Software Dependability:

How Far are We?

Karama Kanoun

LAAS-CNRS

Dependability of Computing Systems: Memories and Future
15-16 April 2010 - Toulouse - France

Why Software Dependability Assessment?

☞ User / customer

- Confidence in the product
- Acceptable failure rate

☞ Developer / Supplier

- During production
  - ☞ Reduce # faults (zero defect)
  - ☞ Optimize development
  - ☞ Increase operational dependability
- During operation
  - ☞ Maintenance planning
- Long term
  - ☞ Improve software dependability of next generations

Approaches to Software Dependability Assessment

☞ Assessment based on software characteristics

- Language, complexity metrics, application domain, …

☞ Assessment based on measurements

- Assessment of the product
- Assessment of the production process

☞ Assessment based on controlled experiments

- Ad hoc vs standardised → benchmarking

Outline of the Presentation

☞ Assessment based on software characteristics

- Language, complexity metrics, application domain, …

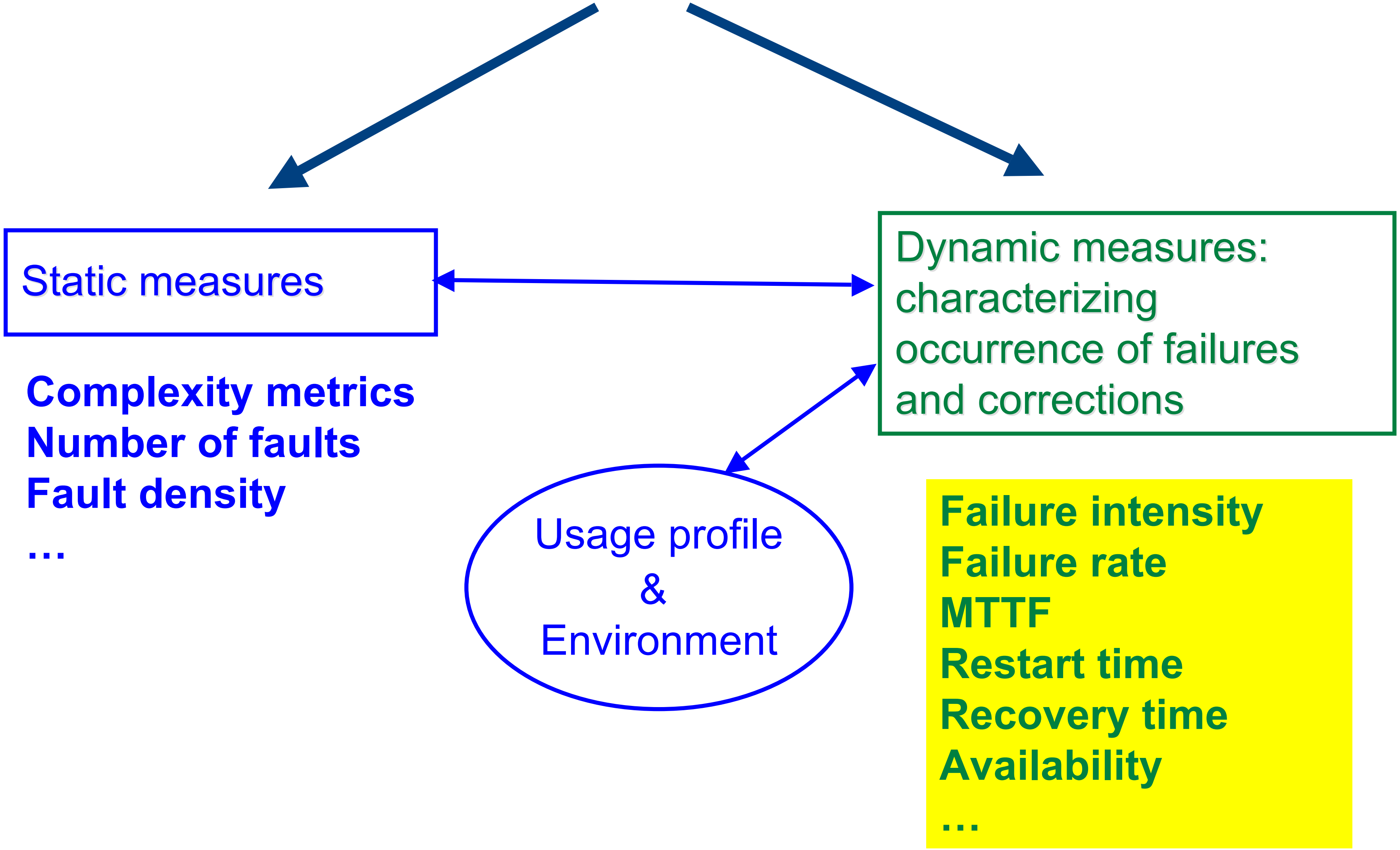
☞ Assessment based on measurements

- Assessment of the product
- Assessment of the production process

☞ Assessment based on controlled experiments

- Ad hoc vs standardized → benchmarking

Dependability Measures?



Static measures

Complexity metrics
Number of faults
Fault density
…

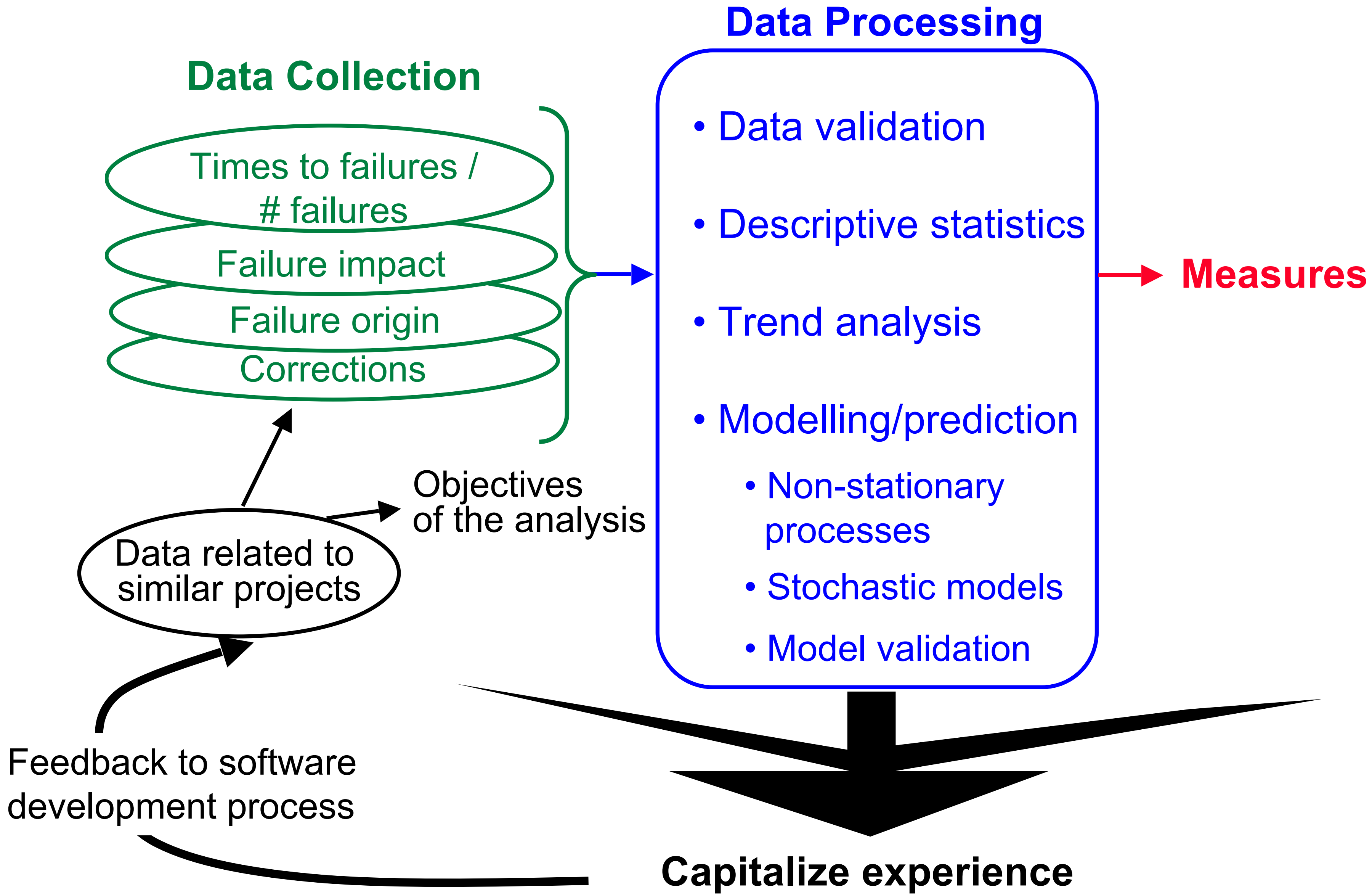
Dynamic measures: characterizing occurrence of failures and corrections

