A First View about
Performance Evaluation and Analysis
« Performance Evaluation as Criteria to Best Computer Systems »

Carlos Jaime BARRIOS HERNÁNDEZ, PhD.
@carlosjaimebh


Pair testing gone wrong
ISTQB BASIC LEVEL
"Not so fast Michael! It says here that you should write the test case before you can execute it."
AG
Andy Glover cartoontester.blogspot.com © 2012

PERFORMANCE EVALUATION

- Often in Computer Science you need to:
  - **Demonstrate** that a new concept, technique, or algorithm is feasible
  - **Show** that a new method is better than an existing method
  - **Understand** the impact of various factors and parameters on the performance, scalability, or robustness of a system

Thinking in Computers…

It is posible to observe a “computer” from:

- Classes of Computer/System
- Program/Application Performance
- Programmer Point of View

Classes of Computers

- Desktop Computers
  - Personal Use, Good Performance to an Single User
  - Third Party Software, Low Cost
- Servers
  - Access Only Via Network
  - Carry Large Workload
  - Single Complex Applications // Many Small Jobs
  - Dependability

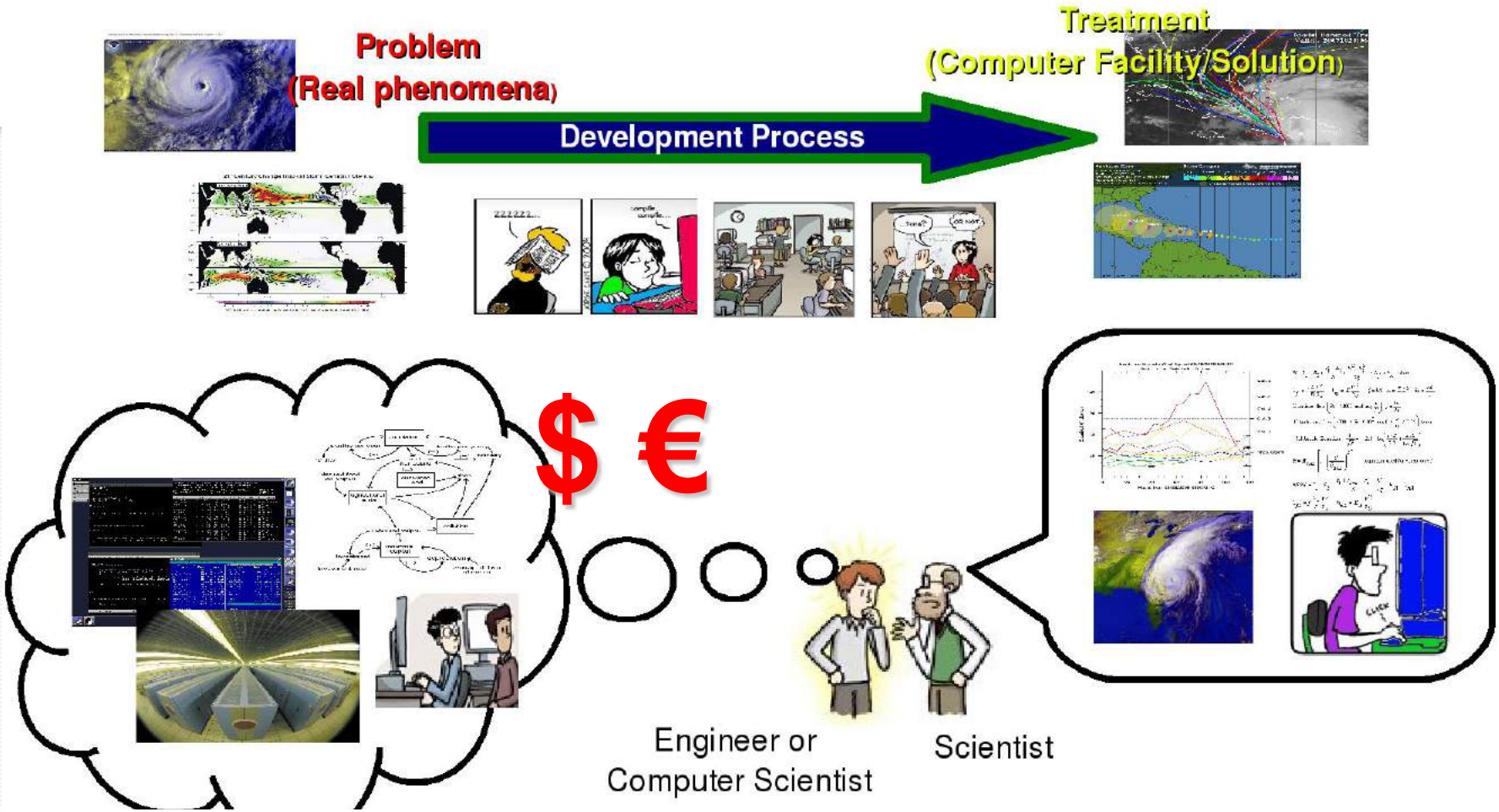
Classes of… (II)

- Supercomputers
  - Large Processing / Memory / Storage
  - High End-Scientific Calculations
  - Peak of Computing Capability
- Data Centers
  - Large Processing / Memory / Storage with Emphasis in Storage
- Embedded Computers
  - Designed to run one application or one set of related application
  - Single System
  - Widest Range of Applications
  - Large Scale Development
  - Minimum Performance
  - Low Fault Tolerance
  - Simplicity
  - Dependability
  - Redundancy

Understanding Performance
Program Point of View

- Depends on a combination of the efectiveness of the algorithms used in the program, the software systems used to create and traslate the program into machine instructions, and the efectiveness of the computer hardware elements.

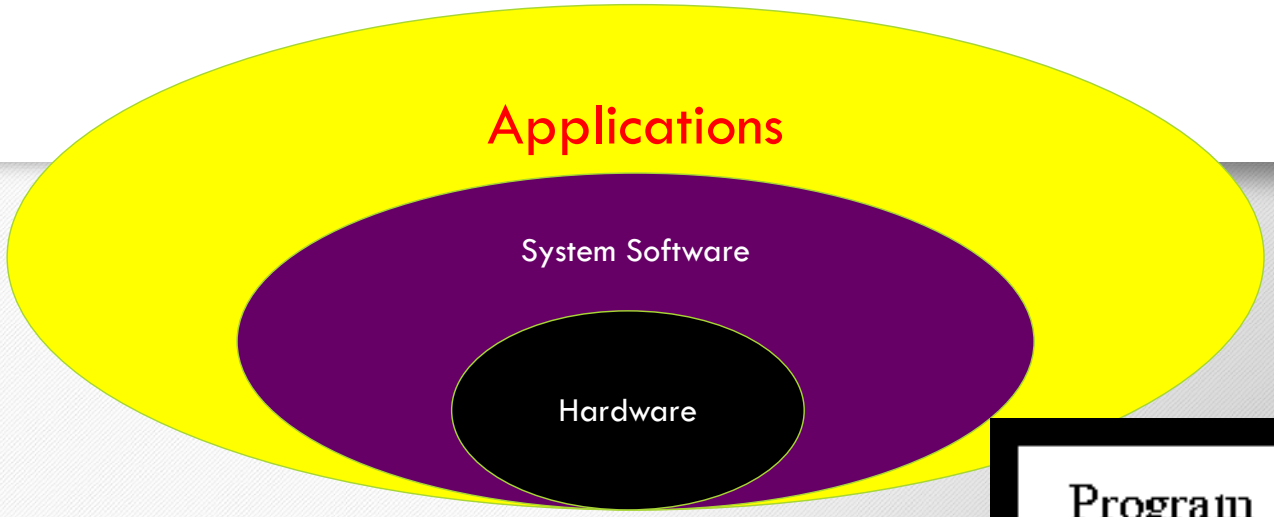
Conception, Development and use of Computer Systems

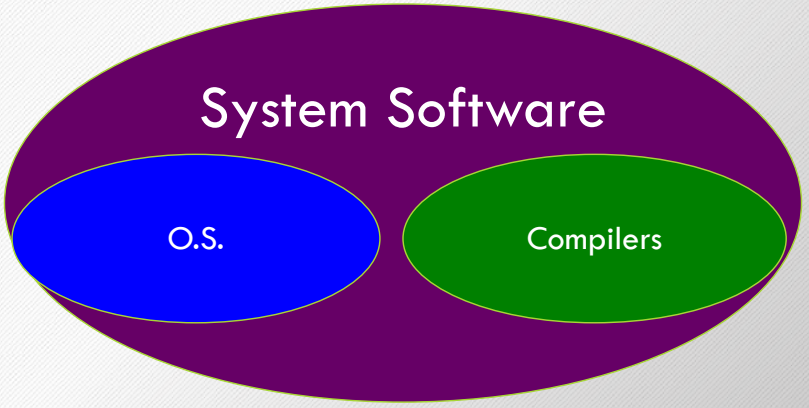


SW and HW Performance

- Algorithm
  - # of Source-Level statements and # of I/O operations executed
- Programming Language, Compiler and Architecture
  - # of Computer Instructions foe each source-level statements
- I/O System (HW and O.S)
  - How fast I/O operations may be executed.

