Performance Evaluation of Computer Systems

Jean-Marc Vincent and Arnaud Legrand [1]

[1] MESCAL Project
Laboratory of Informatics of Grenoble (LIG)
Universities of Grenoble
{Jean-Marc.Vincent,Arnaud.Legrand}@imag.fr
http://mescal.imag.fr/members.php
http://mescal.imag.fr/membres/arnaud.legrand/teaching/2013/M2R_EP.php

Outline

Outline

Research activities in performance evaluation

Teams in Grenoble

- Mescal project : Large systems (clusters and grids)
- Moais project : Interactive parallel systems
- Drakkar team : Networking
- Erods (Sardes) : Middleware
- Verimag : Embedded systems
- etc

Industrial collaborations

- France-Télécom R & D : load injectors, performances of middlewares
- HP-Labs : cluster computing, benchmarking
- Bull : benchmarking, performances analysis
- ST-Microelectronics

Research activities in performance evaluation

Teams in Grenoble

- Mescal project : Large systems (clusters and grids)
- Moais project : Interactive parallel systems
- Drakkar team : Networking
- Erods (Sardes) : Middleware
- Verimag : Embedded systems
- etc

Industrial collaborations

- France-Télécom R & D : load injectors, performances of middlewares
- HP-Labs : cluster computing, benchmarking
- Bull : benchmarking, performances analysis
- ST-Microelectronics

Application context (1)

Complexity of computer systems

- **hierarchy** : level decomposition : OS / Middleware / Application
- **distribution** : asynchronous resources : memory, CPU, network
- **dynamicity** : architecture and environment (reliability, mobility,...)
- **scalability** : number of components (autonomous management)

Typical problems

- Minimize losses in routing policies
- Minimize active waiting in threads scheduling
- Maximize cache hits
- Optimise block sizes in parallel applications
- Maximize troughput of communication systems
- Fix time-outs, reemission periods, ...
- Fix the granularity : pages, blocks, tables, message sizes...
- ...

Application context (1)

Complexity of computer systems

- **hierarchy** : level decomposition : OS / Middleware / Application
- **distribution** : asynchronous resources : memory, CPU, network
- **dynamicity** : architecture and environment (reliability, mobility,...)
- **scalability** : number of components (autonomous management)

Typical problems

- Minimize losses in routing policies
- Minimize active waiting in threads scheduling
- Maximize cache hits
- Optimise block sizes in parallel applications
- Maximize troughput of communication systems
- Fix time-outs, reemission periods, ...
- Fix the granularity : pages, blocks, tables, message sizes...
- ...

Application context (2)

Typical “hot” applications

- **Peer to peer systems** : dimensionning, control
- **Mobile networks** : ad-hoc networking, reactivity, coherence
- **Grids** : resources utilization, scheduling
- etc

Other application domains

- production systems : production lines, logistic,...
- embedded systems
- modelling of complex systems : biology, sociology,...
- etc

Application context (2)

Typical “hot” applications

- **Peer to peer systems** : dimensionning, control
- **Mobile networks** : ad-hoc networking, reactivity, coherence
- **Grids** : resources utilization, scheduling
- etc

Other application domains

- production systems : production lines, logistic,...
- embedded systems
- modelling of complex systems : biology, sociology,...
- etc

Outline

Development of parallel/distributed aplications

Qualitative specifications : Is the result correct ?

- property verification : formal/automatic proofs
- testing : critical dataset

Quantitative specifications: Is the result obtained in an reasonable time ?

- performance model
- performance measurements

Problem identification: localization of the problem

- debugging, log analysis
- performance statistical analysis

Modification and control

- source code / libraries / OS / architecture
- parameters of the system : dimensioning
- control algorithms : tuning

Development of parallel/distributed aplications

Qualitative specifications : Is the result correct ?

- property verification : formal/automatic proofs
- testing : critical dataset

Quantitative specifications: Is the result obtained in an reasonable time ?

- performance model
- performance measurements

Problem identification: localization of the problem

- debugging, log analysis
- performance statistical analysis

Modification and control

- source code / libraries / OS / architecture
- parameters of the system : dimensioning
- control algorithms : tuning

Development of parallel/distributed aplications

Qualitative specifications : Is the result correct ?

- property verification : formal/automatic proofs
- testing : critical dataset

Quantitative specifications: Is the result obtained in an reasonable time ?

- performance model
- performance measurements

Problem identification: localization of the problem

- debugging, log analysis
- performance statistical analysis

Modification and control

- source code / libraries / OS / architecture
- parameters of the system : dimensioning
- control algorithms : tuning

Development of parallel/distributed aplications

Qualitative specifications : Is the result correct ?

- property verification : formal/automatic proofs
- testing : critical dataset

Quantitative specifications: Is the result obtained in an reasonable time ?

- performance model
- performance measurements

Problem identification: localization of the problem

- debugging, log analysis
- performance statistical analysis

Modification and control

- source code / libraries / OS / architecture
- parameters of the system : dimensioning
- control algorithms : tuning

Dual analysis

Understand the behavior of a distributed application

1. identification of distributed patterns, states of the system
2. pattern verification
3. time evaluation
4. global analysis of the execution and performance synthesis
5. system monitoring
6. global cost evaluation for the application user

Understand resources utilization

1. hierarchical model of resources
2. evaluation of utilization at :
   application level; executive runtime;
   operating system; hardware architecture
3. global cost evaluation for the resources manager

Dual analysis

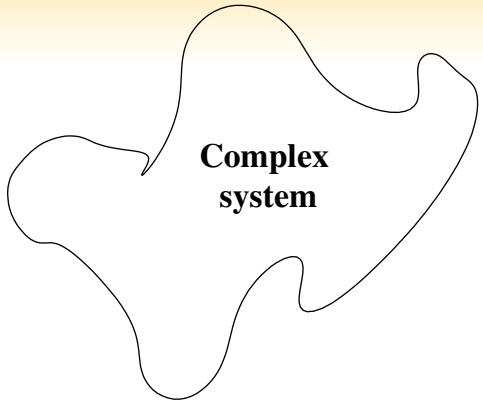
Understand the behavior of a distributed application

1. identification of distributed patterns, states of the system
2. pattern verification
3. time evaluation
4. global analysis of the execution and performance synthesis
5. system monitoring
6. global cost evaluation for the application user

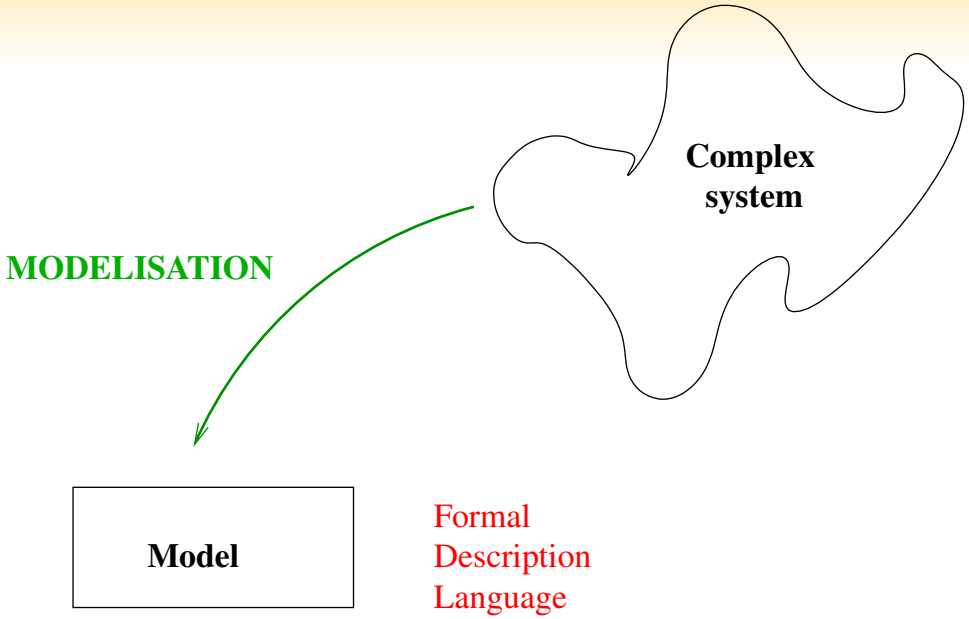
Understand resources utilization

1. hierarchical model of resources
2. evaluation of utilization at :
   application level; executive runtime;
   operating system; hardware architecture
3. global cost evaluation for the resources manager