Usage profile
&
Environment

Failure intensity
Failure rate
MTTF
Restart time
Recovery time
Availability
…

Assessment Based on Measurements

Software Process Improvement (SPI)



Data Processing

- Data validation
- Descriptive statistics
- Trend analysis
- Modelling/prediction
  - Non-stationary processes
  - Stochastic models
  - Model validation

Data Collection
Times to failures / # failures
Failure impact
Failure origin
Corrections
Measures
Objectives of the analysis
Data related to similar projects
Feedback to software development process
Capitalize experience

Benefits from SPI Programmes

- **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

- **IBM** (defect prevention approach):

  Fault density divided by 2 with an increase of 0.5 % of the product resources

- **Motorola** (Arlington Heights), mix of methods:

  Fault density reduction = 50% within 3.5 years

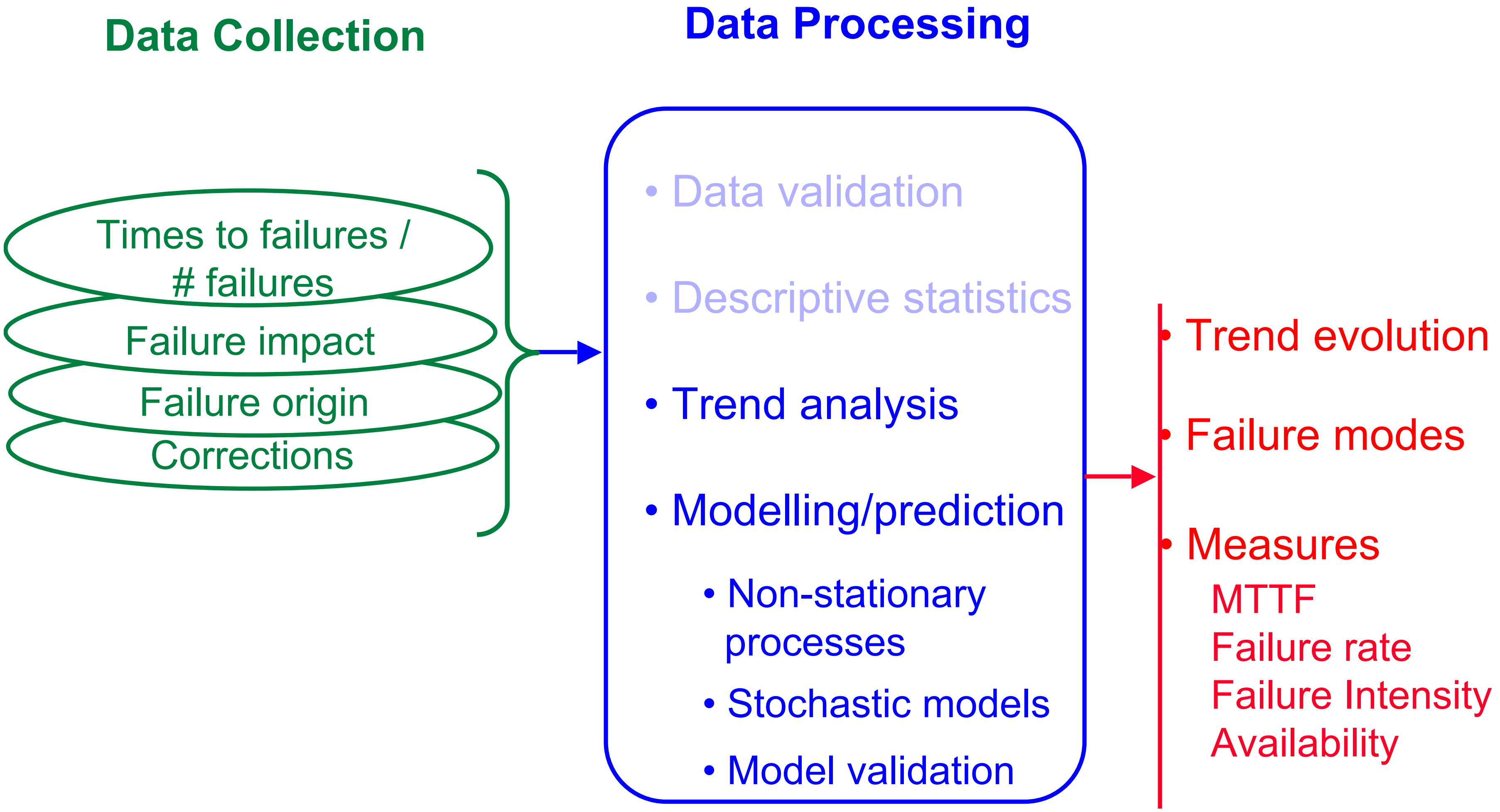
- **Raytheon** (Electronic Systems), CMM:

