Performance evaluation of information systems

Dr. ing. Guillaume Ducoffe

guillaume.ducoffe@ici.ro

What this course is about

- **How "good" does my system perform ?**

Example: How does evolve the QoS as the utilization grows ?

(**congestion collapse, . . .**)

- **What if we try to improve the system ?**

Example: Increasing the capacity of a link in a network

(**comparison, competition between users, . . .**)

- **What should be the size of my components in order to satisfy the SLA's ?**

Example: How many control processor blades should this Cisco router have?

(**system dimensioning, . . .**)

What this course is not about

- How to design an **efficient implementation** of such systems ?

(this requires further notions in algorithmic, advanced programming techniques, computer hardware, etc . . .)

- How to ensure correctness of the implementation (**bug-free**) ?

(cf. model checking, formal verification, etc . . .)

Objectives of this course

- Introduction of the concepts and methodology for performance evaluation

(fundamental notions such as: **metrics, factors, load, congestion, . . .**)

- A general approach: Performance engineering

(**modelization, testing, operational analyses, . . .**)

- Underlying theory: basic and advanced notions in statistics/probabilities

(**queuing theory, Palm calculus, Markov chain, . . .**)

Before I forget . . .

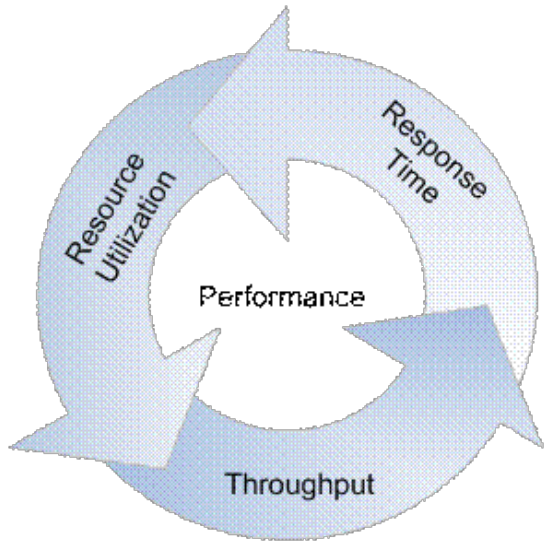
Evaluation for this class:

- final exam counts for 60 % of the grade;
- (some of the) lab exercises count for 40 %.

Warm-up: Performance of a system

Requirements for a system:

- Functional. spec. of functions, I/O's, environments, ...
- **Non functional** $\sim$ QoS. Defined and measured via **metrics**.



Most important metrics: **Response time + Throughput + Resource utilization** (detailed next).

Response time

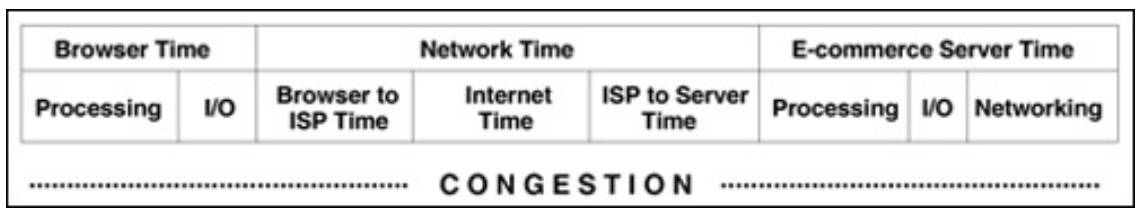
Definition

The time taken to complete a single business transaction. It is usually expressed in seconds.

NB: total response time $= \sum$ response time for every component

2 main categories:

- service time (independent of the load);
- waiting time for resource access (load dependent).



Throughput

Definition

The total number of business transactions completed by the application in unit time (per second or per hour).