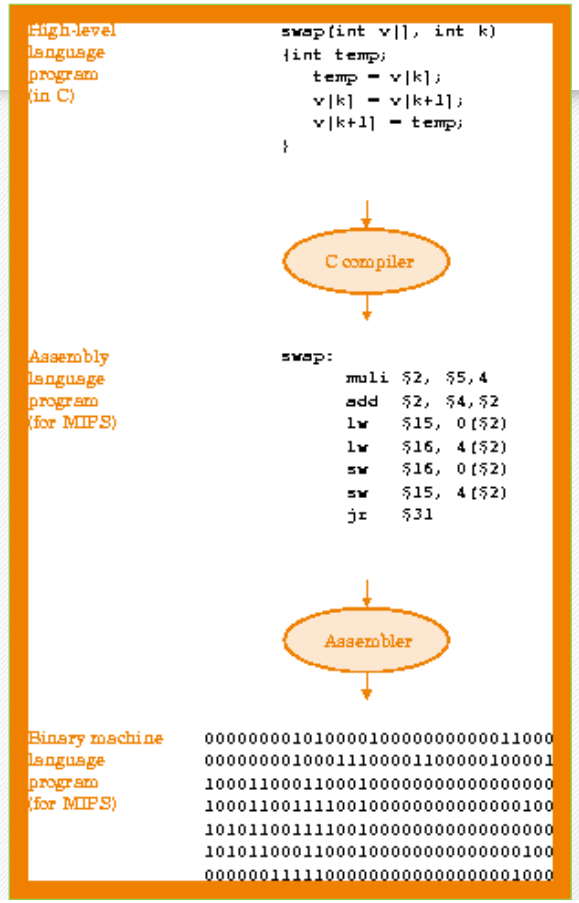
Programmer Point of View





Program	Produced by
Source	Programmer
Assembly	Compiler
Object	Assembler
Executable	Linker
Process	Loader

Some Concepts

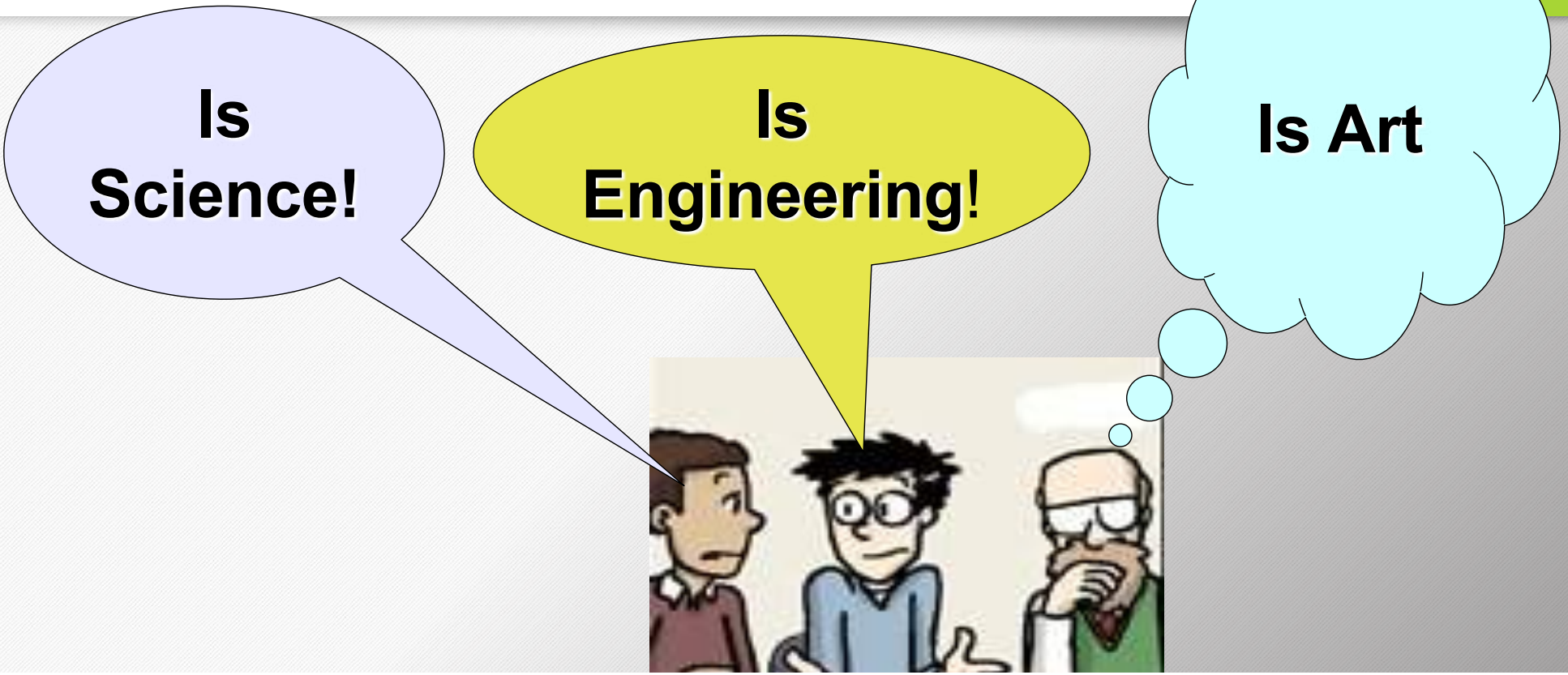


- **Letter:** Binary Digit : bit
- **Instruction:** Command that Computer hardware understand and obeys (Collection of bits)
- **Assembler:** A program that TRANSLATE symbolic version of instructions INTO the binary version.
- **Assembly Language:** A symbolic representation of machine instructions
- **Machine Language:** A Binary representation of machine instructions.

So?

- In all steps of a development process, performance evaluation is mandatory.
  - Monitoring
  - Benchmarking
  - Tests (Non-intrusive / Intrusive)
- Performance Evaluation in any Enginnering Systems project allows to take factible and good decisions

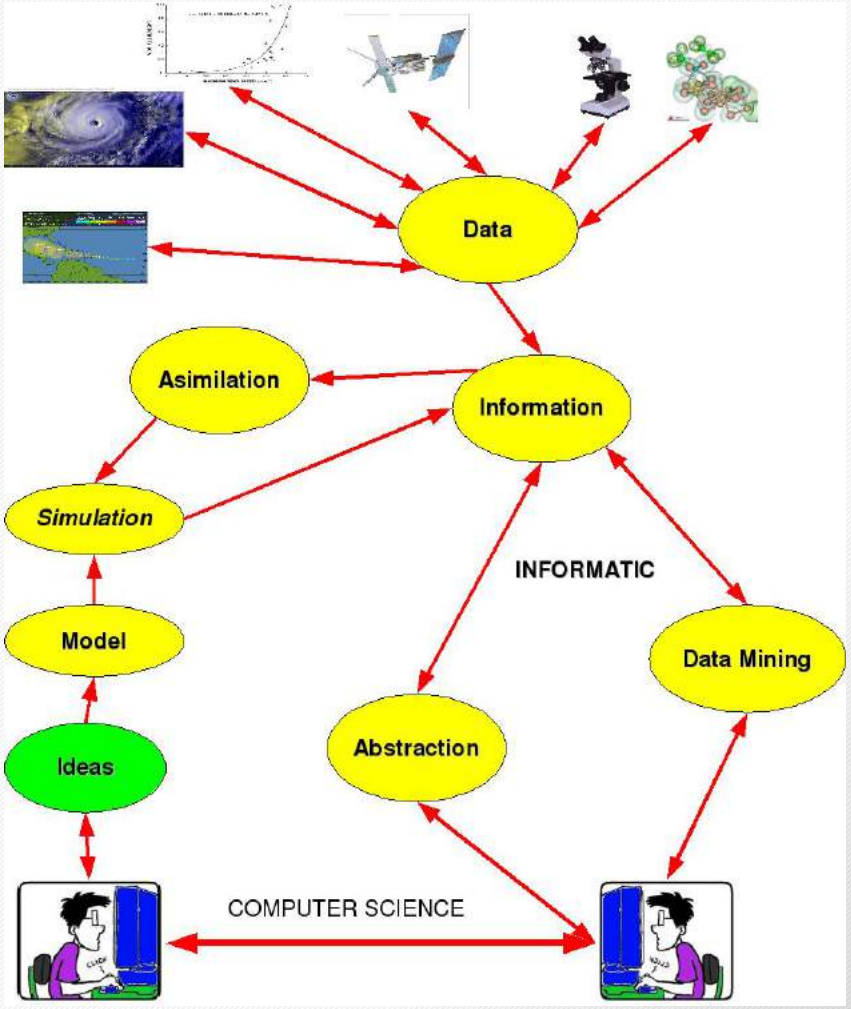
Remember The Computer Science Problem



So.. In performance evaluation context?

