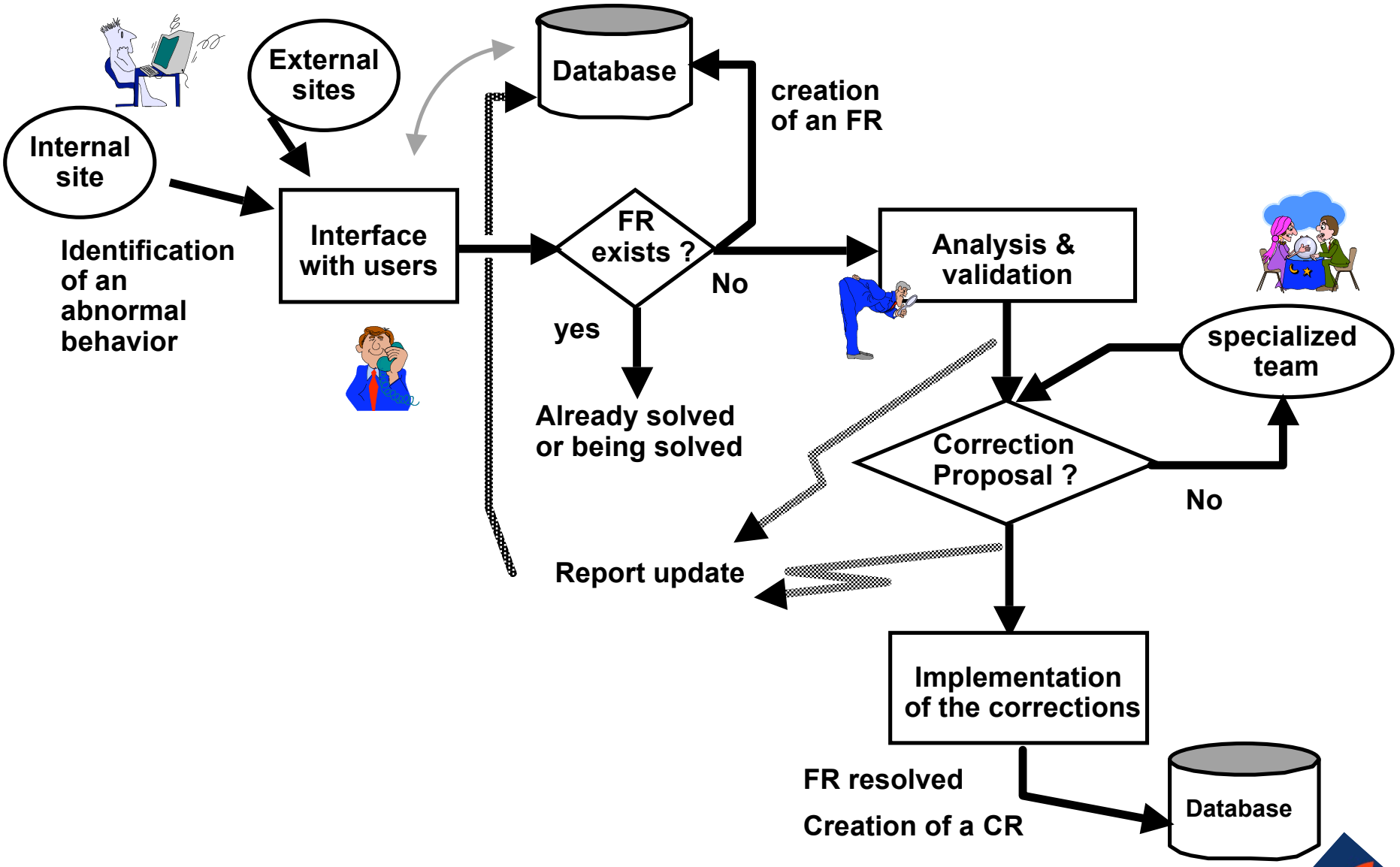
Example 2: a telephone switching system

(analyzed at LAAS)

- 3063 FRs
- Validation ⇒ 1853 Software FRs kept in the database
- Discarded RFs:

Hardware	195	(6%)
Documentation	165	(5%)
Unusable, duplicated, …	716	(24%)
Others	134	(4%)
Total	1210	(39 %)

Life cycle of Failure and Correction Reports (FRs/CRs)



DESCRIPTIVE STATISTICS

- Aim: make syntheses of the observed phenomena
- Simple analyses
  - Fault typology
  - Fault density of components
  - Failure / fault distribution among software components (new, modified, reused)
- Investigation of relationships
  - Fault density / size / complexity
  - Fault density / life cycle phase
  - Nature of faults / life cycle phases
  - Nature of faults / components
  - Number of components affected by changes made to resolve an FR
- Analyses related to the development / debugging process

Analyses related to the development process

- Factors affecting time to locate and solve problems
  - The more FRs circulating, the more time it takes to handle each one
  - Tendency to resolve the easier FRs first, the remaining ones take more time
  - Loss of maintainability with continued changes to resolve faults
  - Introduction of new faults while resolving the old
- Average time to resolve an FR

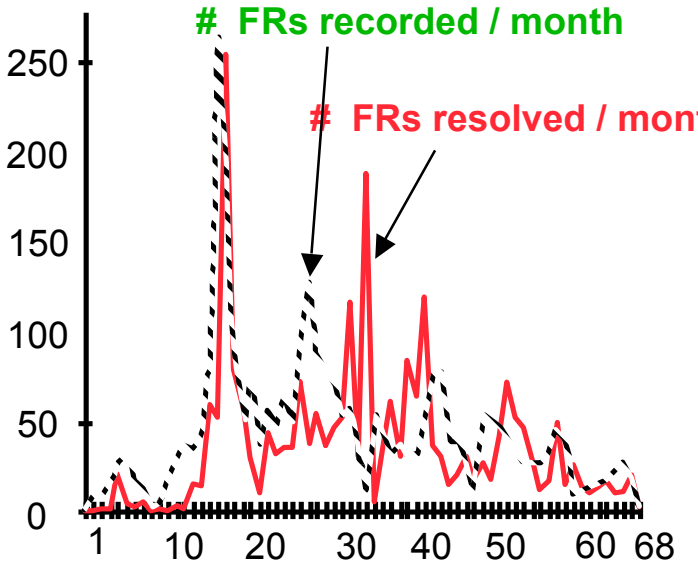
  Modification request time =

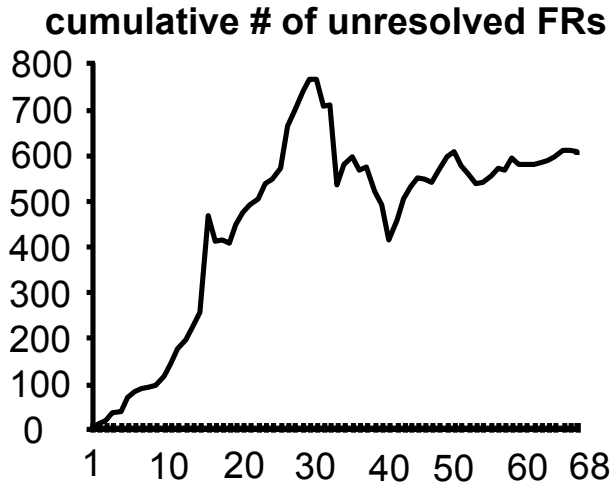
  Time when the FR is resolved - time when it is created

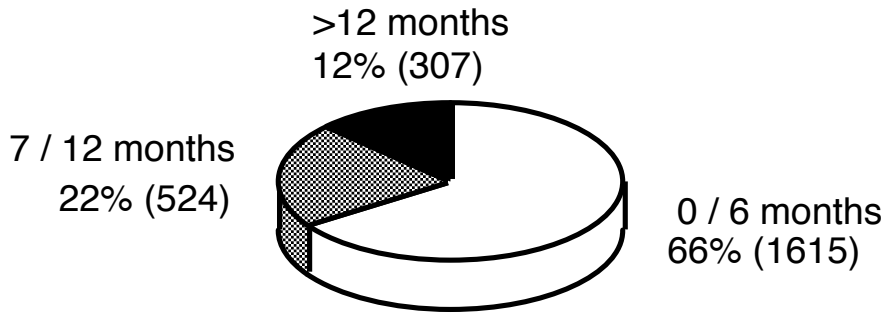
Measures

- Responsiveness of the field support system
- Complexity of maintenance

Case of the switching system of Example 2



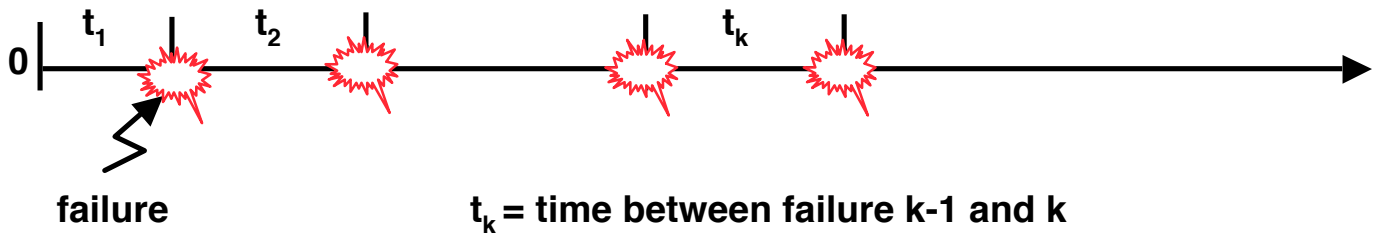




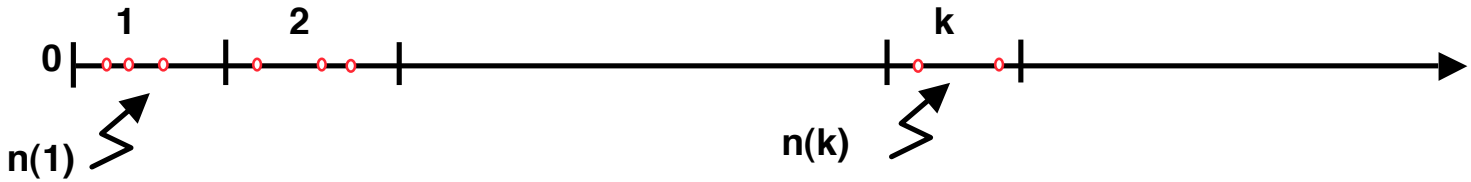
Time to resolve an FR

Data pre-processing for reliability analysis

- **Two kinds of data sets can be extracted from FRs and CRs**
  - Time to failures (or between failures)



  - Grouped data
    - Number of failures per unit of time, n(k)
    - Cumulative number of failures N(k)



Time ?

- Time between failures
  - Execution time
  - Wall clock or Calendar time
  - Number of executions

- Number of failures per unit of time
  - The length of the unit time depends on:
    - accuracy expected for the dependability measures
    - number of observed failures
    - objectives of the study

Example A: Times between failures

- Real-time control system (Musa 1)
  - 136 failures observed during system test (96 days)

 # : number of failures

T_i : times between failures (in seconds)

Dy: day of observation

#	T_i	Dy	#	T_i	Dy	#	T_i	Dy	#	T_i	Dy	#	T_i	Dy	#	T_i	Dy
1	3	1	25	422	31	49	816	56	73	810	64	97	261	72	121	75	80
2	30	2	26	180	32	50	1351	56	74	290	64	98	1800	73	122	482	80
3	113	9	27	10	32	51	148	56	75	300	64	99	865	73	123	5509	81
4	81	10	28	1146	33	52	21	57	76	529	65	100	1435	74	124	100	81
5	115	11	29	600	34	53	233	57	77	281	65	101	30	74	125	10	81
6	9	11	30	15	42	54	134	57	78	160	65	102	143	74	126	1071	83
7	2	17	31	36	42	55	357	57	79	828	66	103	108	74	127	371	83
8	91	20	32	4	46	56	193	59	80	1011	66	104	0	74	128	790	83
9	112	20	33	0	46	57	236	59	81	445	66	105	3110	75	129	6150	83
10	15	20	34	8	46	58	31	59	82	296	66	106	1247	76	130	3321	83
11	138	20	35	227	46	59	369	59	83	1755	67	107	943	76	131	1045	84
12	50	20	36	65	46	60	748	59	84	1064	67	108	700	76	132	648	84
13	77	20	37	476	46	61	0	59	85	1783	68	109	875	77	133	5485	87
14	24	20	38	58	46	62	232	59	86	860	68	110	245	77	134	1160	87
15	108	20	39	457	47	63	330	59	87	983	68	111	729	77	135	1864	88
16	88	20	40	300	47	64	365	61	88	707	69	112	4897	78	136	4116	92
17	670	30	41	97	47	65	1222	62	89	33	69	113	447	79			
18	120	30	42	263	47	66	543	63	90	868	69	114	386	79			
19	26	30	43	452	53	67	10	63	91	724	69	115	446	79			
20	114	30	44	255	53	68	16	63	92	2323	70	116	122	79			
21	325	30	45	197	54	69	529	64	93	2930	71	117	990	79			
22	55	30	46	193	54	70	379	64	94	1461	72	118	948	80			
23	242	31	47	6	54	71	44	64	95	843	72	119	1082	80			
24	68	31	48	79	54	72	129	64	96	12	72	120	22	80			

Example B

Number of failures per unit of time or Cumulative

- Switching system
  - 52 failures in operation (15 months)

i: unit of time (week)

n(i): number of failures per unit of time

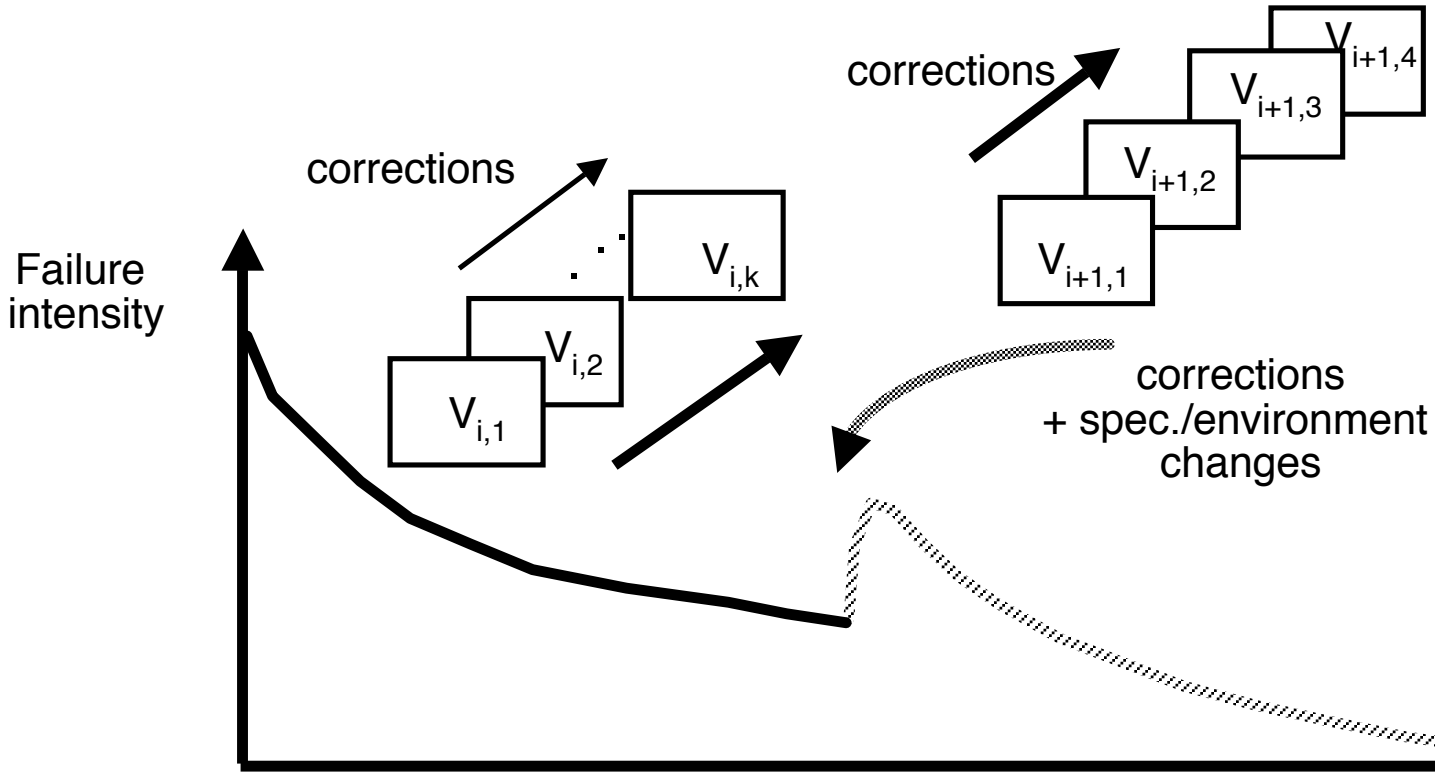
NC(i): cumulative number of failures

NS(i): number of systems in operation at i

i	n(i)	NC(i)	NS(i)	i	n(i)	NC(i)	NS(i)	i	n(i)	NC(i)	NS(i)
1	2	2	4	24	1	30	36	47	0	37	42
2	0	2	10	25	1	31	36	48	0	37	42
3	2	4	10	26	0	31	36	49	0	37	42
4	1	5	10	27	0	31	36	50	1	38	42
5	1	6	10	28	1	32	38	51	0	38	42
6	0	6	12	29	1	32	40	52	1	39	42
7	2	8	12	30	0	32	40	53	1	40	42
8	1	9	12	31	0	32	40	54	0	40	42
9	2	11	12	32	0	32	40	55	0	40	42
10	5	16	12	33	0	32	42	56	1	41	42
11	2	18	13	34	0	32	42	57	1	42	42
12	1	19	13	35	0	32	42	58	6	48	42
13	2	21	13	36	1	33	42	59	0	48	42
14	0	21	13	37	0	33	42	60	0	48	42
15	0	21	21	38	0	33	42	61	0	48	42
16	0	21	21	39	0	33	42	62	1	49	42
17	0	21	21	40	0	33	42	63	0	49	42
18	1	22	21	41	0	33	42	64	0	49	42
19	1	22	21	42	0	33	42	65	0	49	42
20	2	24	28	43	1	34	42	66	1	50	42
21	1	25	28	44	2	36	42	67	0	50	42
22	0	25	28	45	0	36	42				
23	4	29	28	46	1	37	42				

Trend analysis

- Objectives:
  - Analyze software reliability evolution
  - Identify periods of reliability growth and decrease



[See references 9 or 10]

Reliability growth characterization

- Variable: time to failure
  - $T_1, T_2, \ldots, T_n$: time between failure i and i-1
    - Reliability growth: $T_i \underset{st}{\leq} T_k \quad \forall \; i < k$
    - Prob. $\{T_i < x\} \geq$ Prob. $\{T_k \leq x\} \Rightarrow F_{Ti}(x) \geq F_{Tk}(x) \quad \forall \; i < k \quad \forall \; x$
- Variable: number of failures
  - $N(t_1), N(t_2), \ldots, N(t_n)$: cumulative number of failures between 0 and t_i
  - $H(t_i) = E[N(t_i)]$ = expectation of $N(t_i)$
  - If $N(t_i)$ is a Non Homogeneous Poisson Process (NHPP):
    - reliability growth if $H(t_1) + H(t_2) \geq H(t_1 + t_2) \quad \forall \; t_1, t_2 \geq 0$ and $0 \leq t_1 + t_2 \leq T$

      (inequality is strict for at least a pair t_1, t_2)

      N(t) is a subadditive function

Interpretation of Subadditivity

$H(t_1) + H(t_2) \geq H(t_1 + t_2) \quad \forall\, t_1, t_2 \geq 0$ and $0 \leq t_1 + t_2 \leq T$

The number of events in an interval of the form $[0, t_2]$ is larger than the number of events taking place in an interval of the same length beginning later (i.e. in the form of $[T, T+t_2]$ The number of failures is decreasing

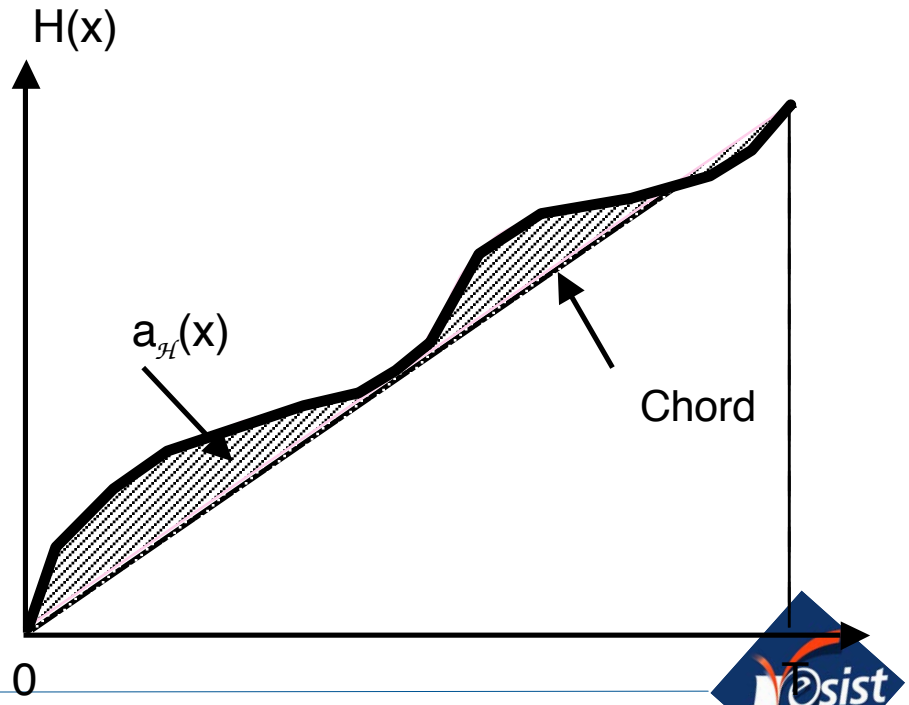
☞ Graphical interpretation

- H(t) = E[N(t)] is subadditive over [0,T] if:

$$a_{\mathcal{H}}(t) = \int_0^t H(x)\,dx - H\left(\frac{t}{2}\right) \geq 0$$

$\forall\, t \geq 0$ and $0 \leq t \leq T$

$a_{\mathcal{H}}(t)$ = subadditivity factor



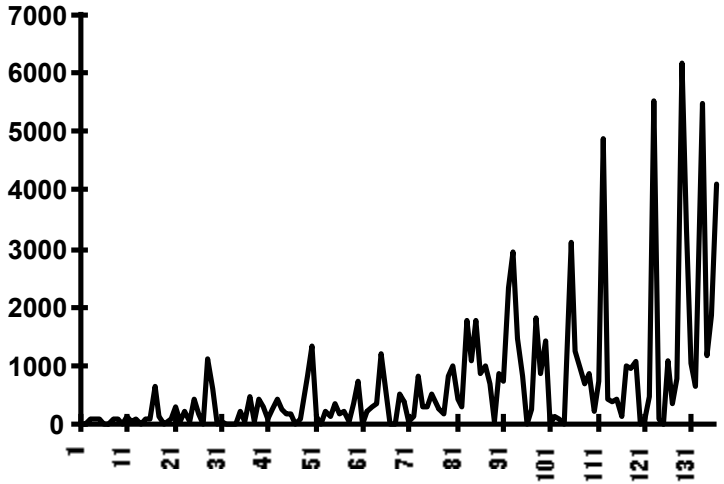
Trend tests

- Means
  - Raw data ⇒ graphical tests
  - Analytical tests ⇒ quantitative indicators
- Raw data
  - Times to successive failures
  - Number of failures per unit of time
  - Cumulative number of failures
- Trend indicators
  - Empirical (arithmetical) means
  - Subadditivity factor
  - Laplace factor

Graphical tests: times to failures (Example A)

Times to failures

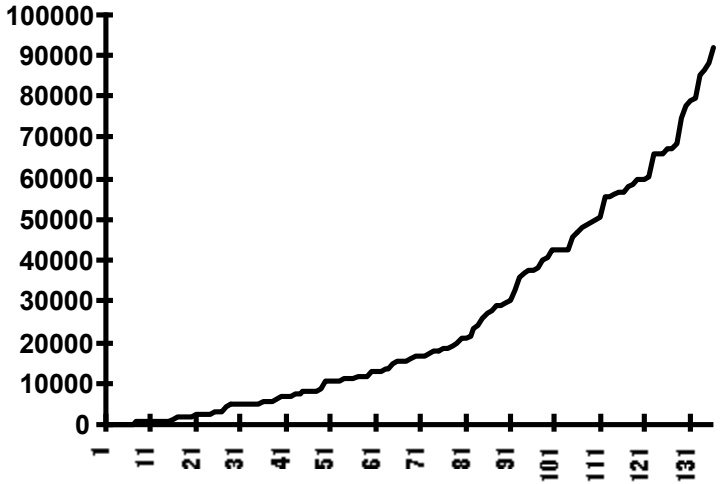
t_i



Failure #

Cumulative times to failures

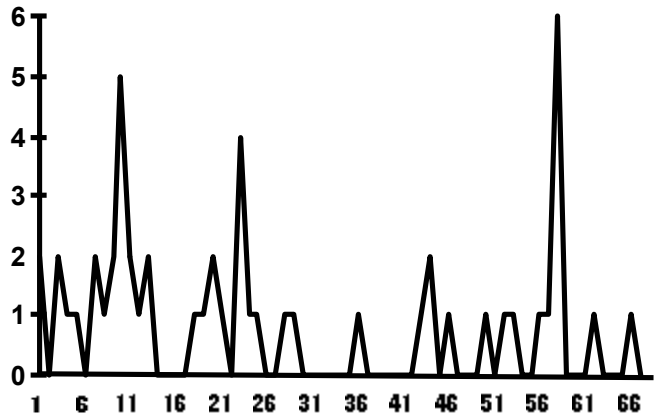
$t_1 + \ldots t_k$



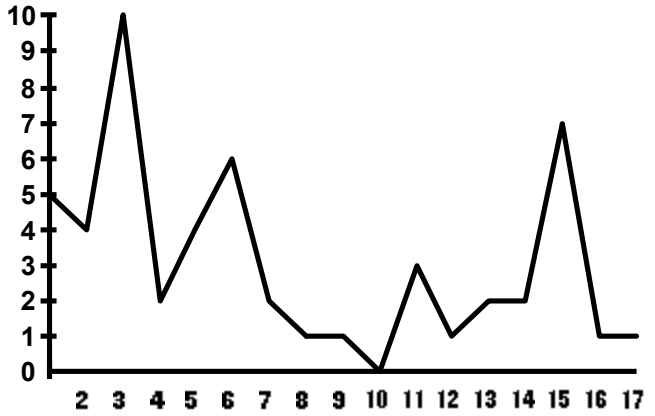
Failure #

Graphical test: grouped data (Example B)

Failure intensity

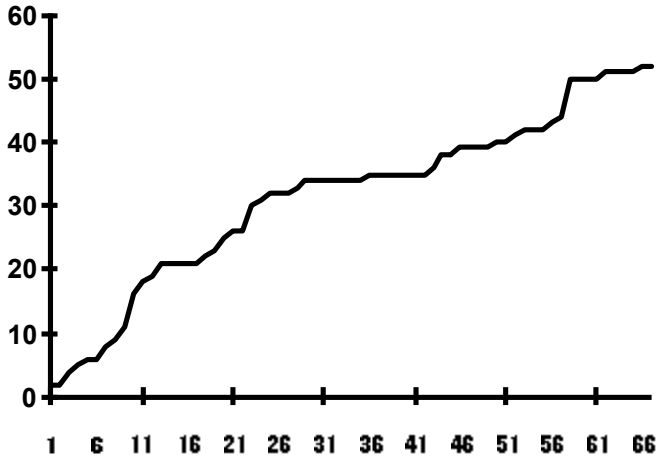


unit of time = one week

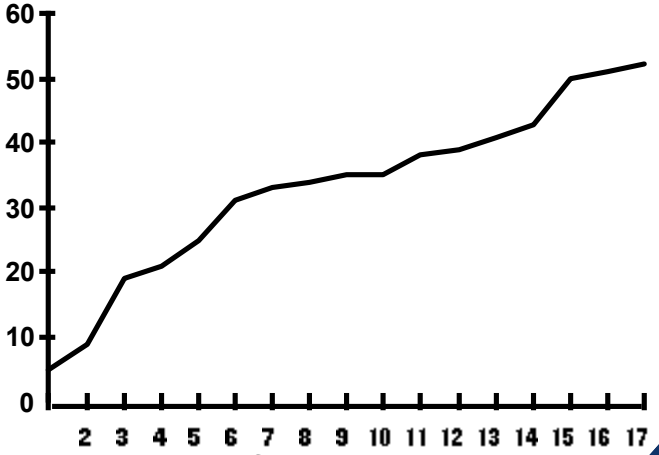


unit of time = 4 weeks

Cumulative number of failures



unit of time = one week



unit of time = 4 weeks

Empirical mean

Global trend

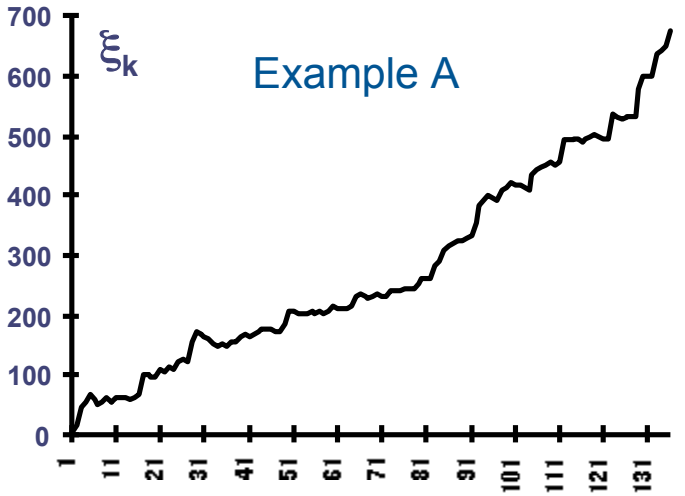
ξ_k : arithmetical mean of the times to failures (from failure 1 to k)

$$\xi_k = \frac{t_1 + t_2 + \ldots t_k}{k}$$

ξ_k constitute a globally increasing series € reliability growth

ξ_k constitute a globally decreasing series € reliability decrease

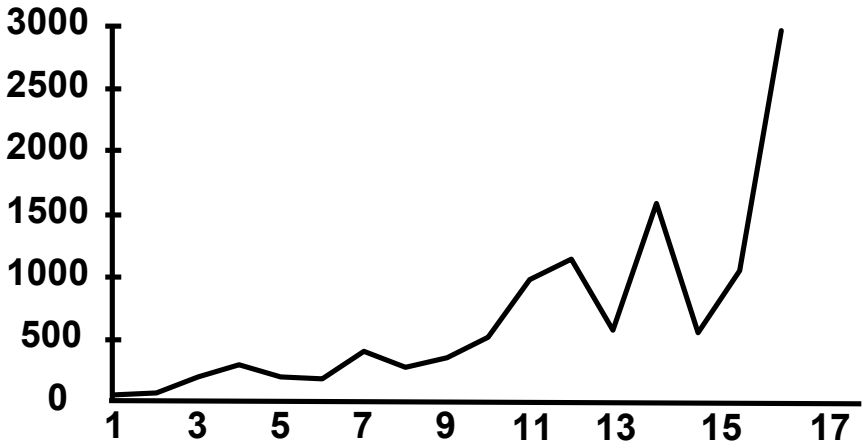
The trend is directly observed on the evolution of ξ_k ⇨