Some examples:

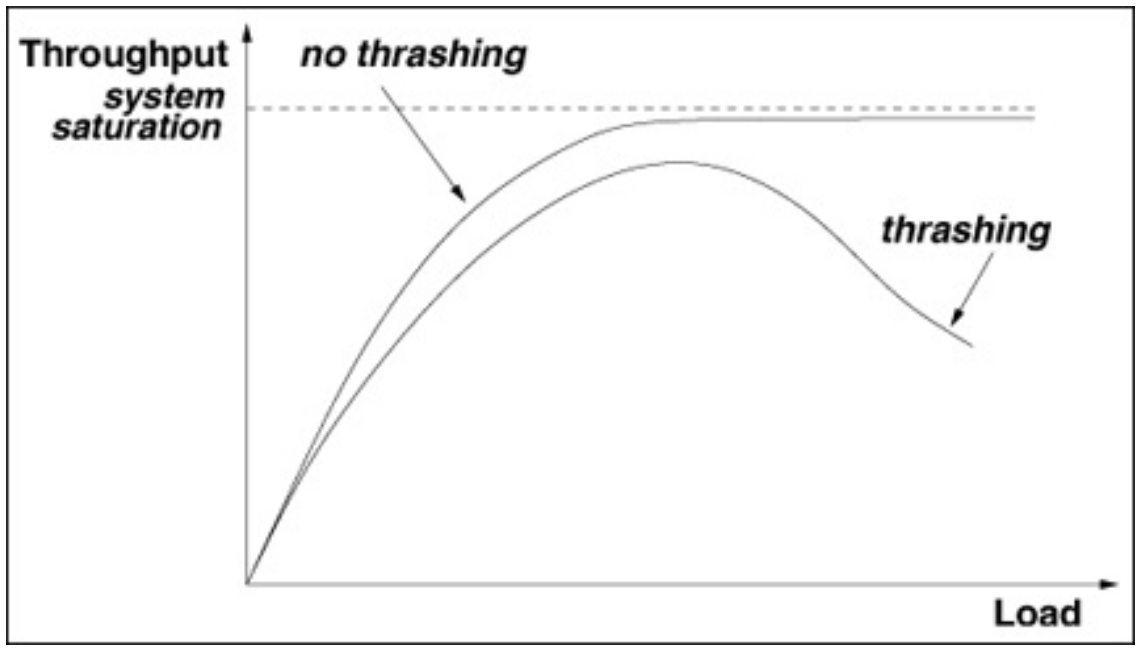
- Transactions per Second (tps)
- HTTP requests/sec
- Page Views per Second
- Bytes/sec
- Sessions per Second
- Searches per Second
- Packets per Second (PPS)
- I/Os per Second, ...

Throughput (cont'd)

Thrashing

- In general, throughput increases with load, then it stagnates when some component is fully used (**bottleneck**).
- In some cases, throughput may decrease with higher load (**thrashing**)!

Example: memory issues



Resource utilization

Definition

The amount of resources consumed by the system during processing a request.

- Some resources are shared: processor, disk (I/O controller), memory, ...

- Utilization of 80% is considered an acceptable limit. Normally utilization of 70% warrants ordering of additional hardware.

Other metrics

- Context dependent

  Main goals of performance evaluation:

  - **comparison**;
  - **system dimensioning**;
  - overload behaviour, ...

Rule of thumb: first define your goals before choosing your metrics.

Other metrics

Availability

Definition

The fraction of time that a system is up and available to its customers.

- A low availability can deteriorate the reputation of a company, make loose clients, etc.
- What does "low" means ? $\Longrightarrow$ context-dependent

Example: a system with 99.99 % availability over a period of thirty days would be unavailable for:

$$(1 - 0.9999) * 30 * 24 * 60 = 4.32 \text{ min.}$$

- Acceptable for a bookstore;
- Unacceptable for a nuclear plant!