  Rework cost divided by 2 after two years of experience

  Productivity increase = 190%

  Product quality: multiplied by 4

Assessment Based on Measurements



Data Collection

- Times to failures / # failures
- Failure impact
- Failure origin
- Corrections

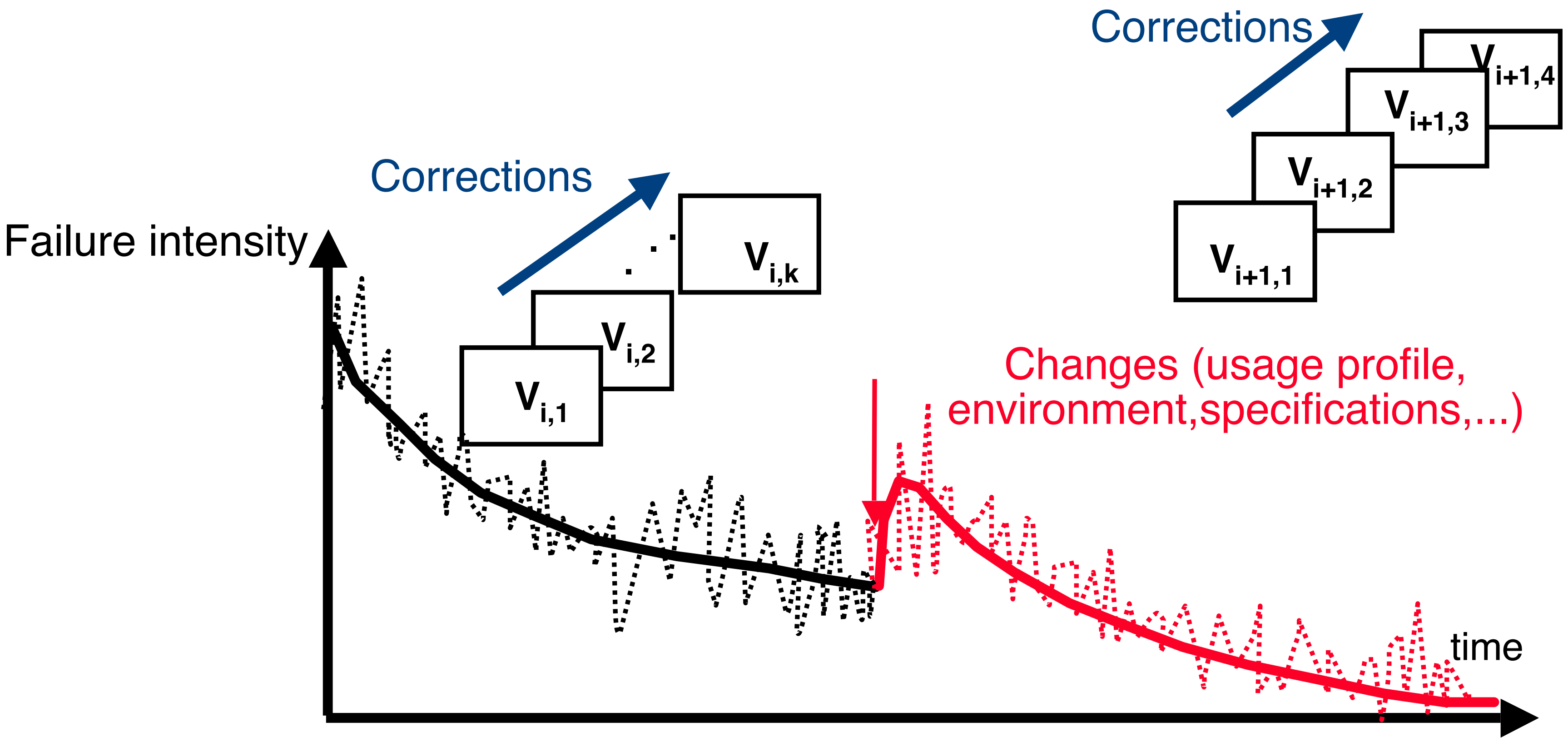
Data Processing

- Data validation
- Descriptive statistics
- Trend analysis
- Modelling/prediction
  - Non-stationary processes
  - Stochastic models
  - Model validation

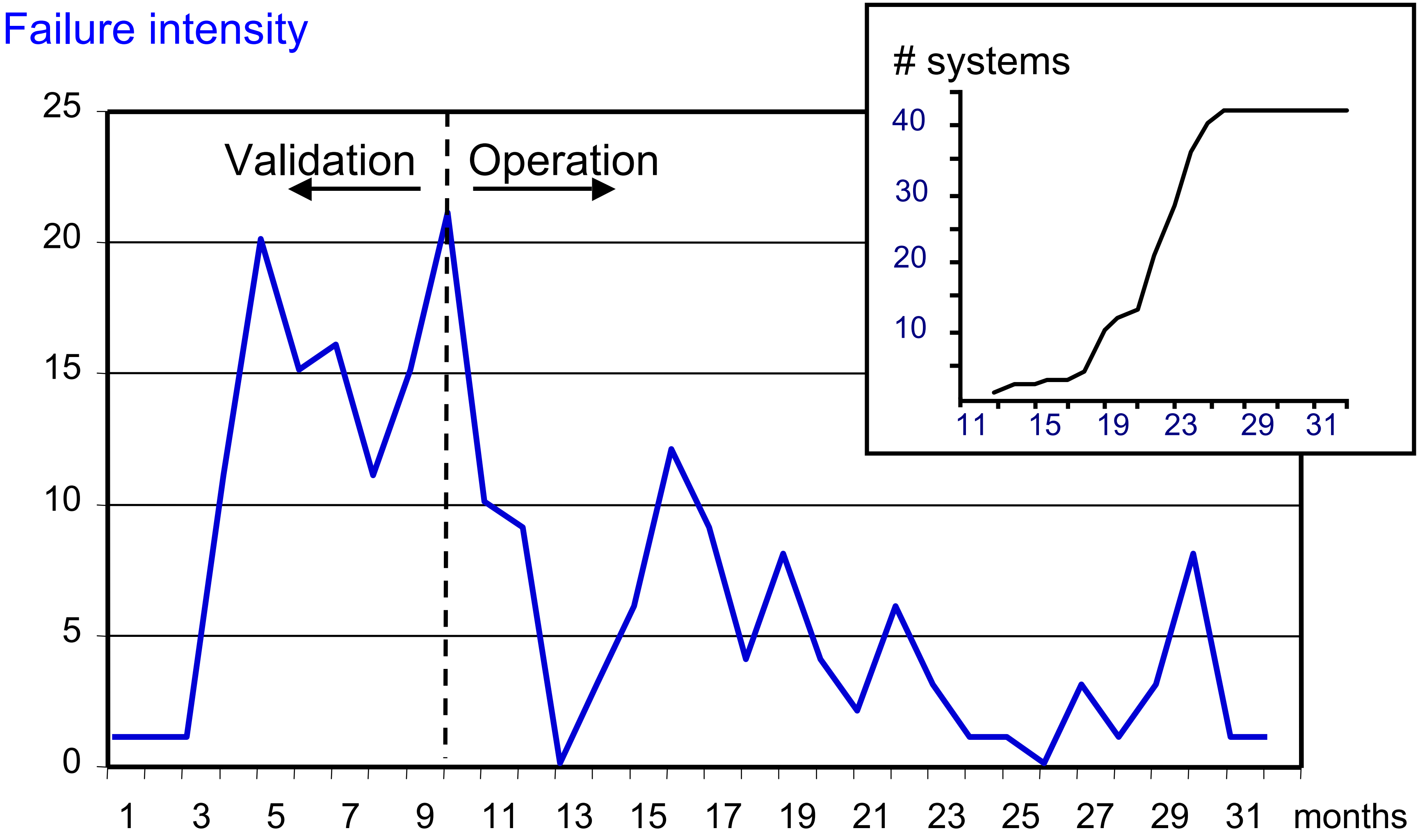
- Trend evolution
- Failure modes
- Measures

  MTTF
  Failure rate
  Failure Intensity
  Availability

Why Trend Analysis?