Empirical mean

- Local trend
  - The data items are grouped into subsets containing m successive data
  - The average is evaluated for each subset
  - The impact of old data items is eliminated

- Example A: m = 8 ⇒ 17 groups (136 failures)



Subadditivity & monotonous growth / decrease

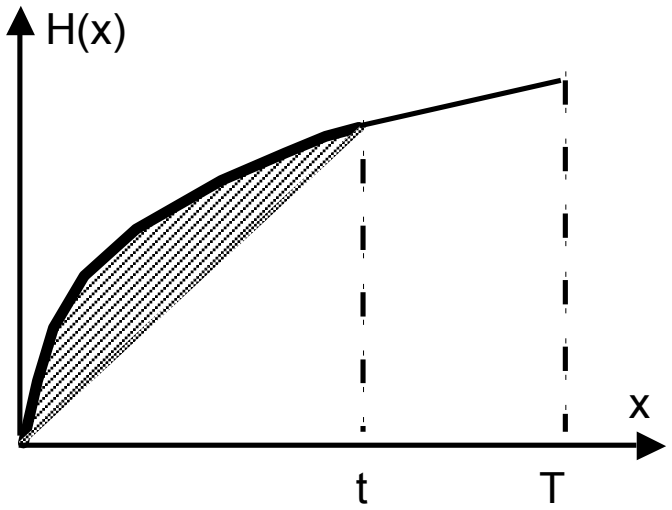
Monotonous growth:

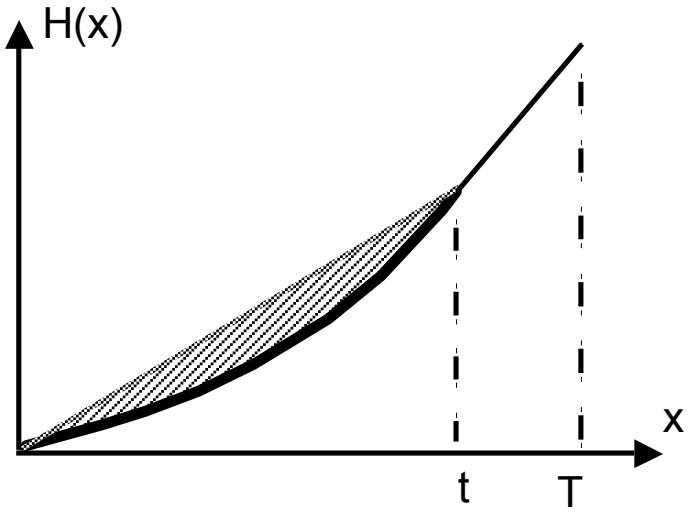
$a_H(x) > 0$ increasing

Monotonous decrease:

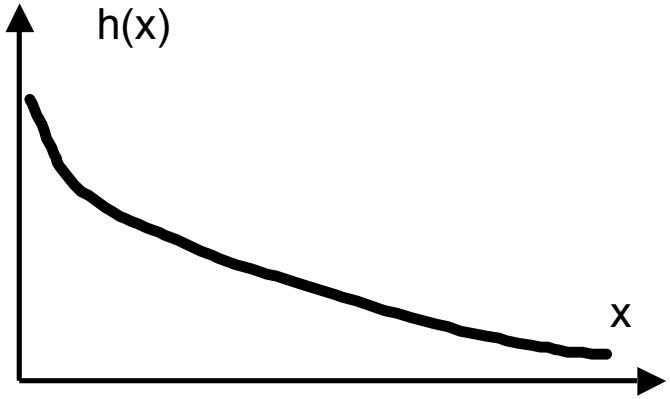
$a_H(x) < 0$ decreasing

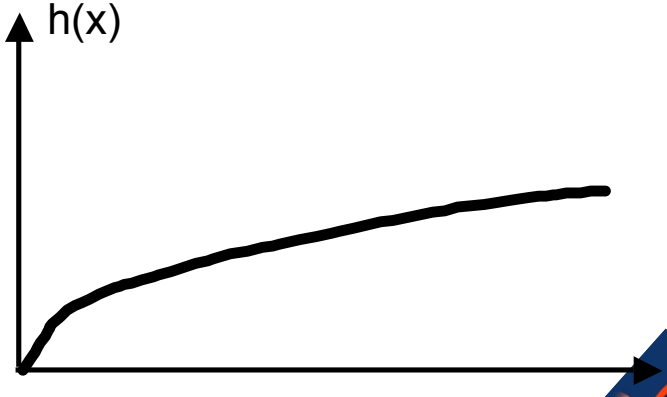
Cumulative number of failures





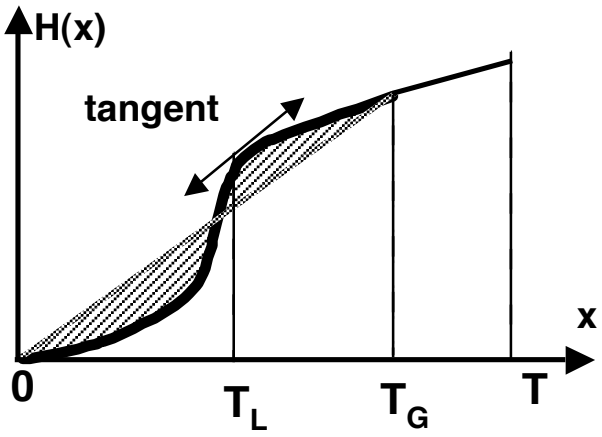
Failure intensity

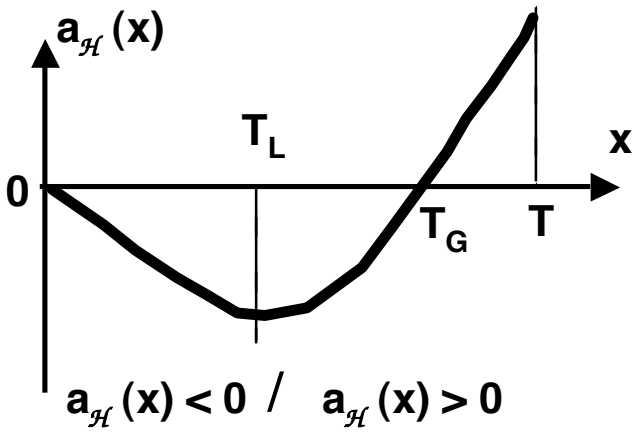




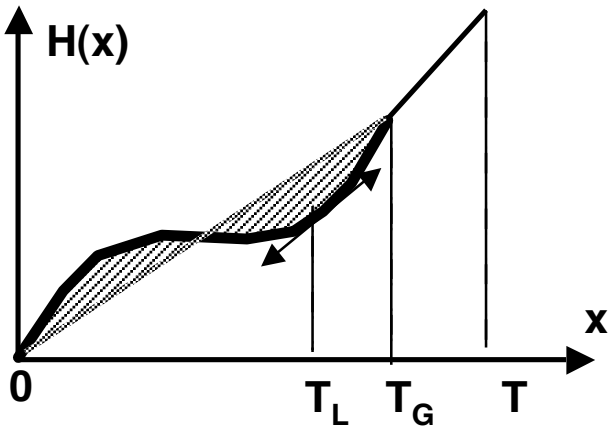
Subadditivity & trend change

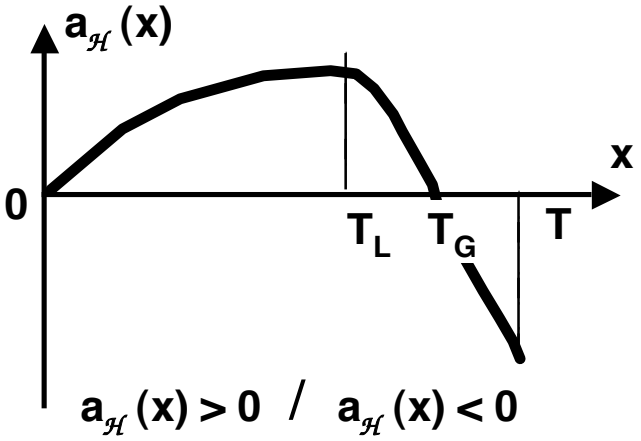
Decrease - Growth





Growth - Decrease

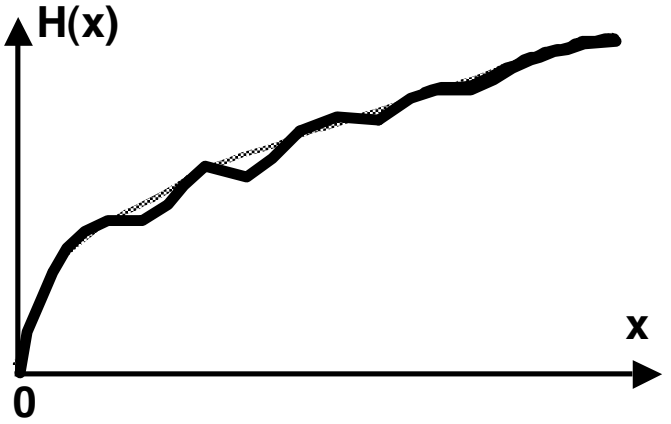


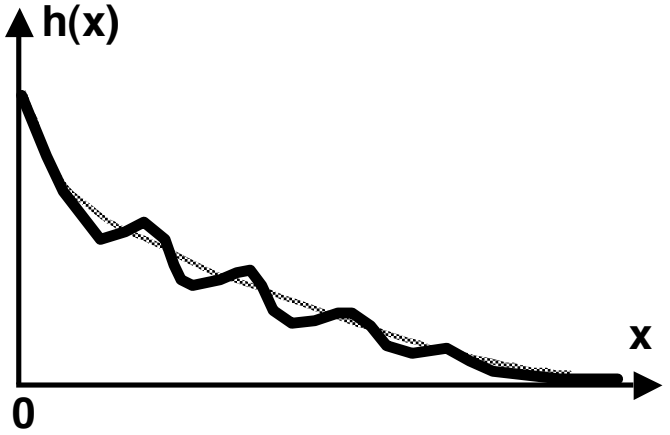


Subadditivity & local trend fluctuations

Example: reliability growth with local fluctuations

$a_{\mathcal{H}}(x) \geq 0$ non decreasing





Laplace factor

- Statistical Test of hypothesis ⇒ Laplace factor u

  Random variable: times to failures T_i (realization of $T_i = t_i$)

$$u(T) = \frac{\frac{1}{N(T)} \sum_{i=1}^{N(T)} \sum_{j=1}^{i} t_j - \frac{T}{2}}{T \sqrt{\frac{1}{12\, N(T)}}}$$

N(T) = # failures in [0,T]

  - $\frac{T}{2}$ = mid of the observation interval
  - $c = \frac{1}{N(T)} \sum_{i=1}^{N(T)} \sum_{j=1}^{i} t_j$ = statistical centre

- In practice

  u > 0 ⇒ global reliability decrease

  u < 0 ⇒ global reliability growth

- Random variable: # failures per unit of time

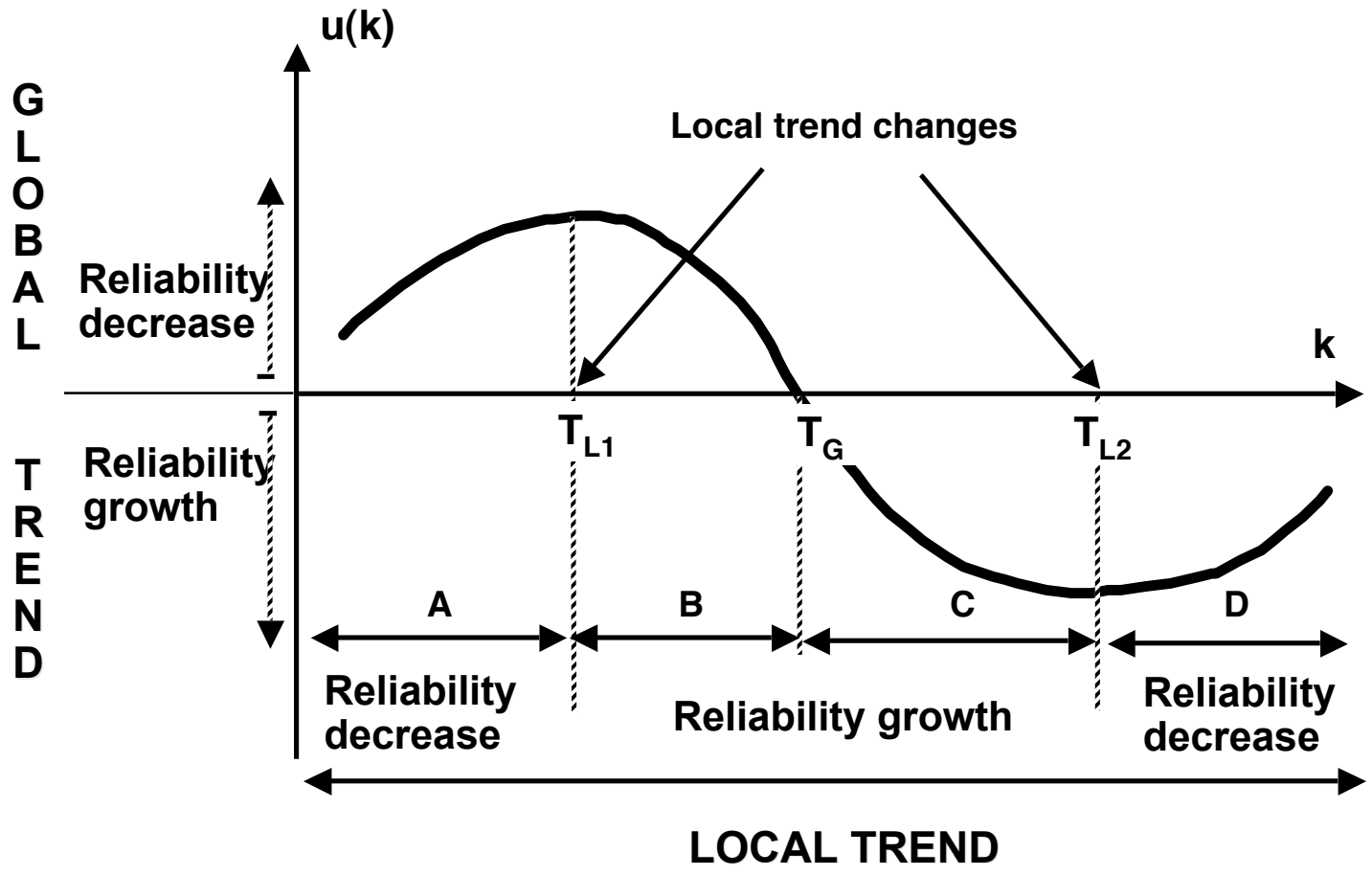
$$u(T) = \frac{\sum_{i=1}^{k} (i-1)\, n(i) \; - \; \frac{k-1}{2} \sum_{i=1}^{k} n(i)}{\sqrt{\frac{k^2-1}{12} \sum_{i=1}^{k} n(i)}}$$

n(i) = # failure during time unit i

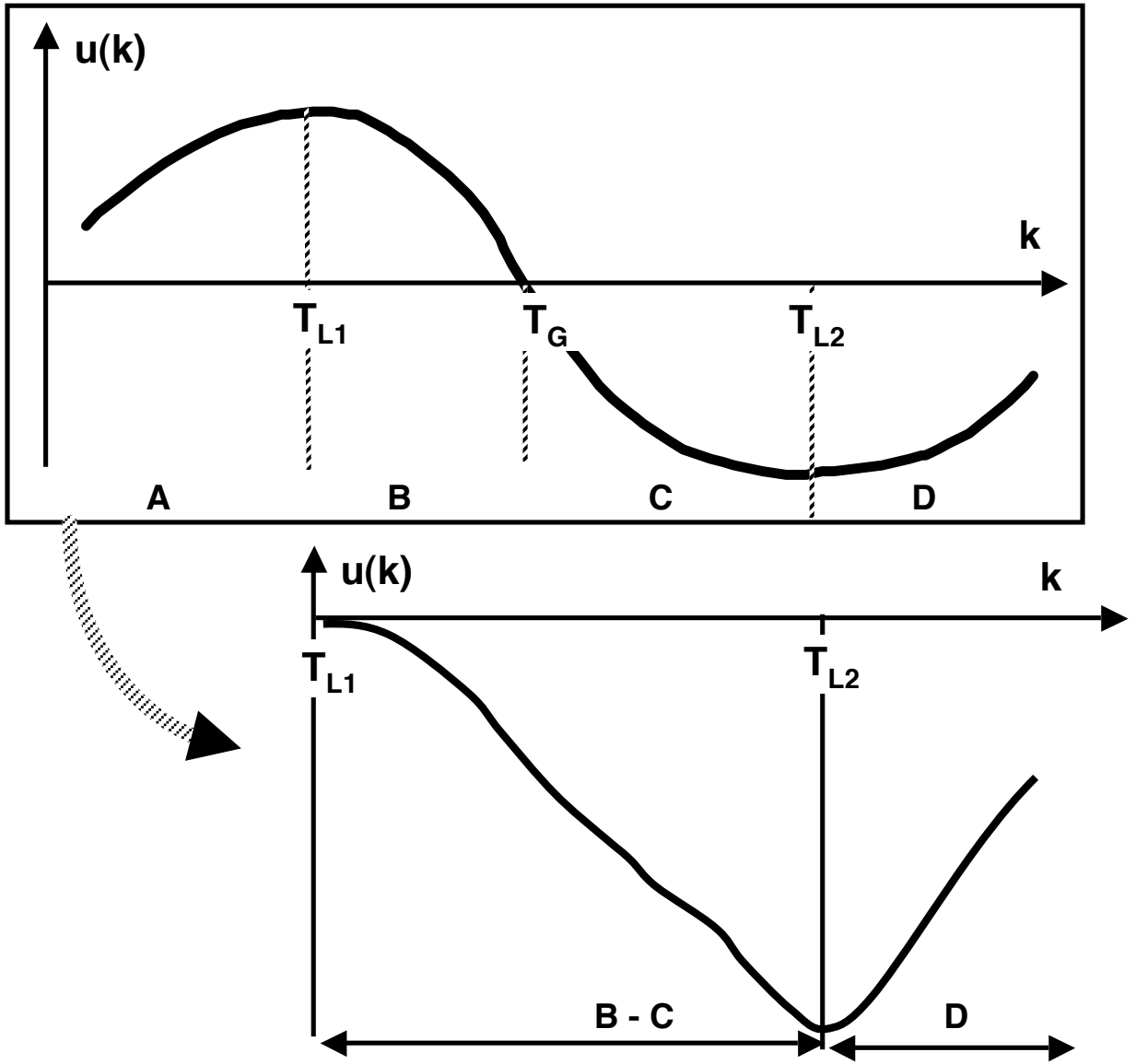
- Can be put in the form:

$$u(T) = - \frac{a_{\mathcal{H}}(T)}{T \sqrt{\frac{1}{12\, N(T)}}}$$

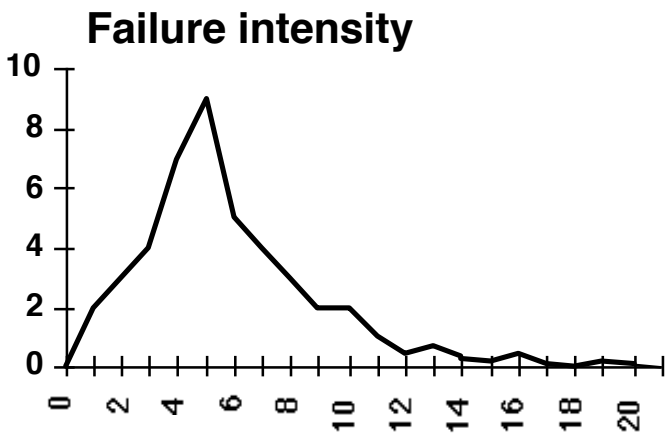
Laplace factor: local and global trend

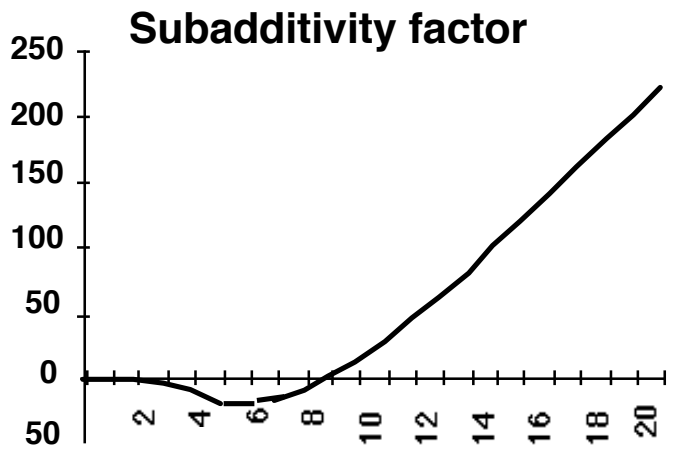


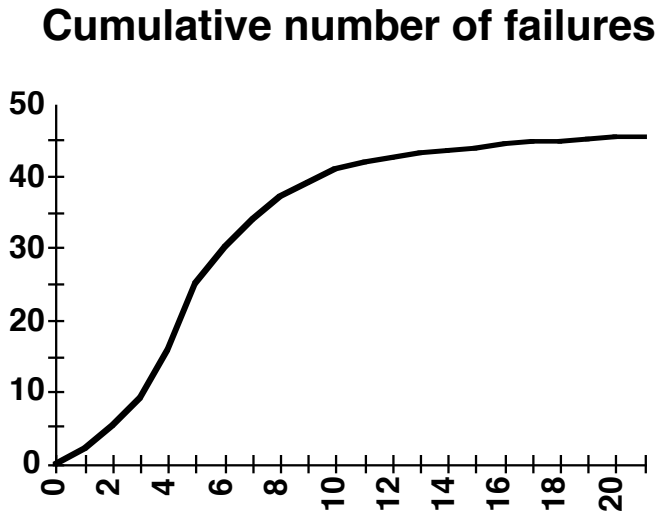
Change of time origin

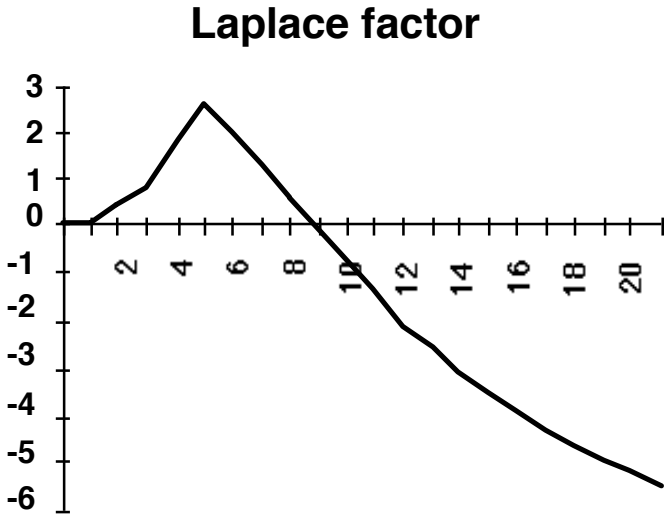


Link : graphical tests - Laplace - Subadditivity

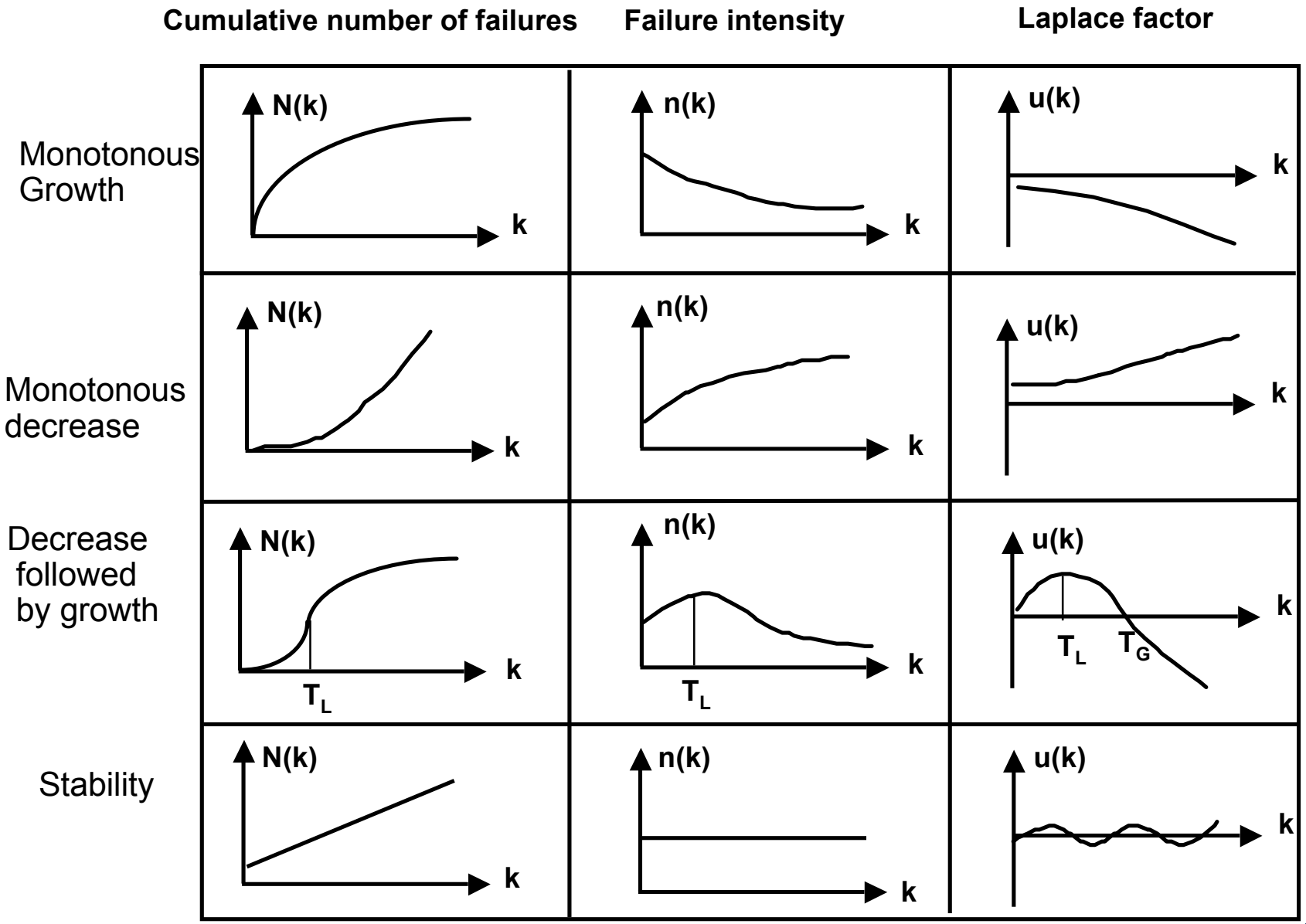








Link between trend indicators



How to use trend test results

- Control of the efficiency of test activities
  - Reliability decrease at the beginning of a new activity: OK
  - Reliability decrease during a relatively long period of time: Pb ?
  - Reliability growth after reliability decrease: OK
  - Sudden reliability growth: caution!
  - Stable reliability: saturation
    - New tests
    - Following phase
    - End of test

- Application of reliability models
  - Trend in accordance with model assumptions

Application to RADC data sets
Rome Air Development Center (USA)