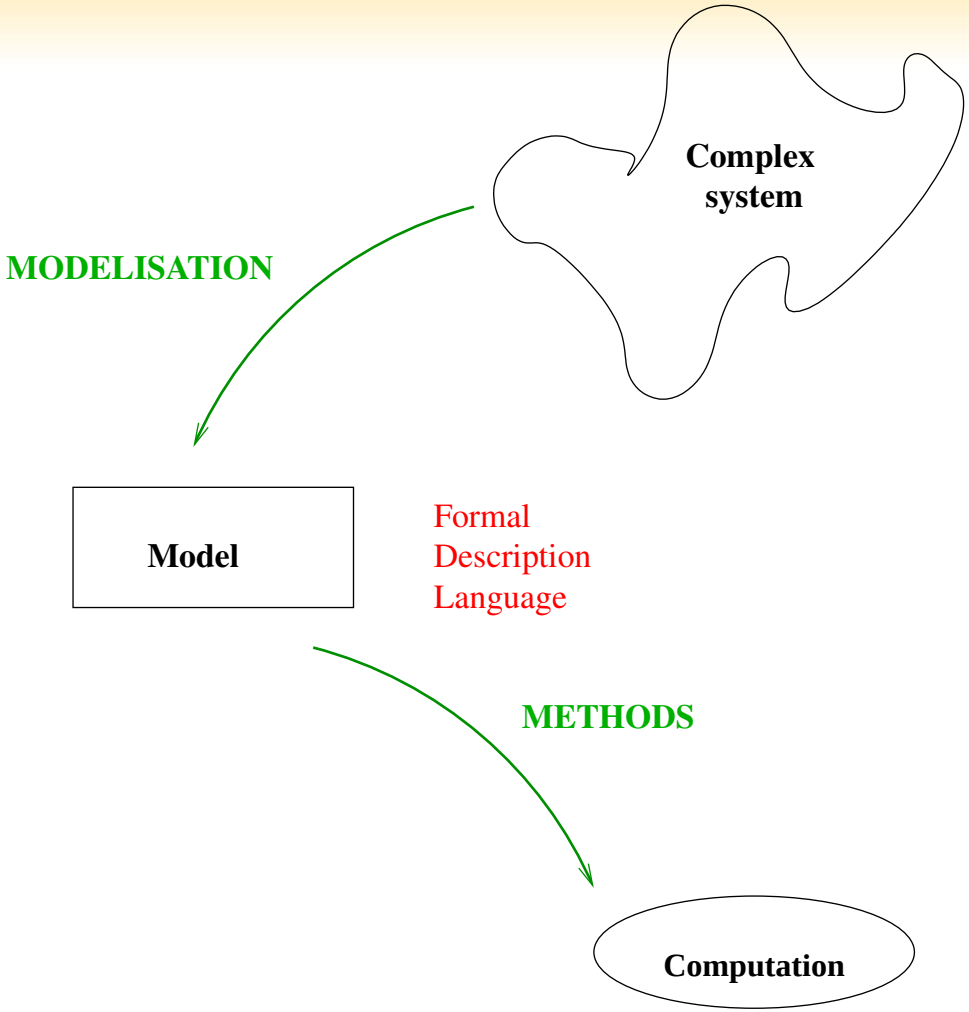
Methodology (1)



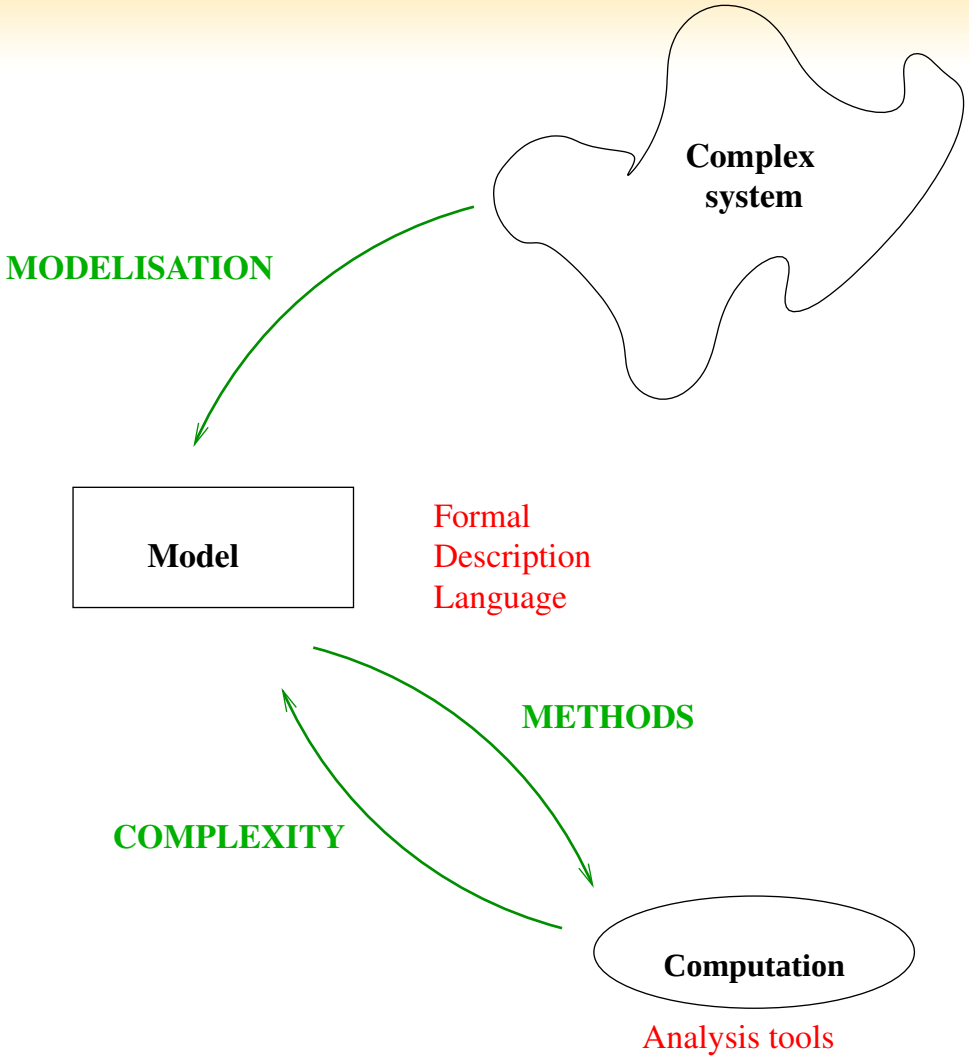
Methodology (1)



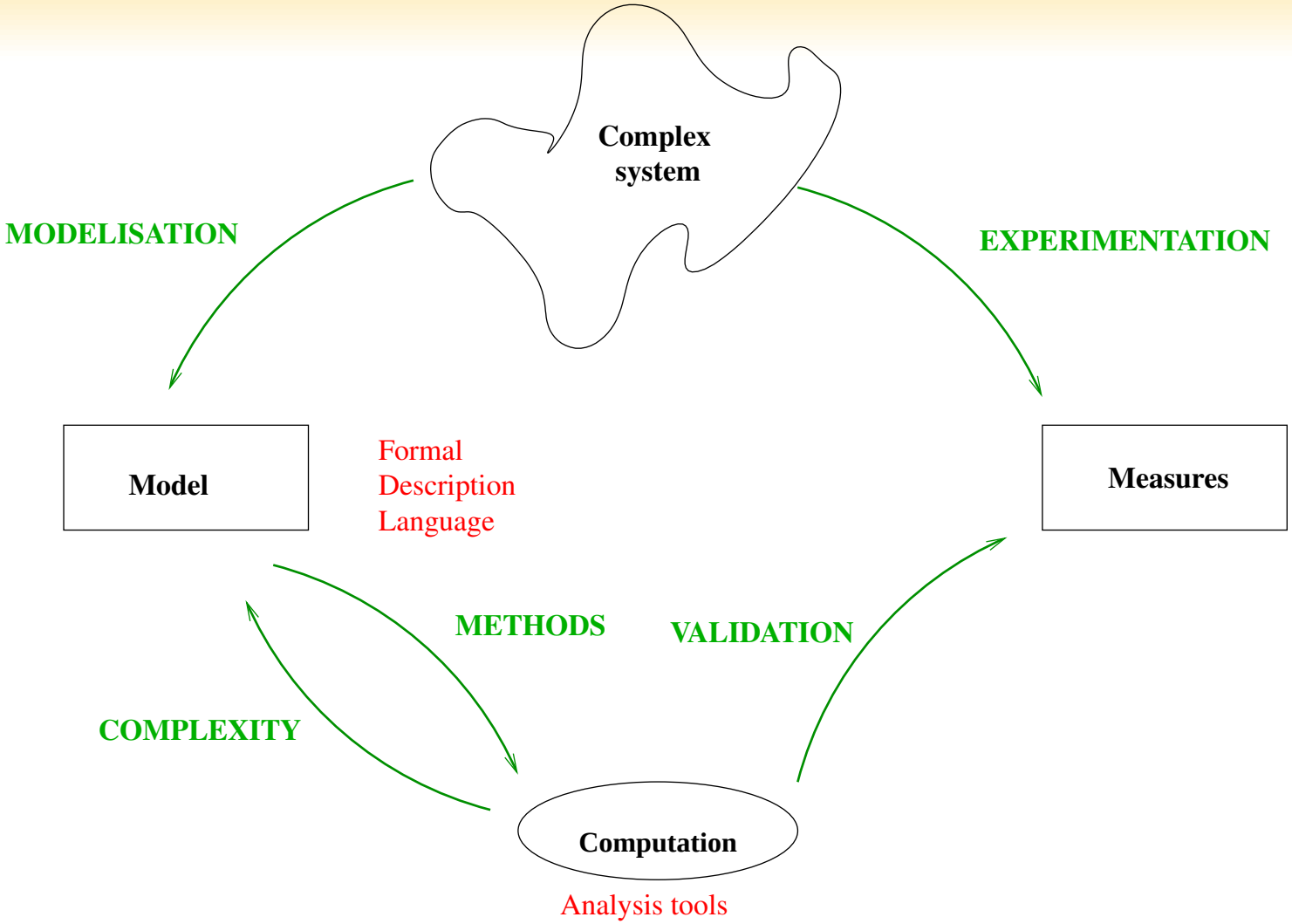
Methodology (1)