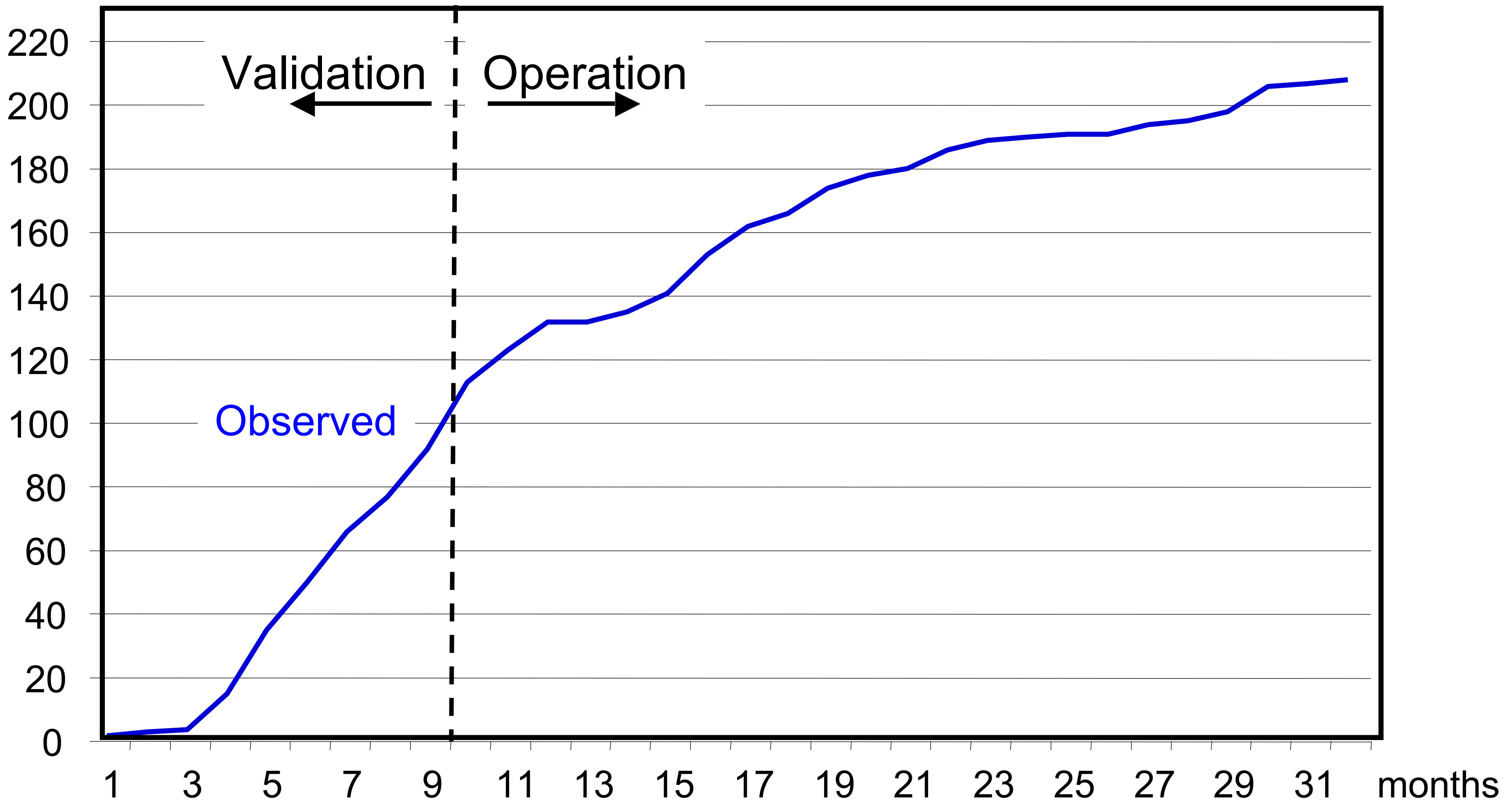


Example: Electronic Switching System



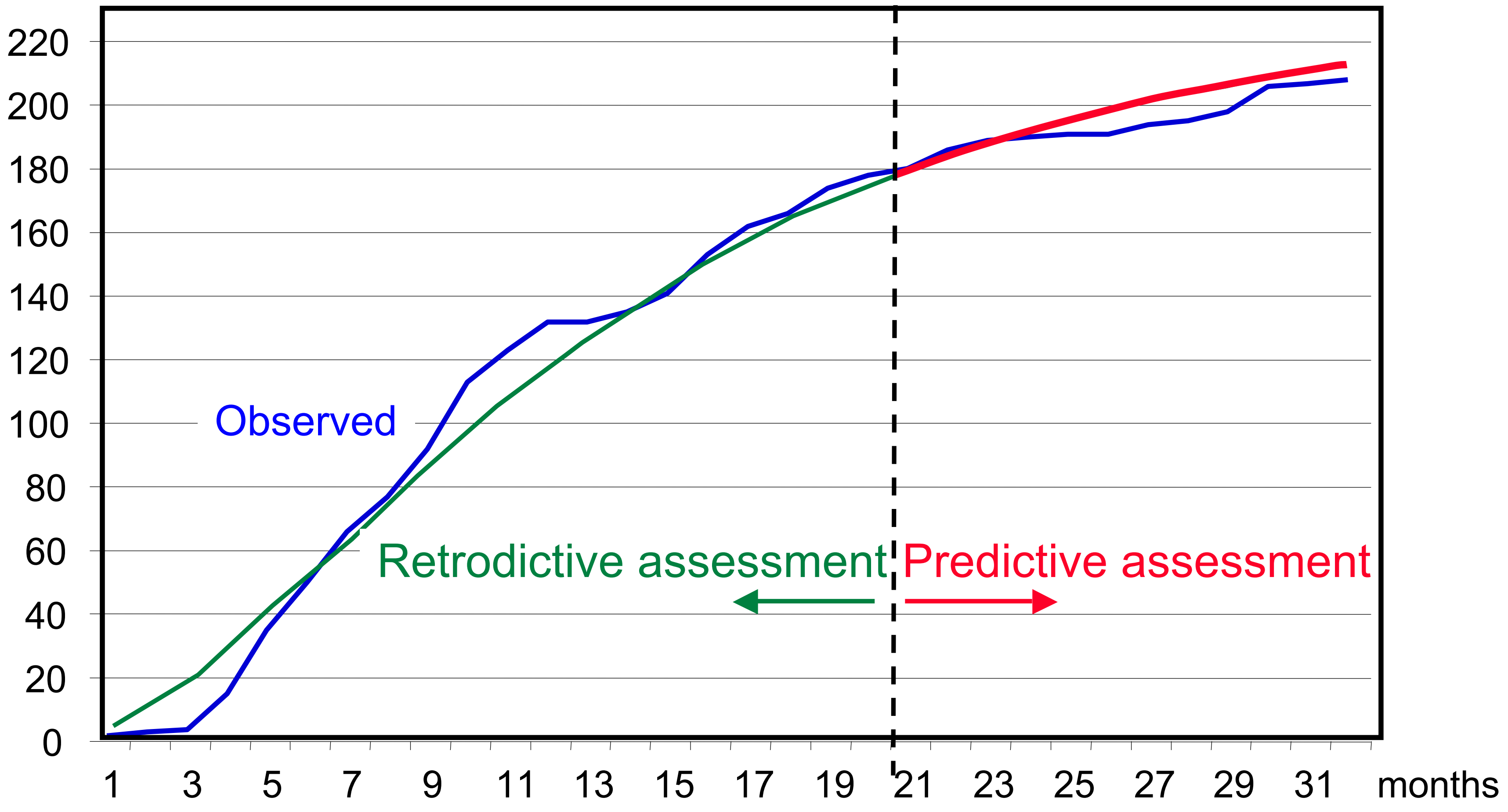
Electronic Switching System (Cont.)

Cumulative number of failures



Electronic Switching System (Cont.)

Cumulative number of failures → Hyperexponential model application
⇒ maintenance planning

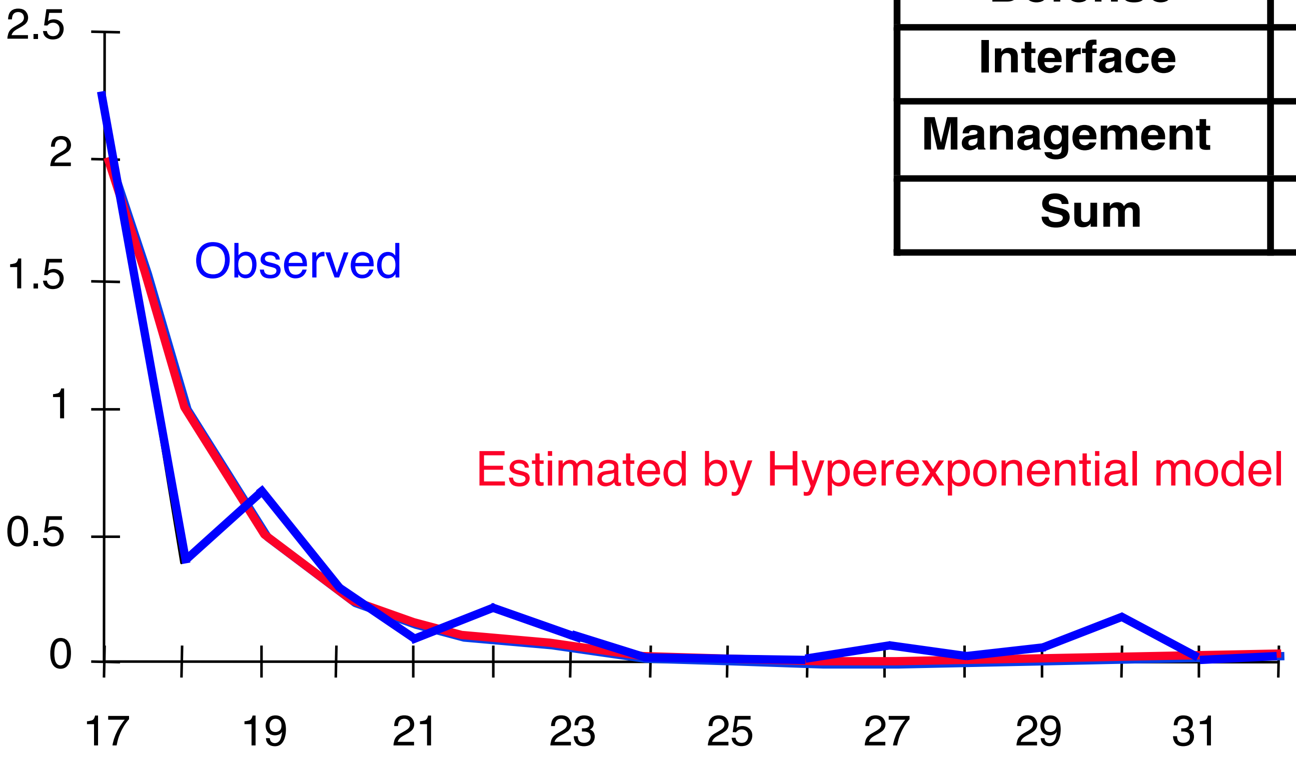


Observed # failures [20-32] = 33 Predicted # failures [21-32] = 37

Electronic Switching System (Cont.)