System Id.	# instructions	# programmers	# failures	Type of system
1	21 700	9	136	RT.C
2	27 700	5	54	RT.C
3	23 400	3	38	RT.C
4	33 500	6	54	RT.C
5	2 445 000	7	831	RT.Com.
6	5 700	275	73	RT.C
7 (14C)	*****	8	36	military
8(17)	61 900	110	38	military
9(27)	126 100	8	41	military
10(40)	180 000	8	101	OS
11A(SS1A)	*****	8	112	OS
11B (SS1B)	*****	unknown	375	OS
11C (SS1C)	*****	unknown	277	TS
12 (SS2)	*****	unknown	192	WP
13 (SS3)	*****	unknown	278	OS
14 (SS4)	*****	unknown	196	

RT: Real-time
C: control
Com. : commercial
WP: word Processing
TS : Time sharing
OS: Operating system
***** : not given

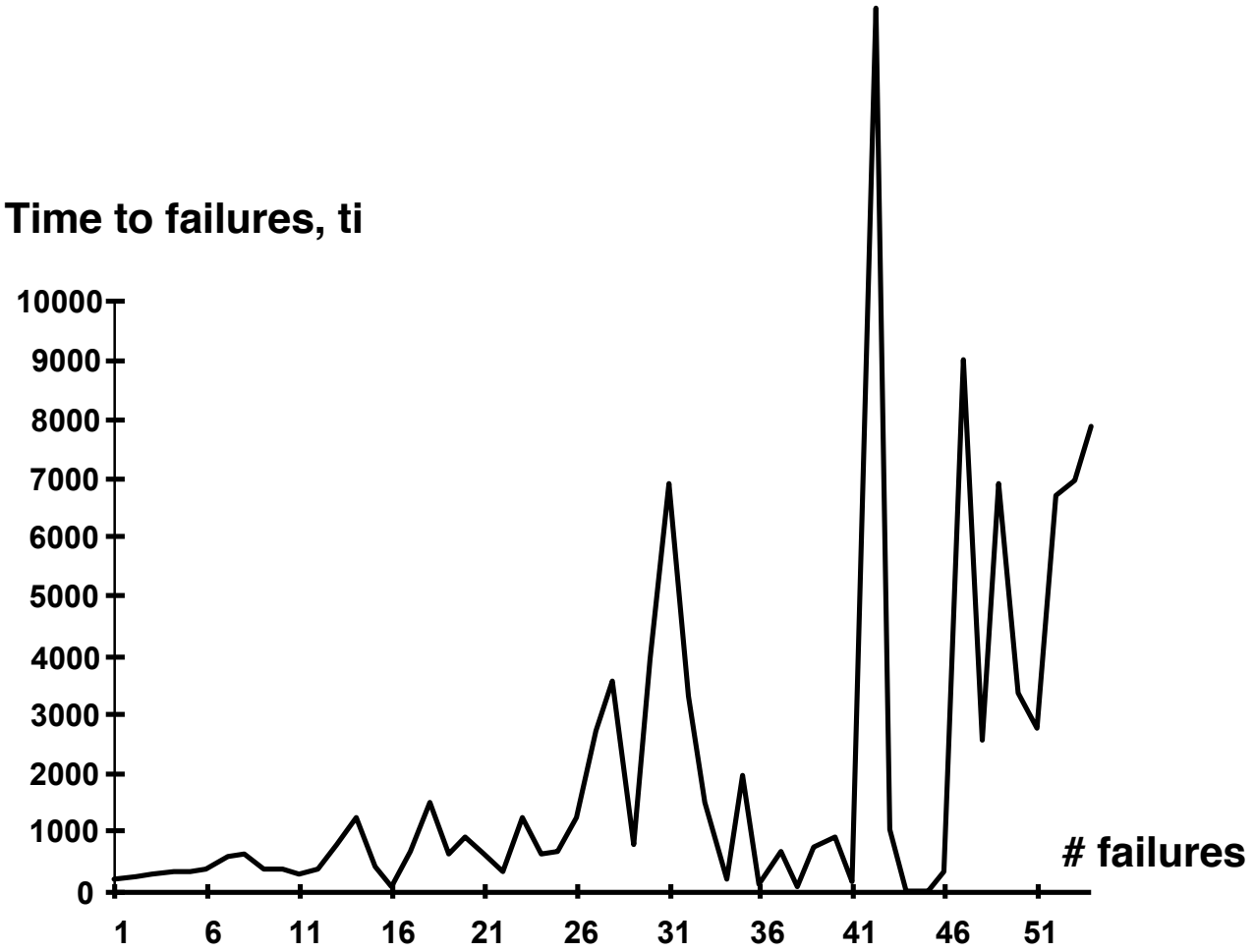
[John Musa], "Software Reliability Data", Rome Air Development Center, NY, USA, 1979

Laplace factor

System	Laplace factor
1	- 9,10
2	- 5,73
3	- 6,13
4	- 8,59
6	- 3,64
7	- 2,14*
8	- 4,65
9	- 5,16
10	- 9,60
11A	- 1,36*
11B	- 0,73*
11C	- 5,15
12	+ 0,74*
13	- 5,64
14	- 1,78*

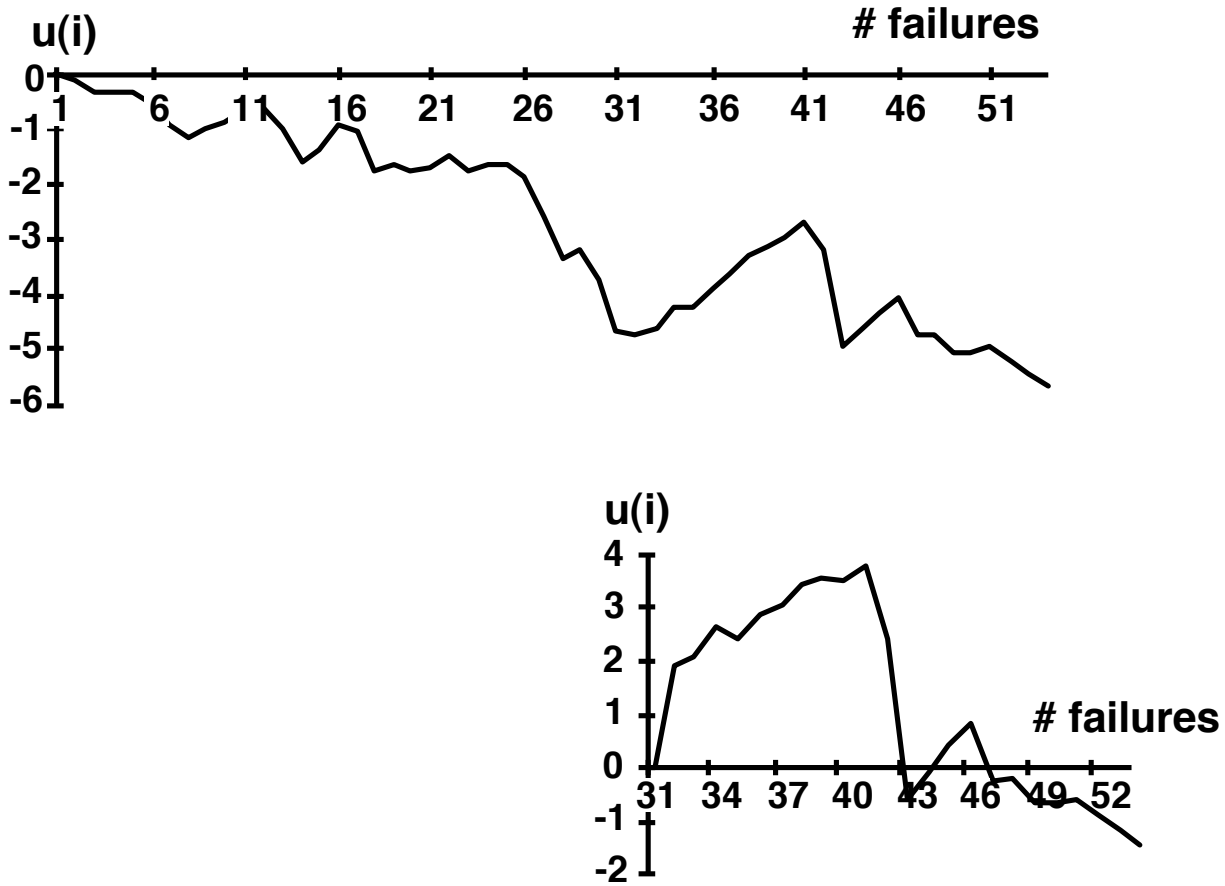
* : stable reliability

System 2: times to failures

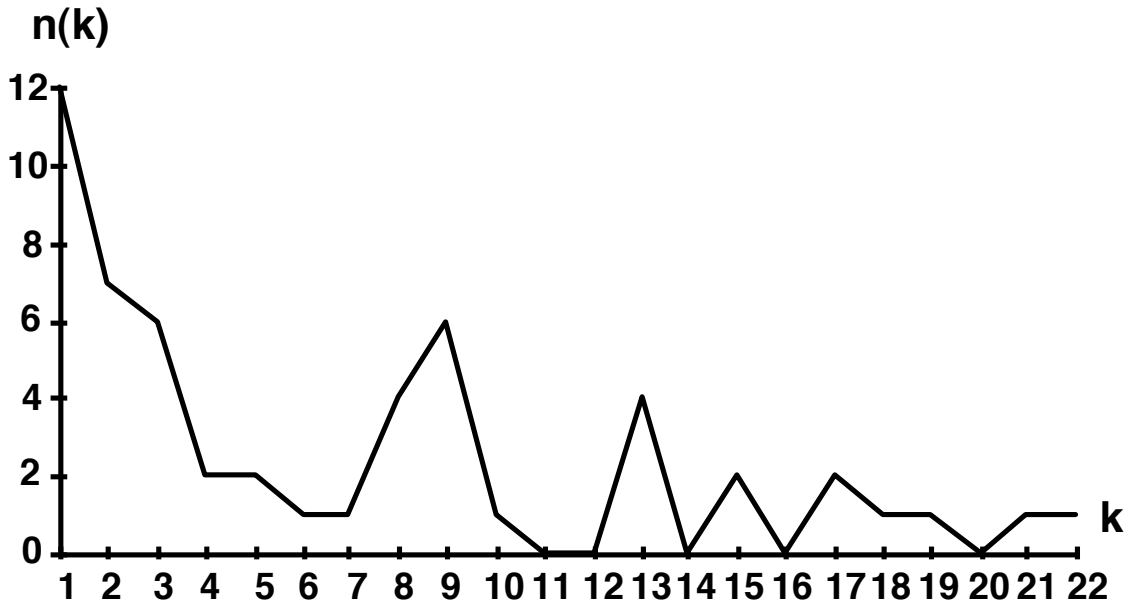


System 2: Laplace factor

- Variable: time to failure



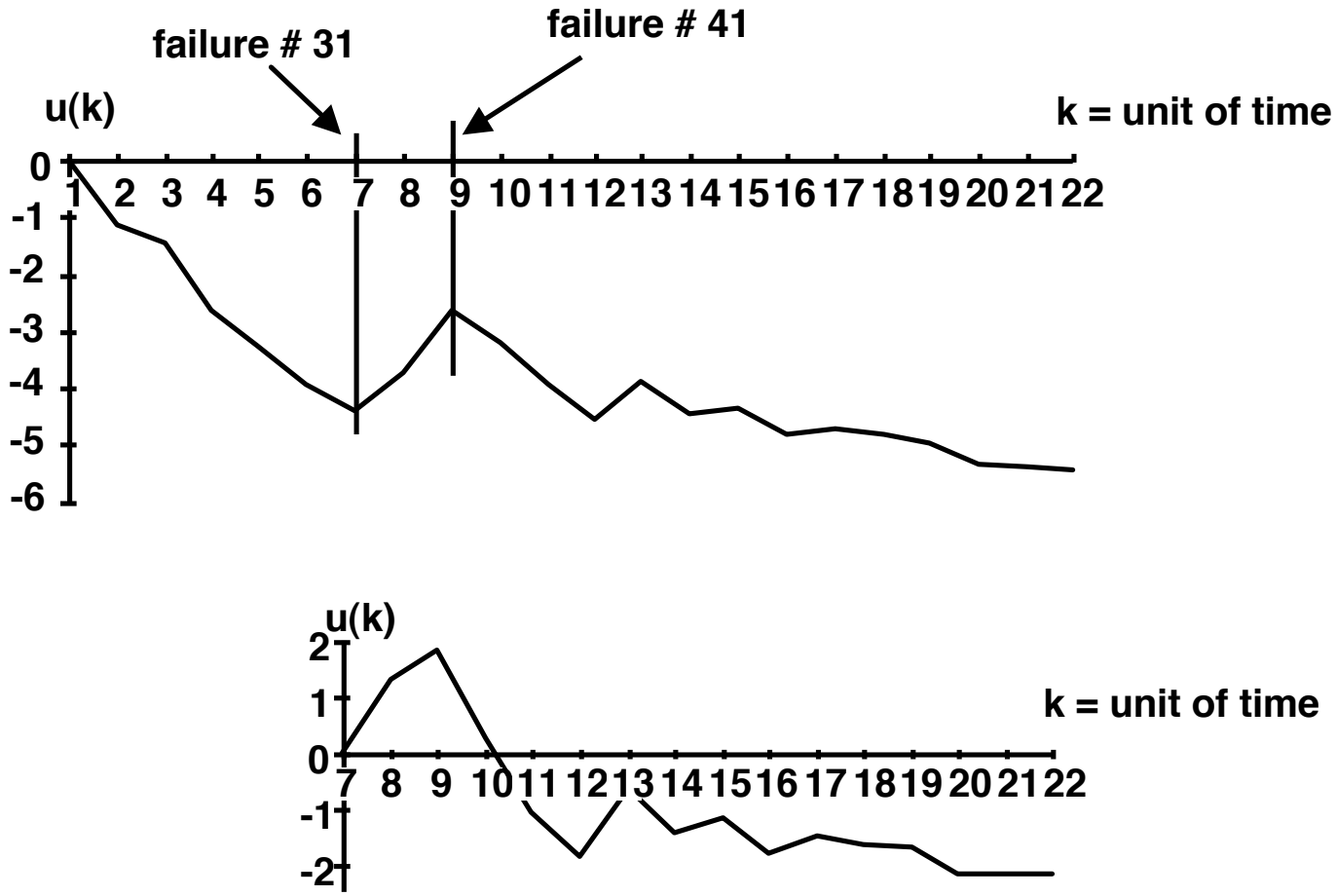
System 2: failure intensity



k: unit of times = 5000 seconds of execution time

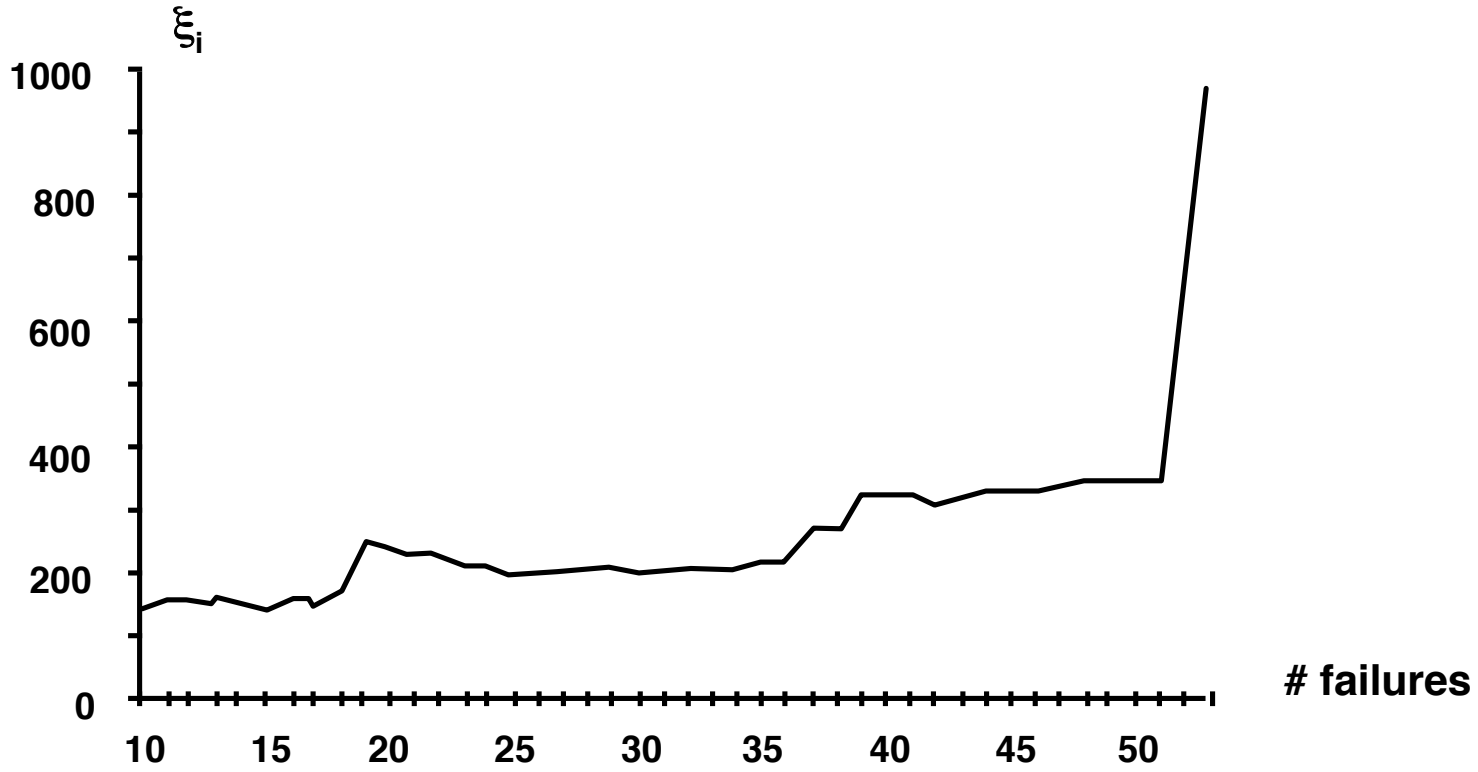
System 2: Laplace factor

- Variable: # failures



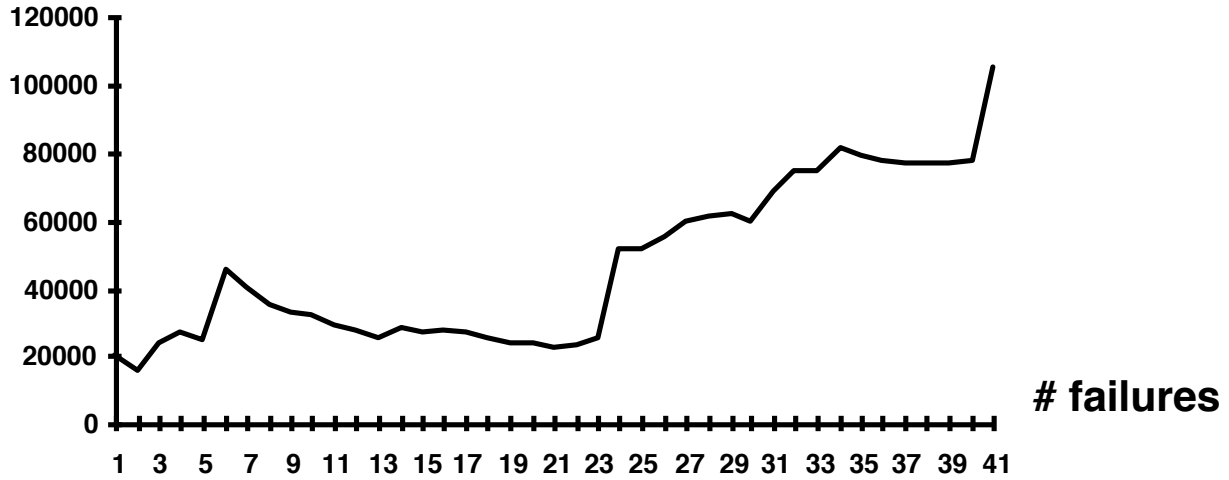
Unit of time = 5000 seconds of execution time

System 4: arithmetical mean

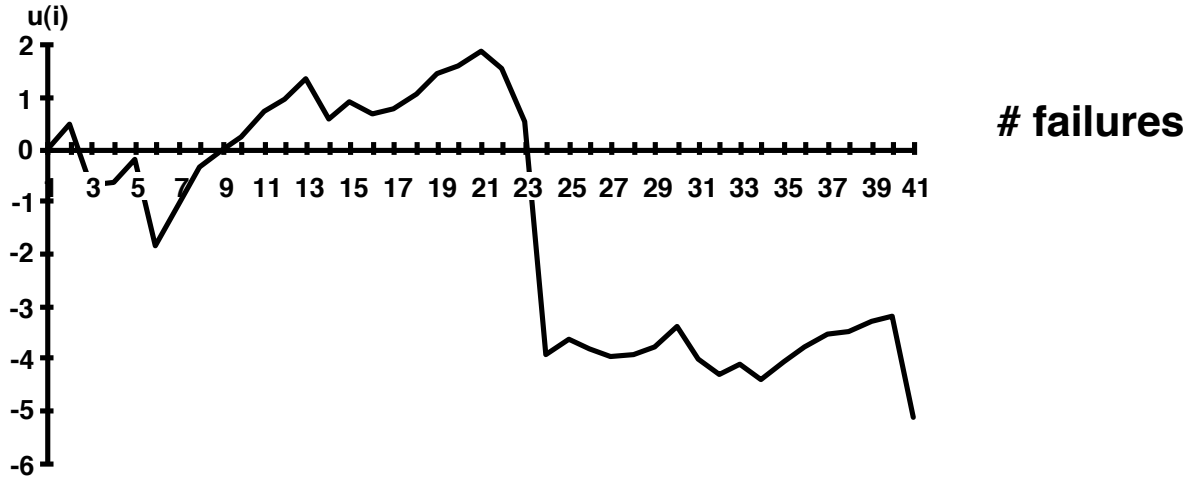


System 9

Arithmetical mean

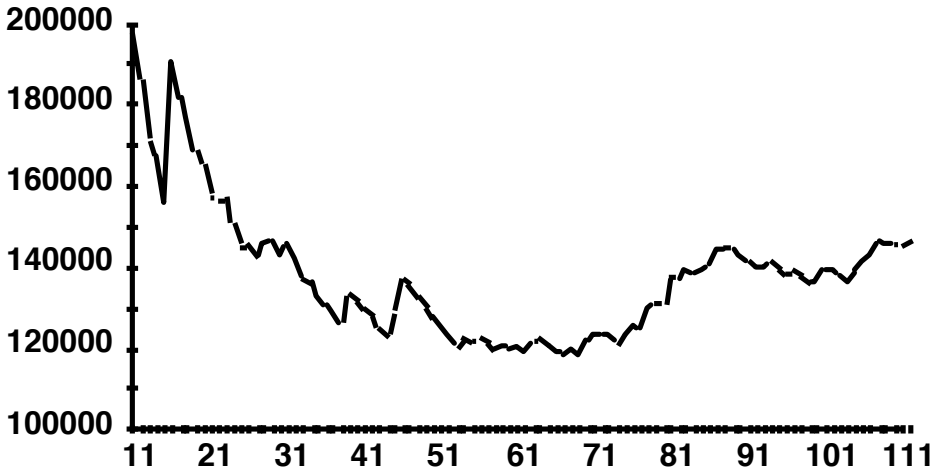


Laplace Factor

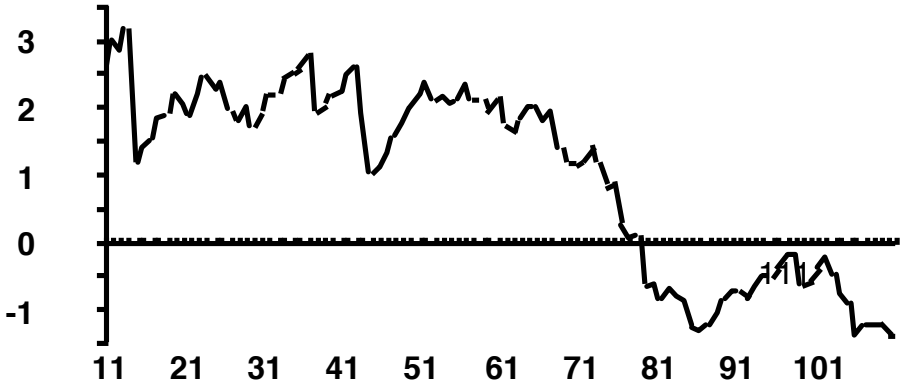


System 11A

Arithmetical mean

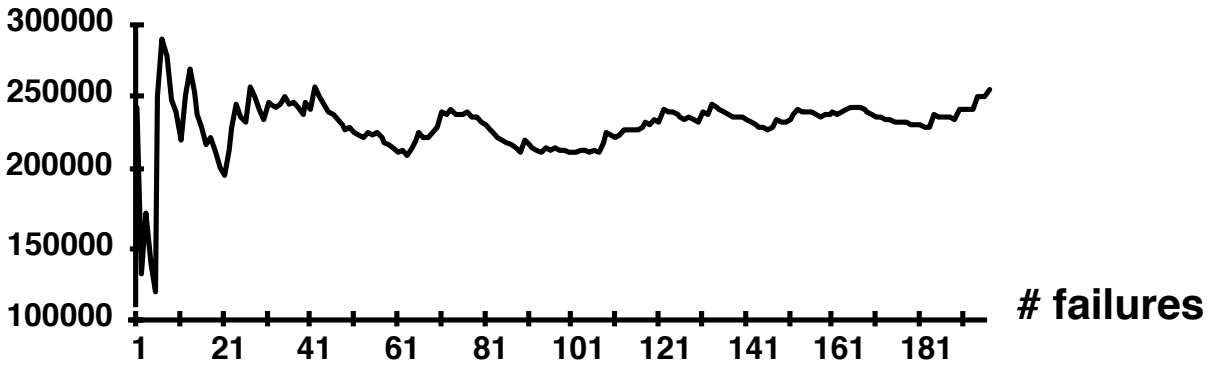


Laplace Factor

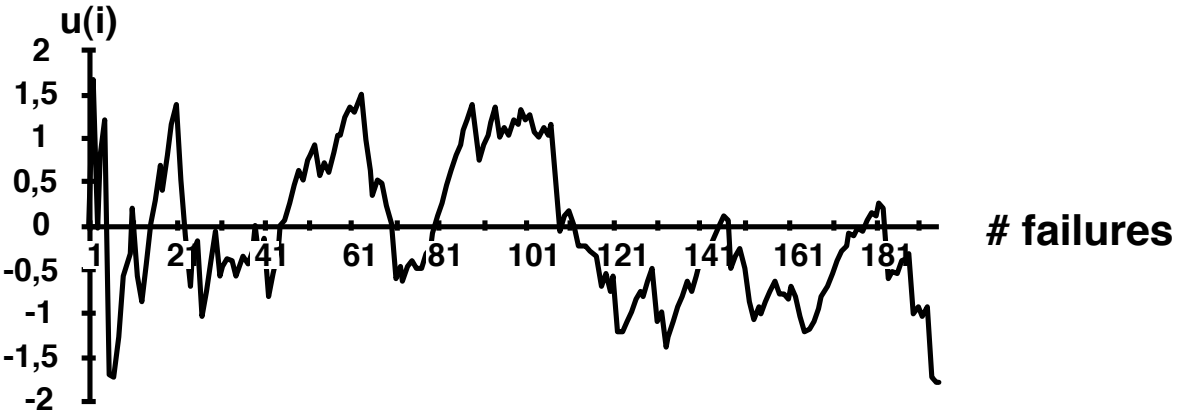


System 14

Arithmetical mean



Laplace factor

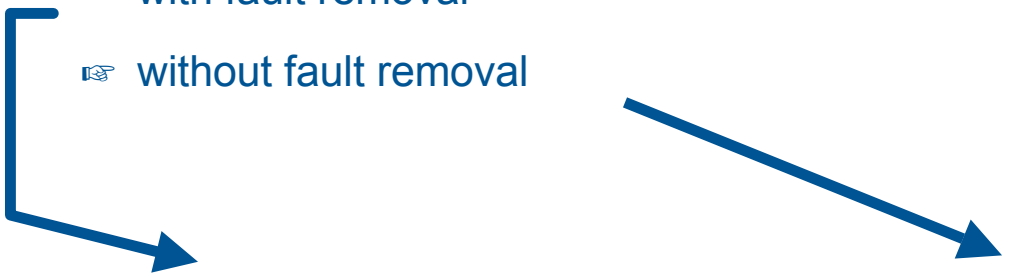


Conclusion

- Some systems can be modeled by an exponential distribution
  - System for which $-2 < u < 2$
- Impact of the operational profile
  - Systems 11 A, B, C are 3 copies of the same program used in different environments
- Benefits from trend analysis
  - Understanding of the underlying processes
  - Follow up of the development process in real-time, fast feedback
  - Helpful for reliability model application

SOFTWARE RELIABILITY EVALUATION

- Objectives
  - Evaluate measures characterizing the software reliability and its evolution
- Methods
  - evaluation from data collected on the software during testing and / or operation
    - with fault removal
    - without fault removal



Reliability growth models

Measures:

- Failure rate
- Failure intensity
- Cumulative number of failures
- MTTF

[See reference 1]

Models in stable reliability

Measures:

- Test duration without failure, required to reach a target reliability
- Probability of accepting/rejecting a piece of software
- Probability of failure in operation

OUTLINE

Reliability growth models

- Presentation of some reliability growth models
- Reliability growth models and trend analysis
- Application of reliability growth models
- Tools