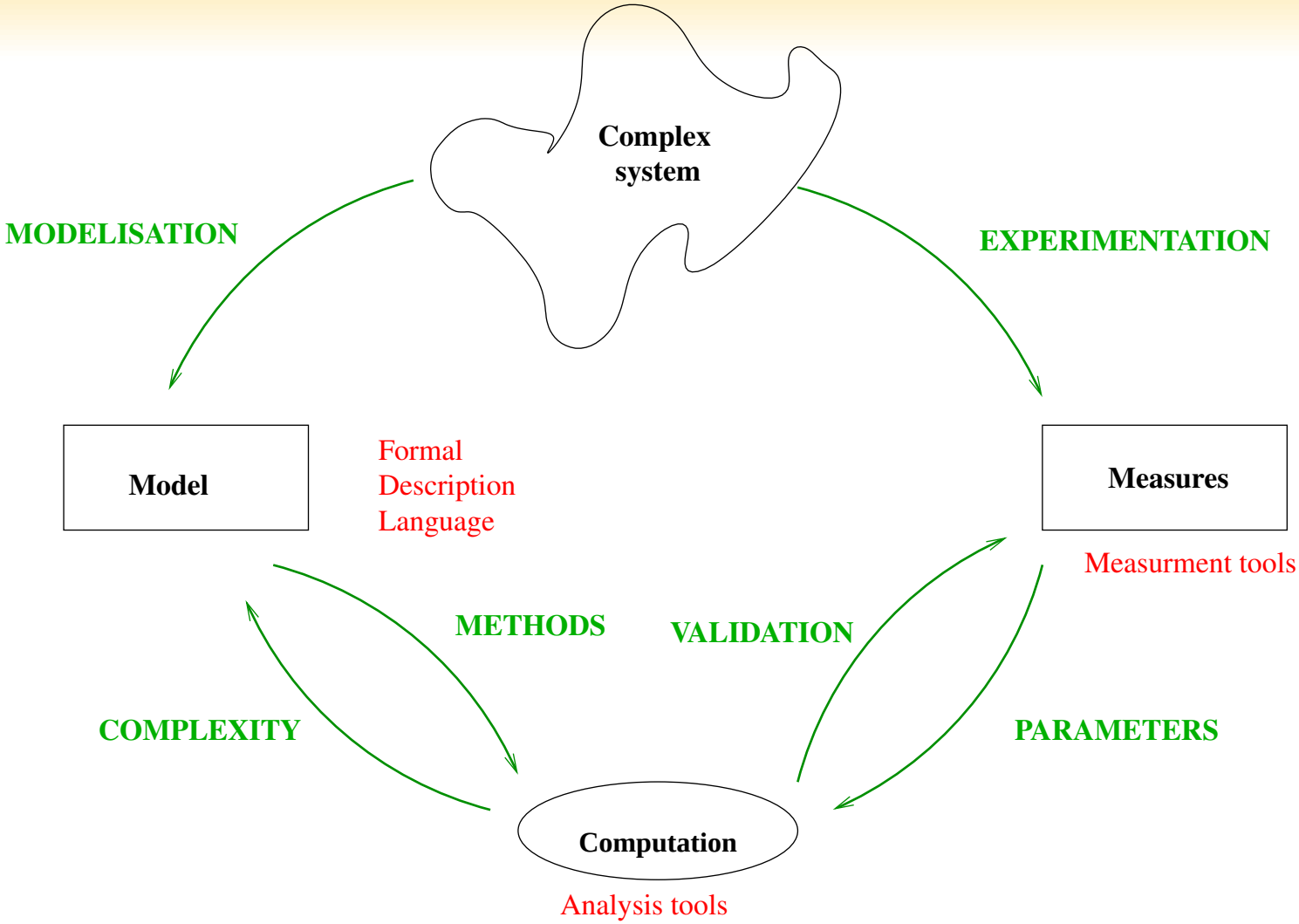
Methodology (1)



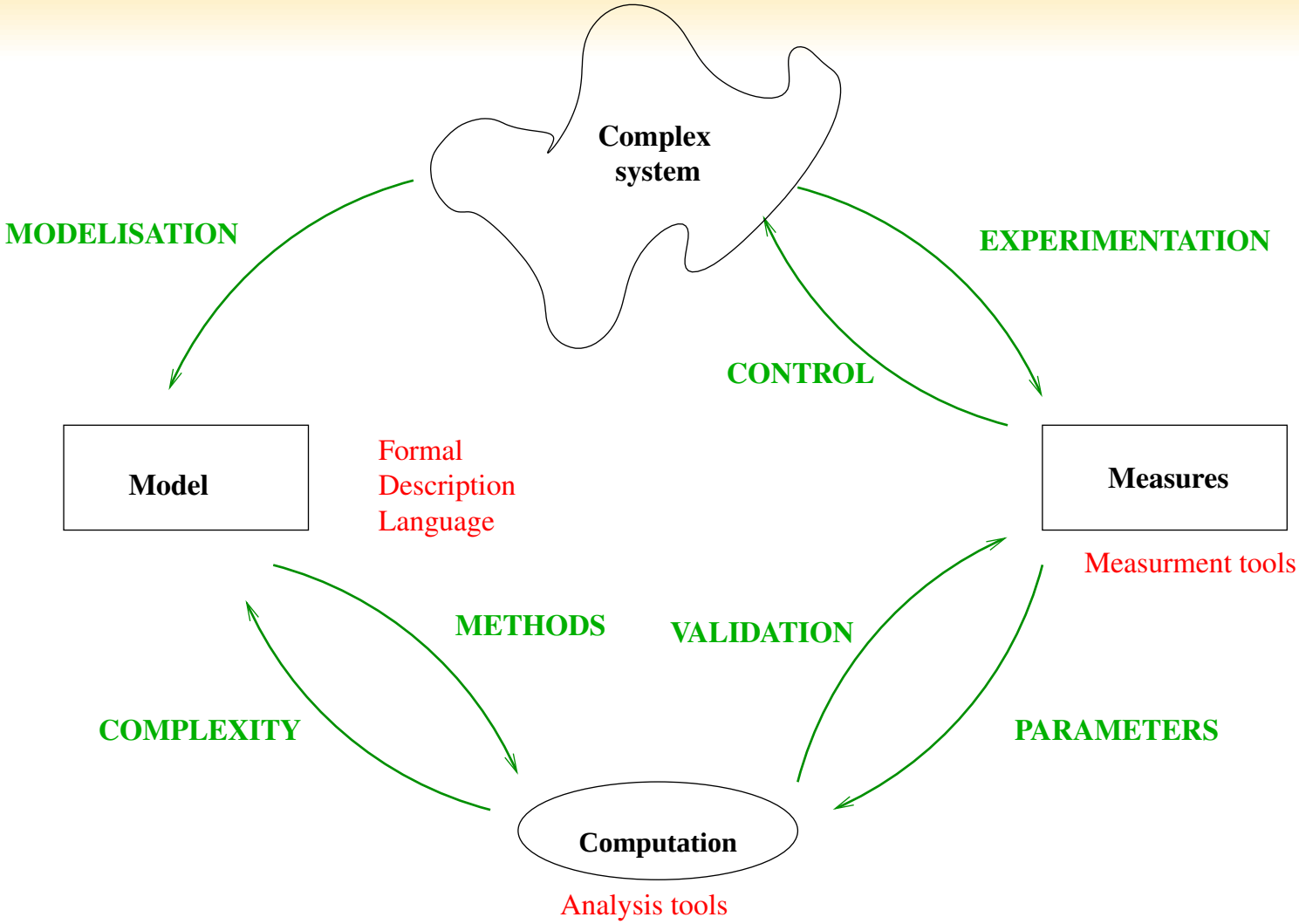
Methodology (1)



Methodology (1)



Methodology (1)



Evaluation methods

From abstraction to physical reality

Model **Method**

Remarks :

Hybrid methods (emulation, load injectors, synthetic programs,...)
Dynamical process of evaluation
Experimentation ⇒ Planning experiments methodology

Evaluation methods

From abstraction to physical reality

Model	Method
Mathematical $\longrightarrow$	Analysis (formal, numerical, approximation)

Remarks :

Hybrid methods (emulation, load injectors, synthetic programs,...)
Dynamical process of evaluation
Experimentation $\Rightarrow$ Planning experiments methodology

Evaluation methods

From abstraction to physical reality

Model		Method
Mathematical	⟶	Analysis (formal, numerical, approximation)
Software	⟶	Simulation (discrete events)

Remarks :

Hybrid methods (emulation, load injectors, synthetic programs,...)
Dynamical process of evaluation
Experimentation ⇒ Planning experiments methodology