Failure intensity and failure rate in operation
(for an average system)

Component	Residual failure rate
Telephony	$1.2\ 10^{-6}$ /h
Defense	$1.4\ 10^{-5}$ /h
Interface	$2.9\ 10^{-5}$ /h
Management	$8.5\ 10^{-6}$ /h
Sum	$5.3\ 10^{-5}$/h

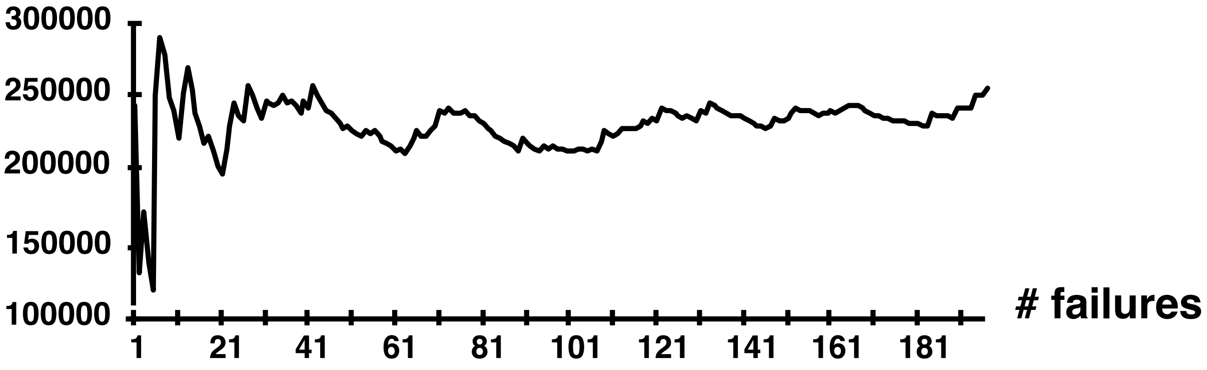


Residual failure rate: $5.7\ 10^{-5}$ /h

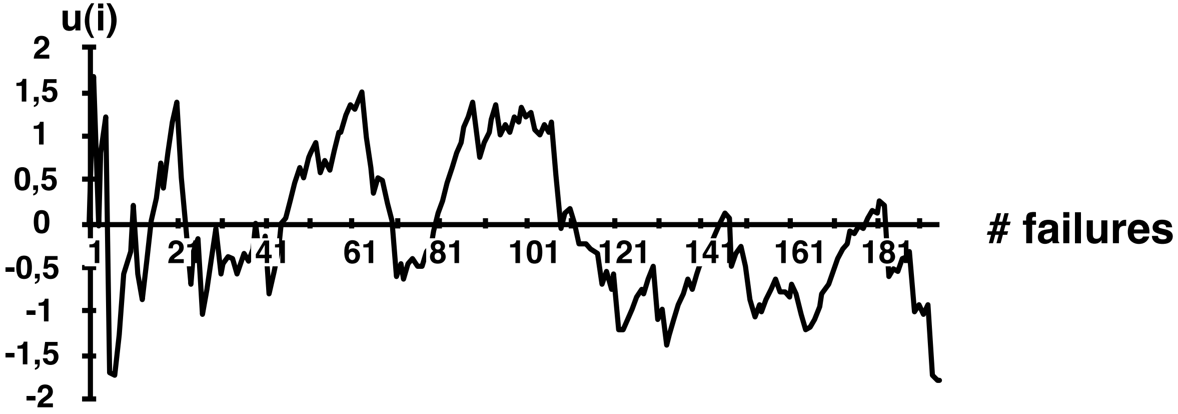
Other Example: Operating System

Observed Time to Failure during operation

Mean
Time to Failure



Trend evolution
= stable dependability



Validity of Results

Early Validation

☞ Trend analysis

→ development

follow-up

~~Assessment~~

End of Validation

☞ Trend analysis

+

☞ Assessment

- operational profile
- enough data

☞ Limits: 10^{-3}/h -10^{-4}/h

Operation

☞ Trend analysis

+

☞ Assessment

High relevance

Examples:

E10-B (Alcatel ESS):

1400 systems, 3 years

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

$\lambda_c = 10^{-7}$/h

Nuclear I&C systems:

8000 systems, 4 years

λ: $3\ 10^{-7}$/h → 10^{-7}/h

$\lambda_c = 4\ 10^{-8}$/h

Research Gaps

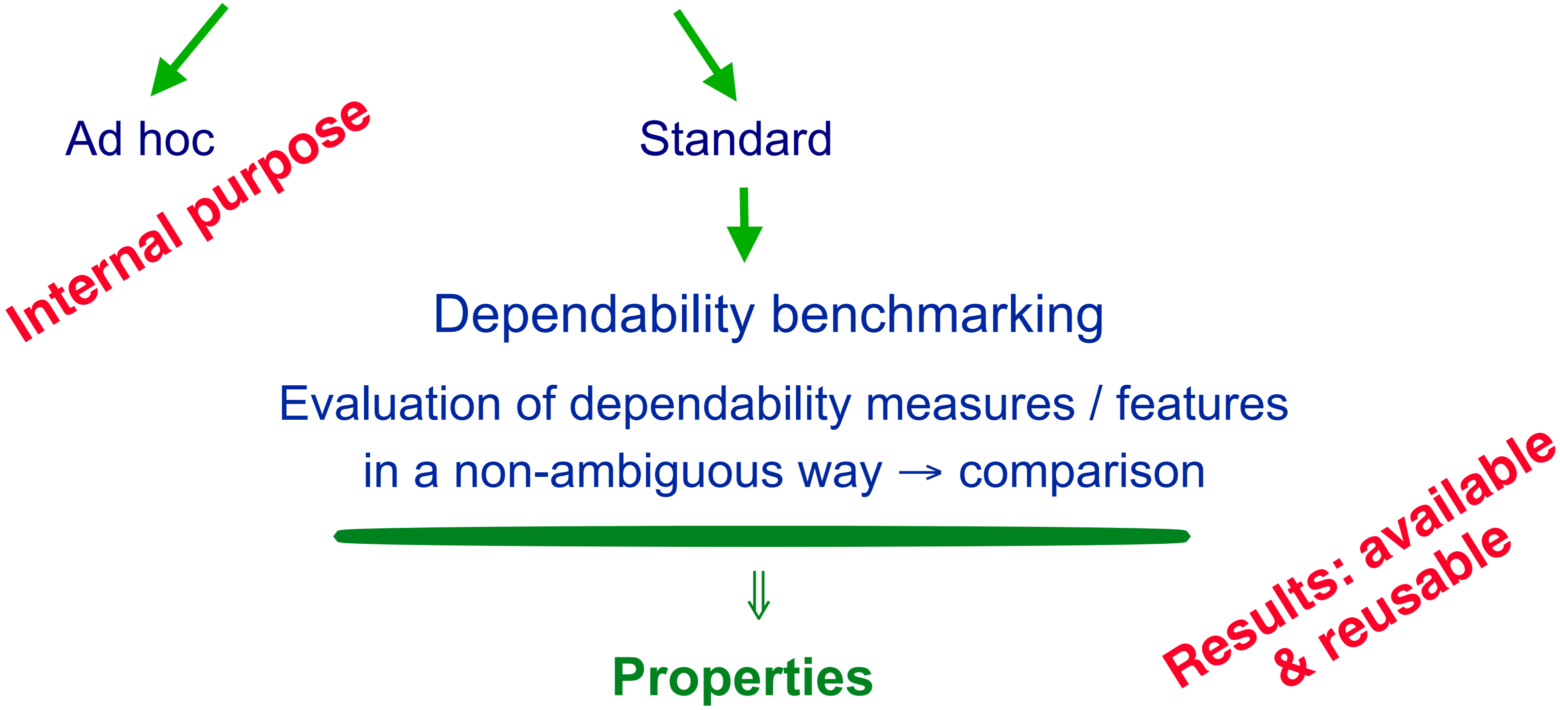
- ☞ Applicability to safety critical systems
  - During development
- ☞ Applicability to new classes of systems
  - Service oriented systems
  - Adaptive and dynamic software systems ⇒ on-line assessment
- ☞ Industry implication
  - Confidentiality ⇒ real-life data
  - Cost (perceptible overhead, invisible immediate benefits)
- ☞ Accumulation of experience ⇒ software process improvement

  ⇒ assessment of the software process
- ☞ Case of Off-The-Shelf software?