Other metrics

Availability

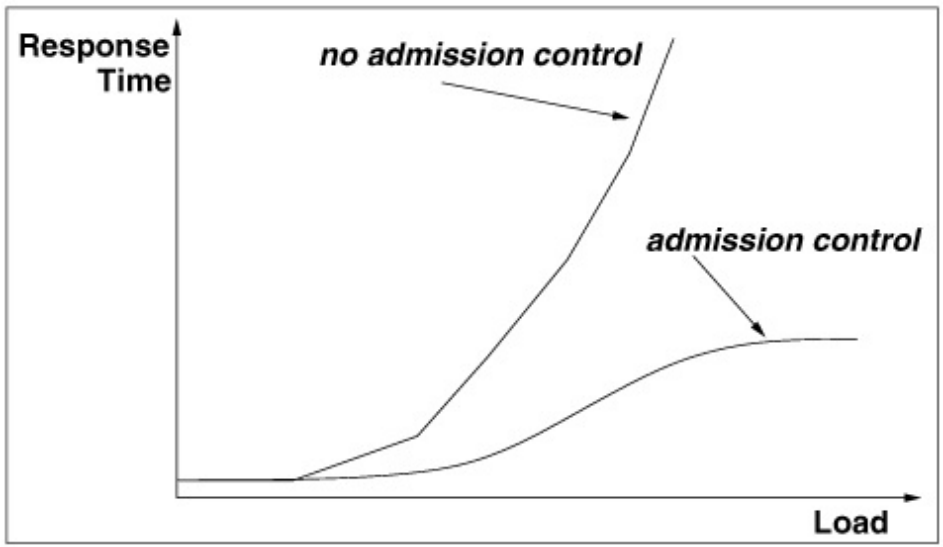
- Main causes for unavailability:
  - **Failures**

    Example: the network connection of a Web site may be down.

    $\Longrightarrow$ failure detection, failure diagnosis, and failure recovery.
  - **Overload**

    Example: the maximum number of connections is reached.

    $\Longrightarrow$ **admission control**



Other metrics

Reliability

Definition

The probability that a system functions properly and continuously over a fixed period of time.

When the period $t \to \infty$ we have:

$$\text{reliability} \to \text{availability}$$

The two are different for short periods of time.

Other metrics

Security

Definition

A combination of three basic attributes:

- *Confidentiality.*
- *Data Integrity.*
- *Non-repudiation.*

- Ensured via authentification protocols and cryptography (beyond the scope of this class).

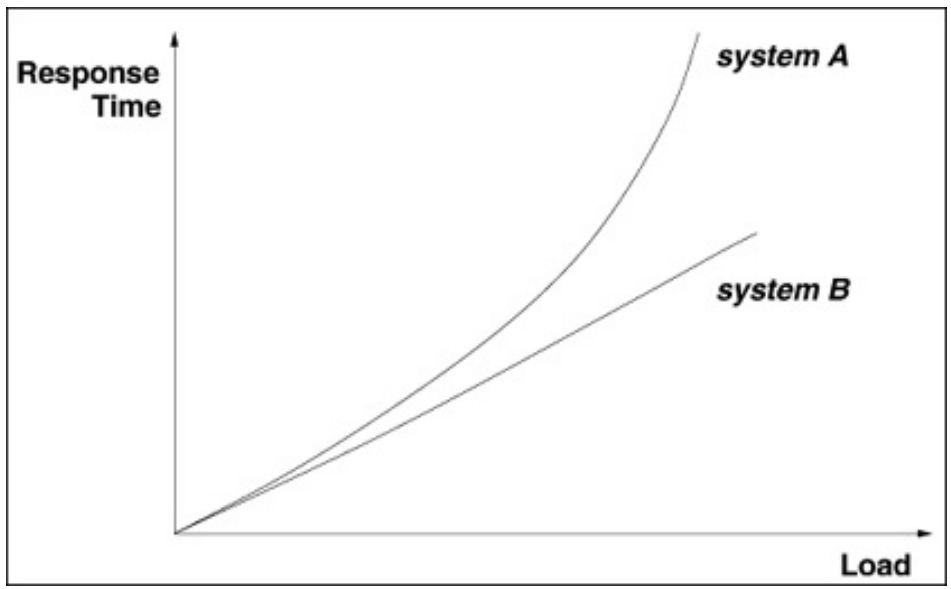
- Encryption is expensive: trade-off between security and performance.

Other metrics

Scalability

Definition

A system is said to be scalable if its performance does not degrade significantly as the load increases.



System B is scalable (linear shape). System A is not.