Models in stable reliability

Other approaches

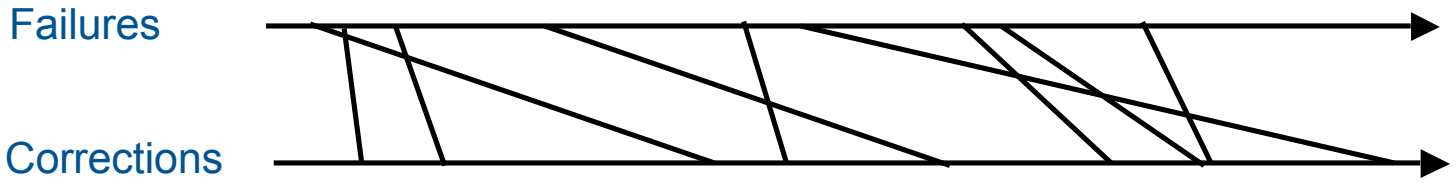
RELIABILITY GROWTH MODELS

Modeling difficulties

- Corrections + specification changes ⇒ varying behavior

  ⇒ absence of repetitive phenomenon ⇒ absence of statistics

- Variations in the usage environment
- No direct relationships between failures and corrections



Objectives of reliability growth models:

Estimation of dependability measures as resulting from the above variations

⇒ restrictive assumption for some models: correction after each failure

RELIABILITY GROWTH MODELS

- Failure rate models
  (Failure rate equations & relationship between successive failure rates)
  - Deterministic, piecewise Poisson Process models: Jelinski Moranda, Musa
  - Stochastic, doubly stochastic process model: Littlewood-Verrall
- Failure intensity models: succession of failures
  (based on Non-Homogeneous Poisson Process (NHPP))
  - Exponential model (Goel Okumoto)
  - Hyperexponential model (Kanoun-Laprie)
  - S-Shaped model (Yamada et al)
- Selection depends on
  - Objectives

    Development follow-up, evaluation of operational MTTF and residual failure rate
  - **Trend displayed by the data set**

Jelinski Moranda model: assumptions

- First software reliability model (1972)
- Assumptions

  H1 : the total number of faults is finite (N_0)

  H2 : No fault introduction while correcting detected faults: each activated fault is corrected before new executions

  H3 : Faults are independent and their manifestation rate is constant

  H4 : Inputs are selected randomly and tests are representative of operational profile

  H5 : All failures are observed

Jelinski Moranda model: equations

- Parameters

  N_0 = total number of faults

  Φ = fault manifestation rate

  $\lambda(i)$ = failure rate of the i-th failure

  Ti = random variable: time between failures i-1 and i (observation = ti)

- Relations

  $\lambda(i) = \Phi [N_0 - (i - 1)] = di / dt \qquad i = 1, 2, \ldots, N_0$

  Prob. $(Ti < ti) = \Phi (N_0 - i + 1]. \exp \{ \Phi (N_0 - i + 1).ti\}$

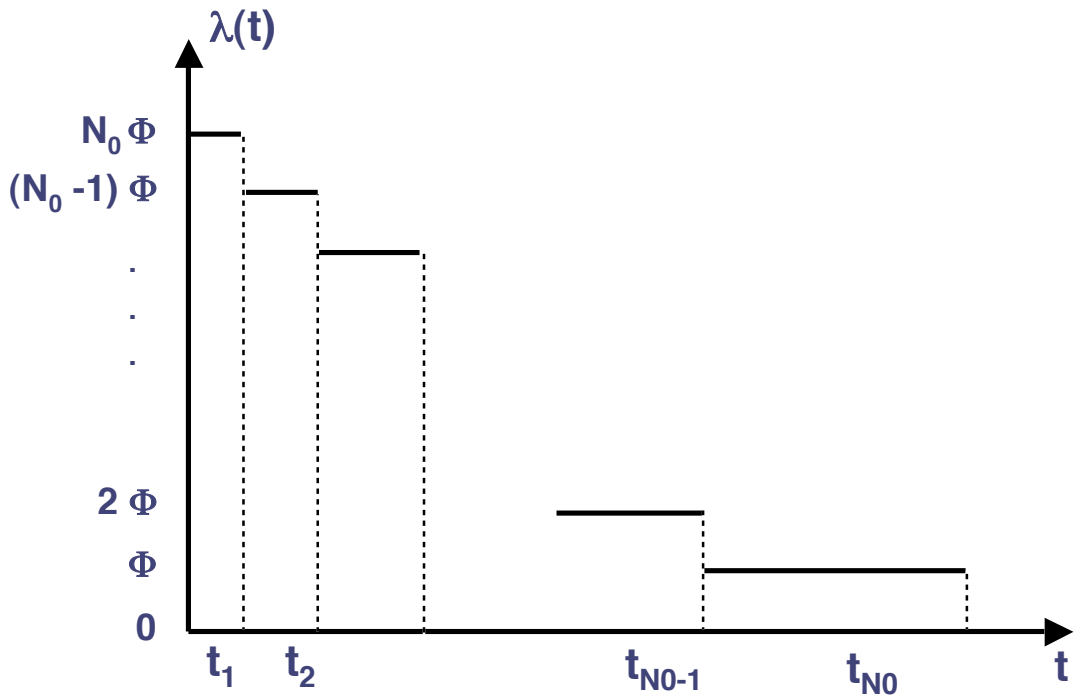
  $$MTTF_i = \frac{1}{\lambda(i)} = \frac{1}{\Phi [N_0 - (i - 1)]}$$

  $N(t) = N_0 [1 - \exp (-\Phi t)]$ = number of faults detected at t

- Parameters to be estimated: N_0 , Φ

Jelinski-Moranda model: λ(t)

- the failure rate is constant and tends to 0 when t tends to ∞



Musa model

- Assumptions similar to the Jelinski-Moranda model
- Parameters definition

  M_0 = number of faults in the software

  N_0 = number of failures

  B = fault reduction factor: number of faults / number of failures $M_0 = B.N_0$

  C = compression factor (execution time in operation / in test)

  Φ = fault manifestation rate

- Relations

  $\lambda(i) = B\ C\ \Phi\ (N_0 - i + 1)$ $\qquad$ $MTTF(i) = \dfrac{1}{B\ .\Phi.(N_0 - i + 1)}$

  $N(t) = N_0\ [\ 1 - \exp(-B\ C\ \Phi\ t)\ ]$ = number of failures observed at t (execution time)

- Parameters to be estimated: N_0 , Φ (B product characteristics; C operational profile)

Littlewood-Verrall model

- Stochastic relationship between the successive failure rates
- Distinction
  - Input uncertainty: λ_i
  - Impact of corrections uncertainty: λ_1 , λ_2 , …, λ_i series of random variables
- Randomness of inputs

  $f(T_i \mid \lambda_i) = \lambda_i \exp.(-\lambda_i t)$

  f : probability density function (pdf)

  T_i : time to failure i since failure i-1 (time to failure i)
- Impact of corrections

$$f(\lambda_i \mid \alpha, \Psi) = \frac{[\Psi(i)]^{\alpha} \lambda_i^{\alpha-1} \exp.(\Psi(i))\lambda_i}{\Gamma(\alpha)}$$

Ψ : programmer skill and programming difficulty

Littlewood-Verrall model

- Distribution of Ti

$$f(t_i|\alpha,\Psi) = \int_0^{\infty} f(t_i|\lambda_i).\, f(\lambda_i|\alpha,\Psi)\, d\lambda_i = \frac{\alpha\,[\Psi(i)]^{\alpha}}{[t_i+\Psi(i)]^{\alpha+1}}$$ Pareto distribution

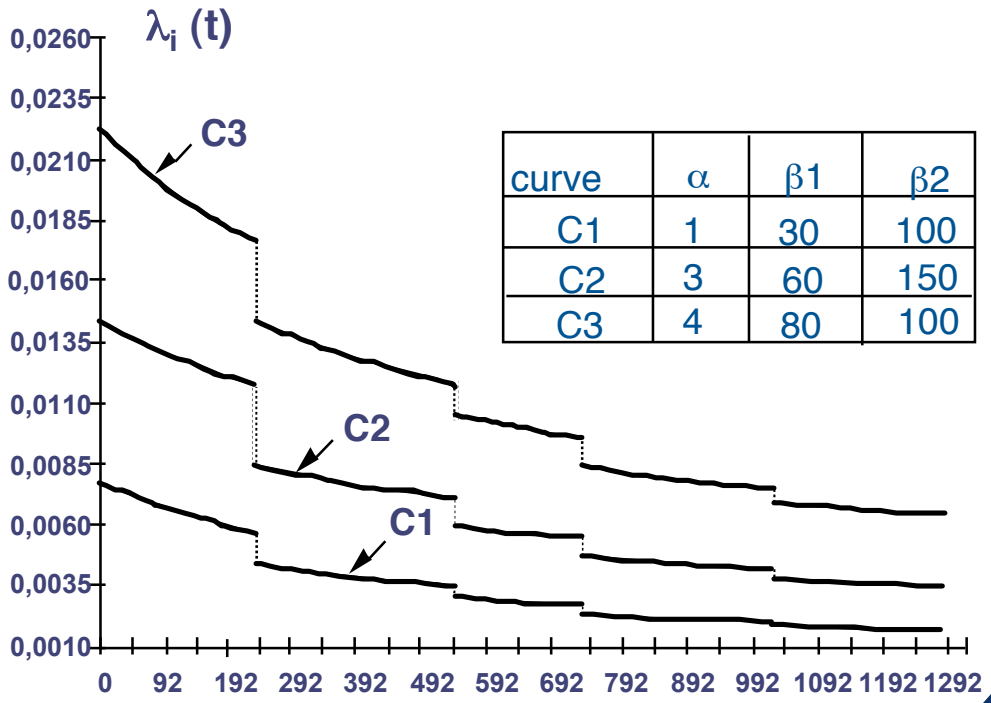
- Reliability growth represented by growth of $\Psi(i)$:

$$\Psi(i) = \beta_1 + \beta_2 \cdot i$$

$$\lambda_i(t) = \frac{\alpha}{t+\Psi(i)}$$

$$MTTF_i = \frac{\Psi(i)}{\alpha - 1}$$

Parameters: α , β_1 , β_2



curve	α	β1	β2
C1	1	30	100
C2	3	60	150
C3	4	80	100

NHPP models

☞ Based on Non-homogeneous Poisson Process (NHPP)

- Definition
  - ☞ $P\{ N(t+dt) - N(t) = 1 \} = h(t)\, dt$
  - ☞ $P\{ N(t+dt) - N(t) \geq 2 \} = o(dt)$
  - ☞ $[N(t_0)], [N(t_1) - N(t_0)] , \ldots, [N(t_n) - N(t_{n-1})], \quad t_0 < t_1 < \ldots < t_n$ are random variables with independent increments

- Properties
  - ☞ number of events on $[t_1, t_2]$

$$E[N(t_2) - N(t_1)] = \int_{t_1}^{t_2} h(t)\, dt = H(t_2) - H(t_1)$$

$$\text{Prob. } \{N(t) = n \mid N(t_0) = n_0 \} = \frac{[H(t) - H(t_0)]^{n - n_0}}{(n - n_0)!} \exp \{- [H(t) - H(t_0)]\} \quad n > n_0 \text{ and } t > t_0$$

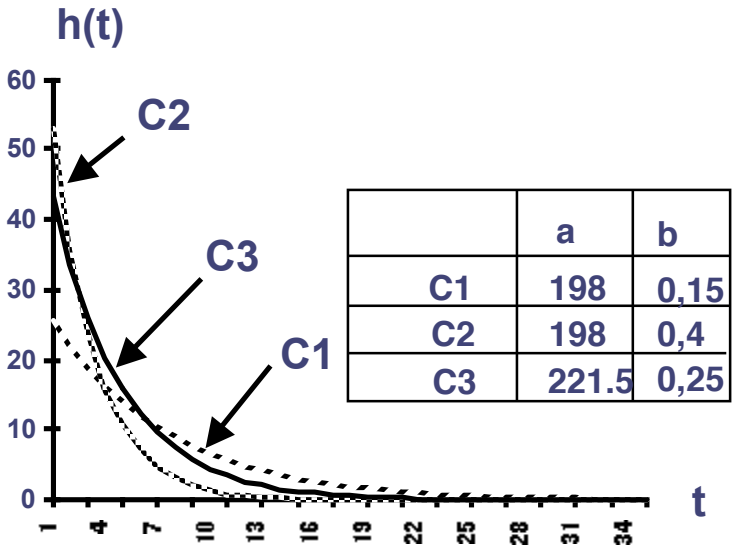
Exponential Model (EXP)

- Failure intensity

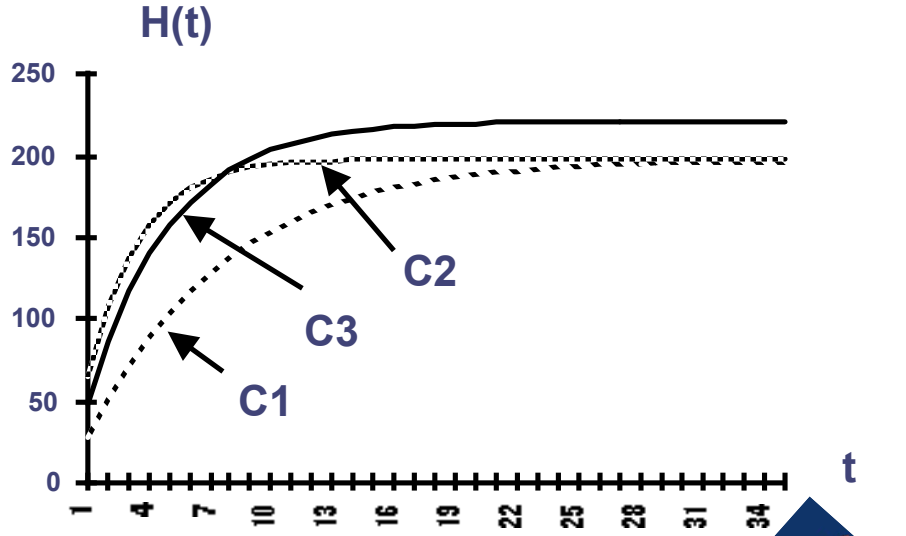
  $h(t) = a\, b \exp(-bt)$ parameters to be estimated: a, b

- Cumulative number of failures

  $H(t) = a\,[\,1 - \exp(-bt)]$



	a	b
C1	198	0,15
C2	198	0,4
C3	221.5	0,25

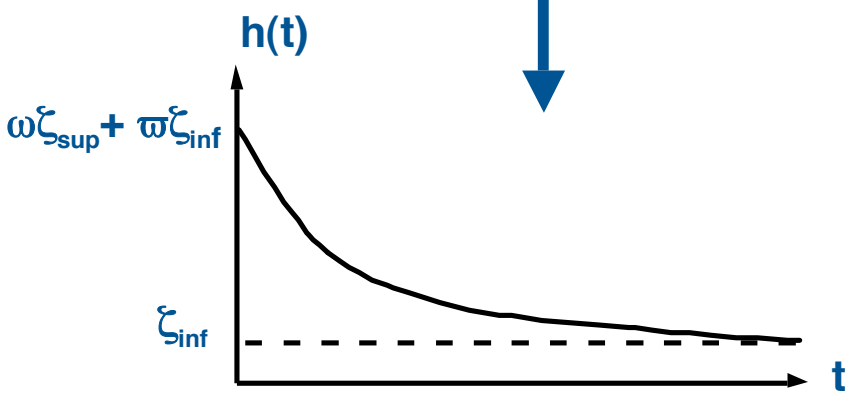


Hyperexponential Model (HE)

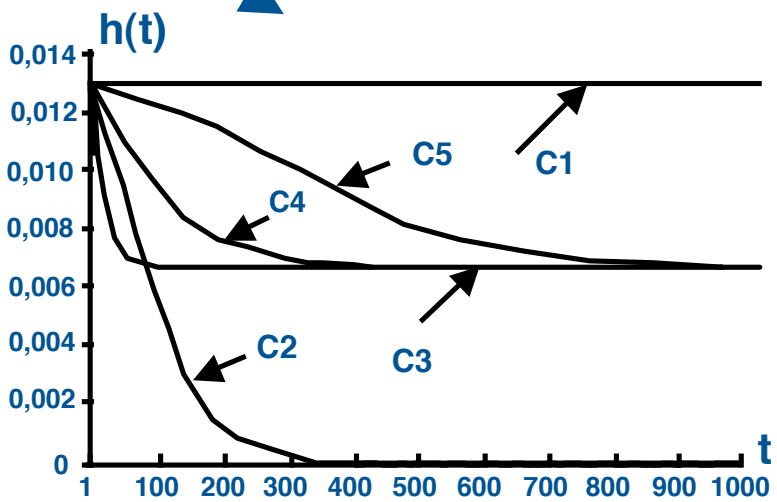
☞ Failure intensity

$$h(t) = \frac{\omega\,\zeta_{sup}\,e^{-\zeta_{sup}t} + \omega\,\zeta_{inf}\,e^{-\zeta_{inf}t}}{\omega\,e^{-\zeta_{sup}t} + \varpi\,e^{-\zeta_{inf}t}}$$

$0 \le \omega \le 1$, $\omega+\varpi = 1$ and $\zeta_{inf} \le \zeta_{sup}$



variation of parameters



ζ_{inf} = **residual failure rate**

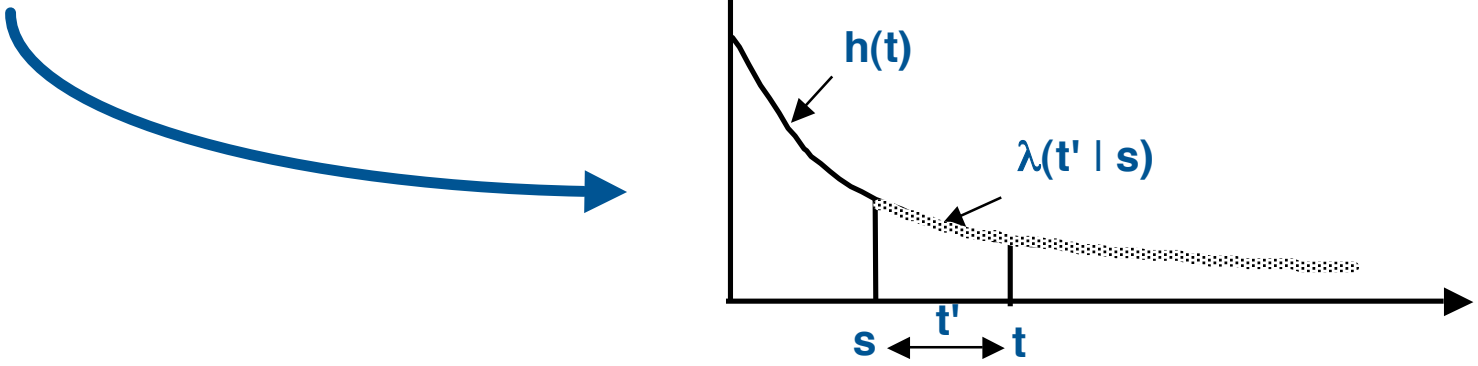
Hyperexponential Model (HE)

- $H(t) = E[N(t)] = -\ln[\ \omega e^{-\zeta_{sup} t} + \varpi e^{-\zeta_{inf} t}]$

$$MTTFi = \frac{\omega \zeta_{sup} e^{-\zeta_{sup} s} + \omega \zeta_{inf} e^{-\zeta_{inf} s}}{\omega e^{-\zeta_{sup} s} + \varpi e^{-\zeta_{inf} s}}$$

- $\lambda(t' \mid s) = h(s+t)$

  s = time of occurrence of failure i



Parameters to be estimated: ω , ζ_{inf} , ζ_{sup}

S-Shaped model (SS)

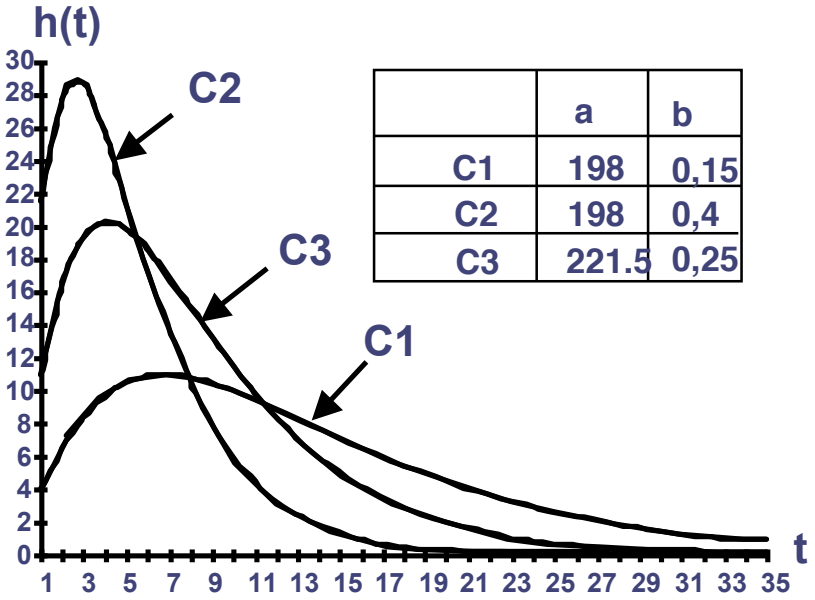
- Failure intensity

$$h(t) = a\, b^2\, t \exp(-bt)$$

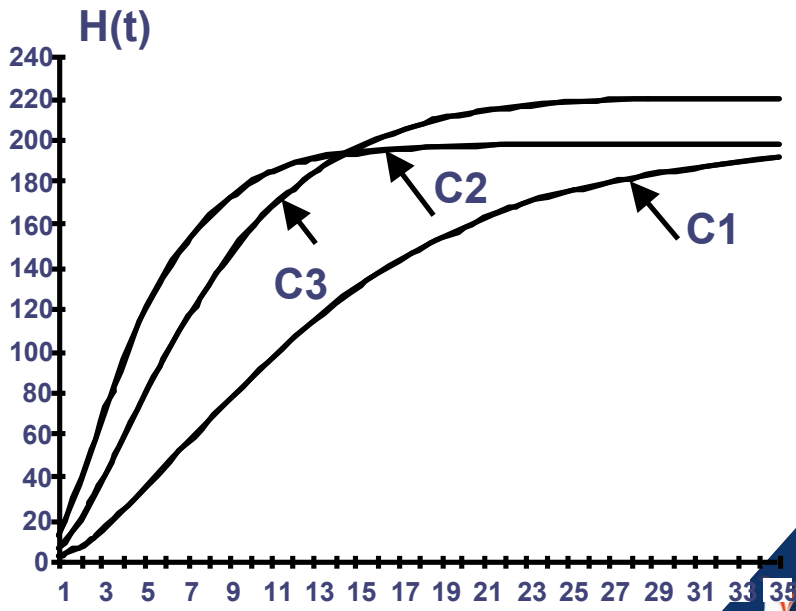
parameters to be estimated : a, b

- Cumulative number of failures

$$H(t) = a\,[\,1 - (1 + b\,t\,)\exp(-bt)]$$



	a	b
C1	198	0,15
C2	198	0,4
C3	221.5	0,25



Model in practice

- ☞ Pre-processing of failure data

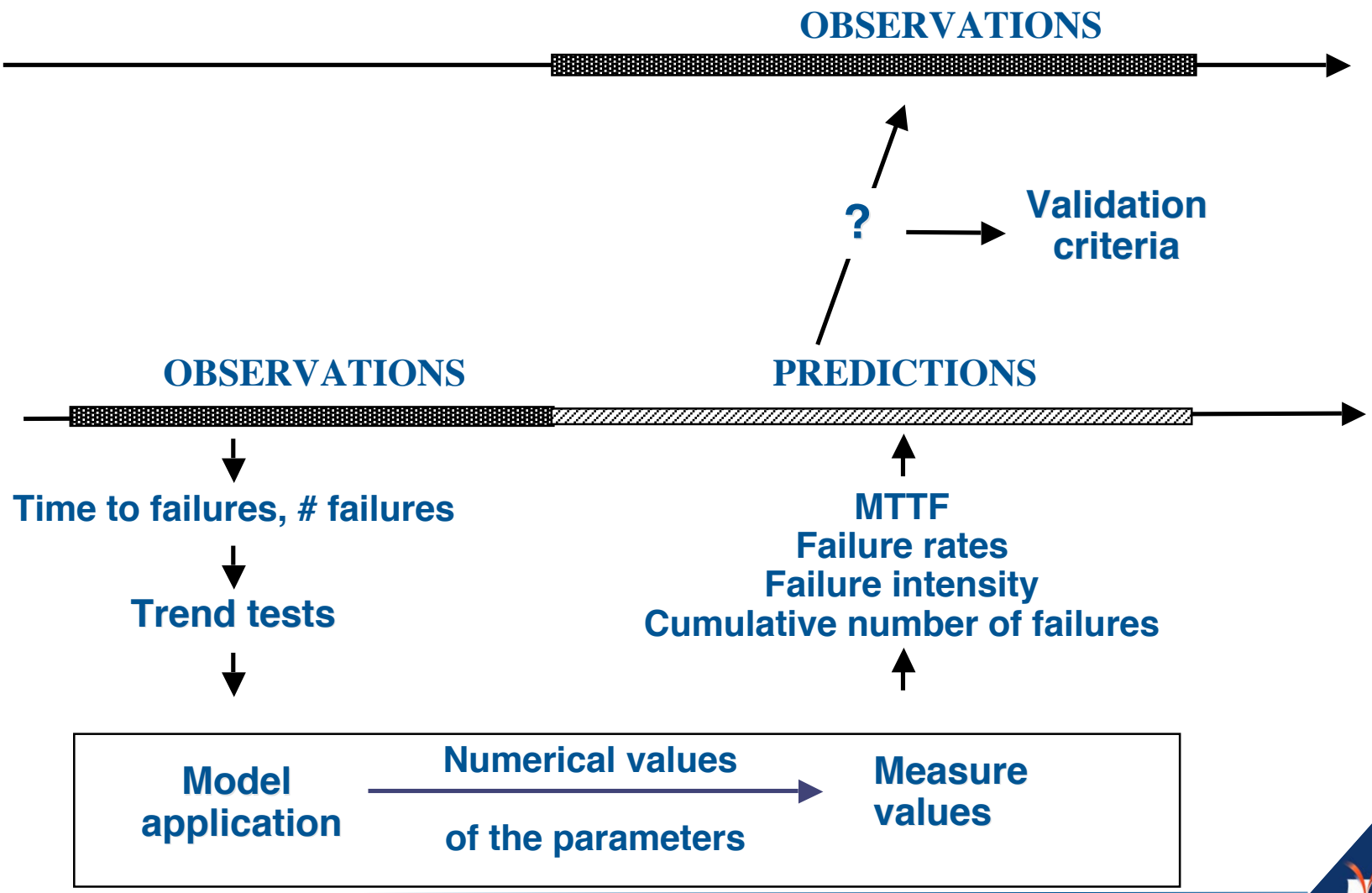
  Trend analysis ⇒ reliability growth ?

