Introduction

Outline

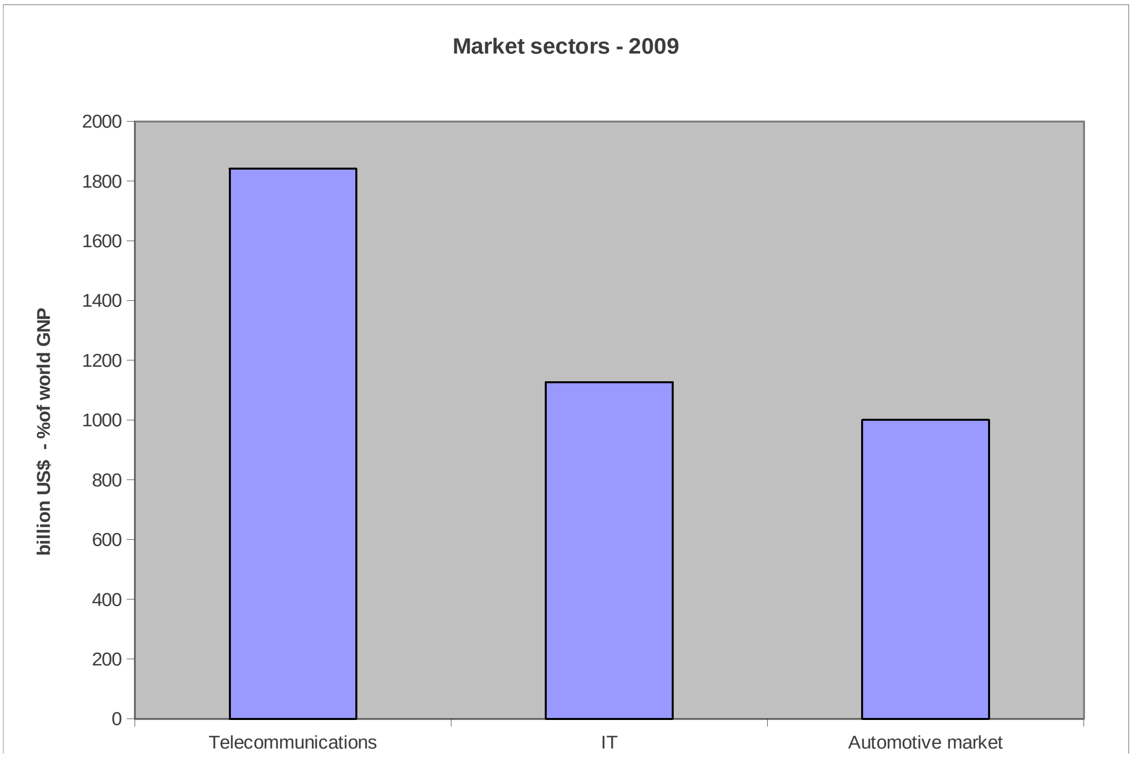
- Motivation
- Failures
- Definition and concepts
- Process and product properties
- Principles
- SE Approaches

Motivation

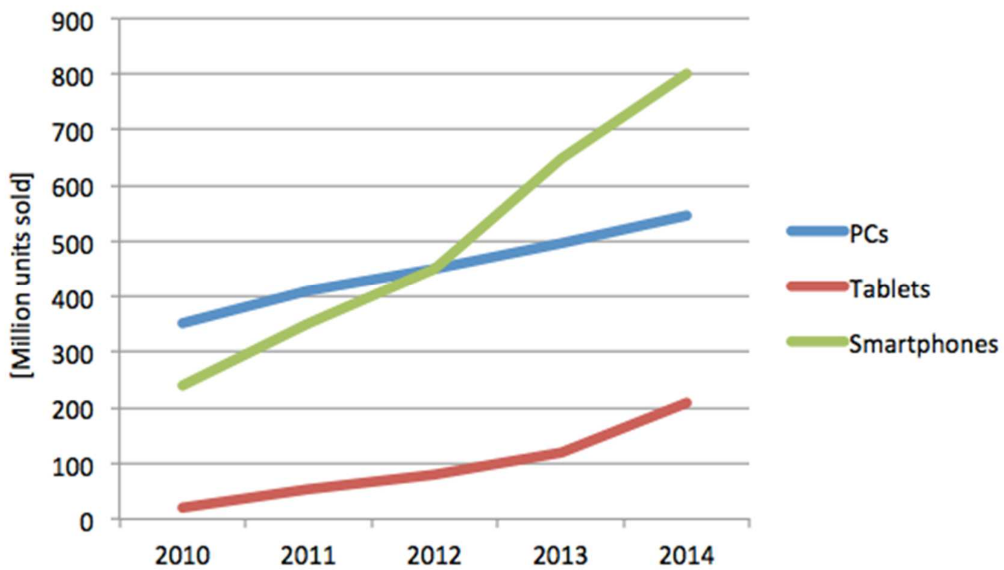
Software and the economy

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled
- Expenditure on software represents a significant fraction of GNP in all developed countries.