Dependability Benchmarking
Off-The-Shelf software

- ☞ No information available from software development

- ☞ Evaluation based on controlled experimentation



Ad hoc

Internal purpose

Standard

Dependability benchmarking

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

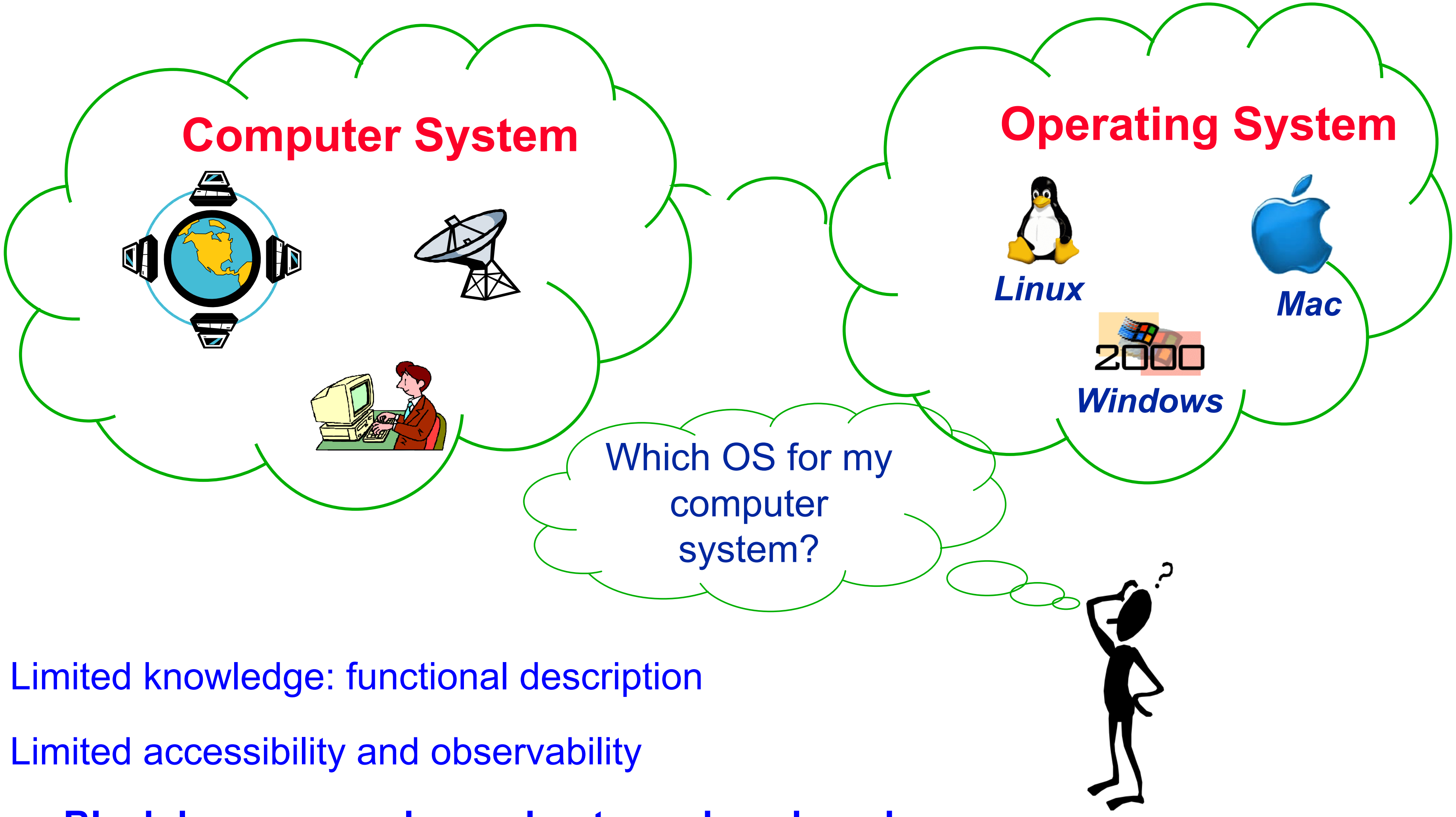
⇓

Properties

Results: available & reusable

Reproducibility, repeatability, portability, representativeness, acceptable cost

Benchmarks of Operating Systems



Computer System

Operating System

Linux

Mac

2000

Windows

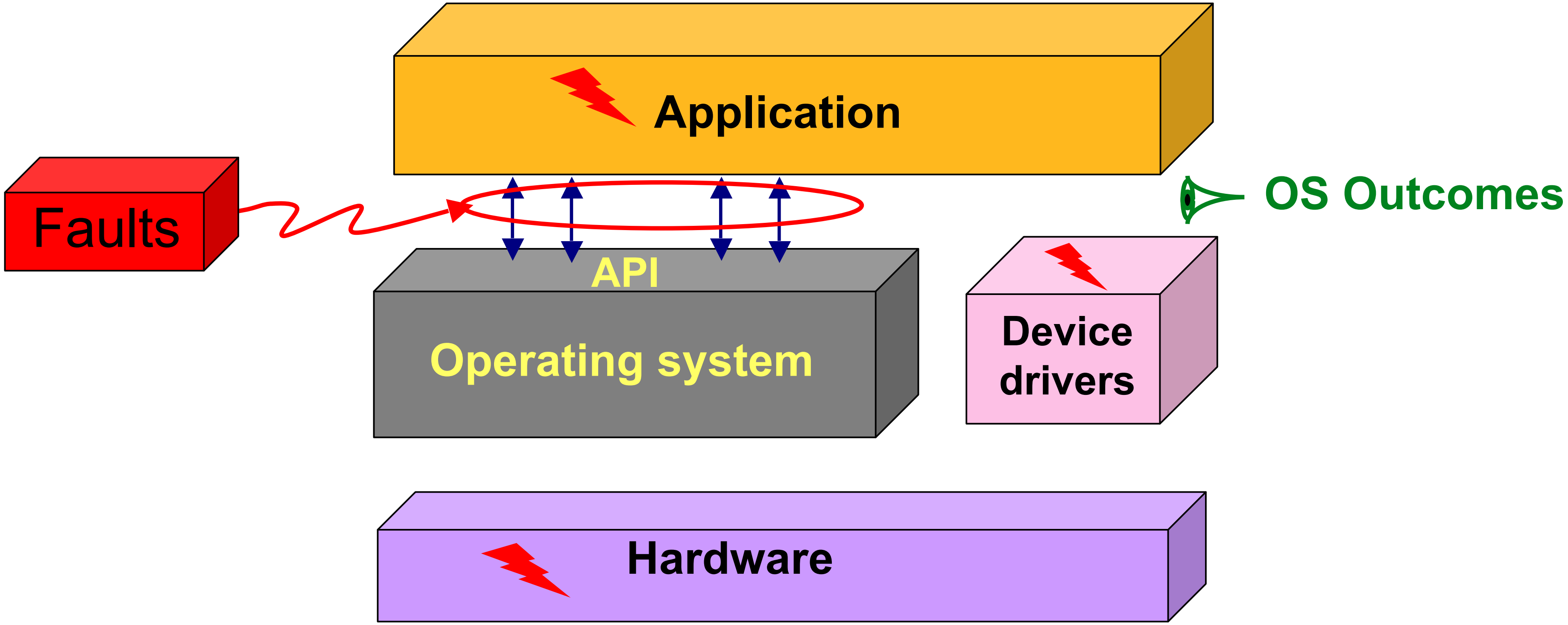
Which OS for my computer system?