Evaluation methods

From abstraction to physical reality

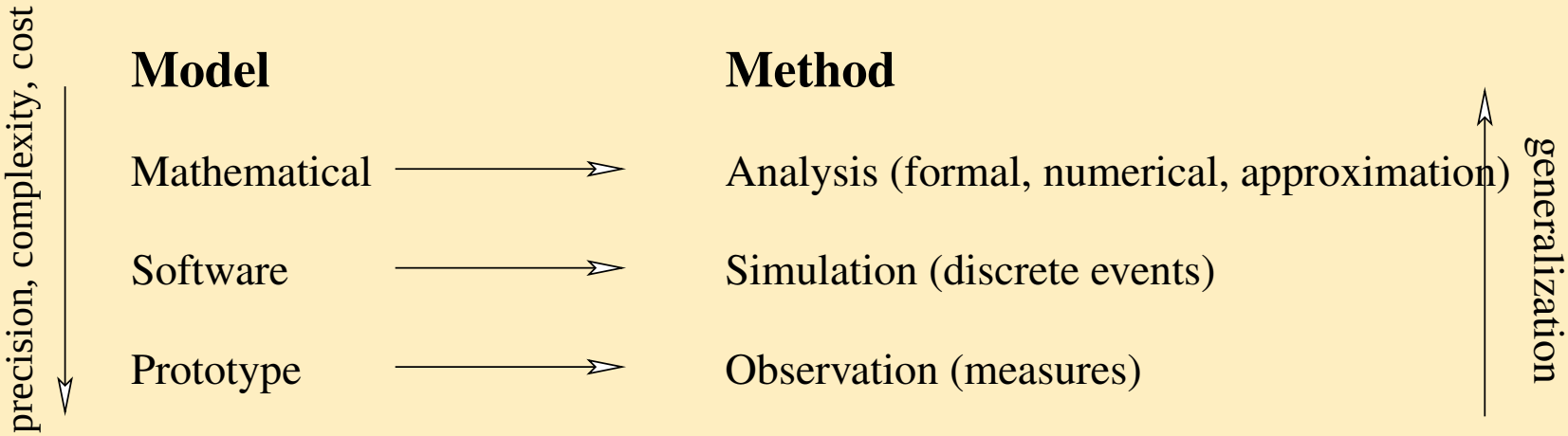
Model		Method
Mathematical	⟶	Analysis (formal, numerical, approximation)
Software	⟶	Simulation (discrete events)
Prototype	⟶	Observation (measures)

Remarks :

Hybrid methods (emulation, load injectors, synthetic programs,...)
Dynamical process of evaluation
Experimentation ⇒ Planning experiments methodology

Evaluation methods

From abstraction to physical reality

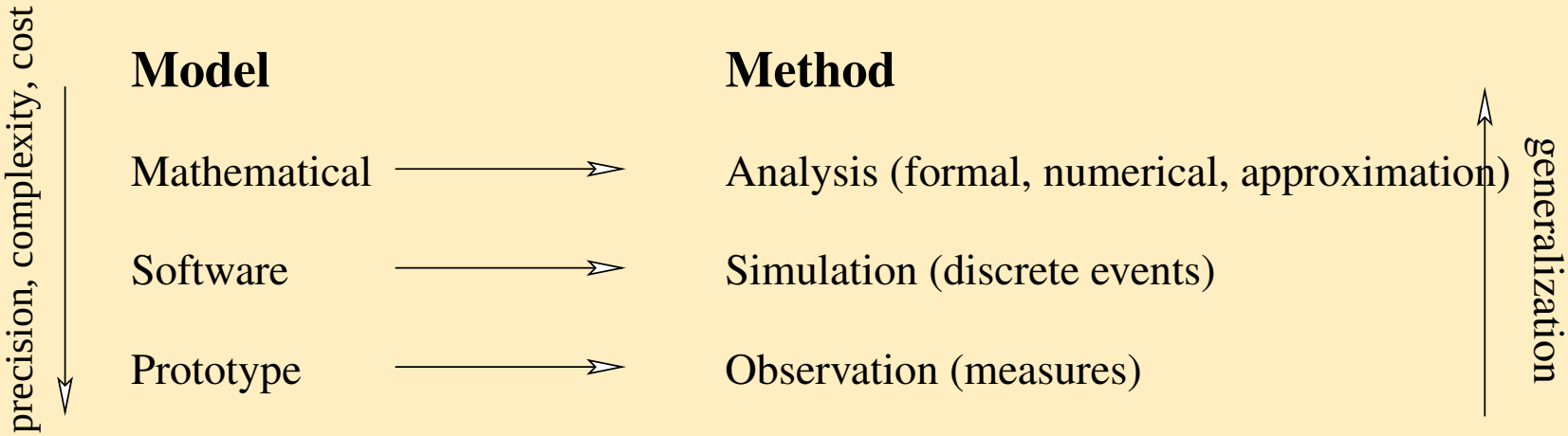


Remarks :

Hybrid methods (emulation, load injectors, synthetic programs,...)
Dynamical process of evaluation
Experimentation ⇒ Planning experiments methodology

Evaluation methods

From abstraction to physical reality



Remarks :

Hybrid methods (emulation, load injectors, synthetic programs,...)
Dynamical process of evaluation
Experimentation ⇒ Planning experiments methodology

Steps for a Performance Evaluation Study (Jain)

1. State the goals of the study : level of decision, investment, optimization, technical,...
2. Define system boundaries.
3. List system services and possible outcomes.
4. Select performance metrics.
5. List system and workload parameters
6. Select factors and their values.
7. Select evaluation techniques.
8. Select the workload.
9. Design the experiments.
10. Analyze and interpret the data.
11. Present the results. Start over, if necessary.

Aim of the course

Objective

1. Be able to analyze and predict performances of parallel/distributed systems
2. Be able to build a software environment that produces the performances indexes.

Methods

1. Specification and identification of problems : modelling
2. Analysis of quantitative models : formal, numerical, simulation
3. Experimentation and statistical data analysis.

Aim of the course

Objective

1. Be able to analyze and predict performances of parallel/distributed systems
2. Be able to build a software environment that produces the performances indexes.

Methods

1. Specification and identification of problems : modelling
2. Analysis of quantitative models : formal, numerical, simulation
3. Experimentation and statistical data analysis.

Organization of the course

Performance evaluation of systems 5 lectures 3h

1. Monday 23/10/2013 (13h45-16h45): JMV+AL] Data presentation, reporting results. Introduction to visualizing data with R and reporting results.
2. Friday 11/10/2013 (13h30-16h45): AL] Measurement on computer systems (benchmarking, observation, tracing, monitoring, profiling).
3. Monday 14/10/2013 (13h30-16h45): AL] Expectation and variance estimation, confidence interval, distribution comparison. Illustration with R.
4. Monday 21/10/2013 (13h30-16h45): AL] Exploring data, checking hypothesis.
5. Monday 04/11/2013 (13h30-15h45): JMV] Design of experiments
6. Monday 18/11/2013 (13h30-15h45): JMV] Design of experiments (2)
7. Monday 25/11/2013 (13h30-15h45): JMV] Design of experiments (3)

Evaluation

Experimental project

Organisation

Team

Arnaud Legrand
Simulation for large scale systems analysis and control

Jean-Marc Vincent
Markovian modeling of systems, simulation and dimensioning

References : text books

- **The Art of Computer Systems Performance Analysis : Techniques for Experimental Design, Measurement, Simulation and Modeling.** Raj Jain *Wiley 1991 (nouvelles versions)*

  Covers the content of the course, a complete book
- **Performance Evaluation** Jean-Yves Le Boudec EPFL electronic book http://ica1www.epfl.ch/perfeval/lectureNotes.htm

  Covers the statistical part of the course
- **Measuring Computer Performance: A Practitioner's Guide** David J. Lilja *Cambridge University press* 2000

  Covers the practical part of measurement and benchmarking
- **Discrete-Event System Simulation** Jerry Banks, John Carson, Barry L. Nelson, David Nicol, *Prentice Hall*, 2004

  Covers the part on simulation

References : journals and conferences

- **General**: JACM, ACM Comp. Surv., JOR, IEEE TSE,...
- **Specialized**: Performance Evaluation, Operation research, MOR, ACM TOMACS, Queueing Systems, DEDS, ...
- **Application**: IEEE TPDS, TC, TN, TAC, Networks,...
- **Theoretical**: Annals of Probability, of Appl. Prob, JAP, Adv. Appl. Prob,...
- **Conferences on performances**: Performance, ACM-SIGMETRICS, TOOLS, MASCOT, INFORMS, ...
- **Conferences on an application domain**: ITC, Europar, IPDPS, Renpar, ...
- **National seminars**: Atelier d'évaluation de performances,...