- Science is formal and based in scientific method (Mathematical description of phenomena), then computer science/informatics is science.
  - Observations and experimentations in controlled or non-controlled environments
- Engineering is based on science (Applied Science, Mathematic tools to implement models), then systems engineering is an applied science.
  - Tests and benchkmarking
- Informatics requires creativity, passion, inspiration and intuition…
  - Contemplation, esthetical views?

The paradigm...



- What process are part of a scientific approach?
- What process are a technological or engineering approach?
- What process are made with intuition, experience or inspiration?

The Focus Problem

- Theoretical:
  - I know everything... but nothing works.
- Practical:
  - Everything works... but I don't know why.
- Theoretical-Practical (Hybrid):
  - Nothing works... and nobody knows why.

In any case, we need predict the behavior of our system

The Importance of Performance in Computer Systems

- Mission-critical applications
- Life support applications
- Homeland security
- Battlefield situations
- Personal communication systems

Experimentation Mess

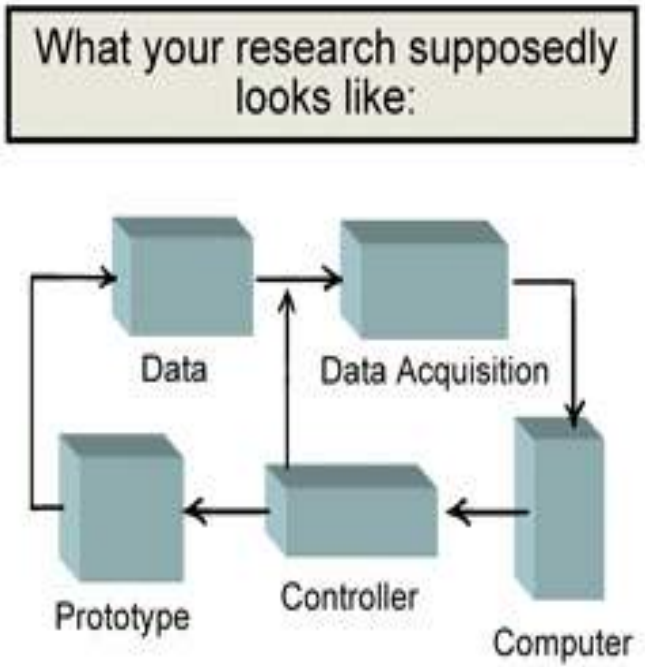


Figure 1. Experimental Diagram

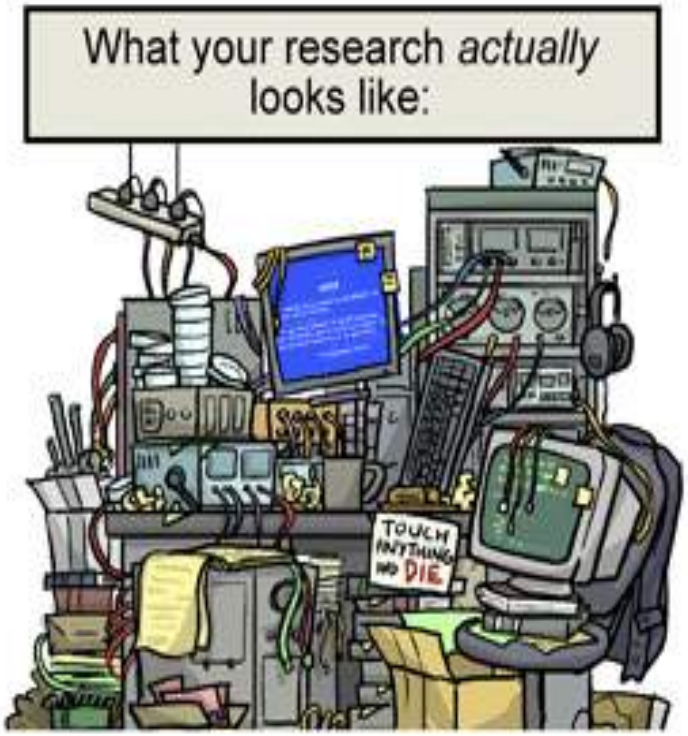
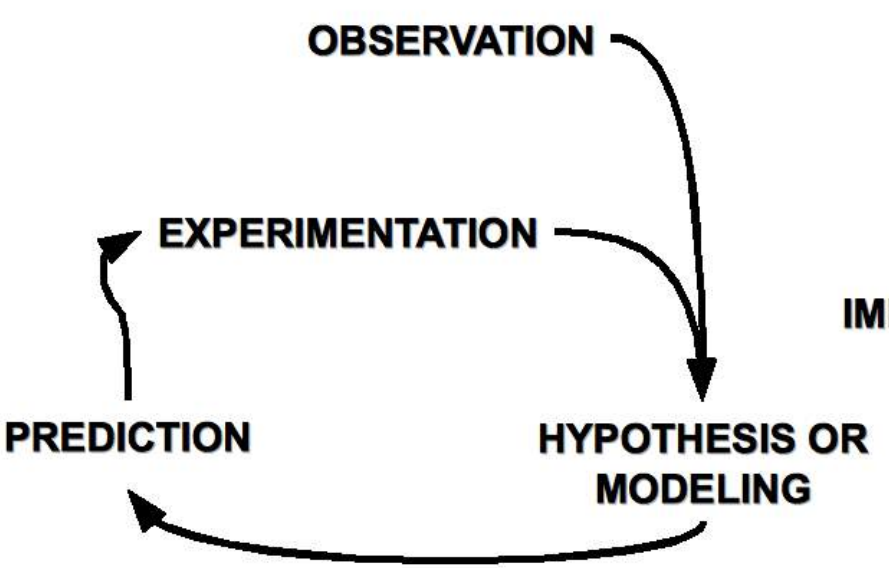


Figure 2. Experimental Mess

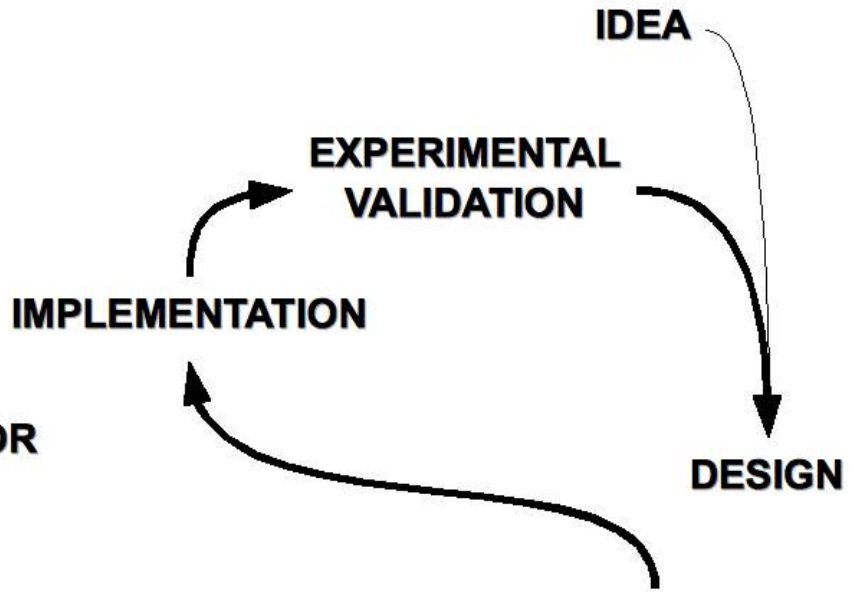
Bad Habits

- No emphasis on design
- Performance evaluation is relegated
  - Absecense of Test Plans
  - Incorrect Metrics to observate
  - Form and Esthetic over Funcional feautures
- No documentation in different levels
  - Developper
  - User
  - Administrator
- Systemic Think forgetfulness

Conception of Systems



Scientific Method



Conception and Design of Systems

- From Special section in Communications of the ACM 50(11), Nov 2007

(Point of view of Systems Conception)

- Theory
  - Abstraction
    - Models
    - Paradigms
    - Methods
    - Algorithms

- Practice
  - Implementation
    - Programs
    - Applications
    - Methodologies
    - Protocols

Science of the Computer Science

- Experimentation (tests) could be confirm or refute the accuracy (efficiency) of a software (system) design.
- Questions and theoric motivations with experiences (tests) produce « good » algorithms and programs.
- Development Cycle of software (systems) include: modeling (design), experimentation (tests – performance evaluation), build (programming)... (It's not a linear cycle).