Other metrics

Extensibility

Definition

The property of a system to easily evolve to cope with new functional and performance requirements.

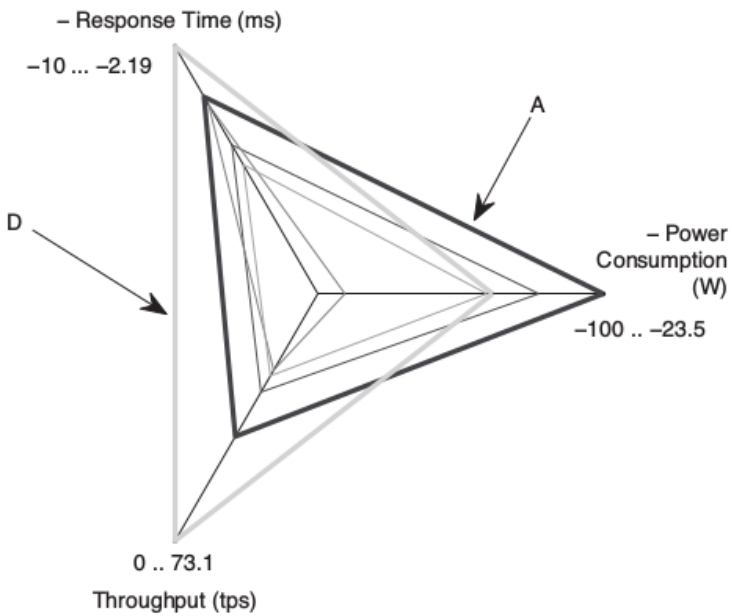
- the needs of system users are almost impossible to completely anticipate
- new laws and regulations, different business models, ...

Multi-dimensional analysis of metrics

- There might not be <u>one</u> "best" solution

$\Longrightarrow$ *partial order*.

- "better" solutions = **non dominated**
- a classical representation: **Kiviat Diagram**



Conditions of evaluation

To be considered during the analysis:

- the **(work)load**.
- external **factors**.
- System **life cycle**.

Why is it important ?

How to take it into account (and to which extent) ?

The offered load

Definition

The quantity of requests that are submitted to the system.

NB: related to resource utilization

typical example: on/off links in a network (green routing)

- performance decreases as the intensity of the load grows

(non linearly, seee the course on queuing theory)

- not all requests are equivalent $\Longrightarrow$ need for a *detailed* intensity

Example: Web server requests

representatives **benchmarks** might be cumbersome to obtain ...

Factors

Definition

System components and load that can potentially impact the performance.

Examples: background activity, multiple users, ...

- **All** factors need to be taken into account

hidden factors can change the conclusions

Example: Simpson paradox

	Week 1 quantity	Week 2 quantity	Total quantity *and* weighted quality
Lisa	0%	75%	**60%**
Bart	**25%**	**100%**	40%

(from Wikipedia)

- *Randomization* can help (*e.g.*, for *Web's transparency*)

System life cycle

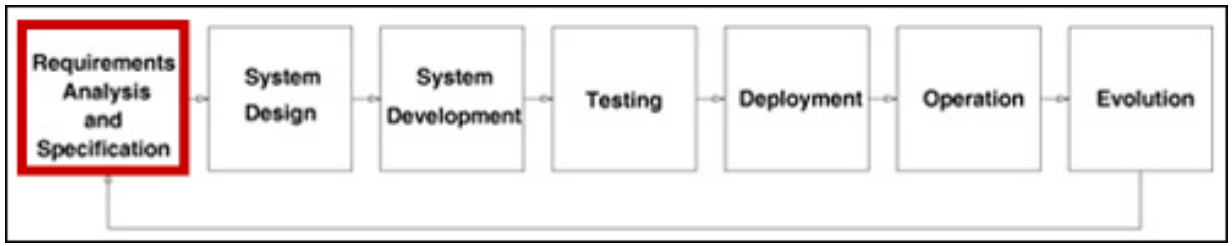
- a common, but misguided approach: addressing performance issues at the *end* of development

Can lead to:

- more expenses
- complicated tuning
- **restart from scratch**
- etc . . .