Outline

Networking

Flow performance

- latency, waiting time, response time
- loss probability
- jitter

Operator performance

- bandwidth utilisation
- achievable throughput
- loss rate

Quality of service

contract between user and provider
service guarantees
tradeoff between utilization and QoS

Networking

Flow performance

- latency, waiting time, response time
- loss probability
- jitter

Operator performance

- bandwidth utilisation
- achievable throughput
- loss rate

Quality of service

contract between user and provider
service guarantees
tradeoff between utilization and QoS

Networking

Flow performance

- latency, waiting time, response time
- loss probability
- jitter

Operator performance

- bandwidth utilisation
- achievable throughput
- loss rate

Quality of service

contract between user and provider
service guarantees
tradeoff between utilization and QoS

Parallel processing

Program execution

- makespan, critical path
- speedup, efficiency
- active waiting, communication overlapping
- throughput

System utilization

- cpu utilization, idle time
- memory occupancy
- communication throughput

Parallel programming and scheduling

granularity of the application
tradeoff between utilization and makespan

Parallel processing

Program execution

- makespan, critical path
- speedup, efficiency
- active waiting, communication overlapping
- throughput

System utilization

- cpu utilization, idle time
- memory occupancy
- communication throughput

Parallel programming and scheduling

granularity of the application
tradeoff between utilization and makespan

Parallel processing

Program execution

- makespan, critical path
- speedup, efficiency
- active waiting, communication overlapping
- throughput

System utilization

- cpu utilization, idle time
- memory occupancy
- communication throughput

Parallel programming and scheduling

granularity of the application
tradeoff between utilization and makespan

Distributed applications

Application

- response time
- reactivity
- throughput (number of processed requests/unit time)
- streaming rate

System utilization

- service availability
- resource utilization
- communication throughput

System security

- reliability (error-free period)
- availability

Distributed applications

Application

- response time
- reactivity
- throughput (number of processed requests/unit time)
- streaming rate

System utilization

- service availability
- resource utilization
- communication throughput

System security

- reliability (error-free period)
- availability

Distributed applications

Application

- response time
- reactivity
- throughput (number of processed requests/unit time)
- streaming rate

System utilization

- service availability
- resource utilization
- communication throughput

System security

- reliability (error-free period)
- availability

Synthesis

User point of view

optimize its own performance
- get the maximum amount of resources for its own purpose
- guarantee the higher quality of service

Resource point of view

Contract between users and resources:
- guarantee of “equity”
- optimize the use of resources
- minimize costs by identifying performance bottlenecks

Tradeoff Performance - Cost

Synthesis

User point of view

optimize its own performance
- get the maximum amount of resources for its own purpose
- guarantee the higher quality of service

Resource point of view

Contract between users and resources:
- guarantee of “equity”
- optimize the use of resources
- minimize costs by identifying performance bottlenecks

Tradeoff Performance - Cost

Synthesis

User point of view

optimize its own performance
- get the maximum amount of resources for its own purpose
- guarantee the higher quality of service

Resource point of view

Contract between users and resources:
- guarantee of “equity”
- optimize the use of resources
- minimize costs by identifying performance bottlenecks

Tradeoff Performance - Cost

Why experiments ?

Design of architectures, softwares

- System debugging (!!)
- Validation of a proposition
- Qualification of a system
- Dimensioning and tuning
- Comparison of systems

Many purposes $\Rightarrow$ different methodologies

Modeling fundamentals

Scientific Method

Falsifiability is the logical possibility that an assertion can be shown false by an observation or a physical experiment. [Popper 1930]

Modeling comes before experimenting

Modeling principles [J-Y LB]

- (Occam:) if two models explain some observations equally well, the simplest one is preferable
- (Dijkstra:) It is when you cannot remove a single piece that your design is complete.
- (Common Sense:) Use the adequate level of sophistication.

Modeling fundamentals

Scientific Method

Falsifiability is the logical possibility that an assertion can be shown false by an observation or a physical experiment. [Popper 1930]

Modeling comes before experimenting

Modeling principles [J-Y LB]

- (Occam:) if two models explain some observations equally well, the simplest one is preferable
- (Dijkstra:) It is when you cannot remove a single piece that your design is complete.
- (Common Sense:) Use the adequate level of sophistication.

Modeling fundamentals

Scientific Method

Falsifiability is the logical possibility that an assertion can be shown false by an observation or a physical experiment. [Popper 1930]

Modeling comes before experimenting

Modeling principles [J-Y LB]

- (Occam:) if two models explain some observations equally well, the simplest one is preferable
- (Dijkstra:) It is when you cannot remove a single piece that your design is complete.
- (Common Sense:) Use the adequate level of sophistication.

Design of models (introduction)

Formulation of the question

Give explicitly the question (specify the context of experimentation)

- Identify parameters (controlled and uncontrolled)
- Identify factors (set levels)
- Specify the response of the experiment

Minimize the number of experiments for a maximum of accuracy

Outline

How to report results from model analysis

Problem : provide "nice" pictures to help the understanding

- **Increases deeply the quality of a paper**
- Show the scientific quality of your research
- Observation leads to open problems
- Pictures generates discussions

Mistakes

- **semantic of graphical objects**
- conventions for graphics reading
- first step in scientific validation

Guidelines for good graphics (Jain)

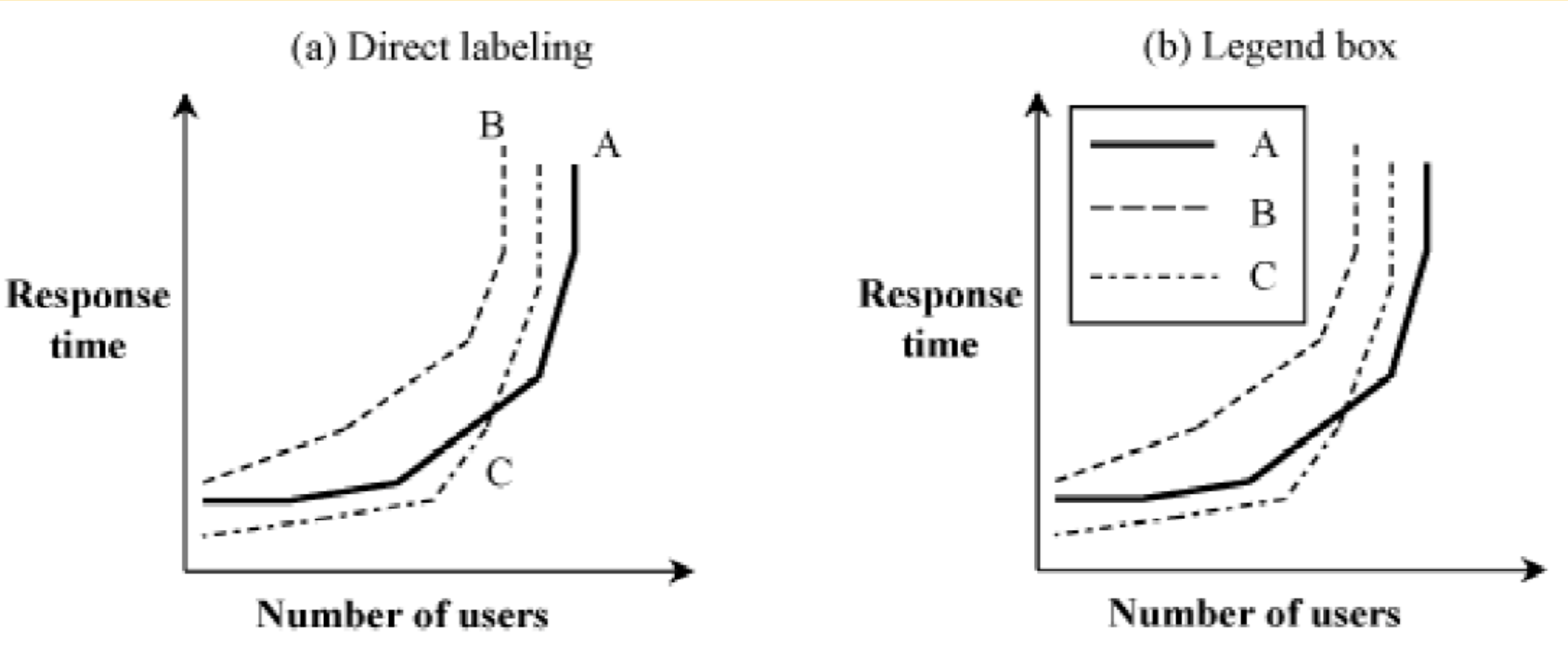
Guidelines for Preparing Good Graphic Charts

Specify the amount of information given by the chart

1. Require Minimum Effort from the Reader
2. Maximize Information
3. Minimize Ink
4. Use Commonly Accepted Practices
5. Make several trials before arriving at the final chart. Different combinations should be tried and the best one selected.