- ☞ Parameter determination from observed failure data
  - Inference procedures
- ☞ Prediction of next failure(s)
  - Evaluation of reliability measures based on observed data
- ☞ Model validation ⇒ confidence in evaluation

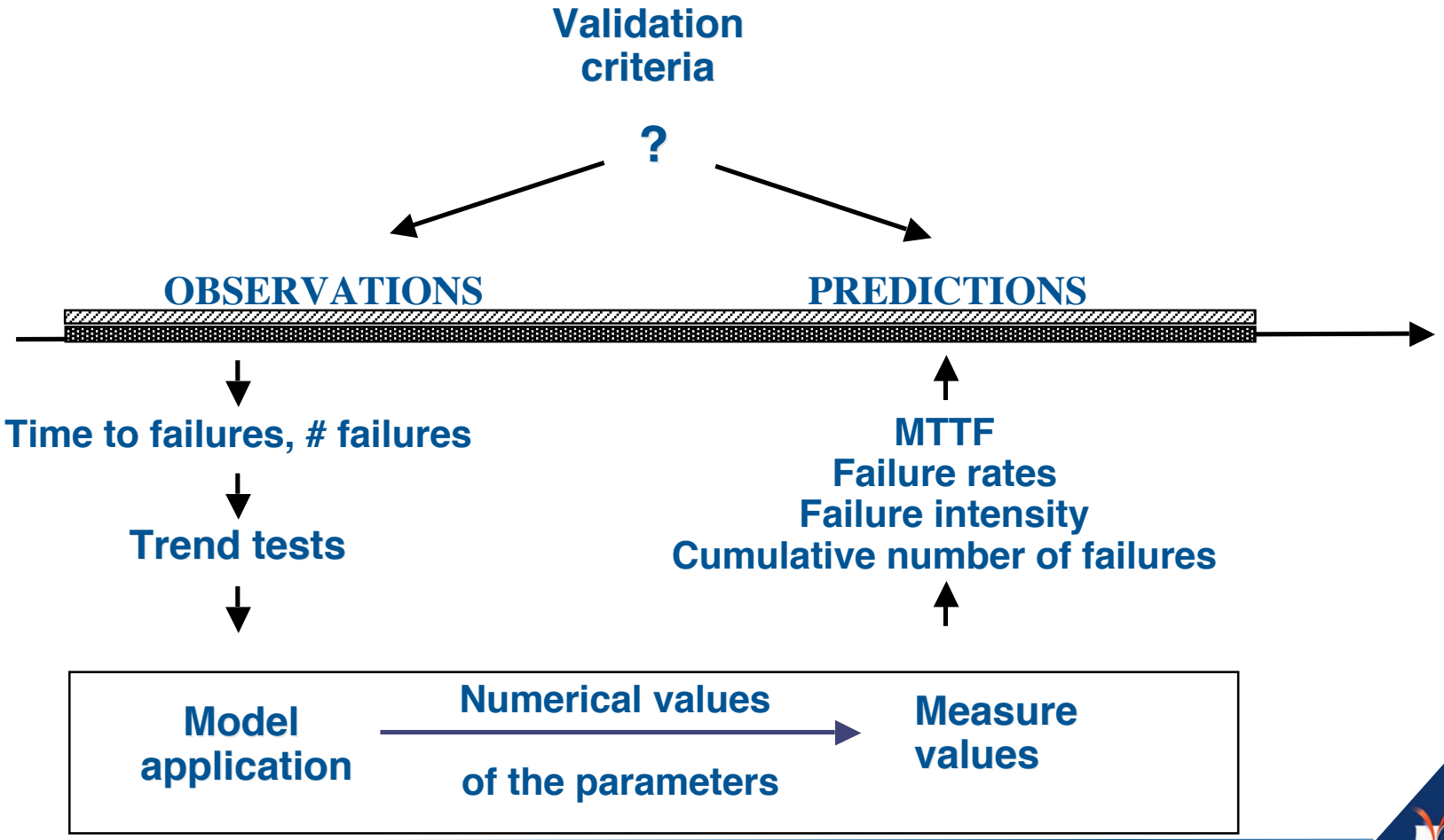
  ⇒ Checking agreement between Predictions / Observations

  - ☞ Predictive analysis
  - ☞ Retrodictive analysis

Model application: predictive analysis

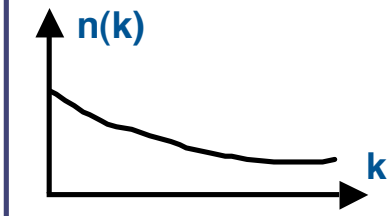
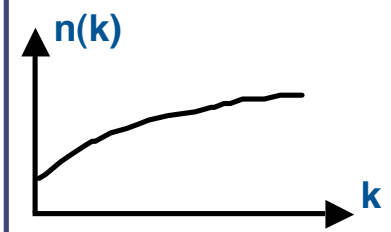


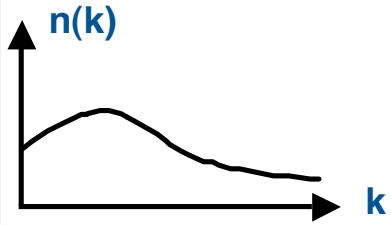
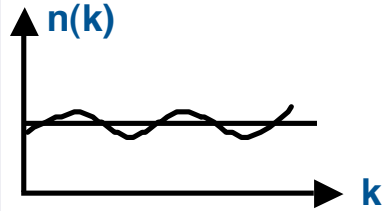
Model application: retrodictive analysis



Trend tests & models

- **Trend test: identification of periods of reliability growth / decrease**
- **Reliability growth models are selected depending on the trend displayed by the observed data set**

Failure intensity	Applicable models
	Models with reliability growth
	

Failure intensity	Applicable models
	Models with reliability decrease followed by reliability growth
	Models in stable reliability

Combined use in real-time of trend tests & models

- Identify the trend
- Apply an appropriate model
- Trust model results as long as the usage conditions are not modified
  - Test of the same function(s)
  - No addition of new users or new sites
  - No specification changes
- In case of significant variation
  - Apply the trend test including the new data items:
    - Reliability growth: trust the previous estimations
    - Reliability decrease: wait for reliability growth
    - Reliability growth after reliability decrease: new data partitioning and application of reliability growth models

Results Validity ?

Unit tests
Static Verification

- Trend analysis

~~Reliability growth models~~

End of Validation

- Trend analysis
  +
- Reliability growth models
  - operational profile ?
  - enough data ?
- Limits: 10^{-3}/h -10^{-4}/h

Operation

- Trend analysis
  +
- Reliability growth models
  or models in stable reliability
  High relevance

Examples:

E10-B (Alcatel ESS):

1400 systems, 3 years

$\lambda = 5\ 10^{-6}$/h — $\lambda_c = 10^{-7}$/h

ABB Atom Nuclear I&C Appli.

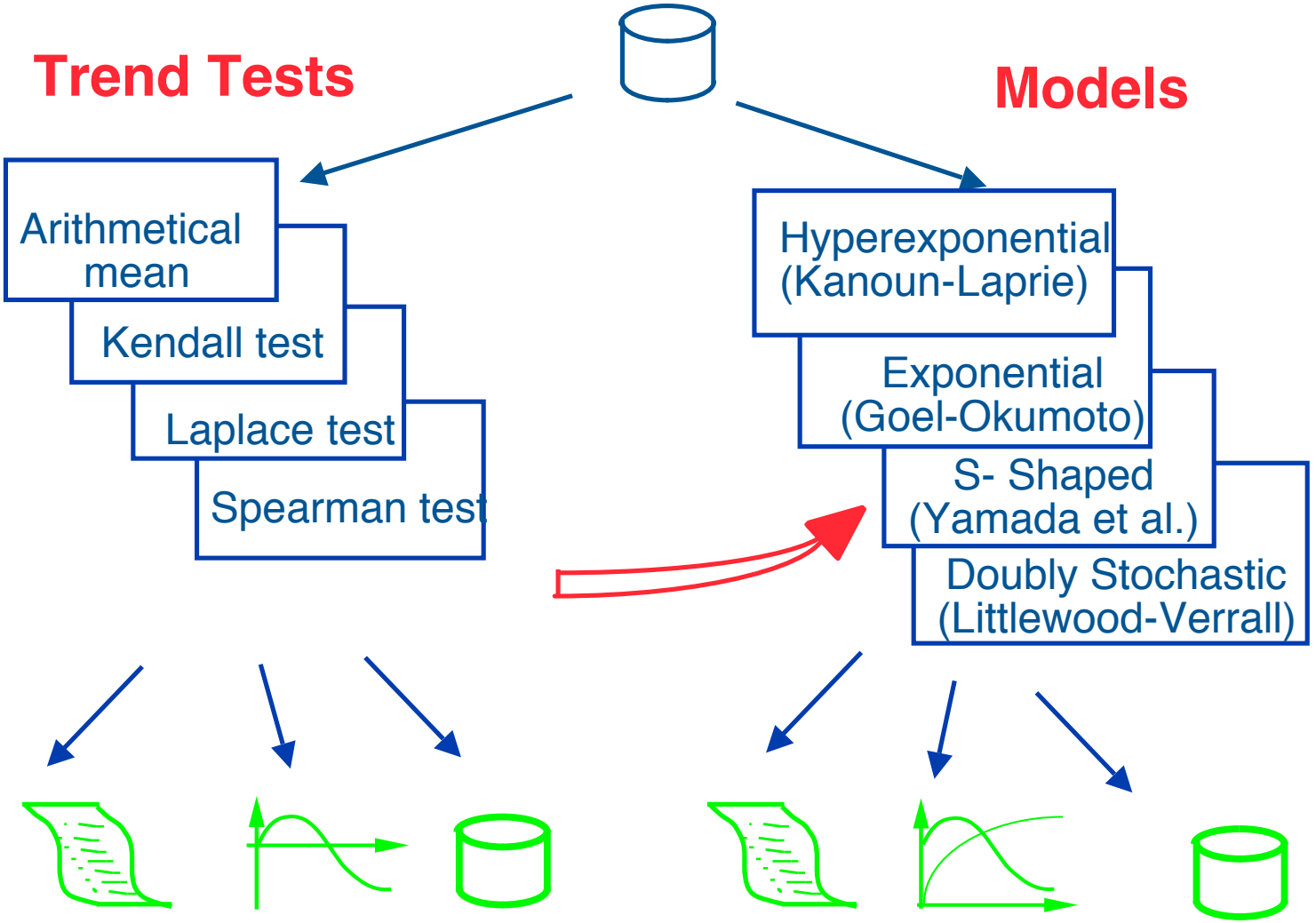
8000 systems, 4 years

λ : $3\ 10^{-7}$ /h — $\lambda_c = 4\ 10^{-8}$/h

Conclusion

- Method
  - Rigorous progressive analysis of the software behavior
  - Deep thoughts about the system and the analyzed data
  - Better results from reliability growth models
- Applicability
  - General method: applicable to hardware design faults
  - Should be integrated to the various phases of the development:
    - early phases: analyses of data and trend tests
    - validation and operational life: application of models (in addition)
- The method has been applied to several real-life systems (hardware and software)
- Needs for tools

Example of Tool: SoRel (developed at LAAS)



[See reference 6]

Experience with SoRel

System	Languages	Volume	Observation	Phases	# Systems	# FR and/or CR
E10-B	Assembler	100 k-bytes	3 years	Val. / Op.	1400	58 FR / 136 CR
TROPICO-R 1500	Assembler	300 k-bytes	27 months	Val. / Op.	15	465 FR/CR
TROPICO-R 4096	Assembler	350 k-bytes	32 months	Val. / Op.	42	210 FR/CR
TROPICO-RS	Assembler	420 k-bytes	47 months	Op.	37	212 FR/CR
TROPICO-RA	CHILL	815 KLOC	68 months	Val. / Op.	146	3063 FR/CR
Telecom. Equipt	PLM-86	5 10^5 inst.	16 months	Val.	4	2150 FR

MODELS IN STABLE RELIABILITY

- Apply when no program evolution nor failure resolution is occurring
- Operational testing (end of validation) — certification

  or when the system is in operation without fault correction
- Residual faults: expected to induce a reduced failure rate
- Two types of inferences
  - Experiments without failures:

    Hypothesis testing evaluate a lower bound on the software reliability

    or an upper bound on the failure probability (for a given confidence level)
  - Experiments with only a few failures observed (all known faults are not fixed)

    1) **Hypothesis testing** (assessment of lower bounds) or

    2) Evaluation of an unbiased estimator of the failure probability per execution

    (the first approach is better when the number of failures is very low)

Reliability evaluation when testing reveals no failure

- Hypothesis testing when testing reveals no failure

  Prob {accepting "$p \leq p_0$" while it is false } $\leq \alpha$

  p = actual probability of failure and p_0 required probability of failure (objectives)

  α = risk error and (1- α) = confidence level

- Amount of execution / time required

  N = number of executions without failure,

  T = test duration without failure

  - Discrete time: $N \geq \dfrac{\ln(\alpha)}{\ln(1-p_0)}$ (results from $(1 - p)^N < \alpha$)

  - Continuous time: $T \geq -\dfrac{\ln(\alpha)}{\lambda_0}$

Measure:

Test duration without failure, required to reach a target reliability objective

Discrete time: Number of program executions without failure

Target failure probability p_0	Risk: α = 10^{-1}	10^{-2}	10^{-3}	10^{-4}
10^{-1}	23	46	69	92
10^{-2}	230	461	691	921
10^{-3}	2303	4605	6908	9210
10^{-4}	23026	46052	69078	92103
10^{-5}	230259	460517	690776	921034
10^{-6}	2302585	4605170	6907755	9210340

Measure:

Test duration without failure, required to reach a target reliability objective

Continuous time: Testing times for some values of λ_0 and α

Target failure rate λ_0 \ Risk α	10^{-1}	10^{-2}	10^{-3}	10^{-4}	Time unit
10^{-1}	1	2	3	4	Days
10^{-2}	10	20	1	1.3	Months
10^{-3}	3.2	6.4	9.6	1	
10^{-4}	2.6	5.3	7.9	10.5	Years
10^{-5}	26.2	52.3	78.9	105.1	
10^{-6}	262.8	525.7	788.6	1051.4	

Other example: stable reliability in operation

- Problem
  - The software system is in operation, some failures have been observed, their consequences are acceptable, even if the faults have been identified
  - Modifications are not performed or, only a few modifications are introduced without perception of any reliability growth / decrease
- Aim

  Evaluate the operational failure rate
- Method

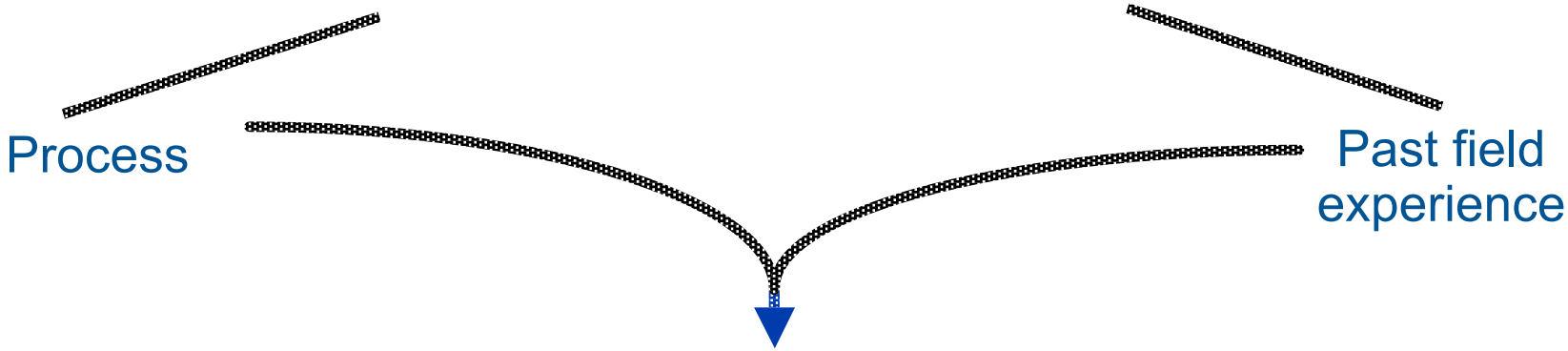
  Constant failure rate, Homogeneous Poisson Process ⇒ Markov process

  Average observed MTTF, associated confidence level

  Usually good results

Product-in-a-process approach

Supplement current approaches to software reliability evaluation with information



Product-in-a-process assessment

Validation of a product= validation of (n+1) th product

with information about: ITSELF + PREVIOUS PRODUCTS

Framework: Bayesian probabilities

θ: conditional probability of failure upon execution / failure rate

Prior and posterior distributions: conjugate distributions

Beta distribution / Gamma distribution

[See reference 4]

$\theta = k_1\ \theta_c + k_2\ \theta_p \qquad k_1 + k_2 = 1$

θ point Bayesian estimate

θ_c conventional estimate (validation of the product in isolation)

θ_p prior estimate (field experience of previous products)

☞ Field produce much more data than validation of new software

$k_2 > k_1 \Rightarrow$ prior estimate dominates conventional estimate

Example:

Satellite control system

$\theta_c = 11.6\ 10^{-3}$/h (6 months)

$\theta_p = 2.8\ 10^{-3}$/h (21 months)

$k_1 = 0.2$; $k_2 = 0.8$

$\Rightarrow\ \theta_p = 4.7\ 10^{-3}$/h

$\Rightarrow$ observed (17 months): $3.9\ 10^{-3}$/h

Conclusion

- Software systems under development and in operation
- With fault removal ⇒ Reliability growth models
  - For several reasons reliability decrease

    (new specifications, environment change, new usage profile, etc.)
  - Identify the trend before model application
  - Good results under certain conditions, for short term objectives
  - Long term objectives ? other new approaches (product-in-a-process approach)
- Without fault removal ⇒ stable reliability
  - Some of the work related to statistical testing could be adapted to operation
  - Two situations: with a few failures or without failures
  - Limitations due to prohibitive test time needed to achieve high reliability objectives
  - Interesting when several systems are under use (example of avionics systems)
  - Test acceleration methods

Off-the shelf software components
Dependability benchmarking

- No information available from component development
- Evaluation based on controlled experimentation

Dependability benchmarking

Evaluation of dependability measures / features
in a non-ambiguous way → comparison

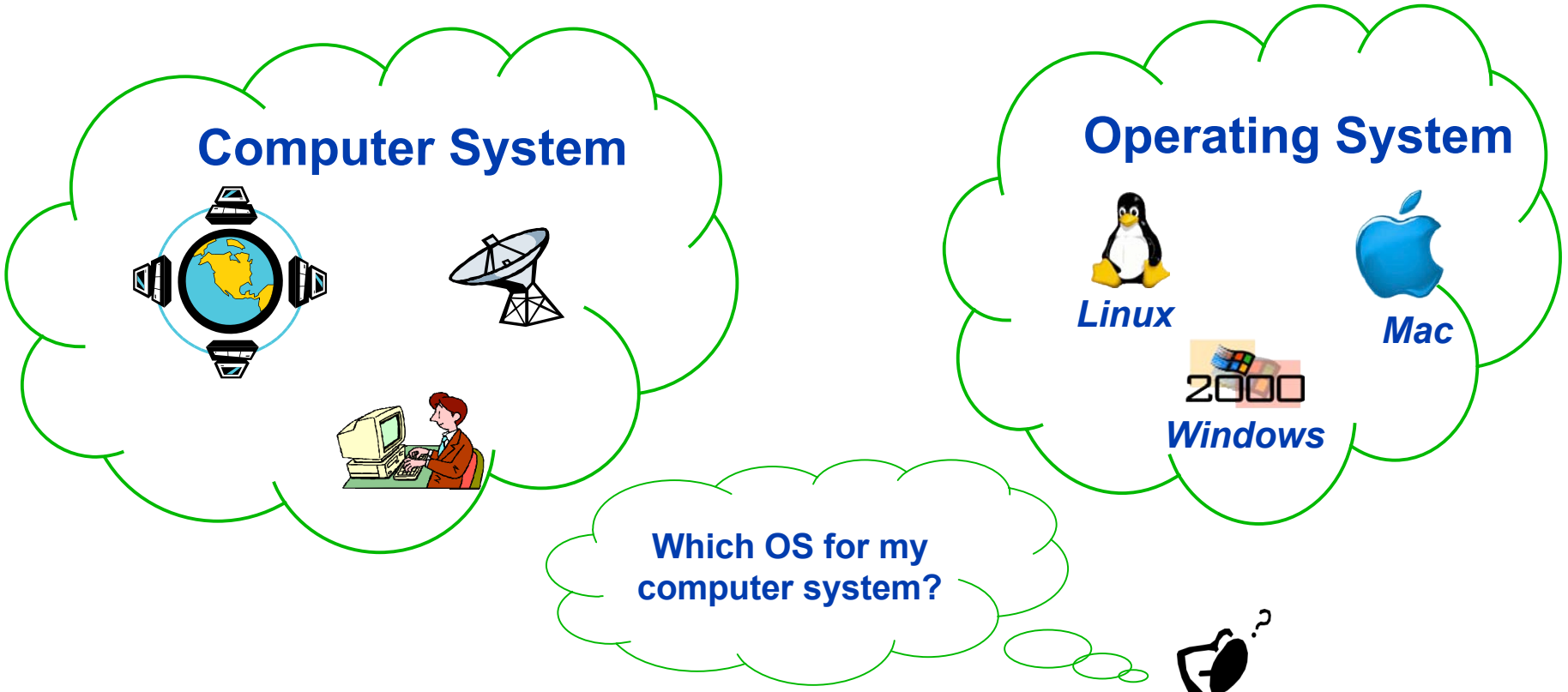
⇓

Properties

Reproducibility, repeatability, portability, representativeness, acceptable cost

[See reference 12]

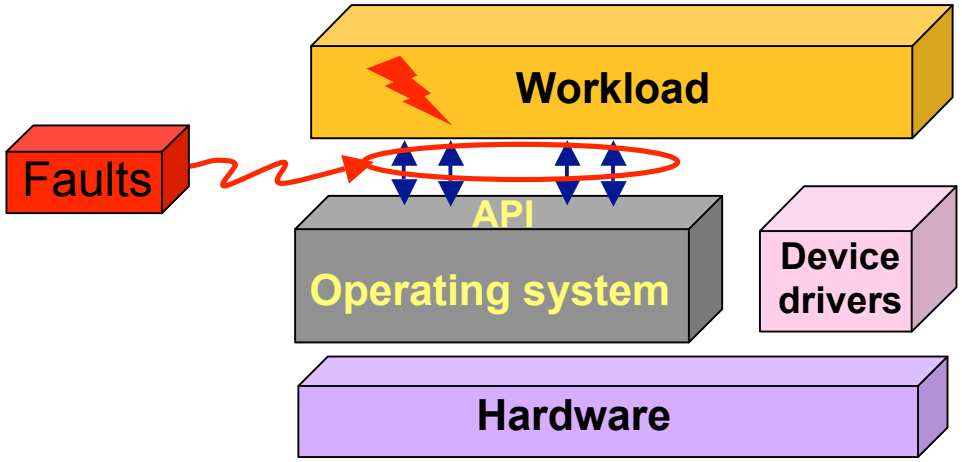
Context: User point of view



- Limited knowledge: functional description
- Limited accessibility and observability
- Limited intrusiveness and interference

⇒ **Black-box approach ⇒ robustness benchmark**

Operating System Benchmarking and Associated Measures



OS Outcomes

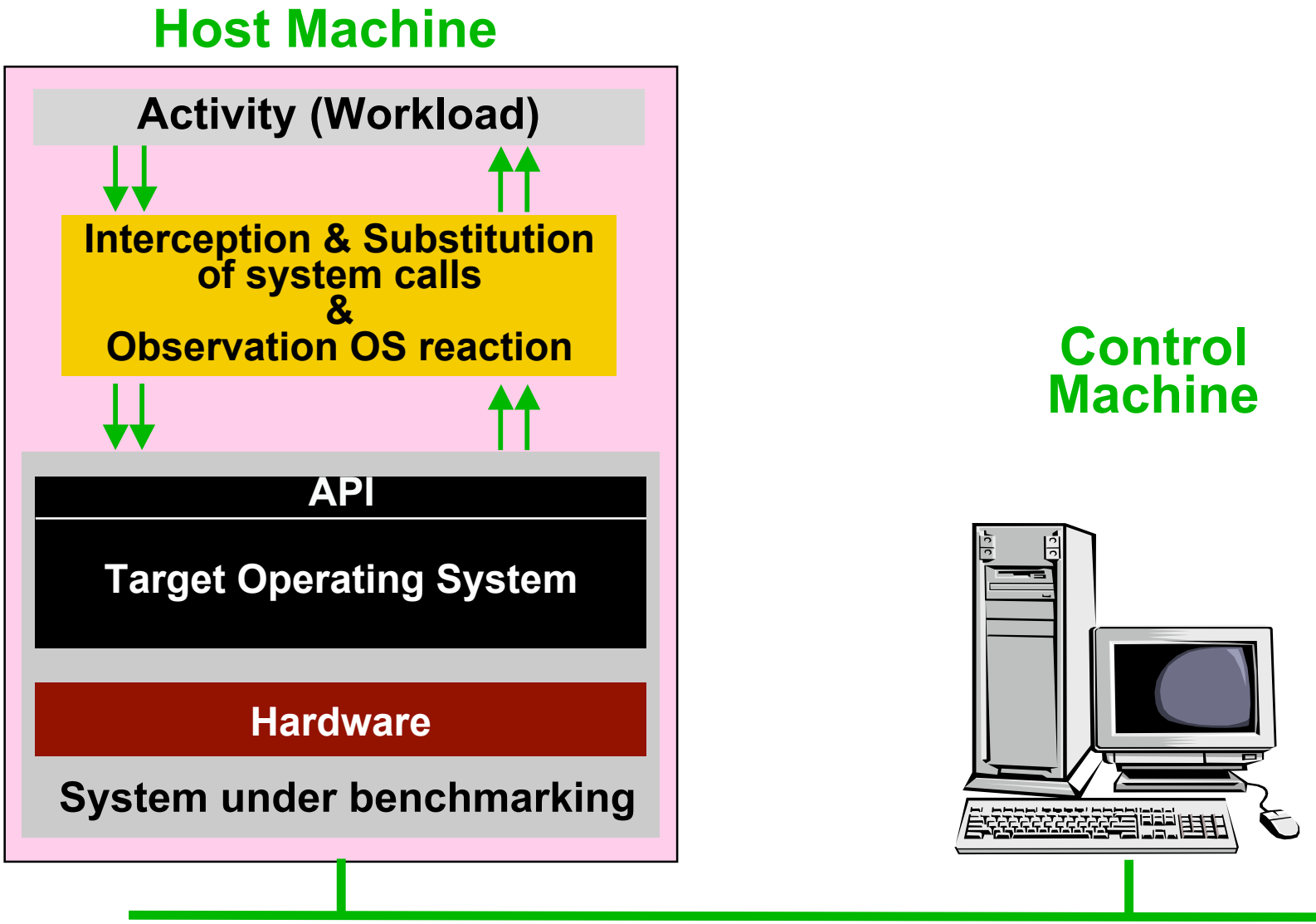
SEr	Error code
SXp	Exception
SPc	Panic
SHg	Hang
SNS	No signaling

Faults = corrupted parameters of system calls

Measures

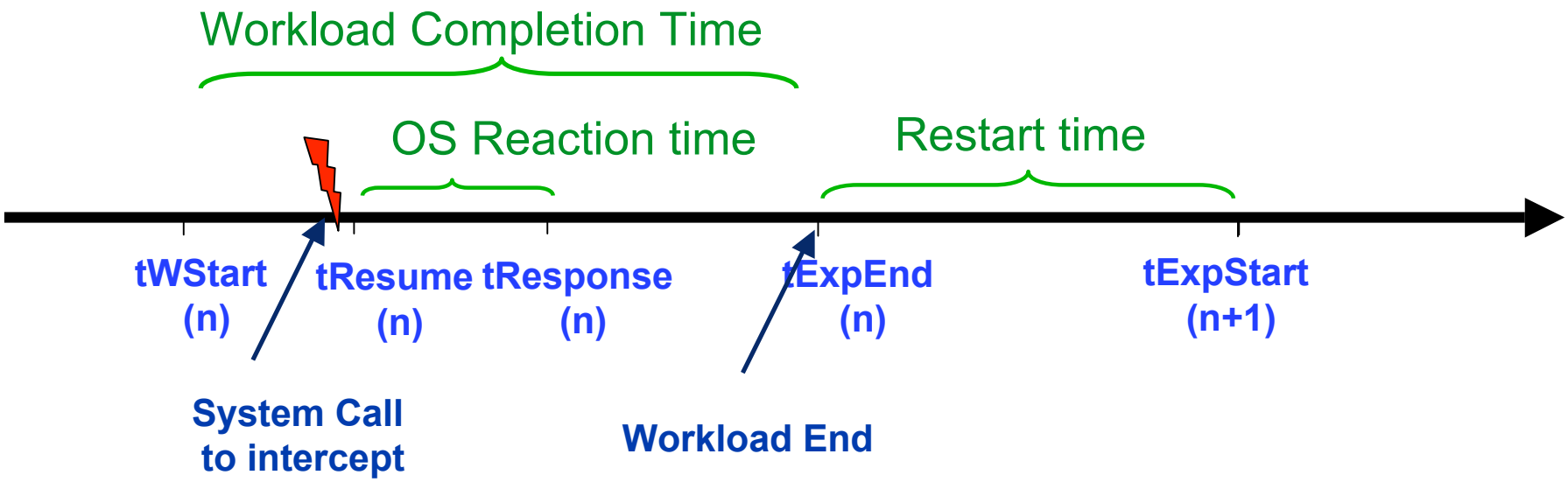
- POS: OS Robustness [%SEr %SXP %SPc %SHg %SNS])
- Texec: OS reaction time in the presence of faults
- Tres: OS Restart time after fault insertion