☞ Limited knowledge: functional description

☞ Limited accessibility and observability

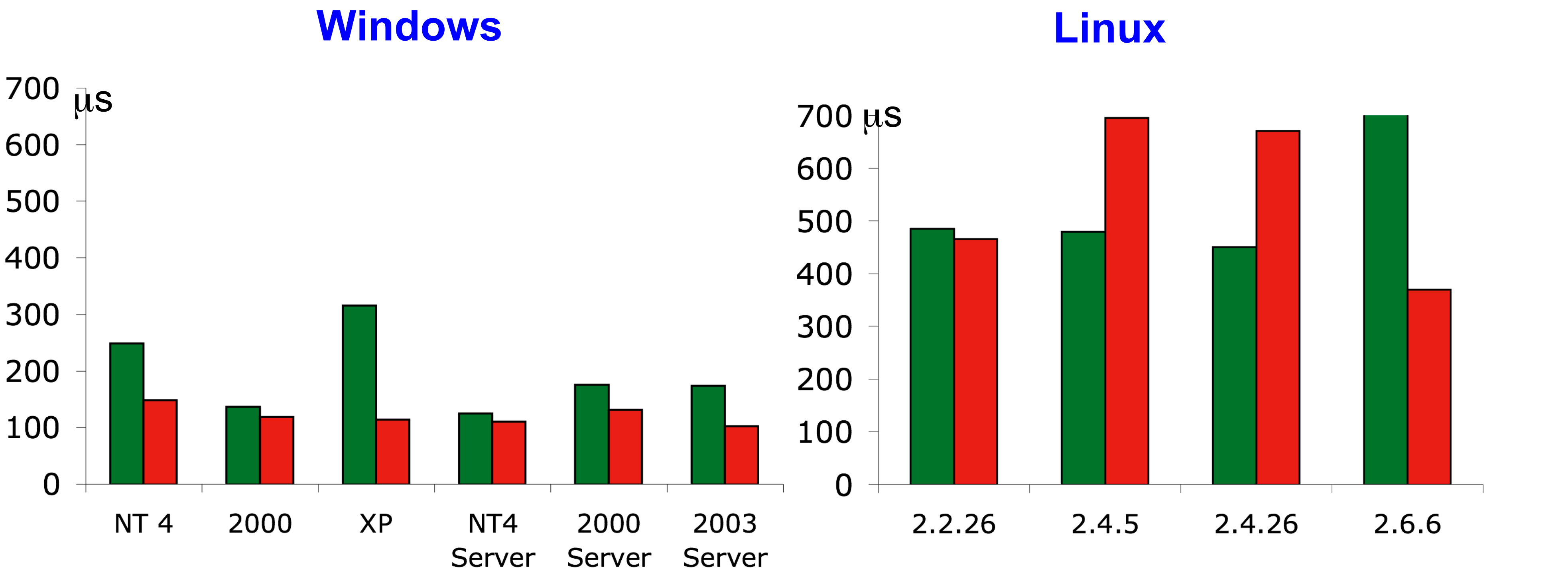
⇒ **Black-box approach ⇒ robustness benchmark**

Robustness Benchmarks



Faults = corrupted system calls

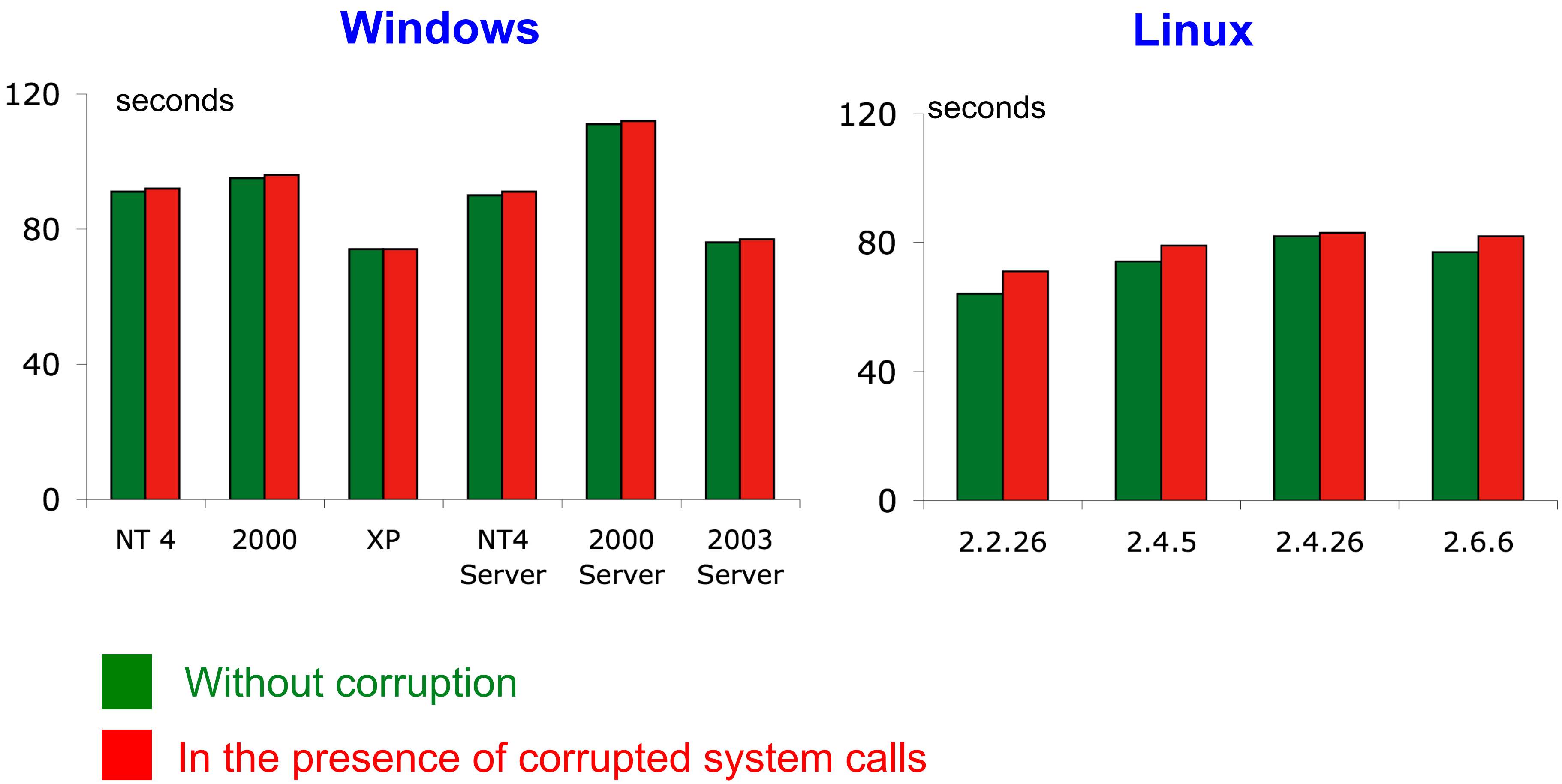
OS Response Time to Faults in the Application