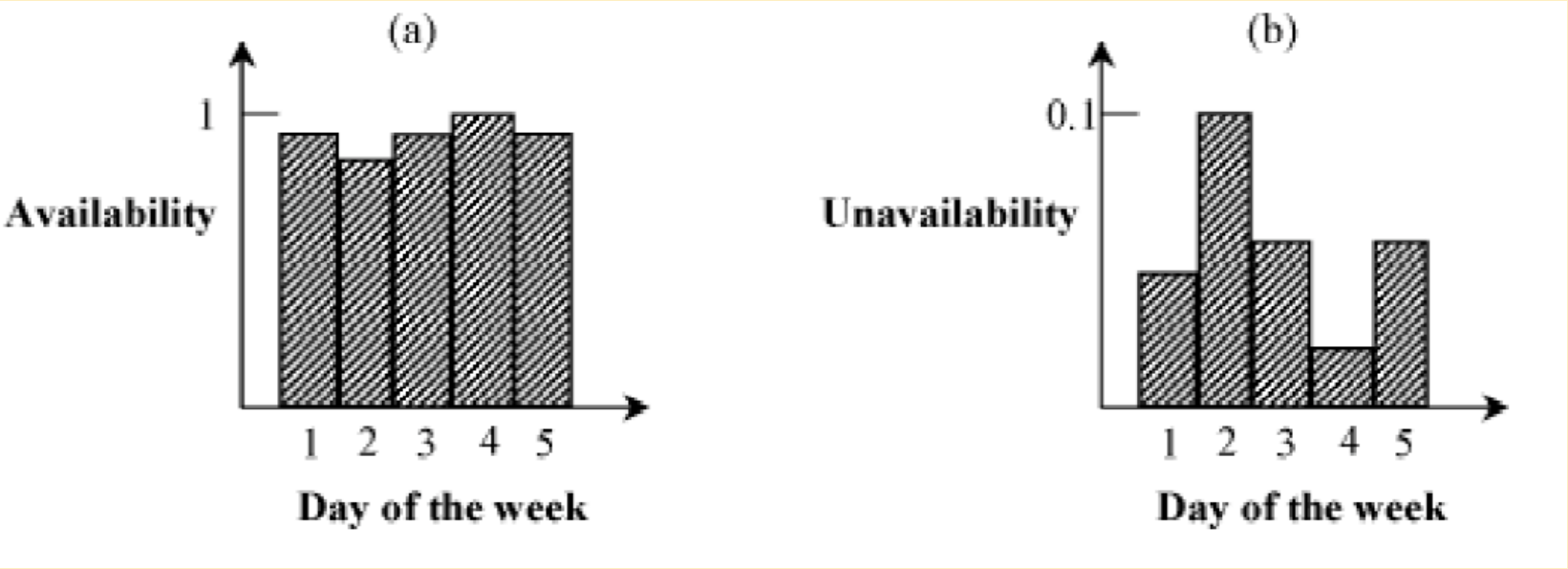
Guidelines for good graphics (Jain)

Minimum effort for the reader



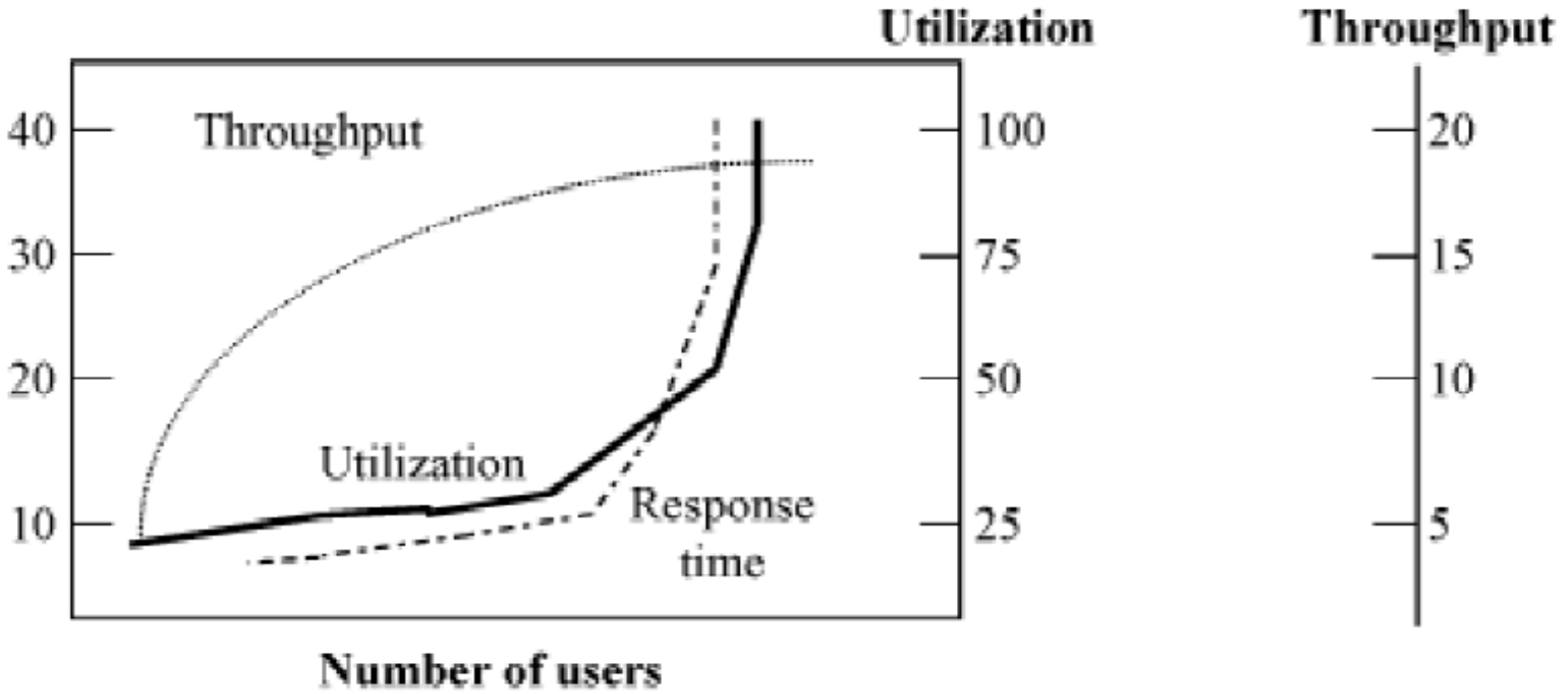
Guidelines for good graphics (Jain)