Observing the Behavior of a System

- Observation
- Measures
  - Metrics
- Replication
- Validation
- Confrontation

Monitoring

Measures

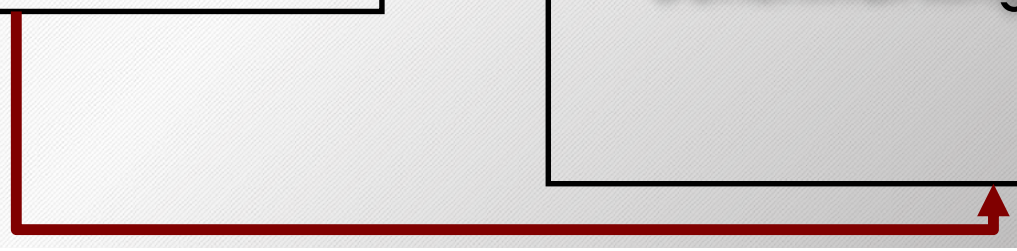
Metrics

Implantation in different environments

Validation

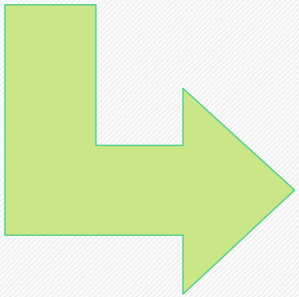
Comparison

Benchmarking



Experimental Computer Science

- Experimental Computer Science includes:
  - Observation
  - Confrontation of hypothesis
  - Reproduction of tests



Performance Evaluation

Performance Evaluation

- Application goal is to run with the maximum performance at least cost.
  - Thus, It's necessary Performance Evaluation
- Performance Evaluation is constant in all life cycle of the Application (or system)
  - Design
  - Building
  - Implementation
  - Implantation
  - Use
  - Actualization

QoS Metrics

- Response time
- Throughput
- Availability
- Reliability
- Security
- Scalability
- Extensibility

Performance Evaluation

(From the Computer Science Problem Heritage)

- Performance Evaluation is a technique:
  - Processes
  - Methodology
  - Tools
- Performance Evaluation is a science:
  - Theoric Basis
  - Experimentation
  - Replication and Validation
- Performance Evaluation is Art:
  - Intuition (Deep Knowledge)
  - Abstraction Capacity
  - Creativity
  - Activity non repetitive
  - Tools

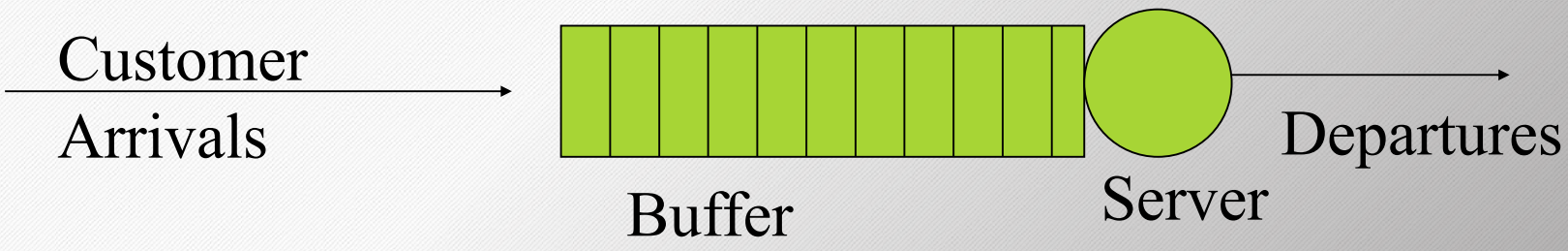
Performance Evaluation allows to know the capacities and limitations of a system.

Modeling, Measuring and Simulating

- Modeling: It allows to build formal abstractions
  - Mathematical Models
  - Analytical Models
  - Causal Models
- Measurement: It allows to characterize
  - Tests, Experiences in environments controlled known.
- Simulating: It allows to observe defined scenarios
  - In according with the modeling.

Analytical Example: Queueing Theory

- Queuing theory is a mathematical technique that specializes in the analysis of queues (e.g., customer arrivals at a bank, jobs arriving at CPU, I/O requests arriving at a disk subsystem, lineup at Tim Hortons)
- General diagram:



Queueing Theory (cont'd)

- The queueing system is characterized by:
  - Arrival process (M, G)
  - Service time process (M, D, G)
  - Number of servers (1 to infinity)
  - Number of buffers (infinite or finite)
- Example notation: M/M/1, M/D/1
- Example notation: M/M/∞, M/G/1/k

Queueing Theory (cont'd)

- There are well-known mathematical results for the mean waiting time and the number of customers in the system for several simple queueing models
- E.g., M/M/1, M/D/1, M/G/1
- Example: M/M/1
  - q = rho/ (1 - rho) where rho = lambda/mu < 1

Queueing Theory (cont'd)

- These simple models can be cascaded in series and in parallel to create arbitrarily large complicated queueing network models
- Two main types:
  - closed queueing network model (finite pop.)
  - open queueing network model (infinite pop.)
- Software packages exist for solving these types of models to determine steady-state performance (e.g., delay, throughput, util.)

Queueing Theory (cont'd)

- These simple models can be cascaded in series and in parallel to create arbitrarily large complicated queueing network models
- Two main types:
  - closed queueing network model (finite pop.)
  - open queueing network model (infinite pop.)
- Software packages exist for solving these types of models to determine steady-state performance (e.g., delay, throughput, util.)

Simulation Example: TCP Throughput

- Can use an existing simulation tool, or design and build your own custom simulator
- Example: ns-2 network simulator
- A discrete-event simulator with detailed TCP protocol models
- Configure network topology and workload
- Run simulation using pseudo-random numbers and produce statistical output

OTHER ISSUES

- Simulation run length
  - choosing a long enough run time to get statistically meaningful results (equilibrium)
- Simulation start-up effects and end effects
  - deciding how much to "chop off" at the start and end of simulations to get proper results
- Replications
  - ensure repeatability of results, and gain greater statistical confidence in the results given

Experimental Example: Benchmarking

- The design of a performance study requires great care in experimental design and methodology
- Need to identify
  - experimental factors to be tested
  - levels (settings) for these factors
  - performance metrics to be used
  - experimental design to be used

FACTORS

- Factors are the main "components" that are varied in an experiment, in order to understand their impact on performance
- Examples: request rate, request size, read/write ratio, num concurrent clients
- Need to choose factors properly, since the number of factors affects size of study

LEVELS

- Levels are the precise settings of the factors that are to be used in an experiment
- Examples: req size S = 1 KB, 10 KB, 1 MB
- Example: num clients C = 10, 20, 30, 40, 50
- Need to choose levels realistically
- Need to cover useful portion of the design space

- MODELING (Analytical Model)
- SIMULATION
- EXPERIMENTATION (TESTS – MEASUREMENT)
  - Tests in controlled systems
  - Tests « On Live » (also controlled)
    - Benchmarking
    - Tracing and Profiling

Solution Techniques

	Technique		
Characteristic	**Analytical**	**Simulation**	**Measurement**
Flexibility	High	High	Low
Cost	Low	Medium	High
Believability	Low	Medium	High
Accuracy	Low	Medium	High

From Measuring Computer Performance: A Practitioner's Guide, David J. Lilja 2004

Performance Evaluation Steps

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

Throughput and Reponse Time

- Throughput: Bandwidth: Number of tasks completed per unit time.
- Reponse Time: Execution Time: The total time required for a computer to complete a task.
  - Disk Accesses
  - I/O Activities
  - Memory Accesses
  - Operating System Overhead
  - CPU execution time

Open Question

- Do the following changes to a computer system increase throughtput, decrease response time, or both?
    1. Replacing the processor in a computer with a faster version
    2. Adding additional processors to a system that uses multiple processors for separate tasks?

Open Answers

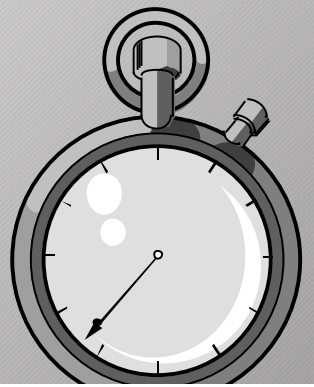
1. Response time and throughput are improved
   - Decreasing response time almost always improves throughput.
2. Thoughtput Increases
   - No ones task gets work done faster.

About the Metrics

- Performance metrics are
  - Count
    - Of how many times an event occurs
  - Duration
    - Of a time interval
  - Size
    - Of some parameter
  - Derivated values from these measurements

PERFORMANCE METRICS

- Performance metrics specify what you want to measure in your performance study
- Examples: response time, throughput, pkt loss
- Must choose your metrics properly and instrument your experiment accordingly



Time-normalized metrics

- « Rate » metrics
  - Normalize metric to common time basis
    - Transactions per second
    - Bytes per second
  - (Number of events) ÷ (time interval over which events occurred)
- « Throughput »
  - Average rate of successful message delivery over a communication channel
- Useful for comparing measurements over different time intervals

Good Metrics Characteristics

- Allows accurate and detailed comparisons
- Leads to correct conclusions
- Is well understood by everyone
- Has a quantitative basis
- A good metric helps avoid erroneous conclusions

Good metrics is

Linear

If metric increases 2x, performance should increase 2x

Reliable

If metric A > metric B

Then, Perf. A > Perf.B

Repeatable

Easy to use

Consistent

Units and definition are constant across systems

Independent

Indepentent to pressure on manufacturers to *optimize* for a particular metric

Performance Metrics Summary

	Clock	MIPS	MFLOPS	SPEC	QUIPS	TIME
Linear					≈☺	☺
Reliable						≈☺
Repeatable	☺	☺	☺	☺	☺	☺
Easy to measure	☺	☺	☺	½☺	☺	☺
Consistent	☺			☺	☺	☺
Independent	☺	☺			☺	☺

From Measuring Computer Performance: A Practitioner's Guide, David J. Lilja 2004

Other metrics

- Response time
  - Elapsed time from request to response
- Throughput
  - Jobs, operations completed per unit time
    - E.g. video frames per second
- Bandwidth
  - Bits per second
- *Ad hoc* metrics
  - Defined for a specific need
    - Requests per transaction

About the means...