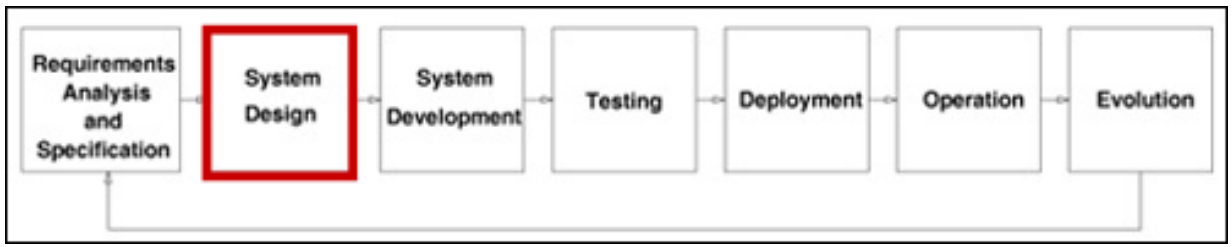
We should address QoS issues at every phase of the life cycle.

System life cycle (cont'd)



- Functional requirements (Non functional is postponed until Testing).
- Stabilizing **Service Level Agreements** (goals)
- Analysis of load intensity

System life cycle (cont'd)

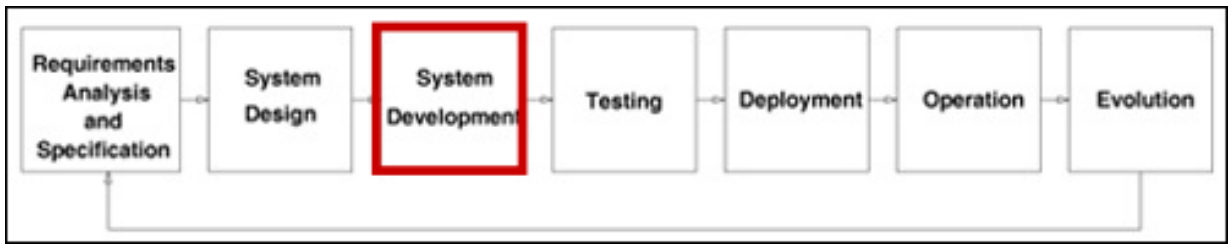


- Stabilizing the Architecture of the System
(system components, data structures, . . .)

- **Performance evaluation** (crucial step): choosing certain components based on the load + SLA's

Example: Research engine

System life cycle (cont'd)

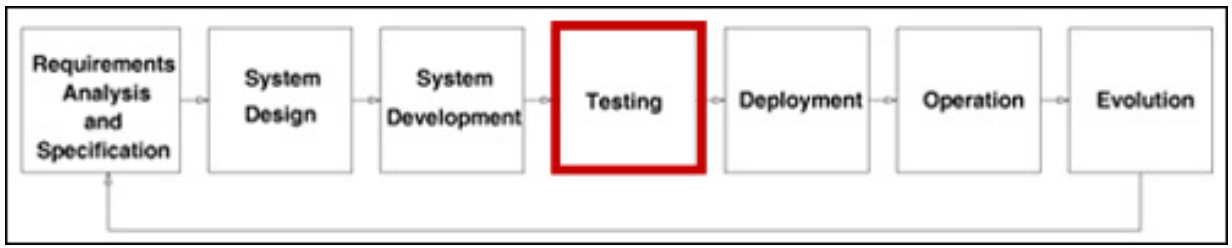


- Implementation of the components

(new or reused from somewhere else. . .)