Minimize Ink



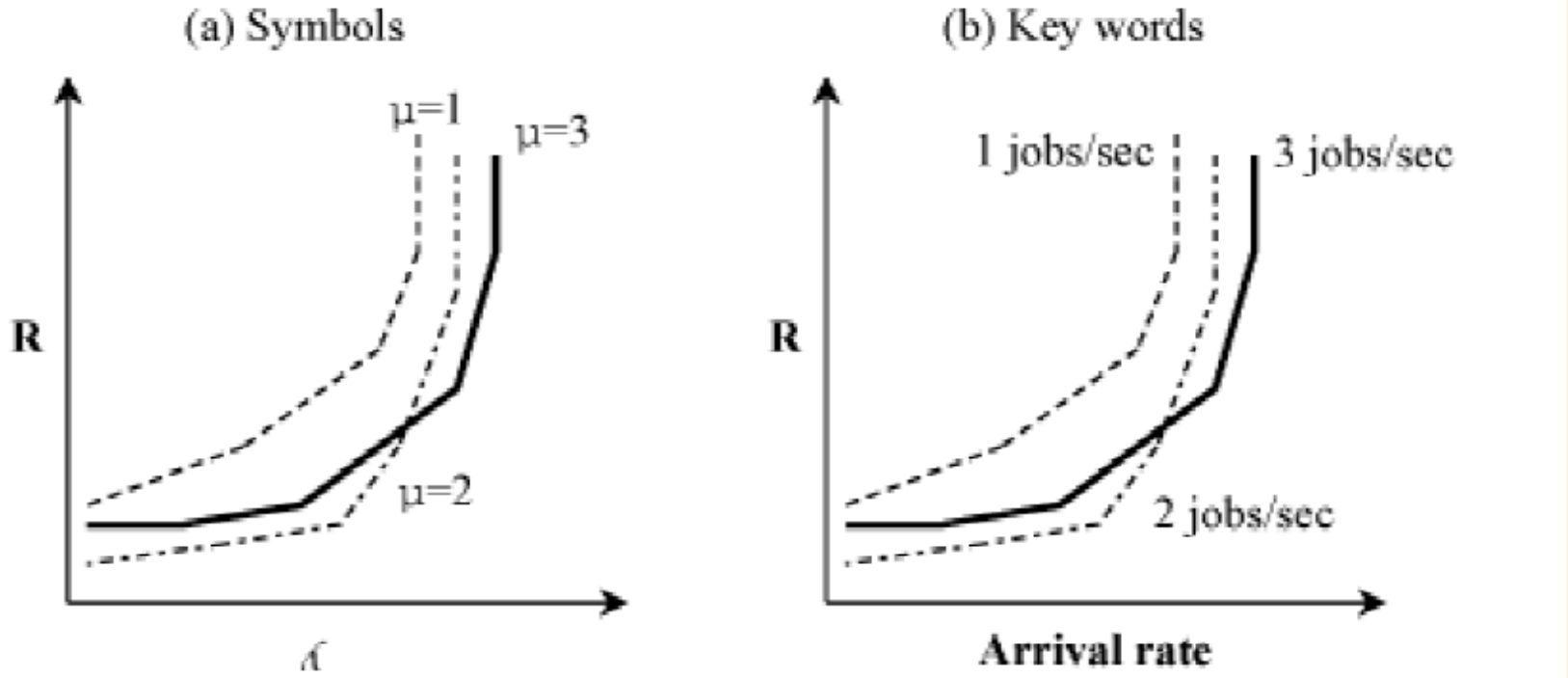
Common mistakes

Multiple scaling, Too much information



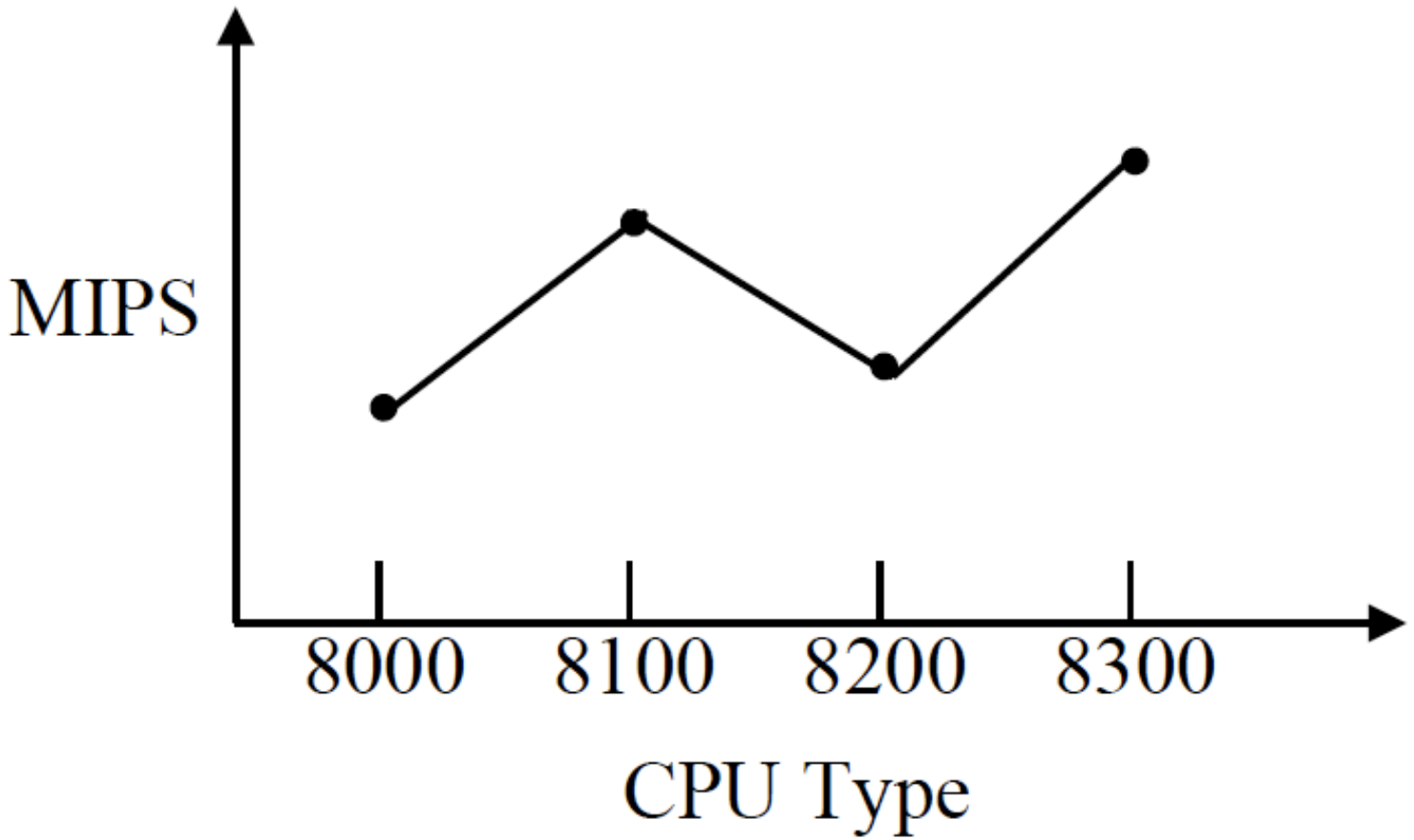
Common mistakes

Cryptic information



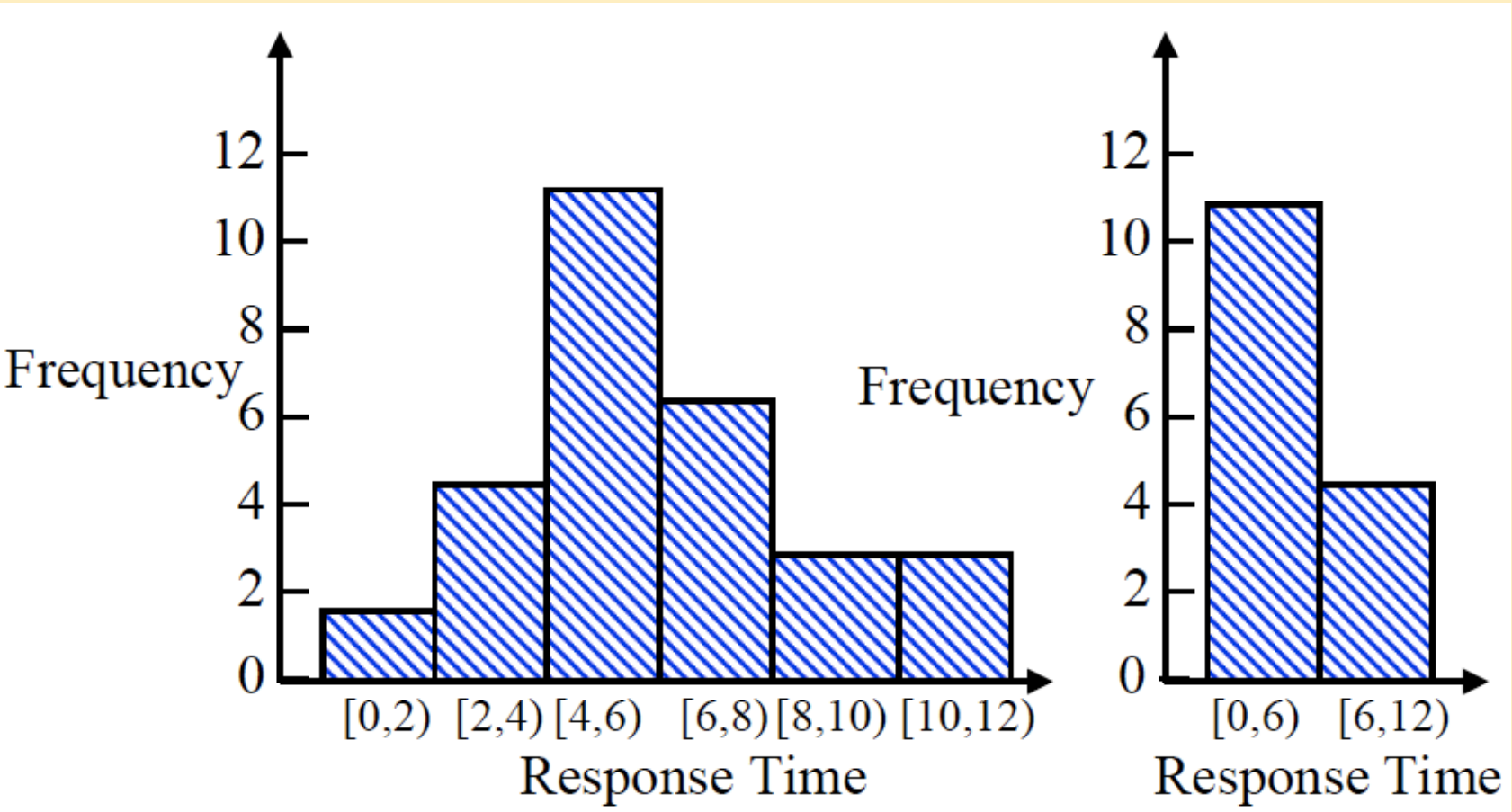
Common mistakes

Non-relevant graphic objects



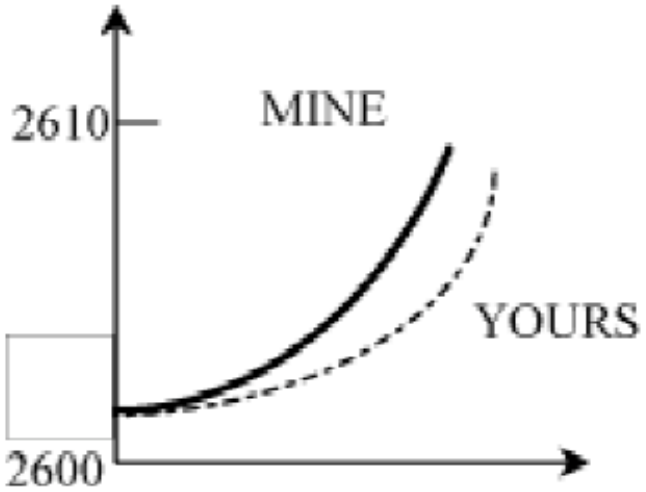
Common mistakes

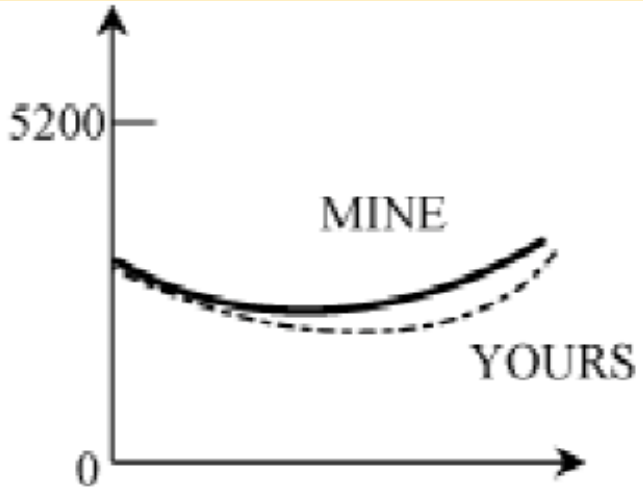
Non-relevant graphic objects

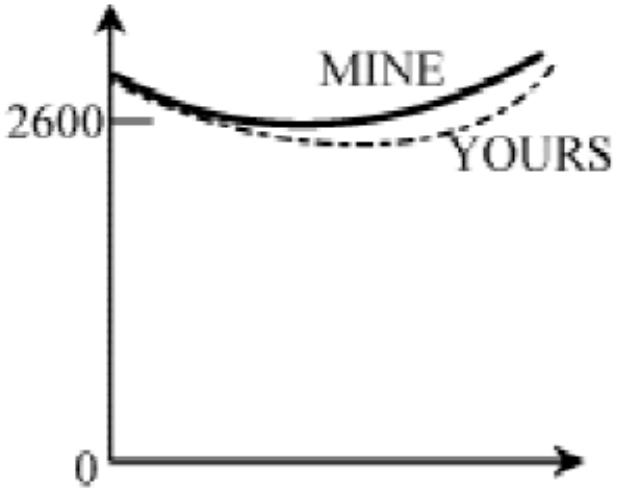


Common mistakes

Howto cheat ?