- Performance in systems is multidimensional
  - CPU time
  - I/O time
  - Network time
  - Read/Write speedup
  - Disk Access
  - Storage Capacity
  - Interactions of various components
  - ...

About measurement tools and methodologies...

- Actually, measurement tools are based in ***events***:
  - Some predefined change to system state
- Event definition depends on metric being measured
  - Memory reference
  - Disk access
  - Change in a register's state
  - Network message
  - Processor interrupt

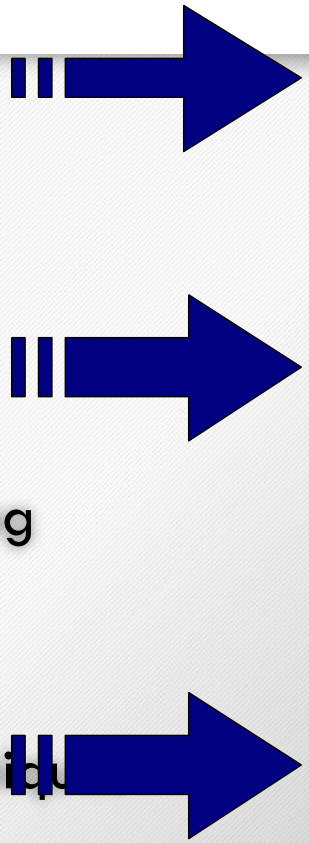
Some measurement techniques comparaison

	Event count	Tracing	Sampling
Resolution	Exact count	Detailed info	Statistical summary
Overhead	Low	High	Constant
Perturbation	~ #events	High	Fixed

From Measuring Computer Performance: A Practitioner's Guide, David J. Lilja 2004

What Evaluate?

- Infrastructure
  - Monitoring
  - Benchmarking
  - Emulating
- Applications
  - Monitoring
  - Benchmarking
  - Tracing and Profiling
- Users
  - Monitoring
  - Organization Techniqu



- Accuracy
  - In accord with your needs)
- Efficiency
  - In accord with the available resources
- Fault tolerance
- Security and Safety

Benchmark and Workload

- Benchmark:
  - Result of running a computer program, or a set of programs, in order to assess the relative performance of an object by running a number of standard tests and trials against it *(wikipedia)*
- Workload:
  - Quantified effort
    - Adition Instructions
    - Hybrid Instructions
    - Syntetics Programs
    - Kernels
    - Benchmark Applications

The Critical Behaviors to Evaluate

- Data Transfer
  - High Bandwidth Data Transfer implies heterogenity, dynamicity, concurrence and so on.
- File System Sensibility
  - I/O Sensibility
- Adaptation and Effectiveness
- Scalability
- Fault Tolerance
- Security
- Energy Consumption
- ... and the « Human invervention »
- Processing is critical but...

Response Time Breakdown

- Service time (does not depend on the load)
- Congestion (load-dependent)

Browser Time		Network Time			E-commerce Server Time		
Processing	I/O	Browser to ISP Time	Internet Time	ISP to Server Time	Processing	I/O	Networking

..... CONGESTION .....

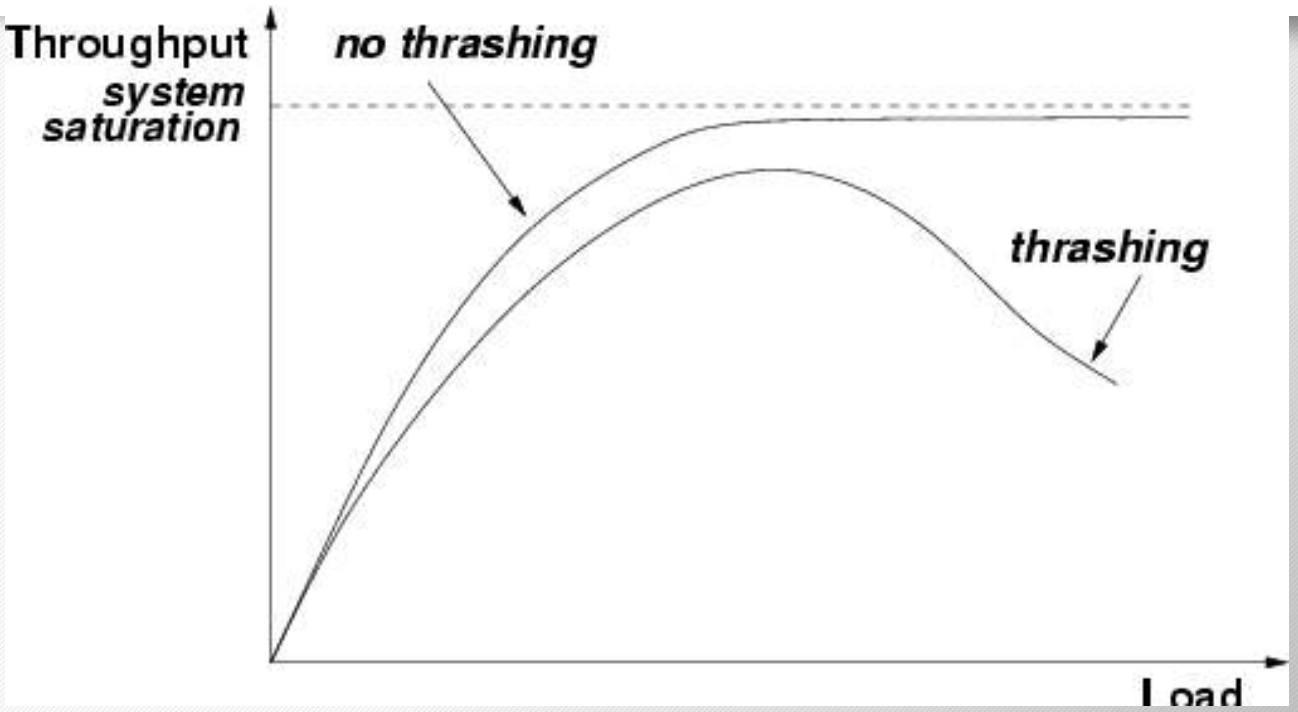
Throughput

- Measured in units of work completed over time. It’s a rate.
  - I/O’s/sec
  - Page downloads/sec
  - HTTP requests/sec
  - Jobs/sec
  - Transactions per second (tps)

Throughput Example

- An I/O operation at a disk of an OLTP system takes 10 msec on average.
  - What is the maximum throughput of the disk?
  - What is the throughput of the disk if it receives I/O requests at a rate of 80 requests/sec?