Experimental setup



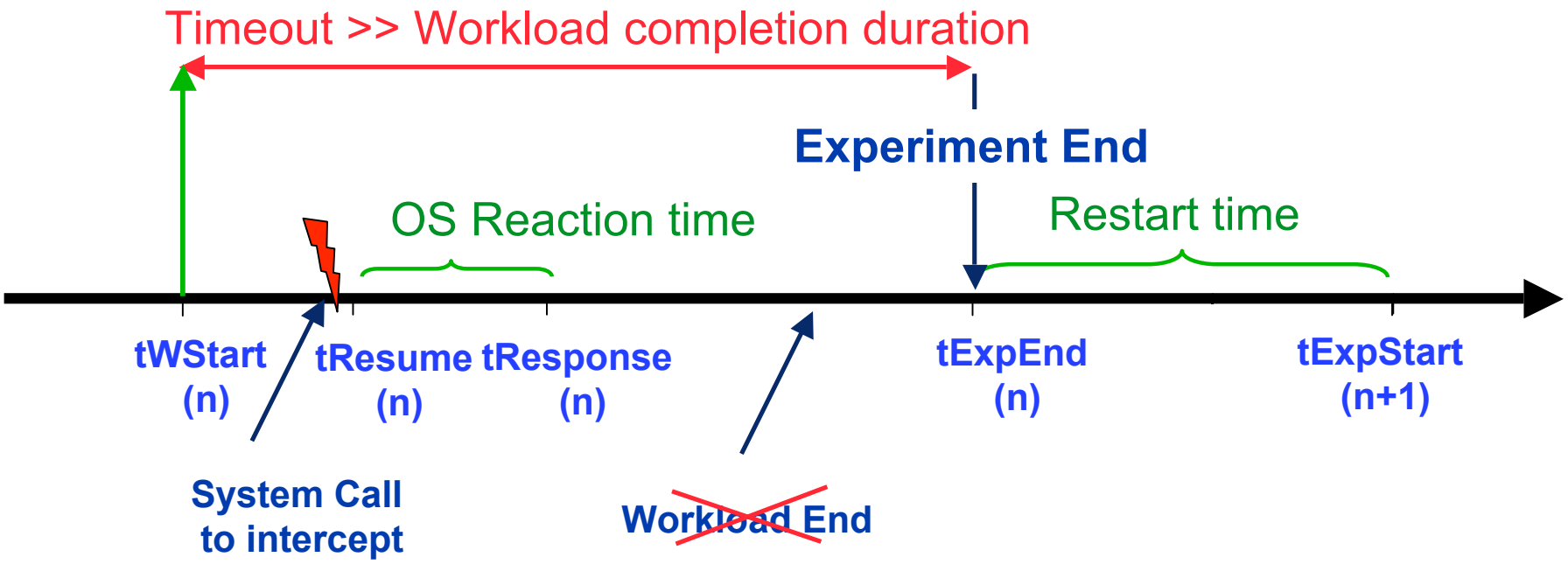
Measurements

Experiments with Workload completion

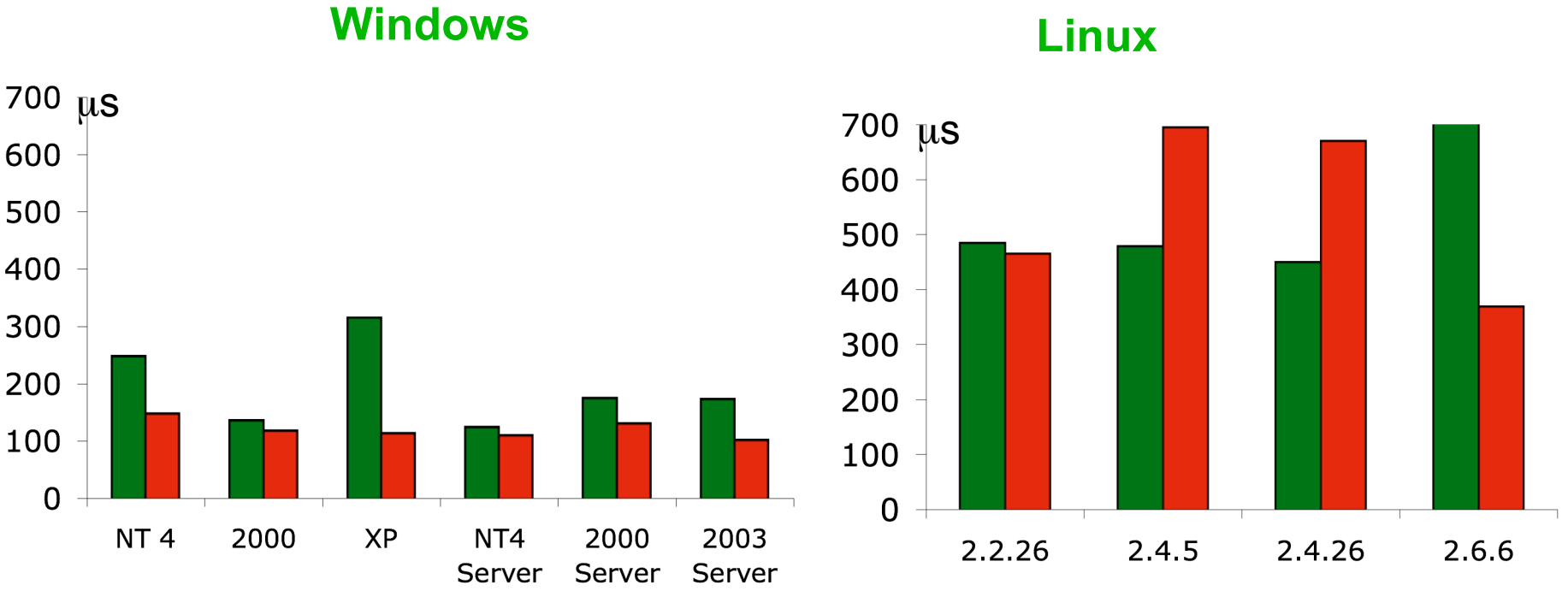


Measurements

Experiments without Workload completion



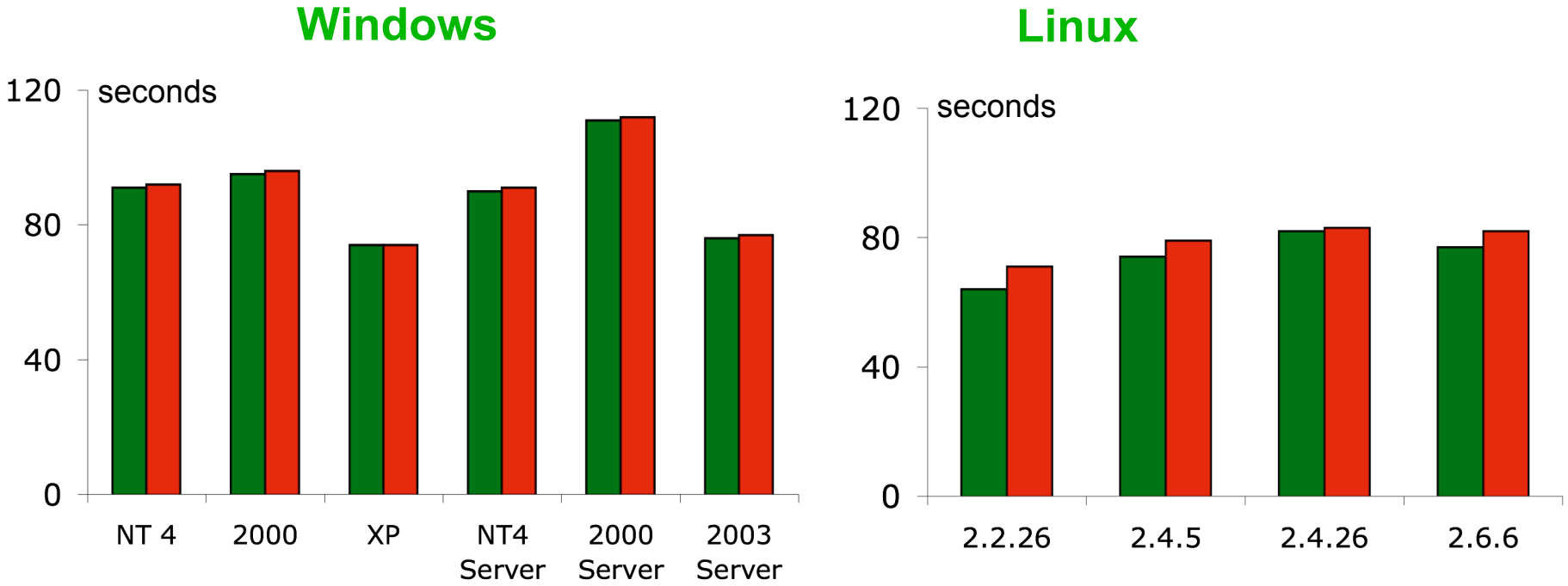
OS reaction time (Workload = PostMark)



In the presence of faults

Without parameter corruption

OS Restart time

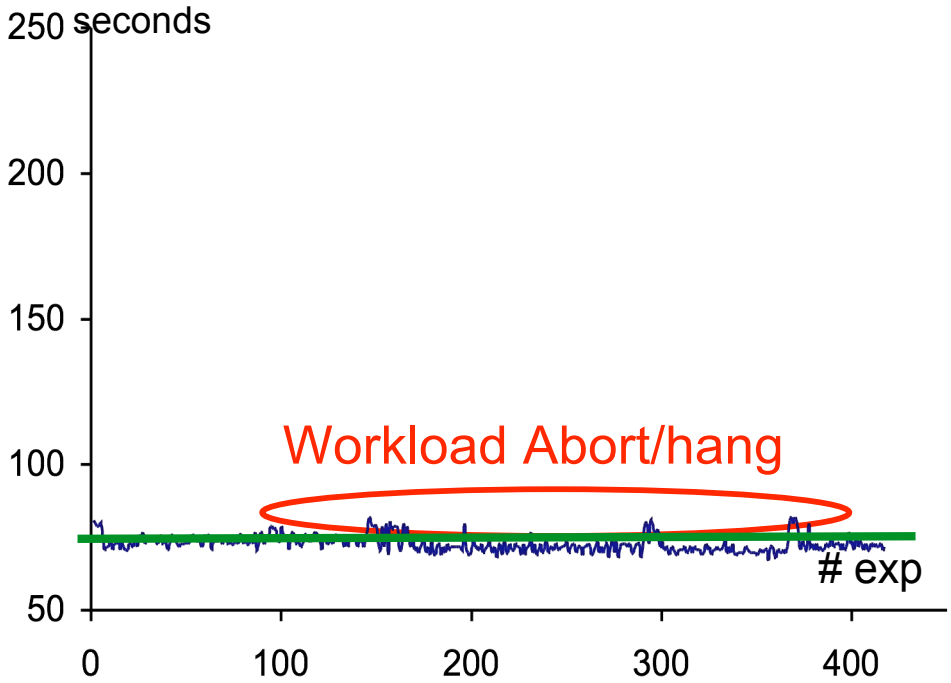


In the presence of faults

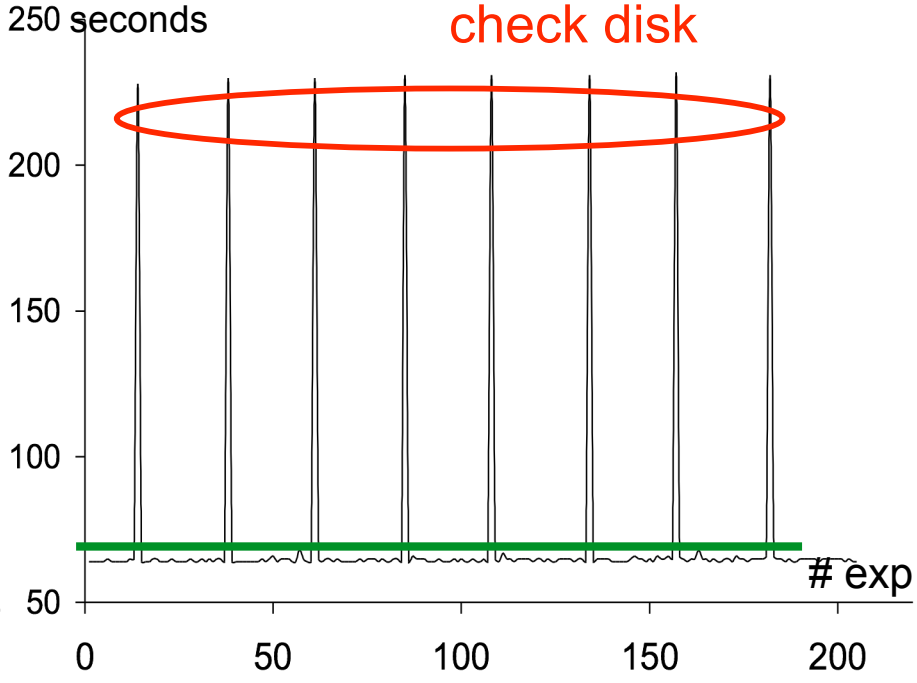
Without parameter corruption

Detailed OS Restart times

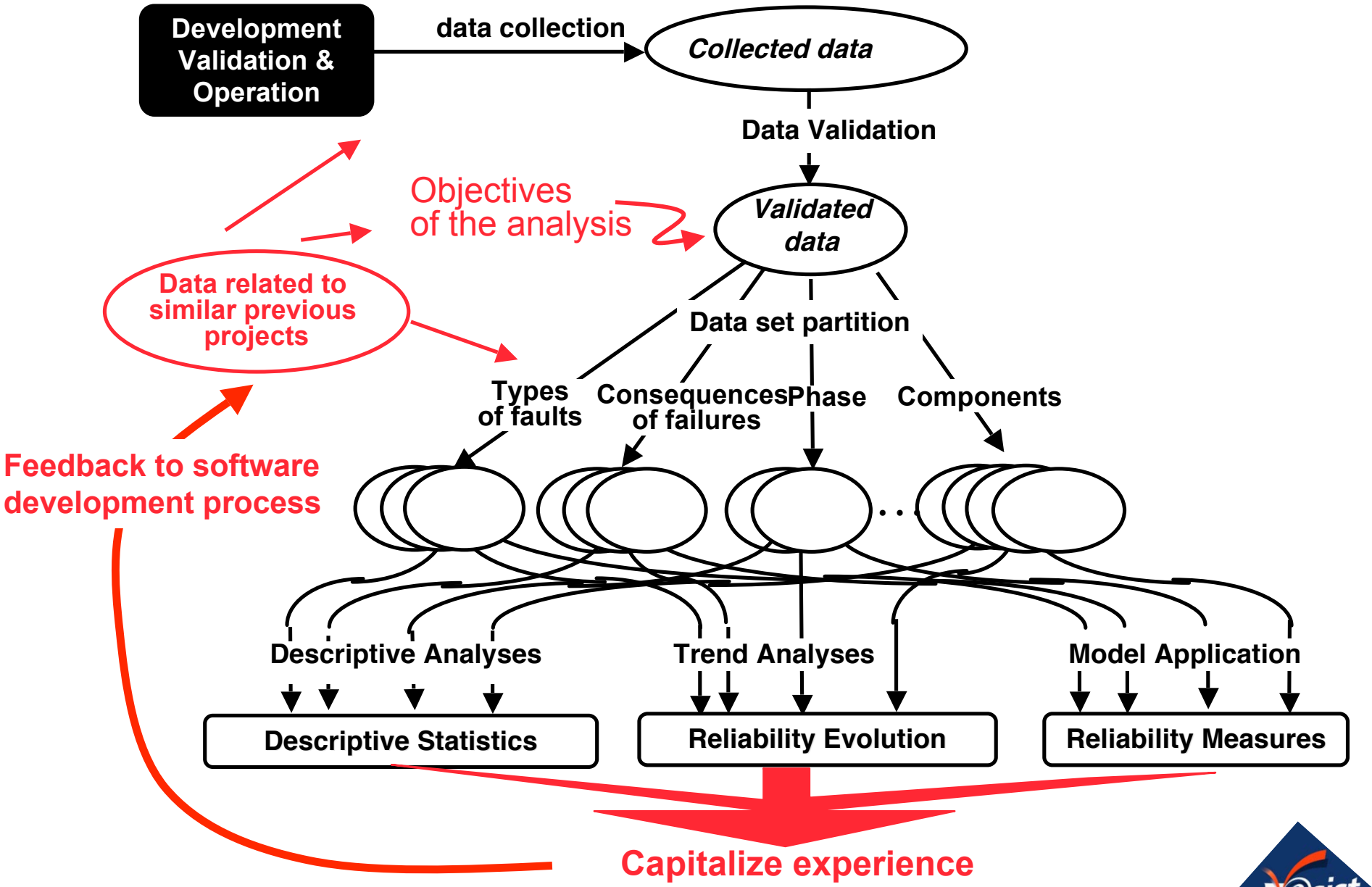
Windows XP



Linux 2.2.26



Summary


Development Validation & Operation
data collection
Collected data
Data Validation
Validated data
Objectives of the analysis
Data related to similar previous projects
Data set partition
Types of faults
Consequences of failures
Phase
Components
Feedback to software development process
Descriptive Analyses
Trend Analyses
Model Application
Descriptive Statistics
Reliability Evolution
Reliability Measures
Capitalize experience

SOFTWARE PROCESS IMPROVEMENT (SPI) (The maturity process)

- To obtain consistent quality of the software
  ⇒ control the production process ⇒ improve the software process
- The engineering method:
  - Observe existing solutions
  - Propose better solutions
  - Build / develop
  - Measure and analyze
  - Repeat the process until no more improvements possible

  ⇒ evolutionary / continuing improvement oriented approach

  Models for **process maturity** or **organization maturity**

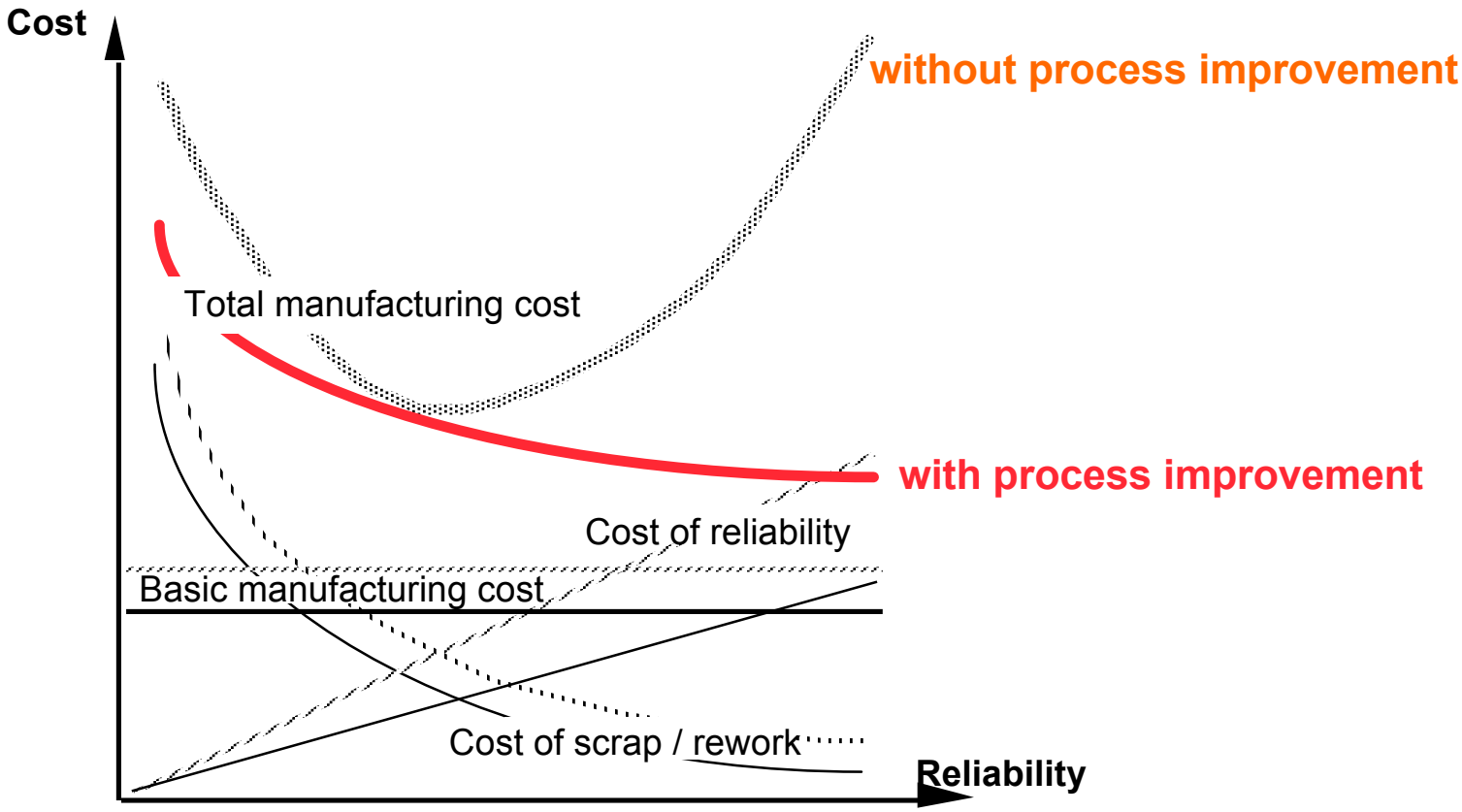
  Aim: assess the organization maturity level

[See reference 11]

Some existing methods / models

- Crosby: **Satisfaction by Quality Scheme to software development**
- Weinberg: The **Software Engineering Culture Patterns**
- Humphrey: **A Maturity Framework** ⇒ The Capability Maturity Model
- Other approaches:
  - AT&T: **Quality Program**
  - Fujitsu: **Concurrent-Development Process Model**
  - IBM: **The Cleanroom Software Development Process**
  - IBM Communication Systems: **The Defect Prevention Process**
  - ODC (Orthogonal Defect Classification)
  - etc.

Cost and reliability evolution, taking into account process improvement



PROCESS IMPROVEMENT → **RELIABILITY IMPROVEMENT**

PROCESS IMPROVEMENT → **COST REDUCTION**

Example of benefits from SPI introduction

- **IBM** (cleanroom approach):
  Productivity increase = 70% for development and 100% for testing

- **IBM** (defect prevention approach):
  Fault density divided by 2 with an increase of 0.5 % of the product resources

- **Fujitsu** (concurrent development process):
  Release cycle reduction = 75 %

- **AT&T**(quality program):
  Customer reported problems divided by 10
  Maintenance program divided by 10
  System test interval divided by 2
  New product introduction interval divided by 3
  Importance of operational profile (principal cost in SRE): ≠ test efficiency

Example of benefits from SPI introduction (Cont'd)

- Raytheon (Electronic Systems), CMM:
  - Rework cost divided by 2 after two years of experience
  - Productivity increase = 190%
  - Product quality: multiplied by 4

- Raytheon (Equipment Division), CMM:
  - Rework cost divided by 4 (elimination of $15.8 million in rework cost)
  - Productivity multiplied by 2
  - Return on investment 7.7-to-1

- Hughes Aircraft (Software Engineering Division, Fullerton CA) :
  - 1987: level 2 ⇒ recommendations & actions ⇒ level 3 in 1990
  - Return on investment of process improvement initiative: 5-to-1

- Motorola (Arlington Heights), mix of methods:
  - Fault density reduction = 50 within 3.5 years

CASE STUDIES

- TROPICO-R 1500 [See reference 3]

  Reliability analysis and evaluation

- TROPICO-R 4096 [See reference 7]

  Software decomposition

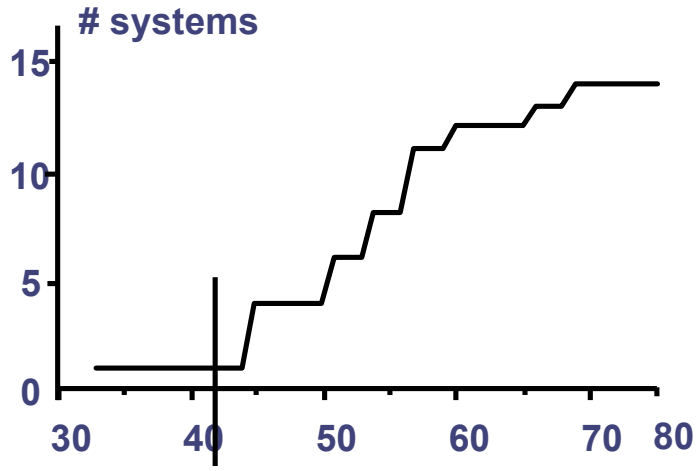
  Reliability analysis and evaluation

- Three generations of TROPICO-R [See reference 8]

  Comparative evolution: fault density and reliability

TROPICO-R 1500

- Characteristics
  - Language: Assembly
  - Size: 300 k-bytes
  - Validation: 10 months, 297 failures / corrections
  - Field trial: 4 months, 55 failures/corrections
  - Operation: 13 months, 109 failures/corrections
  - Total : 461
- Data
  - Number of failures / unit of time
    - unit of time: 10 days
    - observation duration: 81 units of times
  - Times to failures
    - operational life only



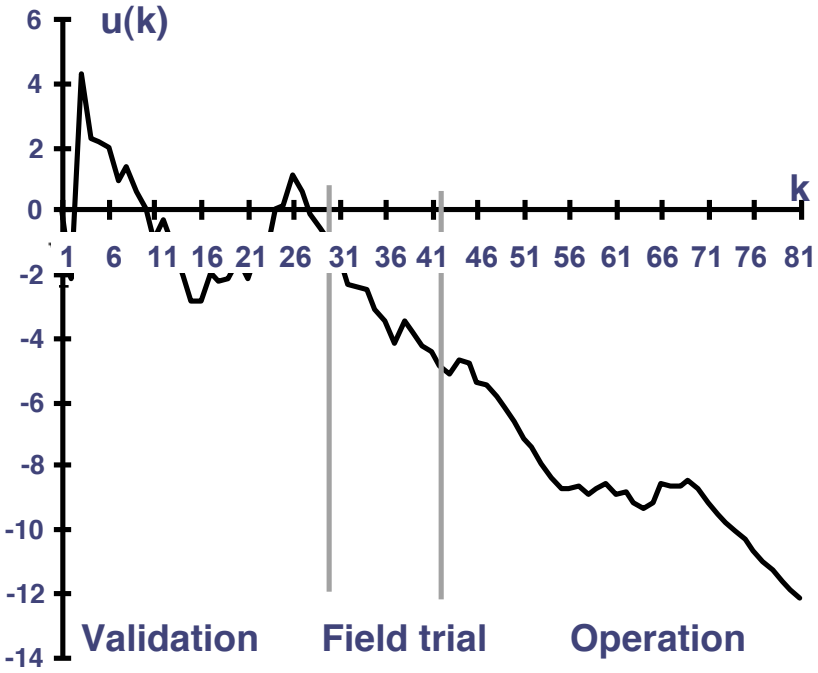
Data set

CNF: cumulative # of failures

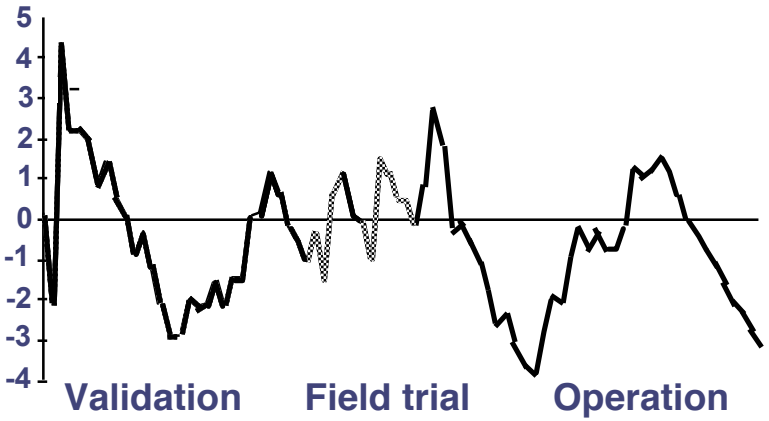
Validation		Field test		Operation	
u. time	CNF	u. time	CNF	u. time	CNF
1	7	31	301	43	356
2	8	32	302	44	367
3	36	33	310	45	373
4	45	34	317	46	373
5	60	35	319	47	378
6	74	36	323	48	381
7	82	37	324	49	383
8	98	38	338	50	384
9	106	39	342	51	384
10	115	40	345	52	387
11	120	41	350	53	387
12	134	42	352	54	387
13	139			55	388
14	142			56	393
15	145			57	398
16	153			58	400
17	157			59	407
18	174			60	413
19	183			61	414
20	196			62	417
21	200			63	419
22	214			64	420
23	223			65	429
24	246			66	440
25	257			67	443
26	277			68	448
27	283			69	454
28	286			70	456
29	292			71	456
30	297			72	457
				73	458
				74	459
				75	459
				76	459
				77	459
				78	460
				79	460
				80	460
				81	461

Test de Laplace & data partitioning

whole phases



Per phase

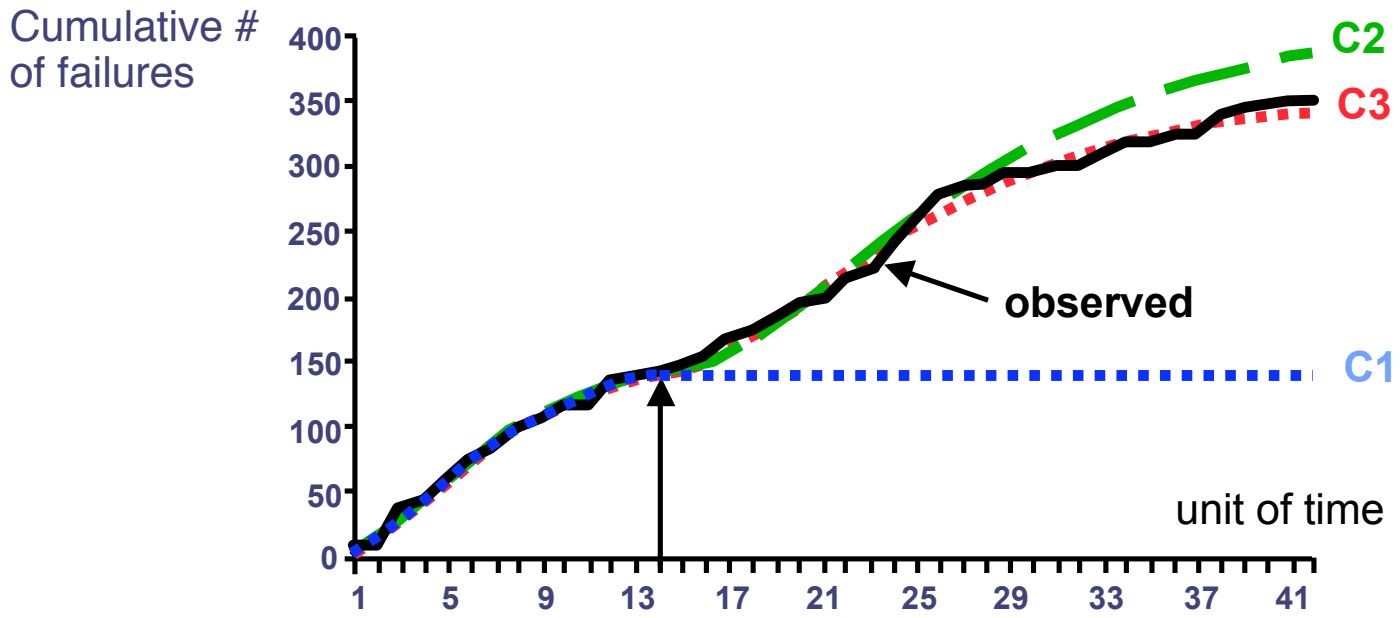


Data Partitioning

- **Validation &field trial**
  - P1 : {1 , 14}
  - P2 : {15 , 42}
- **Operation**
  - P3 : {43 , 54}
  - P4 : {55 , 81}

Model Application
(Number of failures)