- **Performance evaluation**:
  - execution of the decisions taken at the previous step
  - data collection for Testing

System life cycle (cont'd)



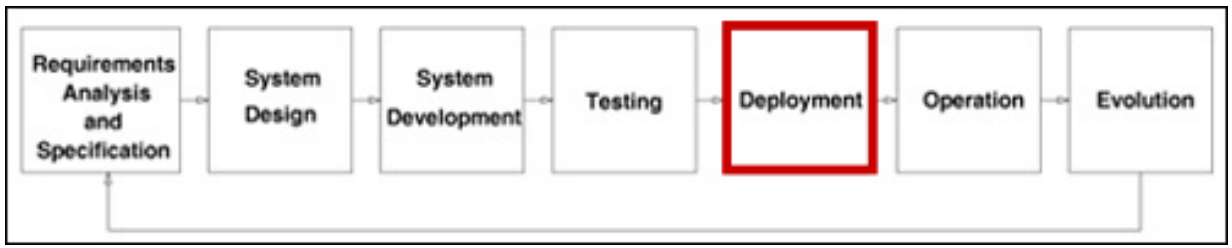
- **Unit tests** for every component as soon as it becomes possible (in parallel with Development)

- Then, testing subsystems (gradually) and finally, the full system.

Functional testing is more time consuming than Non functional testing (in general)

It is unrealistic (and expensive) to test every possible scenario

System life cycle (cont'd)

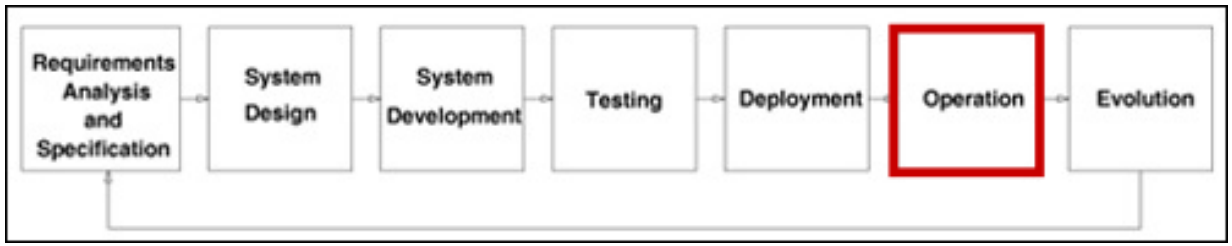


- Configuration parameters depend on the SLA's

Example: number of TCP connections, size of databases, ...

- We make performance predictions for multiple scenarios (partly derived from load analysis)

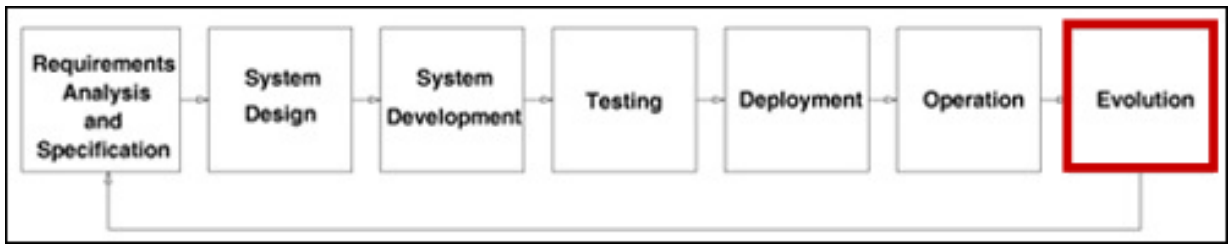
System life cycle (cont'd)



- Constant auditing of the system:
  - load (anormal peak periods ?)
  - external metrics (from the client's side):
    response time, throughput, ...
  - internal metrics (helpful for system improvements):
    resource utilization, ...
  - *availability*

Some adjustments might be necessary ...

System life cycle (cont'd)



- Changes in the system environment

NB: related to Extensability

- Design of **predictive models** for QoS impact.

Performance patterns

Definition

Traits that are common to multiple settings.

- Extremely helpful for the evaluator !
- Classical performance patterns:
  - Bottlenecks
  - Congestion collapse
  - Latent congestion collapse
  - Competing users (w/o congestion control)

Bottlenecks

Definition

The weakest components of the system.

- Usual suspects: processors, network connection, I/O disk, etc.

- simple estimation:

  overall performance "=" behaviour of the bottlenecks

Example (thanks Wikipedia !): a gaming computer with a powerful CPU is bottlenecked by a lower-end GPU.