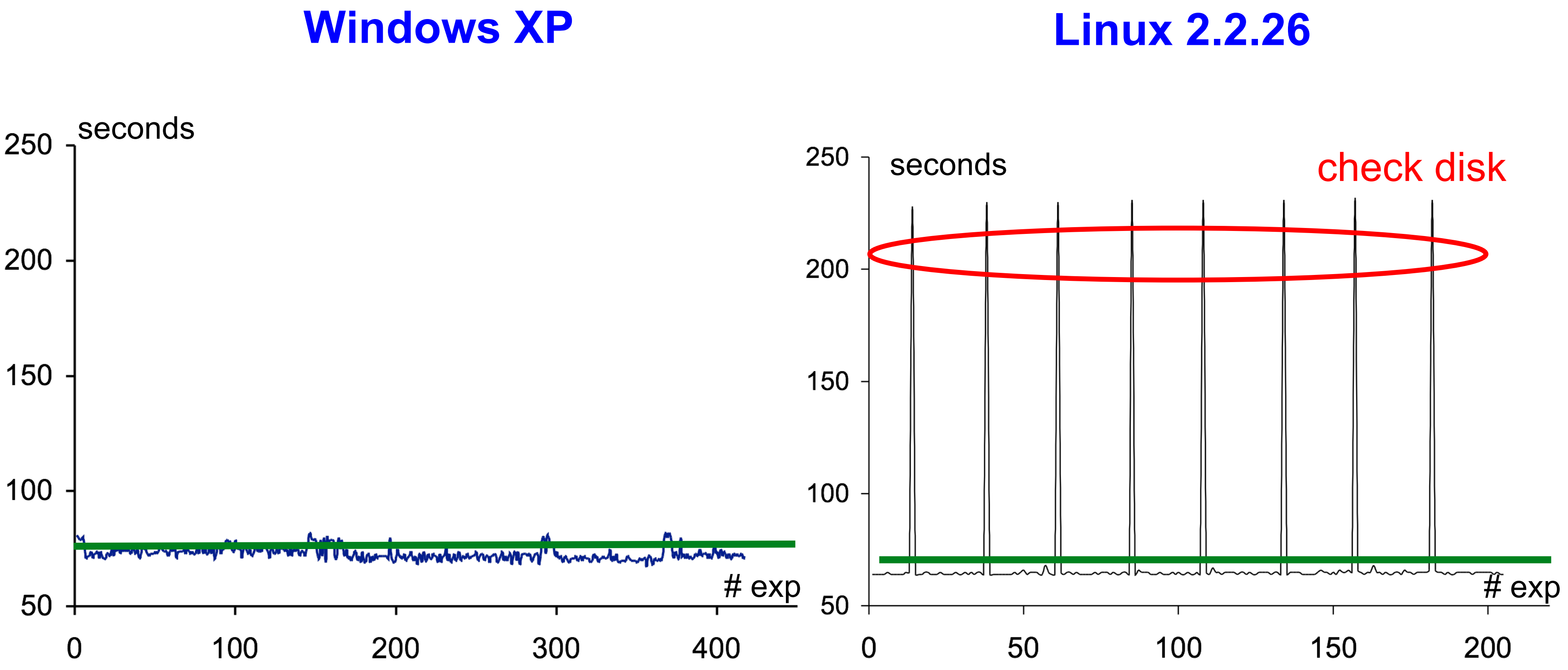
🟩 Without corruption

🟥 In the presence of corrupted system calls

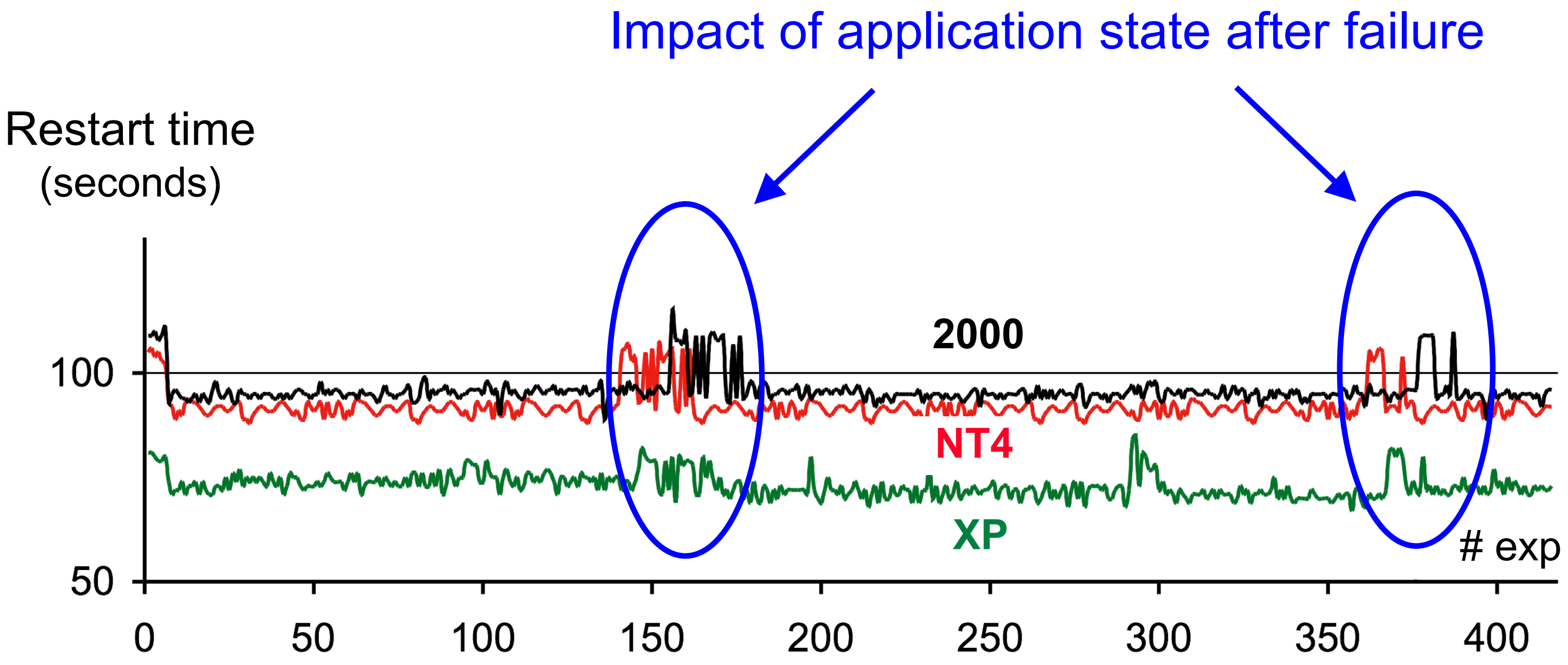
Mean Restart Time



Detailed Restart Time



More on Windows family



Benchmark Characteristics

- ☞ A benchmark should not replace software test and validation
- ☞ Non-intrusiveness ⇒ robustness benchmarks

  (faults injected outside the benchmark target)
- ☞ Make use of available inputs and outputs → impact on measures
- ☞ Balance between cost and degree of confidence
- ☞ # dependability benchmark measures >

  # performance benchmark measures

⇒ Lack of maturity

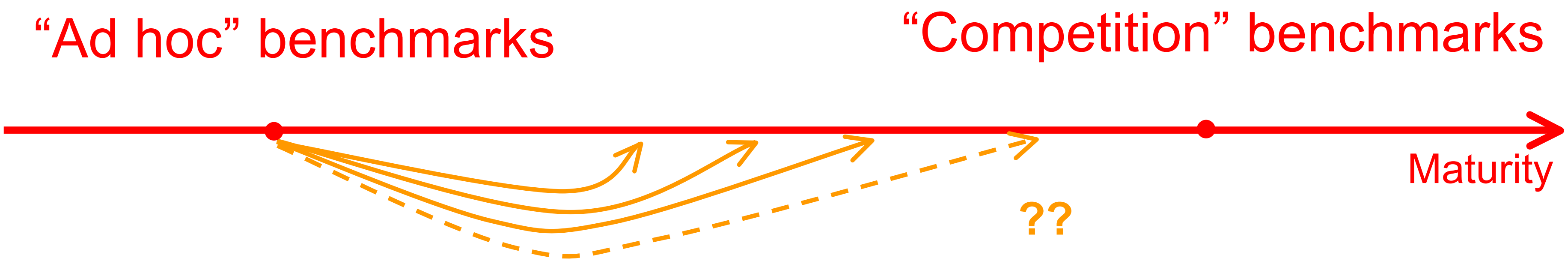
Maturity

☞ Dependability benchmarks

- Infancy
- Isolated work
- Not explicitly addressed
- Acceptability?

☞ Performance benchmarks

- Mature domain
- Cooperative work
- Integrated to system development
- Accepted by all actors for competitive system comparison



“Ad hoc” benchmarks

“Competition” benchmarks

Maturity

??

Ultimate objective:
more reliable software, faster, and cheaper!

Software Dependability:

How Far are We?

Karama Kanoun

LAAS-CNRS

Dependability of Computing Systems: Memories and Future
15-16 April 2010 - Toulouse - France