- Validation & field trial, application of the S-Shaped model

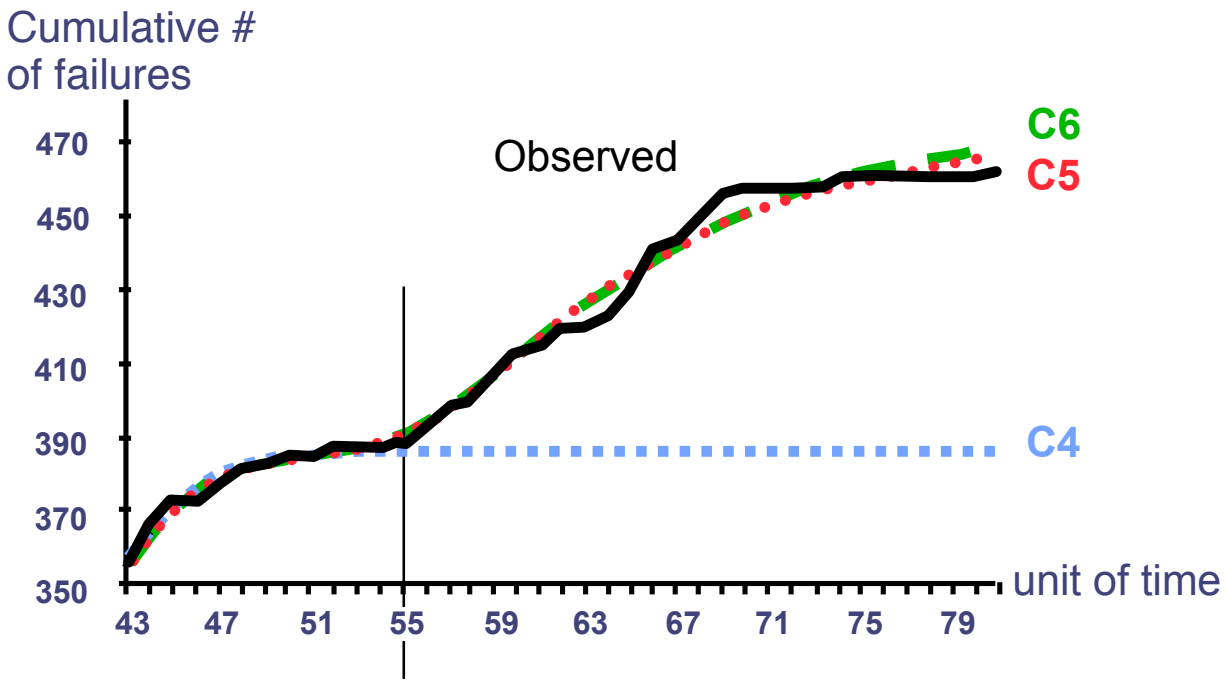


	$R_{9,14}$	$R_{28,42}$	$R_{30,42}$
C1	2,6		
C2		28,4	31,2
C3			5,8

C1 : calibrated from {1,8}
C2 : calibrated from {15,27}
C3 : calibrated from {15,29}

Model application
(Number of failures)

- Operational life, application of the S-Shaped model (SS)



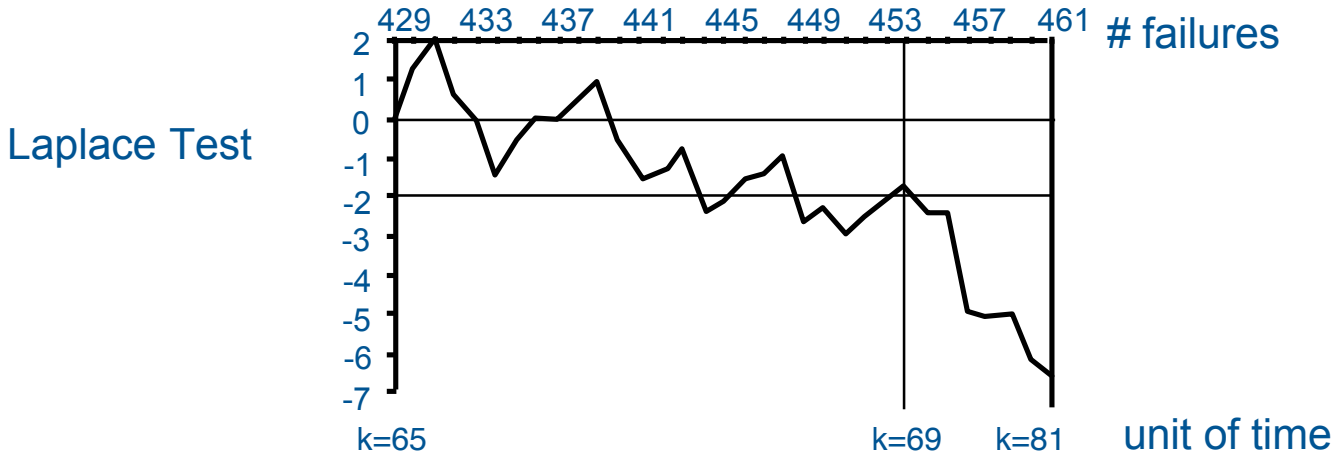
Residue	$R_{51,55}$	$R_{74,81}$	$R_{76,81}$
C4	1,8		
C5		4,3	5,3
C6			3,5

Prediction for next quarter (all systems)

2 failures the next month
& 1 failure / month the next two months

Model application
(times to failures)

☞ Operational life, average system, application of the Hyperexponential model



Software residual failure rate for an average system

$\lambda_{sof} = 1{,}3\ 10^{-4}$ /h (all consequences)

Hardware failure rate (known from a different study)

$\lambda_{har} = 4\ 10^{-6}$ /h (leading to system unavailability)

⇨ $\lambda_{har} \ll \lambda_{sof}$

⇨ **apply reliability growth models to failures leading to total unavailability**

Model application according to software components & to failure consequences

- Other switching system E-10-B
  - Hyperexponential model

Component	$\lambda_r(10^{-7}$ / h)
Telephony	7,5
Defense	27,4
Exploitation	7,3
Executive	8,3
All corrections	47,5

Consequence	$\lambda_r(10^{-7}$ / h)
General unavailability	1,2
Partial unavailability	7,9
Exploitation treat. delay	3,7
Loss of a hardware unit	3,1
All failures	38,2

TROPICO-R 4096

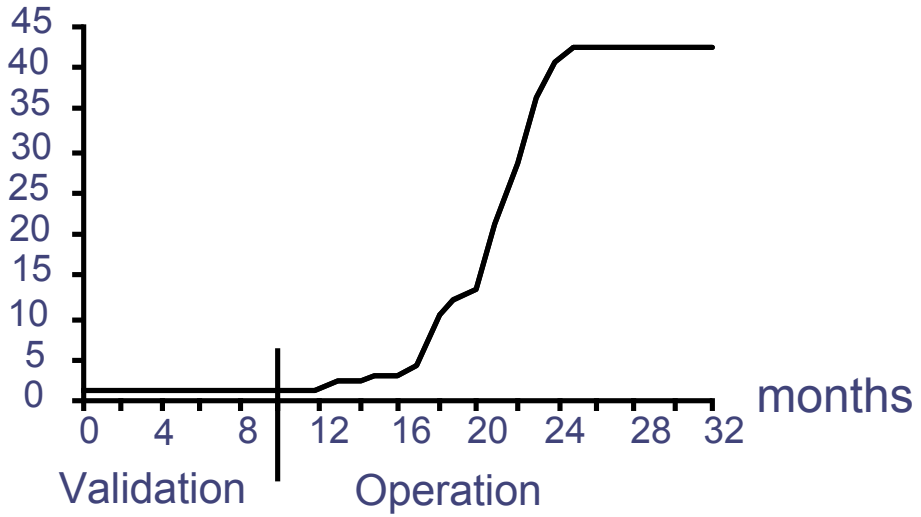
- Characteristics
  - Language: Assembly
  - Size: 335 k-bytes
  - Validation : 8 months, 76 failures / corrections
  - Operation: 24 months, 134 failures/corrections
  - Total: 210
- Data
  - Number of failures / unit of time
    - observation period: 32 months
  - Times to failures
    - for operational life

Software decomposition and # of systems

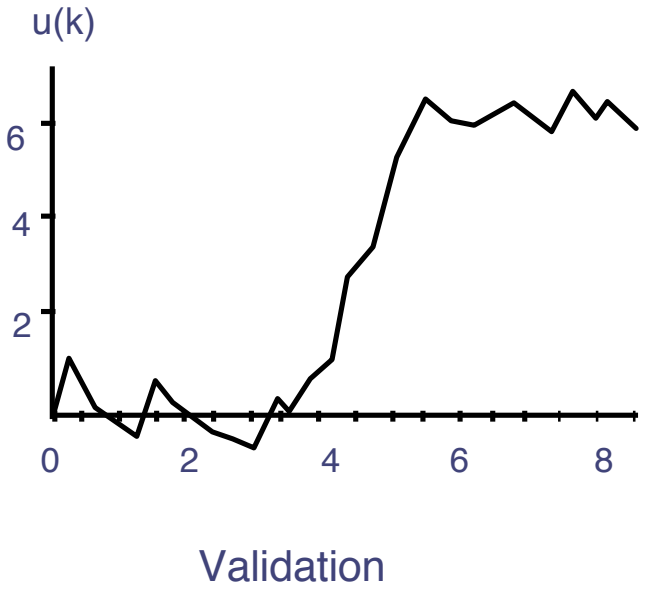
- Decomposition

	Volume	# failures
Telephony	75 k-bytes	74 (34 - 40)
Defense	117 k-bytes	67 (20 - 47)
Interface	115 k-bytes	61 (20 - 41)
Management	44 k-bytes	31 (13 - 18)

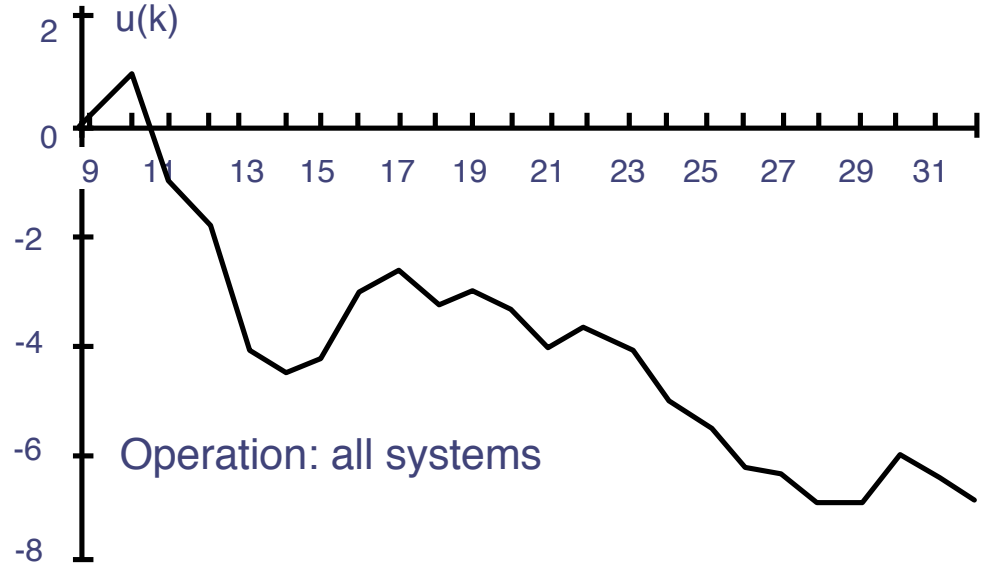
- Number of systems

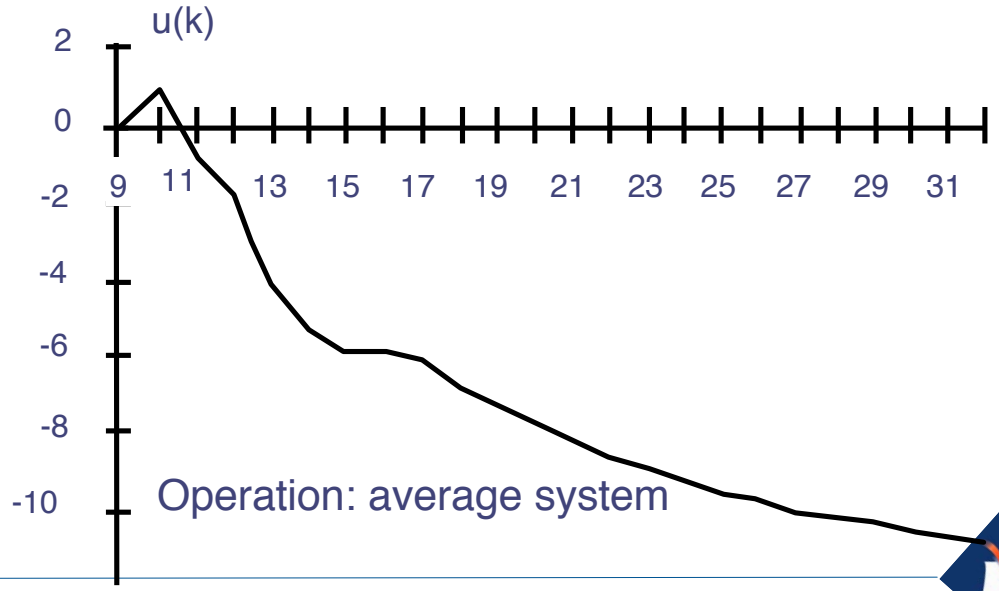


Laplace Test

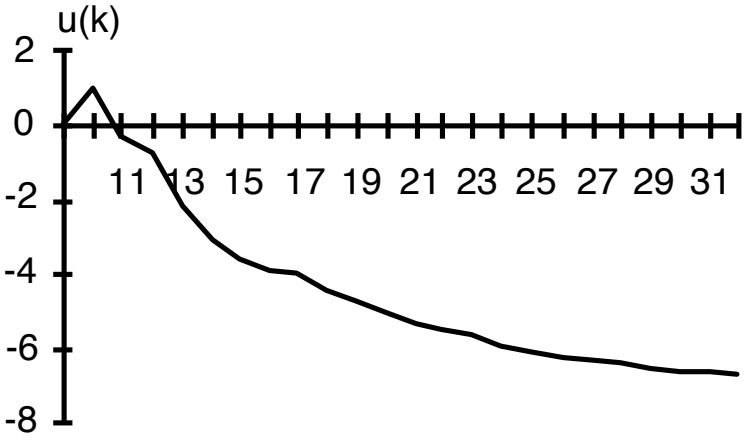


Validation

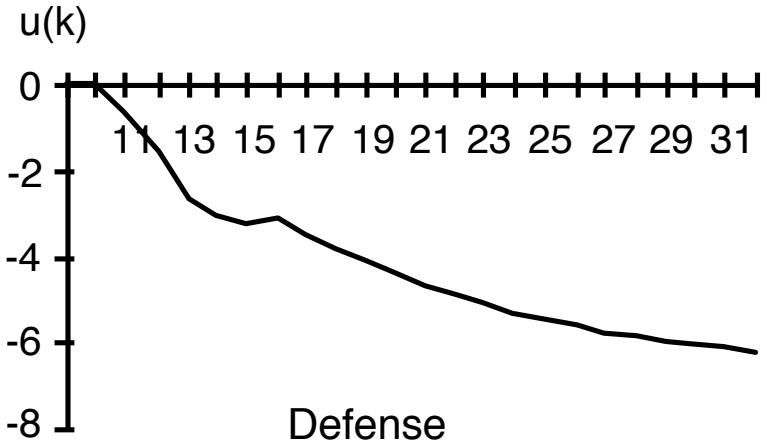




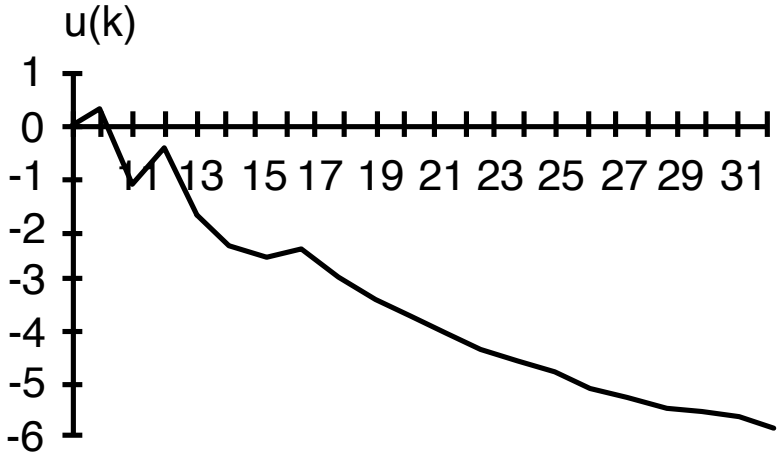
Laplace Test for the software components



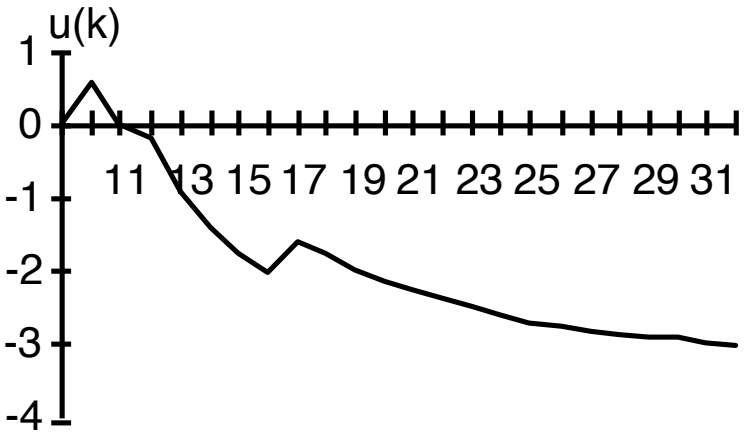
Telephony



Defense

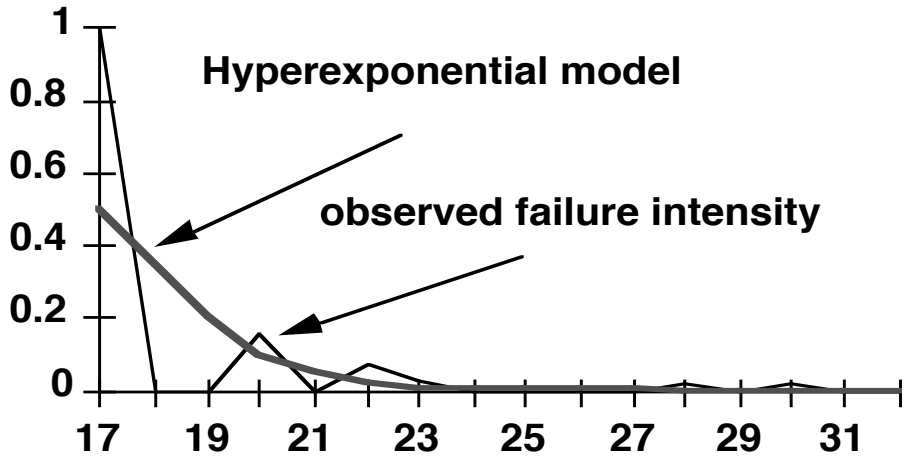


Interface

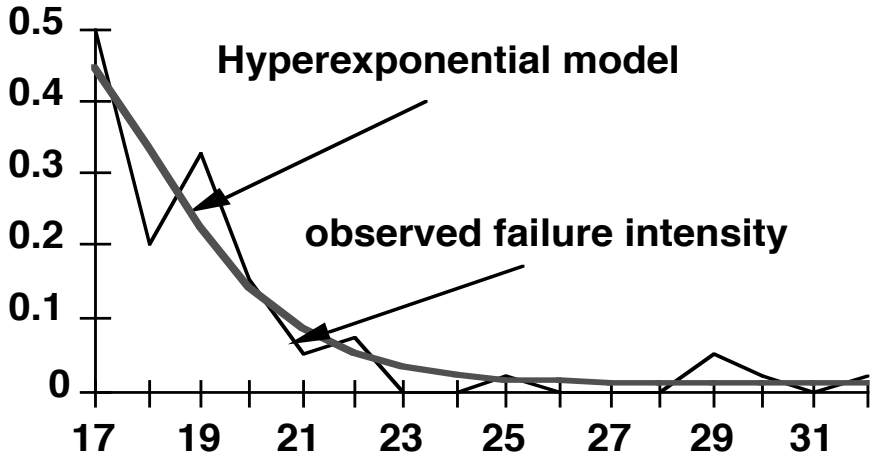


Management

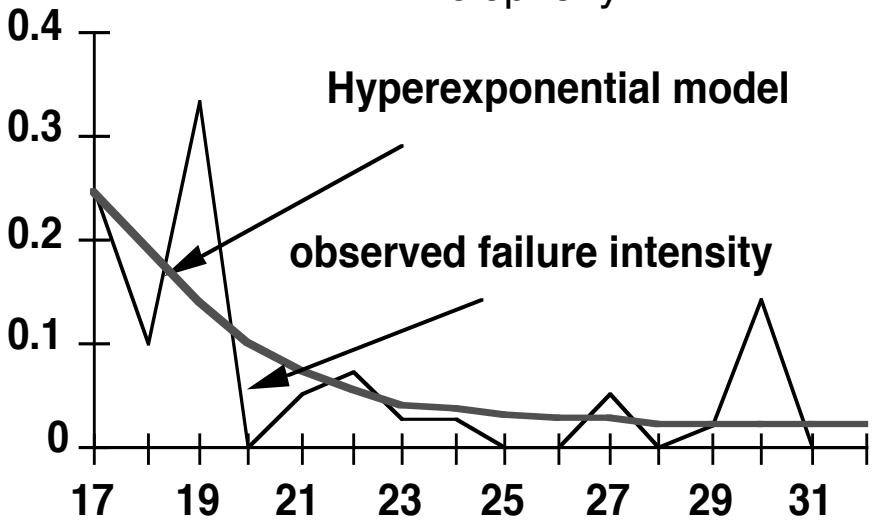
Failure intensity: Hyperexponential model application



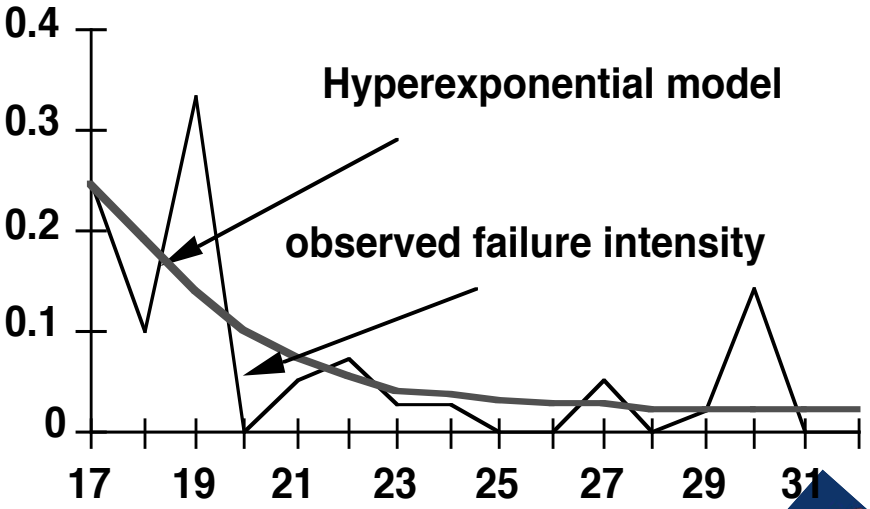
Telephony



Defense



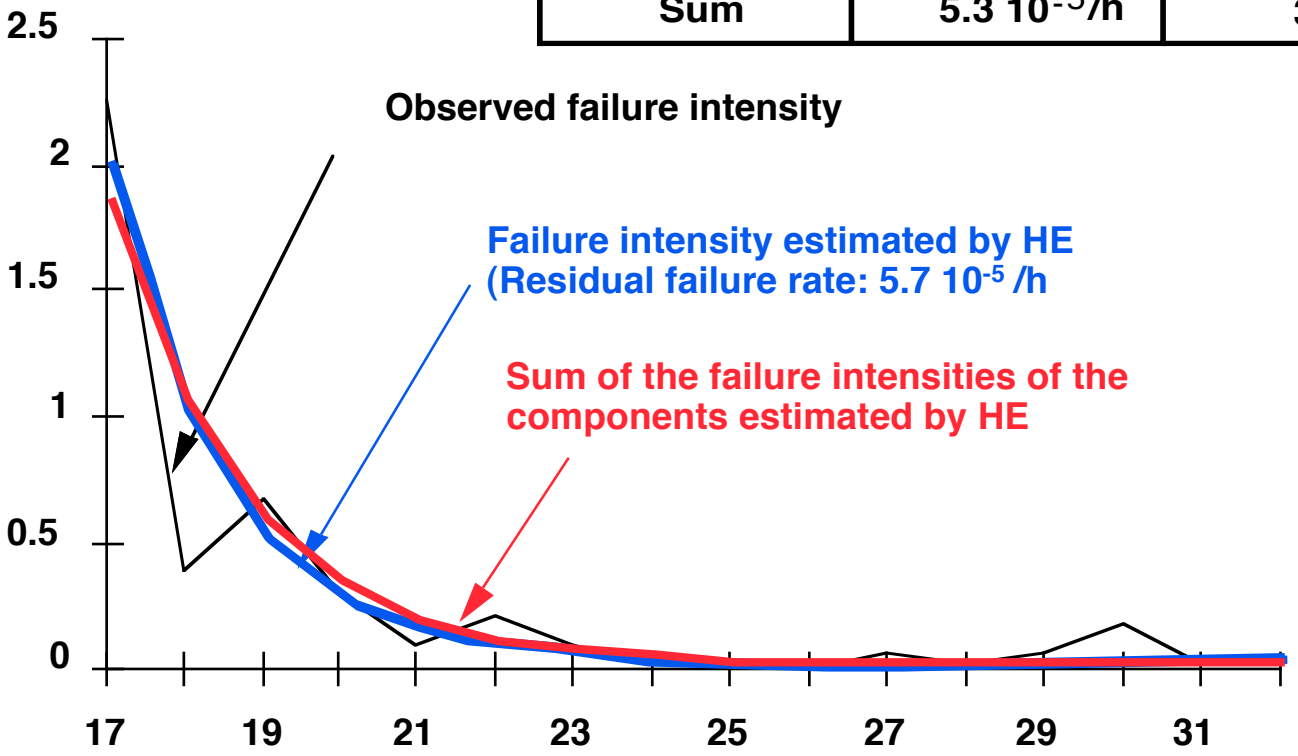
Interface



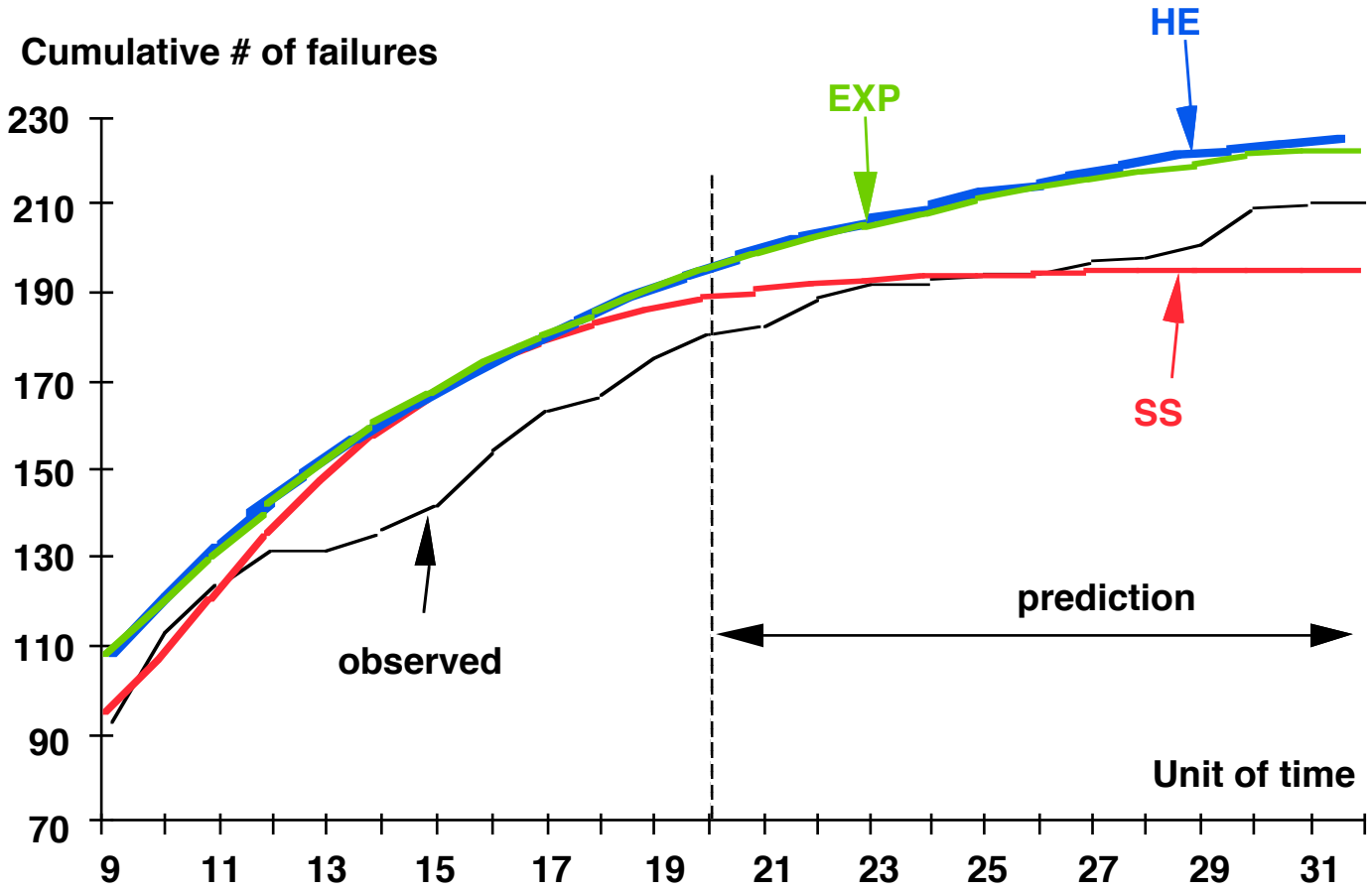
Management

Residual failure rates (Hyperexponential model)