It is important and useful to identify the bottlenecks

Bottlenecks (cont'd)

- Any system or application will hit a bottleneck if the work arrives at a sufficiently fast pace.

- Removing one bottleneck may cause some other bottleneck to appear !

Example: High Performance Web Sites

- 14 consecutive bottlenecks identified, *i.e.*, DNS lookups, then script parsing, etc.
- eventually bottlenecks are CPU speed + network connection (much more desirable).

Congestion collapse

- Bottleneck implies a maximum **capacity**

- **Congestion** occurs whenever load $>$ capacity
(unavoidable !)

- **Congestion collapse** means a reduction in system utility

Example: message losses in a saturated network

Some feedback mechanisms can help preventing the collapse (*e.g.*, TCP)

Latent congestion collapse

- Bottlenecks sometimes protect from congestion collapse.

Removing the bottleneck $\Longrightarrow$ congestion collapse

("*give more, get less*" paradox)

Example: a museum audio guide to download

- Bottleneck is the transfer rate;
- However, increasing the rate also increases the load!
- Packet collisions become more frequent, hence the congestion collapse.

Competition Side Effects

Definition

The performance of one user influences other users.

- A paradox:

increasing resources $\Longrightarrow$ increasing load for some users $\Longrightarrow$ decreasing performance for some users !

- If in addition there is no congestion control then the following can happen:

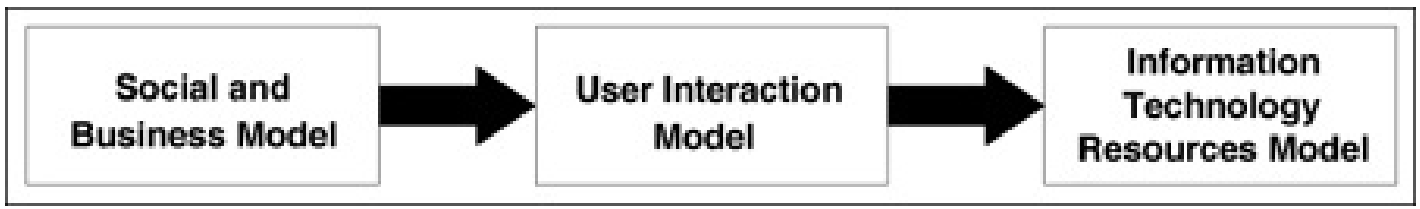
increasing resources $\Longrightarrow$ performance deterioration or stagnation for all users !

- Cf. Lab #1 for an example.

A reference model for information systems

- Integration of IT System with social environment

- Accounting for three distinct models:



this is important for defining the high level requirements.

A reference model for information systems (cont'd)

• System designers and developers depend on the social and business model that is to be supported by the system.

- **Social model**;
- **Business model**.

Example: case of a bank

- Social model: privacy + accessibility policies
- Business model: location of ATM's, number of bank accounts, banking transactions, etc.

A reference model for information systems (cont'd)