Throughput example

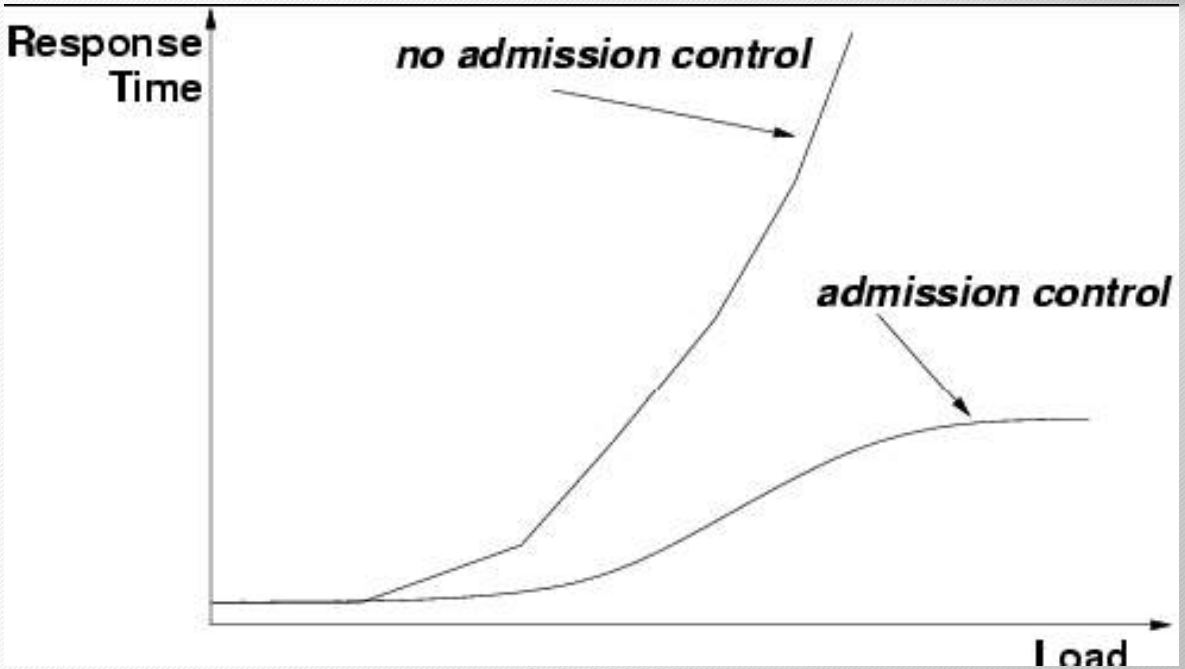


Availability

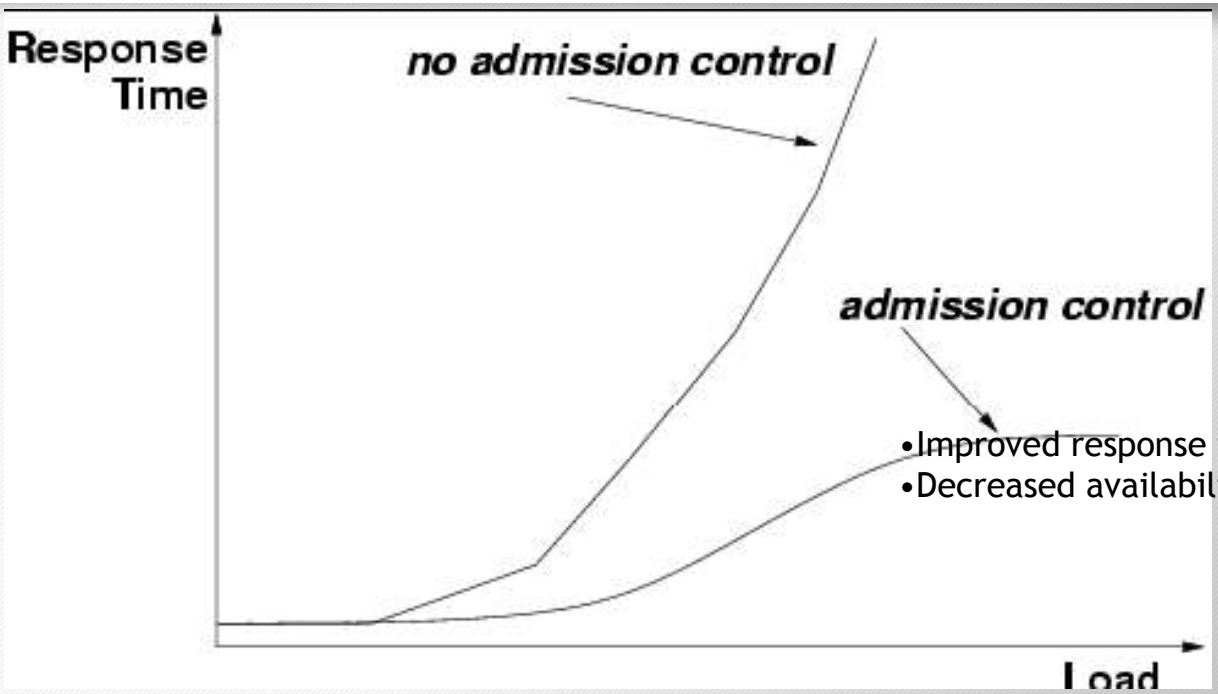
- Fraction of time a system is available (i.e.,operational).
  - Service interruptions can damage the reputation of a company, may endanger lives, and may cause financial disasters.
  - A system with 99.99% availability over 30 days is unavailable (1-0.9999) x 30 x 24 x 60 = 4.32 minutes.

Availability Problems

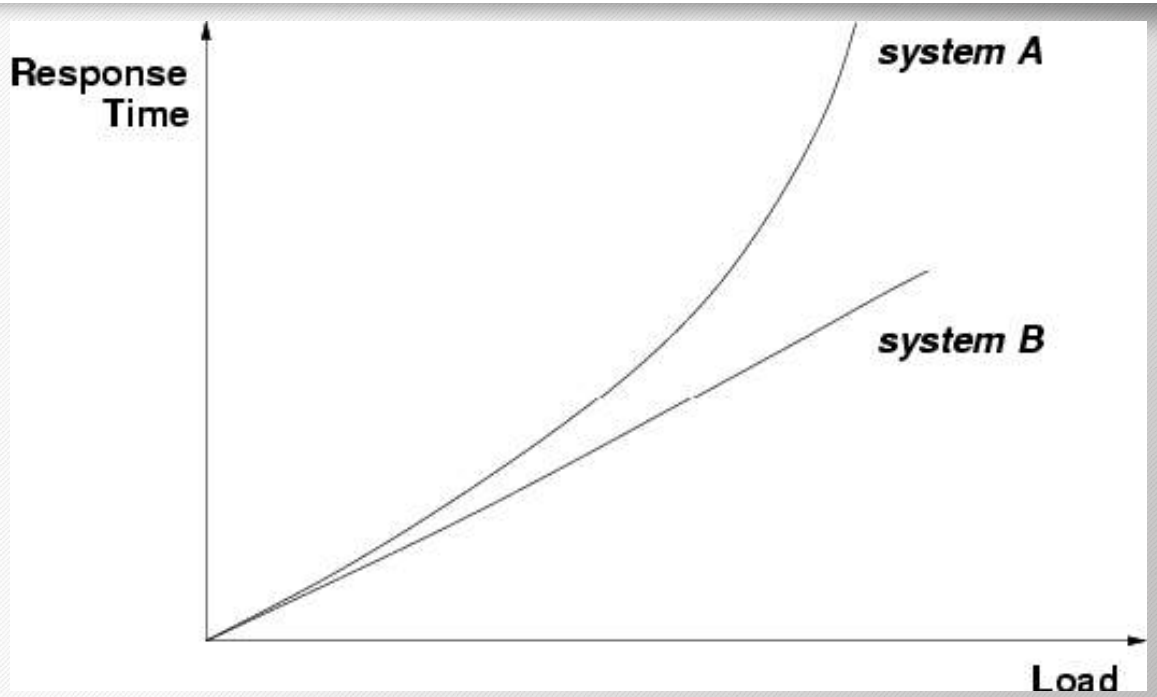
Admission Control



Admission Control



Scalability



System A is not scalable

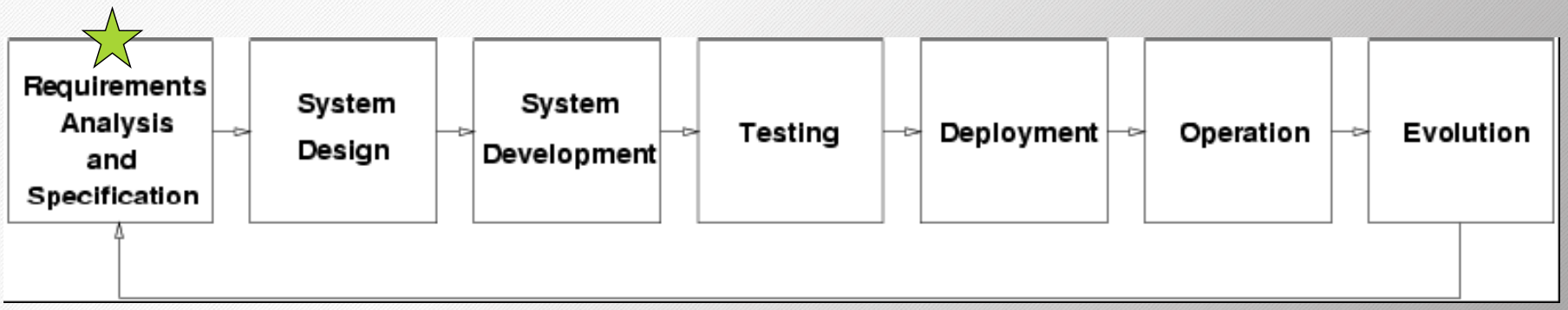
Extensibility

- Property of a system to constantly evolve to meet functional and performance requirements.
  - Autonomic computing, self-managing systems, self-healing systems.

Computer System Lifecycle

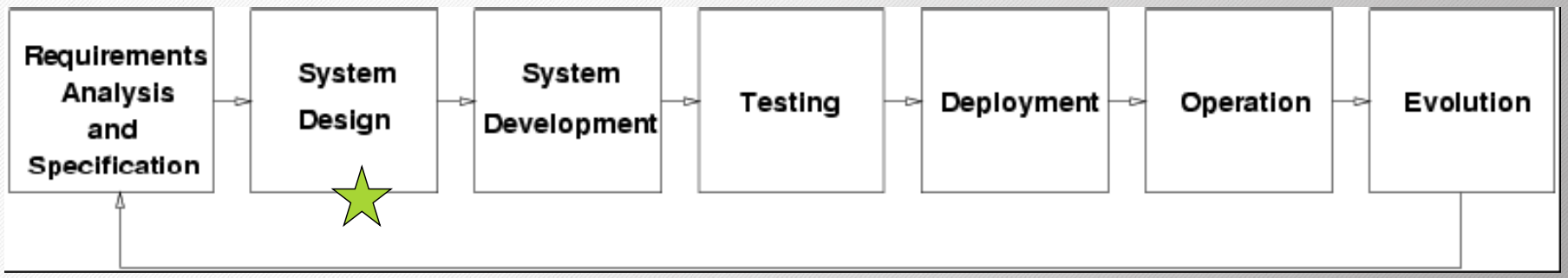
•Functional requirements: what the system has to do and on what type of platforms.