	Residual failure rate	Size (Kb)
Telephony	$1.2\ 10^{-6}$ /h	75
Defense	$1.4\ 10^{-5}$ /h	103
Interface	$2.9\ 10^{-5}$ /h	115
Management	$8.5\ 10^{-6}$ /h	42
Sum	$5.3\ 10^{-5}$/h	335



Maintenance planning



Estimated # failures from 20 to 32:

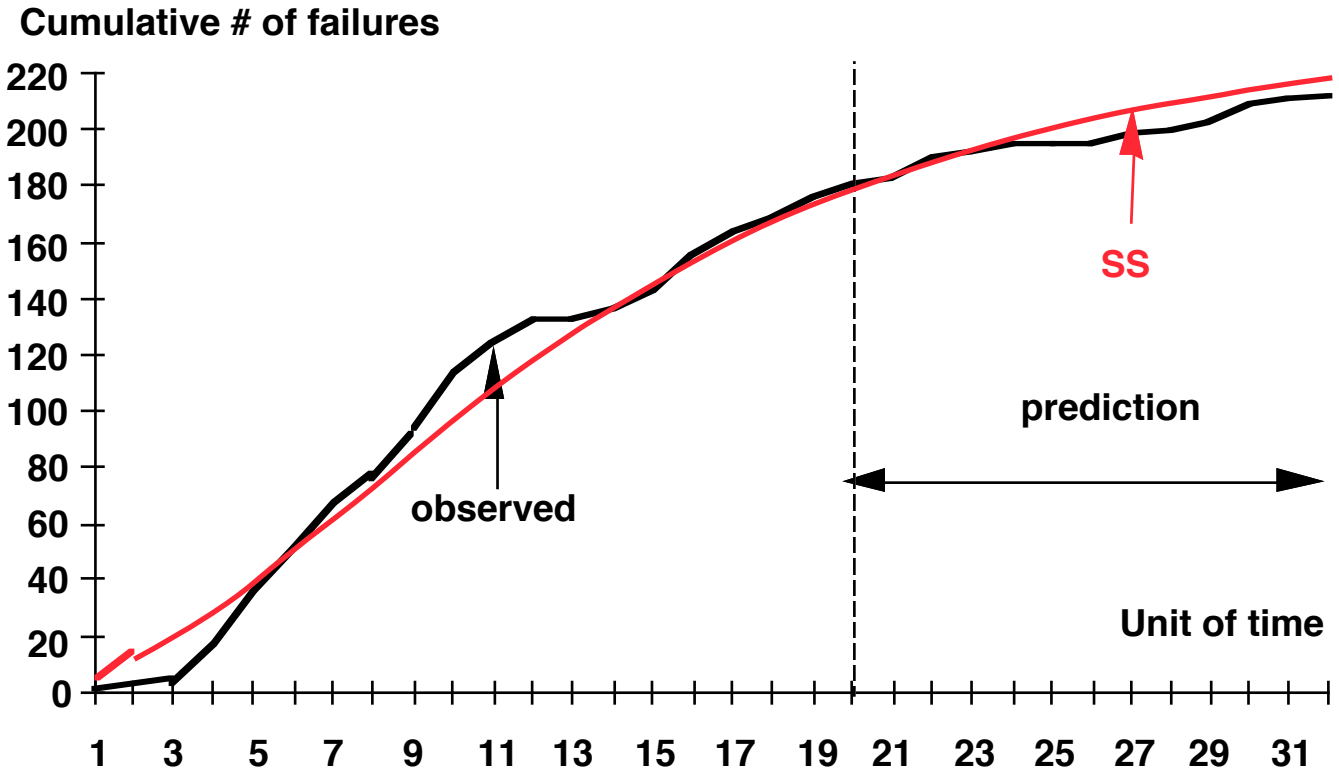
Exponential: 33

Hyperexponential: 37

S-Shaped: 9

Observed: 34

Maintenance planning



Estimated # failures from 20 to 32: 40

Observed: 34

Software Reliability Analysis of Three Successive Generations of a Switching System

Outline

- The products investigated
- Data collected
- Statistics on failures and faults
- Residual failure rates
- Conclusion

Products & Software

- Three products
  - TROPICO-R 1500 (PRA)
  - TROPICO-R 4096 (PRB)
  - TROPICO-RS (PRC)

- Software components (Applicative & Executive software)
  - Elementary Implementation Blocks (EIB)
  - Functions
    - Telephony (TEL)
    - Defense (DEF)
    - Interface (INT)
    - Management (MAN)

Software decomposition and size

PRA

	# EIB	size (Kbytes)
TEL	6	72
DEF	9	93
INT	10	113
MAN	4	42
Sum	29	320

PRB

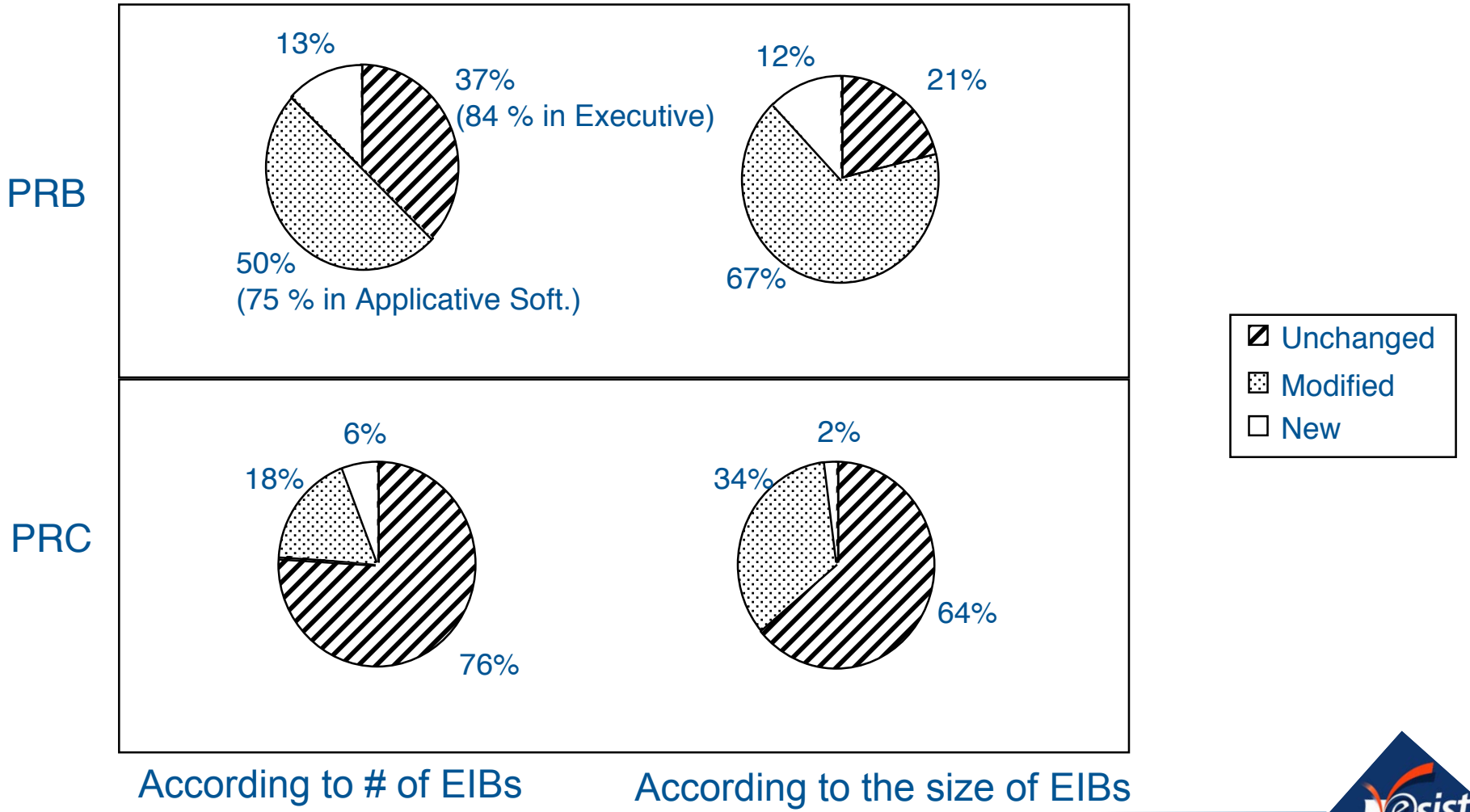
	# EIB	size (Kbytes)
TEL	6	75
DEF	12	117
INT	10	115
MAN	4	44
Sum	32	351

PRC

	# EIB	size (Kbytes)
TEL	8	111
DEF	12	130
INT	10	129
MAN	4	51
Sum	34	421

EIB Distribution

- Two types of EIBs: new — reused (modified / unchanged)



Test environment and failure data

- Software test program

  Steps: unit tests, integration tests, validation tests, field tests

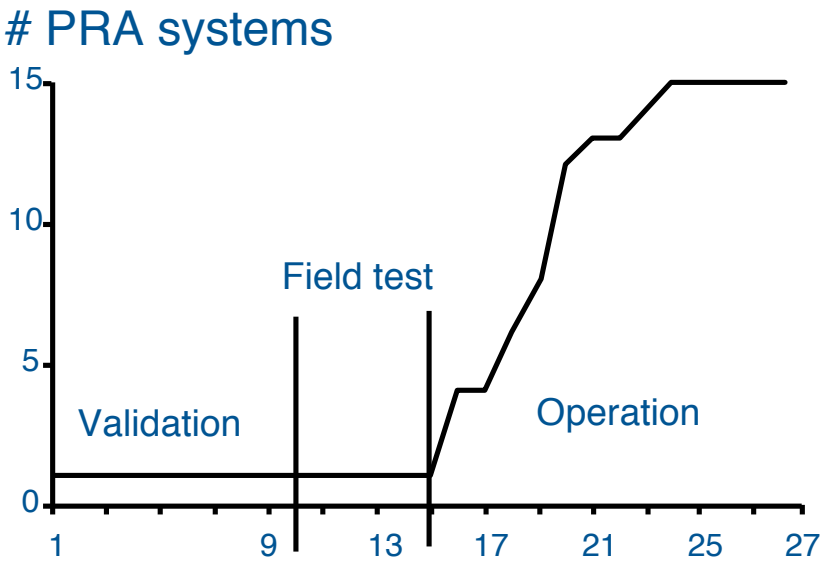
  Validation tests: functional, quality, performance, overload tests

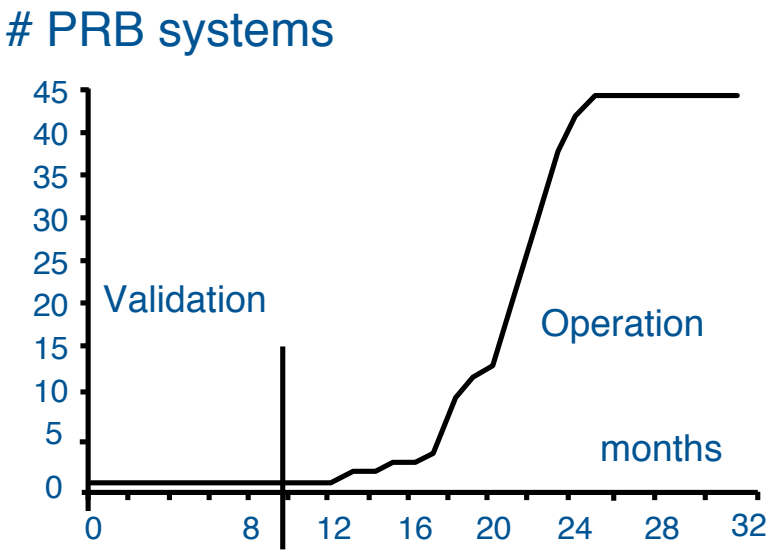
- Failure reports & Trouble reports (FRs & TRs)
  - Date of failure occurrence (static analysis Þ date of detection)
  - Description of system configuration in which the failure was observed
  - Type: hardware, software, documentation, affected EIBs
  - Analysis: identification— classification of faults (coding, specification, etc.)
  - Solutions
  - Regression testing

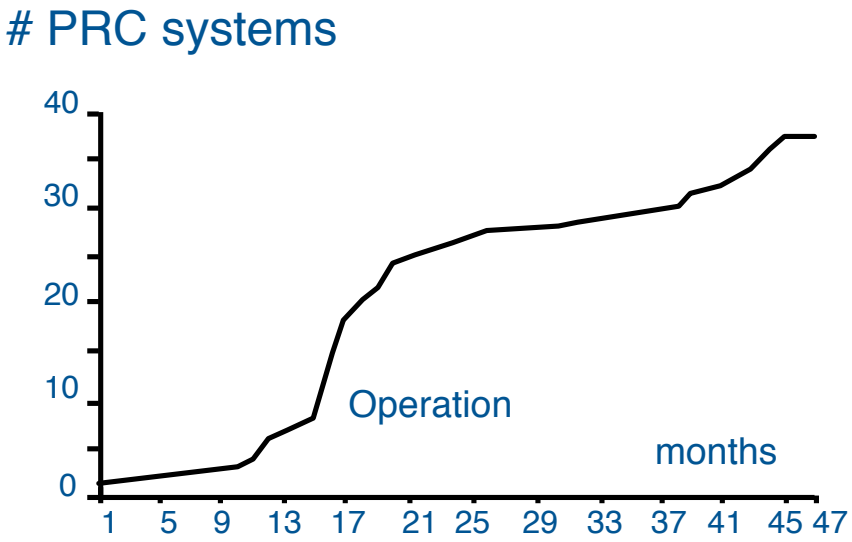
- An FR is a failure report and also a correction report

- Rediscoveries are not recorded

Data Collection







Statistics on Failures and Faults

	# FR (# TR)	# CF
PRA	465	637
PRB	210	282
PRC	212 (105)	394

☞ >70 % of failures led to modification of one EIB

# corrected EIBs	# FR in PRA	# FR in PRB	#FR+TR in PRC
1	362 (77.8%)	165 (78.6%)	228 (71.9%)
2	72 (15.5%)	33 (15.7%)	69 (21.8%)
≥ 3	31 (6.7%)	12 (5.7%)	20 (6.3%)

☞ identify EIBs which are dependent w.r.t failure occurrence

(2 pairs of strongly dependent EIBs)

Statistics on Failures and Faults (cont'd)

PRA

	# FR	# CF	Size
TEL	146	190	72
DEF	138	164	93
INT	170	191	113
MAN	78	92	42
Sum	532	637	320

PRB

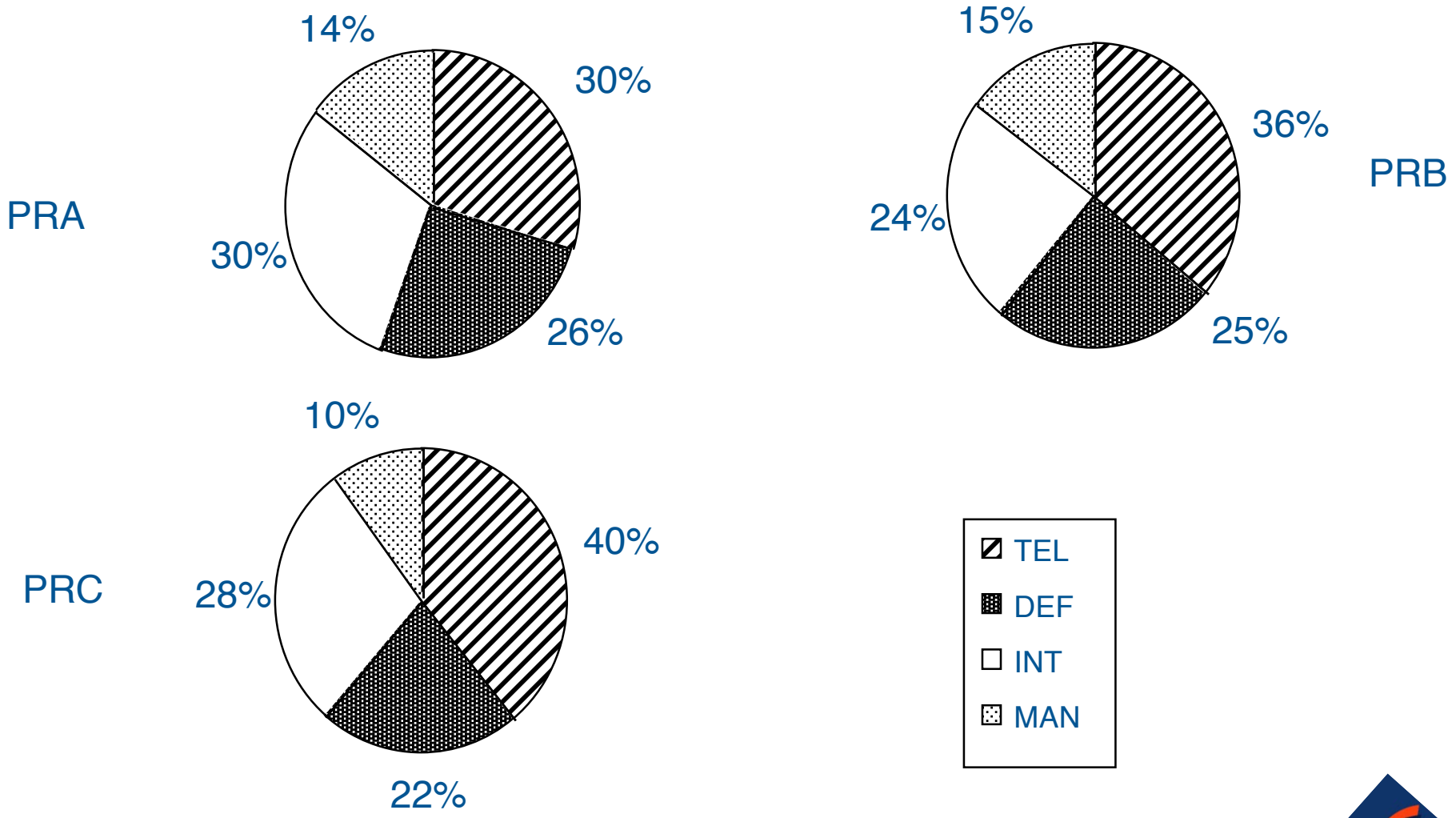
	# FR	# CF	Size
TEL	74	102	75
DEF	67	71	117
INT	61	68	115
MAN	31	41	44
Sum	233	282	351

PRC

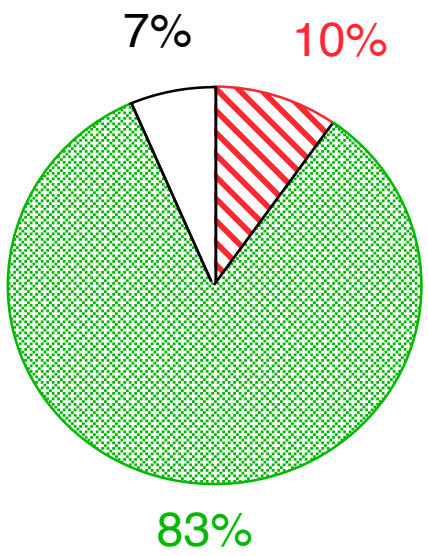
	# FR (# TR)	# CF	Size
TEL	65 (52)	155	111
DEF	63 (21)	88	130
INT	72 (27)	112	129
MAN	25 (10)	40	51
Sum	225 (110)	395	421

☞ 90% of FRs led to modification of only one Function

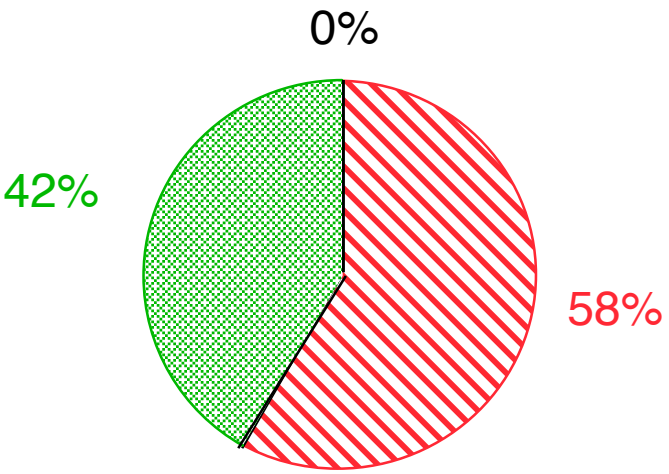
Distribution of faults among functions



Distribution of faults per EIB type



PRB



PRC

Unchanged
Modified
New

Average fault density

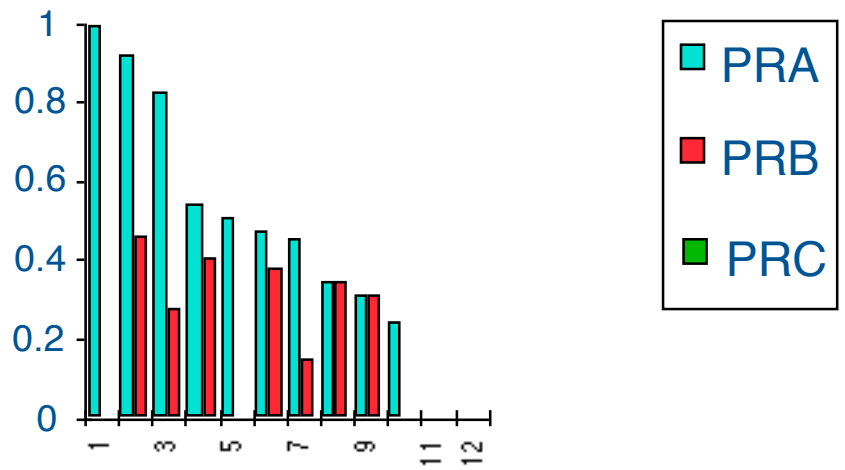
- versus EIB size

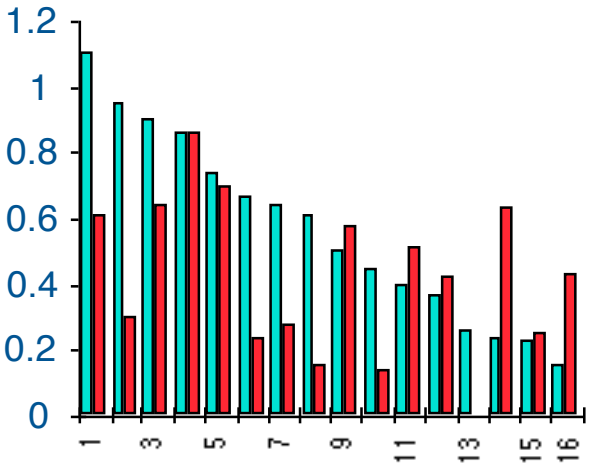
Size	PRA	PRB	PRC (all faults)	PRC (only faults relative to FRs)
EIB size > 15 Kb	1.80	1.08	0.99	0.65
10 Kb <EIB size< 15 Kb	2.02	0.68	0.58	0.47
5 Kb <EIB size< 10 Kb	2.31	0.60	0.96	0.52
EIB size< 5 Kb	2.56	0.71	0.62	0.56
Average fault density	2.1	0.76	0.79	0.55

- data collected during operation

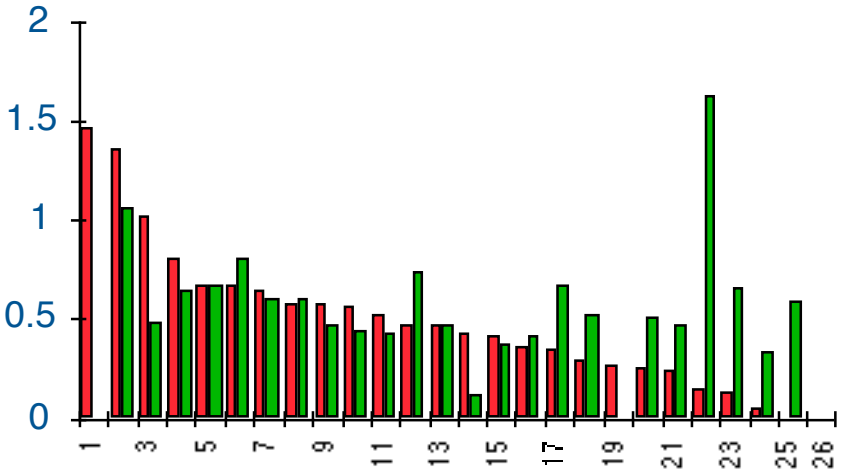
	PRA	PRB	PRC
After 13 months	0.34	0.35	0.3
After 24 months	-	0.47	0.6

Fault density evolution / EIB type (Operation)

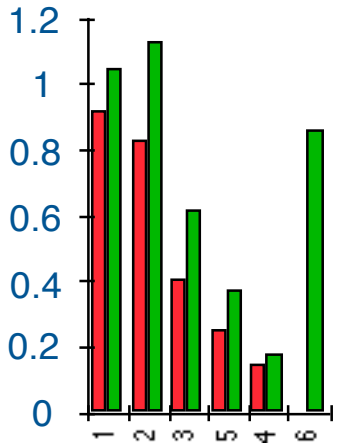




1 year



Unchanged EIBs



Modified EIBs

2 years

Residual failure rates

	PRA	PRB	PRC
TEL	$2.6\ 10^{-5}$ / h	$1.2\ 10^{-6}$ / h	$4.3\ 10^{-5}$ / h
DEF	$4.3\ 10^{-5}$ / h	$1.4\ 10^{-5}$ / h	$1.9\ 10^{-5}$ / h
INT	$4.2\ 10^{-5}$ / h	$2.9\ 10^{-5}$ / h	$3.2\ 10^{-5}$ / h
MAN	$1.4\ 10^{-6}$ / h	$8.5\ 10^{-6}$ / h	$9.9\ 10^{-6}$ / h
Sum	$1.124\ 10^{-4}$ / h	$5.27\ 10^{-5}$ / h	$1.03\ 10^{-4}$ / h

Conclusion

- PRA & PRB
  - Similar development environment ⇒ reliability improvement
- PRC
  - Learning process interrupted ⇒ reliability improvement?
- PRA, PRB & PRC
  - Residual failure rates: same order of magnitude
  - Failure rate of the software = sum of the failure rates of its components
- Additional experimental studies

⇒ factors impacting the reliability of a family of products

References

1. Handbook of Software Reliability Engineering, *Edited by* Michael R. Lyu, Published by IEEE Computer Society Press and McGraw-Hill Book Company, 1996.

2. Software Reliability Engineering: More Reliable Software Faster and Cheaper, 2nd Edition, John Musa, September 2004.

3. A method for software reliability analysis and prediction, application to the TROPICO-R switching system, K Kanoun, M. R. Bastos Martini, J. Moreira de Souza, IEEE Transactions on Software Engineering, N° 4, pp. 334-344, April 1991.

4. For a product-in-a-process approach to software reliability evaluation,J. C. Laprie, Third IEEE International Symposium on Software Reliability Engineering (ISSRE'92), Research-Triangle Park (USA), October 7-10 1992, pp.134-13

5. Operational Profiles in Software-Reliability Engineering, John D. Musa, IEEE Software 10 (2), pp. 4-32, 1993.

6. SoRel: a tool for reliability growth analysis and prediction from statistical failure data, K. Kanoun, M. Kaâniche, J. C. Laprie and S. Metge, 23rd IEEE International Symposium on Fault-Tolerant Computing (FTCS'23), Toulouse, France, June 22-24, 1993, pp.654-659.

7. Experience in software reliability: from data collection to quantitative evaluation, K. Kanoun, M. Kaâniche, and J. C. Laprie, 4th International Symposium on Software Reliability Engineering, Denver (USA), 3-6 November 1993, pp.234-245.

8. Software failure data analysis of tthree successive generations of a switching system, M. Kaâniche, K. Kanoun, M. Cukier, M. R. Martini, 1st European Dependable Computing Conference (EDCC-1), Berlin, Germany, 4-6 October 1994, pp. 473-490.

9. Software Reliability Trend Analyses: From Theoretical to Practical Considerations, K. Kanoun and J. C. Laprie, IEEE Transactions on Software Engineering, Vol.20, N°9, pp.740-747, September 1994.

10. Trend Analysis, Kanoun, K. and J.-C. Laprie, in Handbook of Software Reliability Engineering, Ed. M. Lyu, Mc Graw Hill, Chapter 10, pp. 401-437, 1996. Freely available at: http://www.cse.cuhk.edu.hk/~lyu/book/reliability/

11. A measurement-based framework for software reliability improvement, K. Kanoun, Annals of Software Reliability, Vol.11, N°1, pp.89-106, November 2001.

12. Windows and Linux Robustness Benchmarks With Respect to Application Erroneous Behavior, K. Kanoun, Y. Crouzet, A. Kalakech, A. E. Rugina, in Dependability Benchmarking for Computer Systems, Chapter 12, pp. 277-254. Editors: Karama Kanoun and Lisa Spainhower, IEEE Computer Society and WILEY, August 2008.