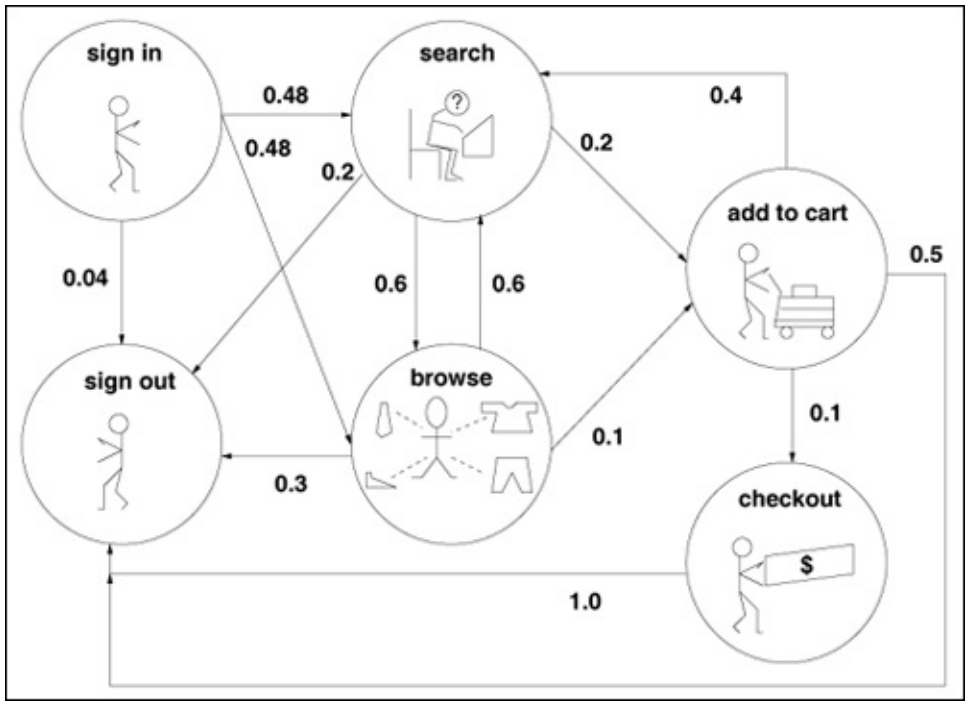
User interaction model

Definition

The interactions between customers and the system

- Information about requests:
  - nature;
  - frequency;
  - consecutiveness, ...
- Customer Behavior Model Graph (CBMG)

Related to Markov theory (see later in the course)

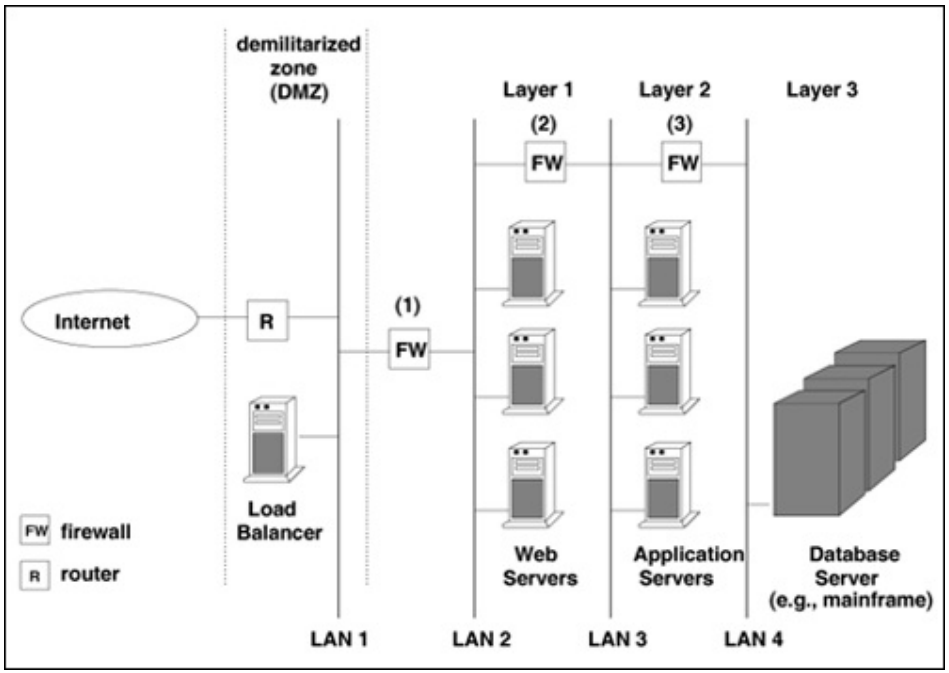


A reference model for information systems (cont'd)

IT resources model

- IT infrastructure: processors, storage subsystems, networks, routers, firewalls, ...

- Technical support of previous models.



Concluding remarks

Steps to follow . . .

- Design a reference model;
- Follow the System life cycle;
- Define the goals of your performance evaluation;
- Identify the factors;
- Choose your metrics;
- Evaluate the workload;
- Recognize the classical patterns
  (bottlenecks, congestion collapse, . . .)

Next courses

- Modelization of information systems
- Analytical models: **queuing theory**
- Palm calculus (Little's law).

Questions