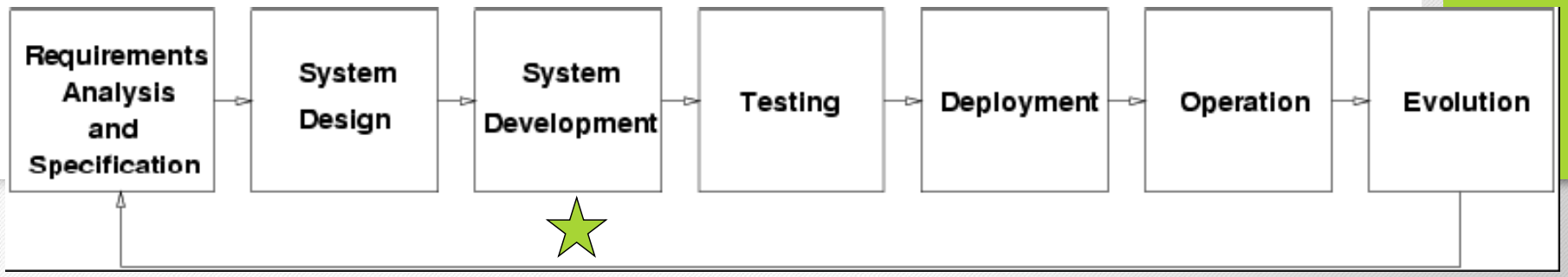
•Non-functional requirements: how well the system has to accomplish its functions. Service Level Agreements (SLA) are established.
In many cases, non-functional requirements have been neglected or considered only at system test time!



Computer System Lifecycle

How will the requirements be met?
- System architecture
- System broken down into components
- Major data structures, files, and databases are designed.
- Interfaces between components are specified





Components are implemented.
- some are new
- some are re-used
- some are adapted

Components are interconnected to form a system

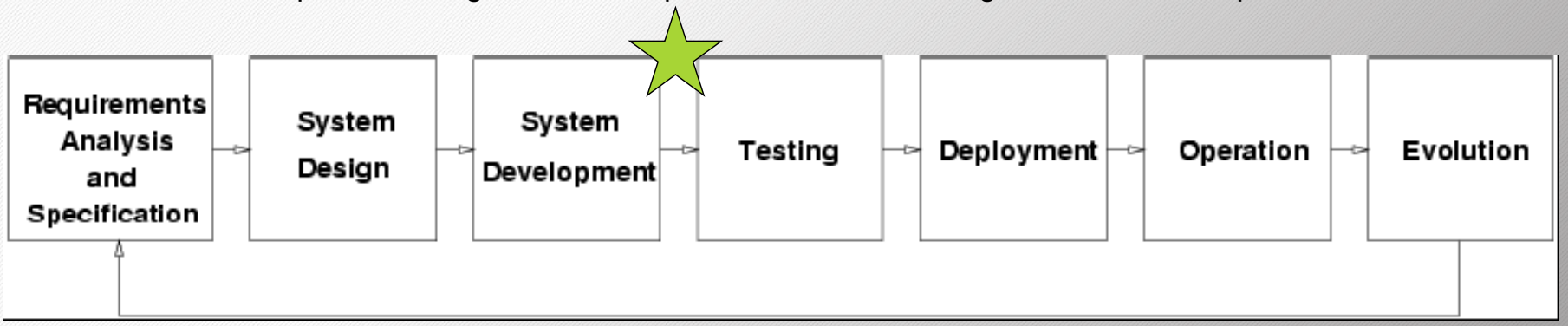
Components should be instrumented as they are built

Computer System Lifecycle

Concurrent with system development, as components become available (unit testing)

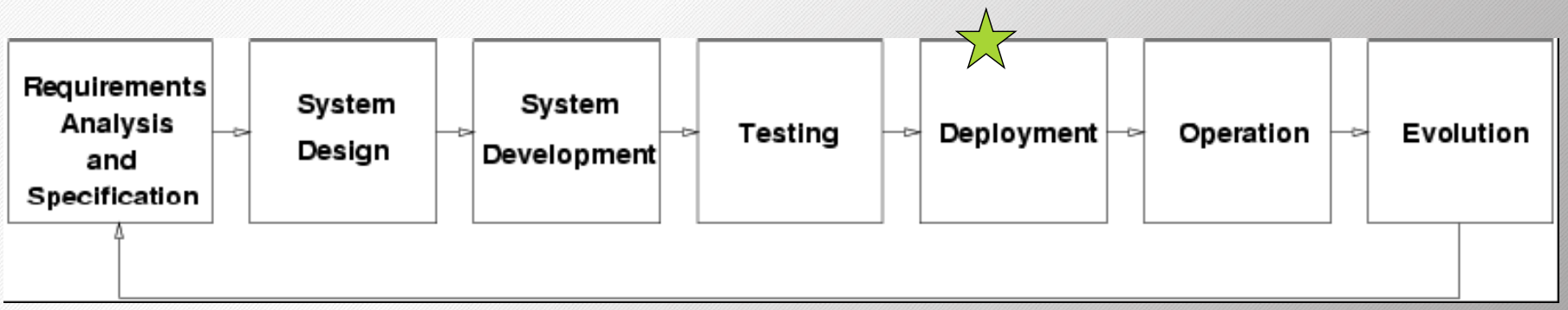
Integrated tests are carried out when the entire system is ready.

Often, more time is spent in testing functional requirements than in testing non-functional requirements.



Computer System Lifecycle

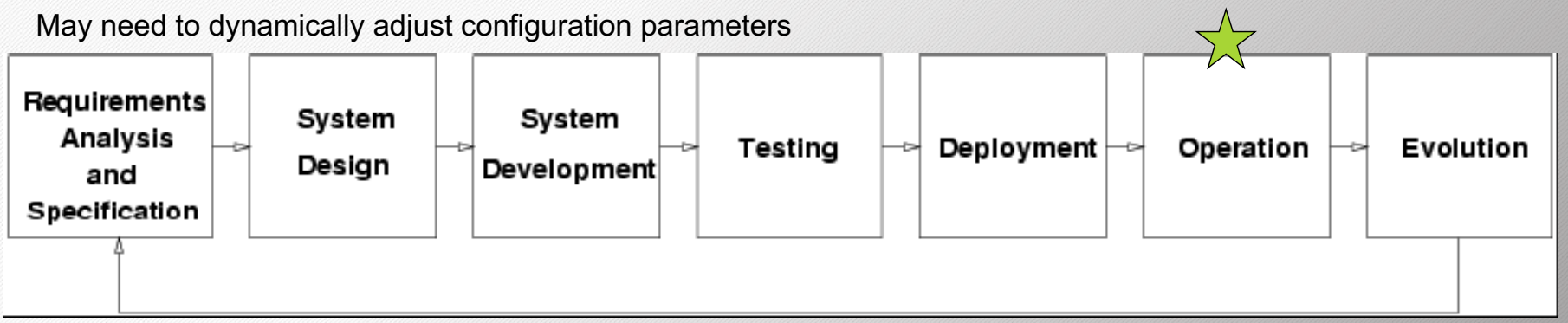
- Configuration parameters have to be set in order to meet the SLAs.
  - e.g., TCP parameters, database poolsize, maximum number of threads, etc.