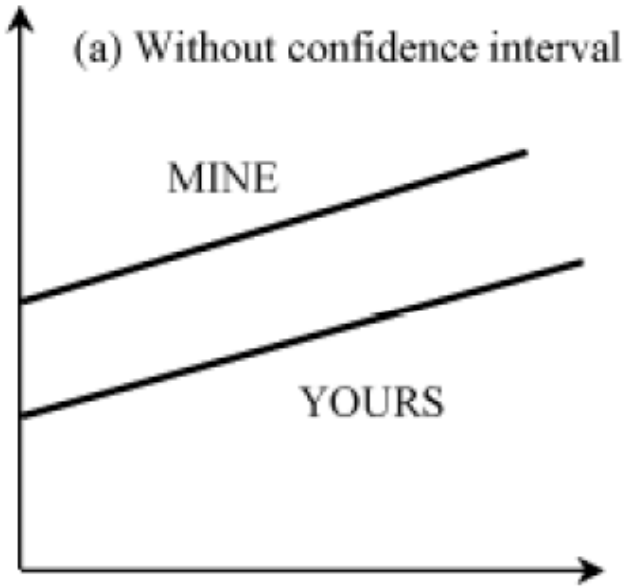


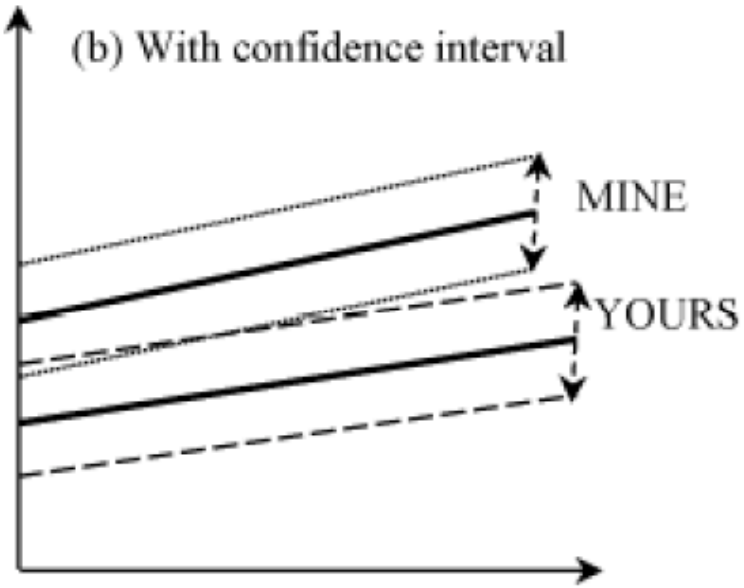




Common mistakes

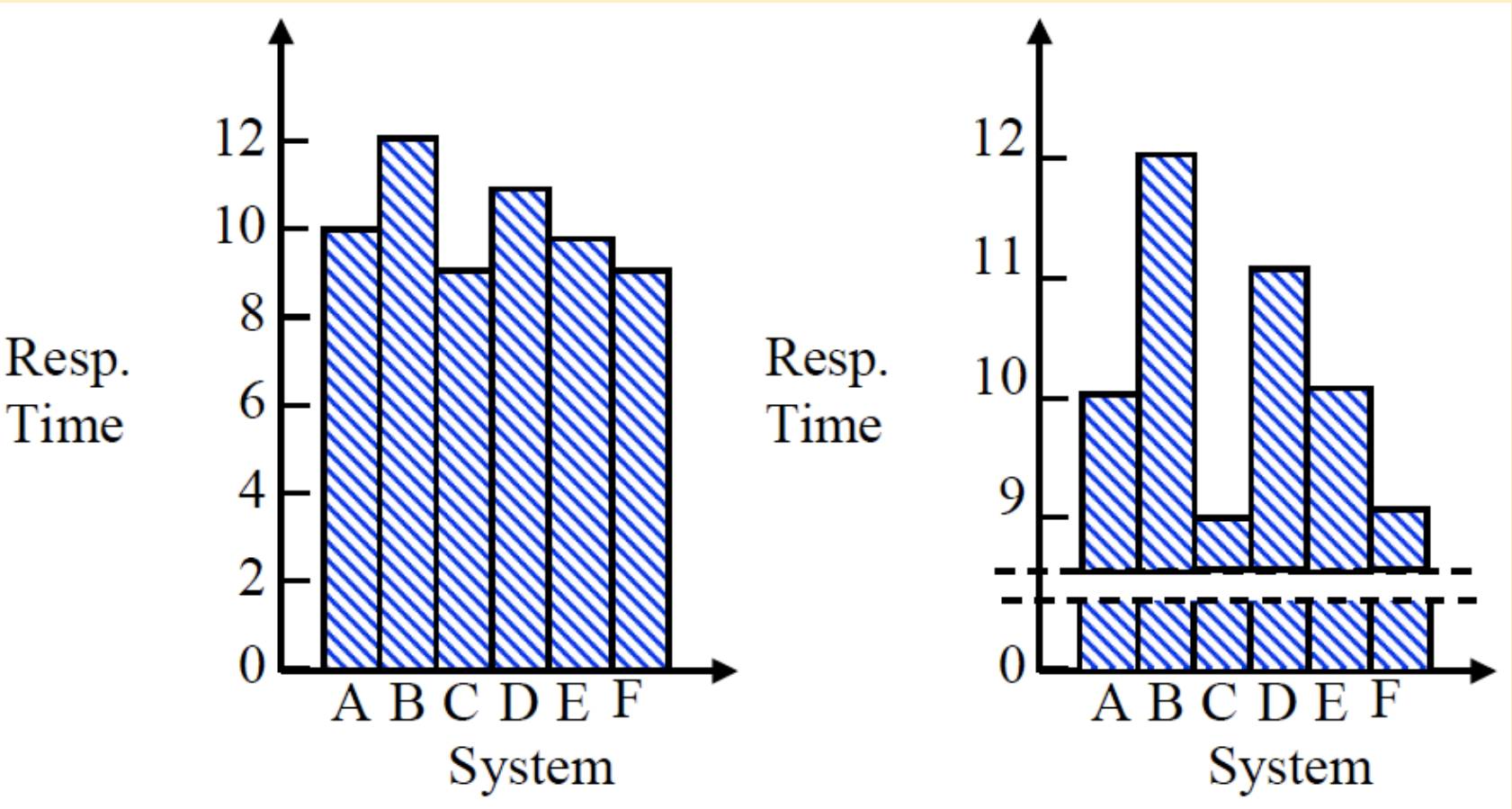
Howto cheat ?





Common mistakes

Howto cheat ?



Checklist for good graphics (Jain)

1. Are both coordinate axes shown and labeled?
2. Are the axes labels self-explanatory and concise?
3. Are the scales and divisions shown on both axes?
4. Are the minimum and maximum of the ranges shown on the axes appropriate to present the maximum information.
5. Is the number of curves reasonably small? A line chart should have no more than six curves.
6. Do all graphs use the same scale? Multiple scales on the same chart are confusing. If two charts are being compared, use the same scale if possible.
7. Is there no curve that can be removed without reducing the information?
8. Are the curves on a line chart individually labeled?
9. Are the cells in a bar chart individually labeled?
10. Are all symbols on a graph accompanied by appropriate textual explanations?
11. If the curves cross, are the line patterns different to avoid confusion?

Checklist for good graphics (Jain)

12. Are the units of measurement indicated?
13. Is the horizontal scale increasing from left to right?
14. Is the vertical scale increasing from bottom to top?
15. Are the grid lines aiding in reading the curve?
16. Does this whole chart add to the information available to the reader?
17. Are the scales contiguous? Breaks in the scale should be avoided or clearly shown.
18. Is the order of bars in a bar chart systematic? Alphabetic, temporal, best-to-worst ordering is to be preferred over random placement.
19. If the vertical axis represents a random quantity, are confidence intervals shown?
20. For bar charts with unequal class interval, is the area and width representative of the frequency and interval?
21. Do the variables plotted on this chart give more information than other alternatives?

Checklist for good graphics (Jain)

22. Are there no curves, symbols, or texts on the graph that can be removed without affecting the information?
23. Is there a title for the whole chart?
24. Is the chart title self-explanatory and concise?
25. Does that chart clearly bring out the intended message?
26. Is the figure referenced and discussed in the text of the report?