Computer System Lifecycle

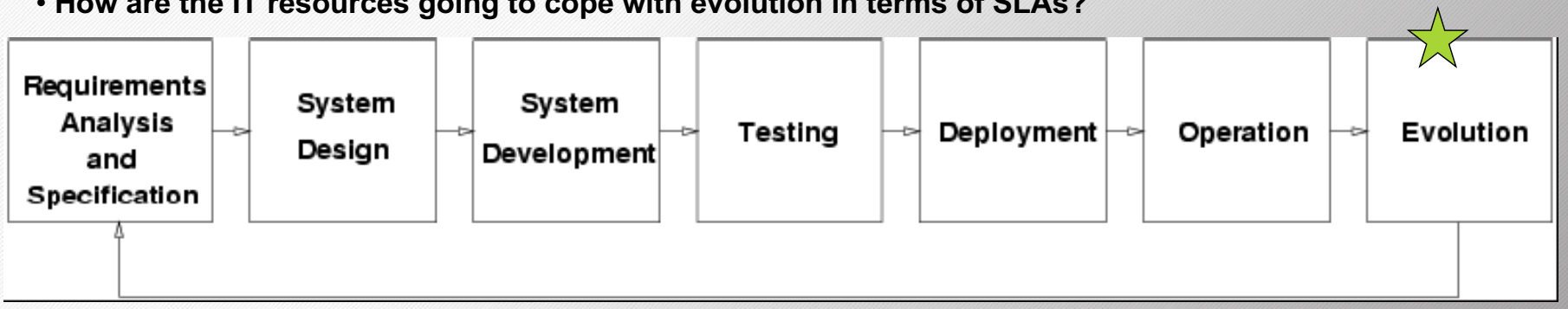
Constant monitoring to check if the system is meeting demands:
- workload (peak periods, unusual patterns)
- external metrics (user-perceived)
- internal metrics (help to detect bottlenecks and to fine tune the system)
- availability (external and internal)

May need to dynamically adjust configuration parameters



Computer System Lifecycle

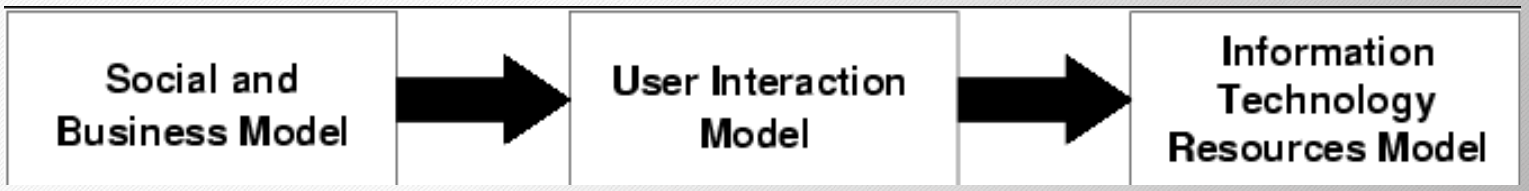
- **Systems may need to evolve to cope with new laws and Regulations (e.g., HIPPA)**
- **Systems may need to evolve to provide new functions (e.g., sale of downloadable MP3 music in addition to CDs)**
- **How are the IT resources going to cope with evolution in terms of SLAs?**



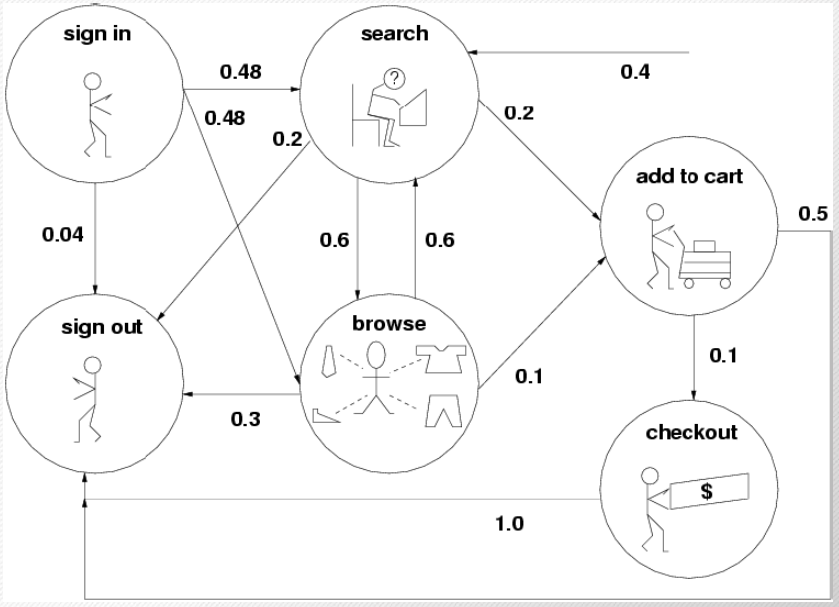
Reference Model for IT

Business Model:
- number of branches
- number and location of ATMs
- number of accounts of each type
-business evolution plans (e.g., mergers)

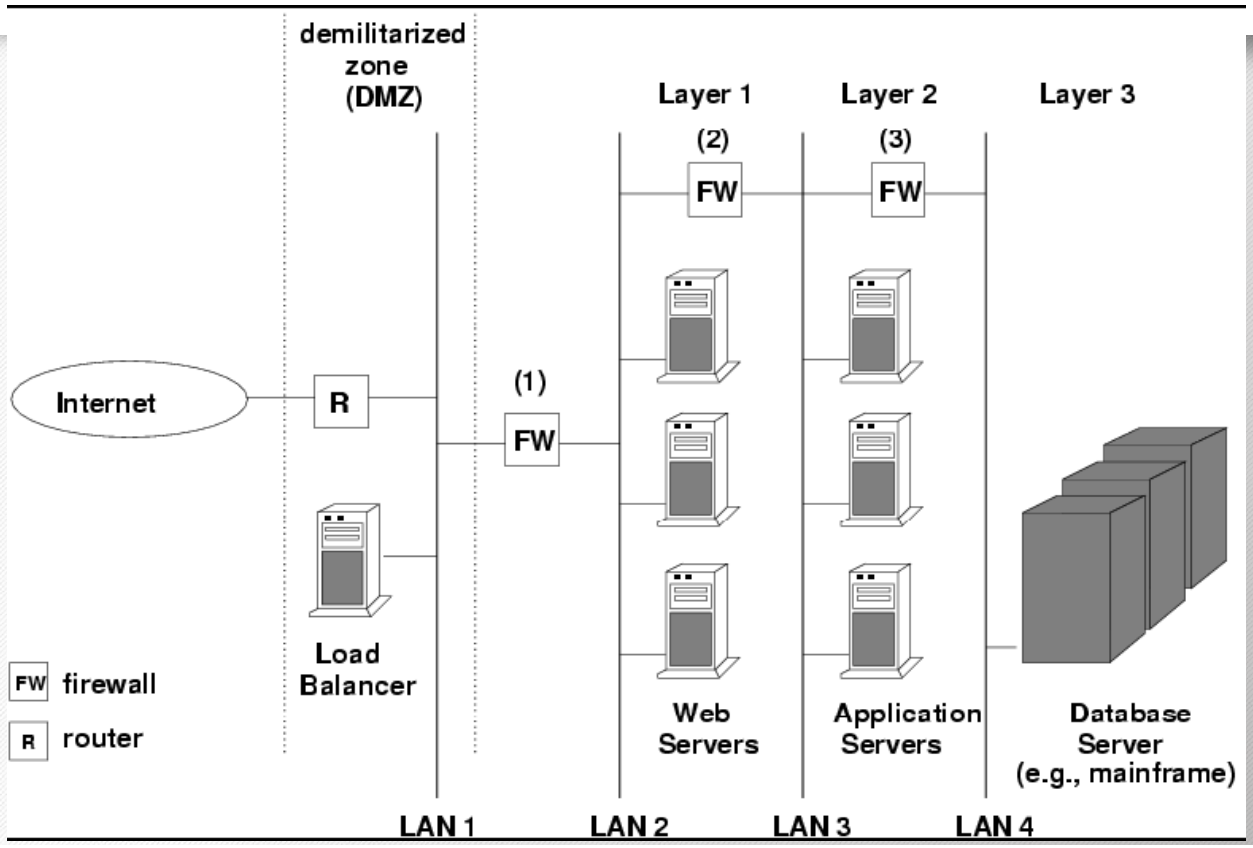
Social Model
- privacy policy
- accessibility policy



User Model:
Customer Behavior Model Graph

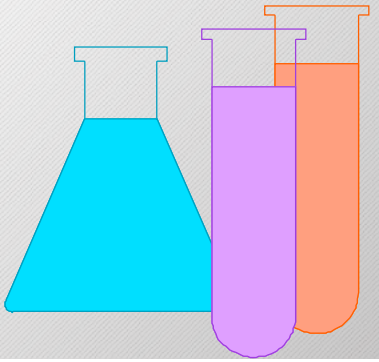


IT Infrastructure: Example



EXPERIMENTAL DESIGN

- Experimental design refers to the organizational structure of your experiment
- Need to methodically go through factors and levels to get the full range of experimental results desired
- There are several "classical" approaches to experimental design



EXAMPLES

- One factor at a time
  - vary only one factor through its levels to see what the impact is on performance
- Two factors at a time
  - vary two factors to see not only their individual effects, but also their interaction effects, if any
- Full factorial
  - try every possible combination of factors and levels to see full range of performance results

Open Questions

- Performance Evaluation of Systems is REALLY important... then,
  - How to increase the level of accuracy of performance models?
    - Of course, it's necessary the definition of metrics and build tools.
  - How to implementate realistic models to performance evaluation?
    - On live process
  - How to integrate the needs of scientifics and enginner/computer science scientist in performance evaluation?

Recommended Lectures

- **The Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling de Raj Jain .Wiley Computer Publishing, John Wiley & Sons, Inc.**
- **Measuring Computer Performance: A Practitioner's Guide, David J. Lilja 2004**

MIPS Reference Cards

- *inst.cs.berkeley.edu/~cs61c/resources/**MIPS**_Green_Sheet.pdf*
- *refcards.com/docs/waetzigj/**mips**/**mips**ref.pdf*
- *www2.engr.arizona.edu/~ece369/.../spim/**MIPSReference**.pdf*

"Houston, We've Had a Problem"