ICT Market – world – 2009



Sales



1.http://en.wiserearth.org/article/61670dc8dd5c5a3adba285f39bbf8145, last visited : Feb, 1

Sales, 2010

CPUS: 10Gb

Mobile phones: 1G

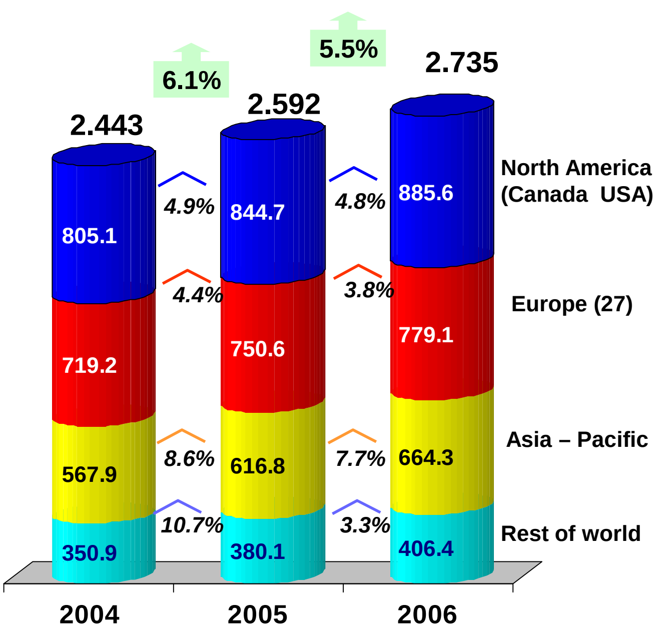
Computers: 300M

Tv sets: 250M

Vehicles: 60M

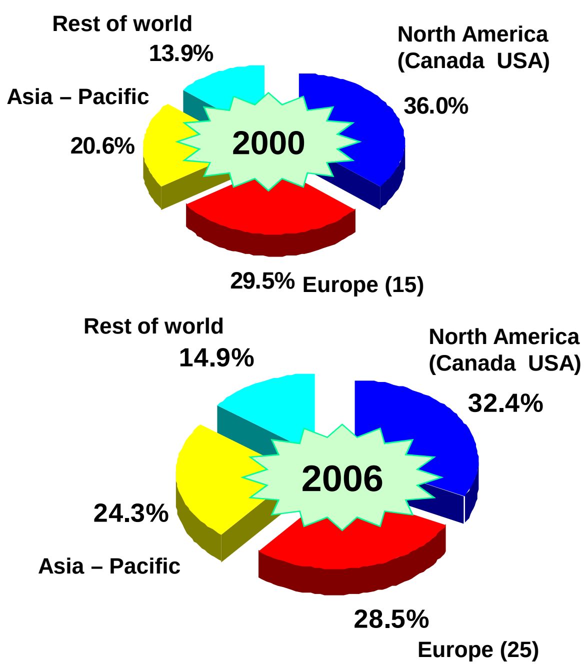
ICT market, per area

Mld$, yearly variations

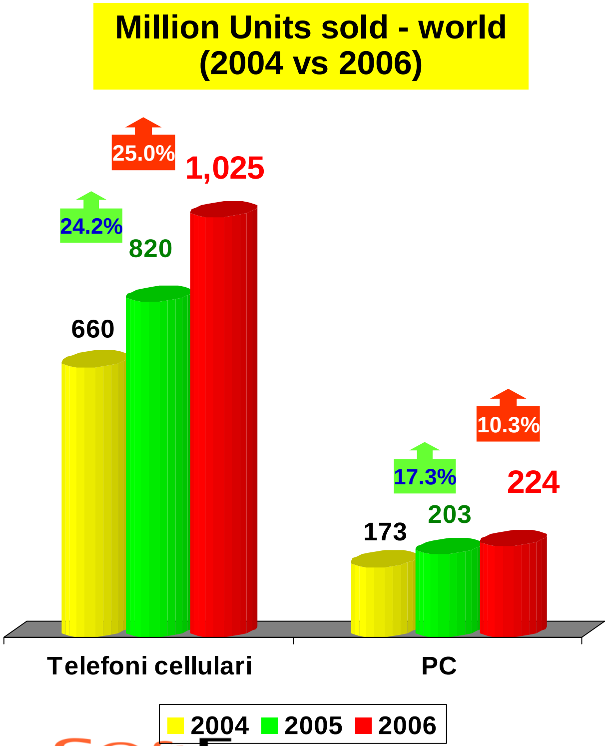


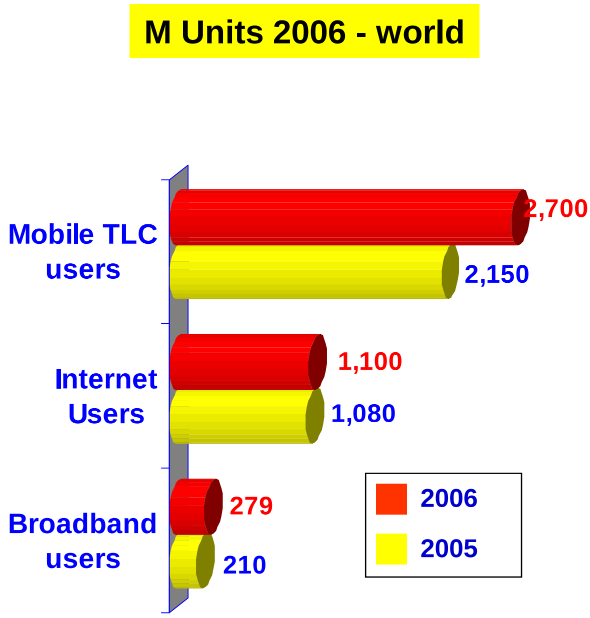
Fonte: AITech - Assinform / NetConsulting

world ICT market shares



ICT Diffusion, world





Fonte: AITech - Assinform / NetConsulting

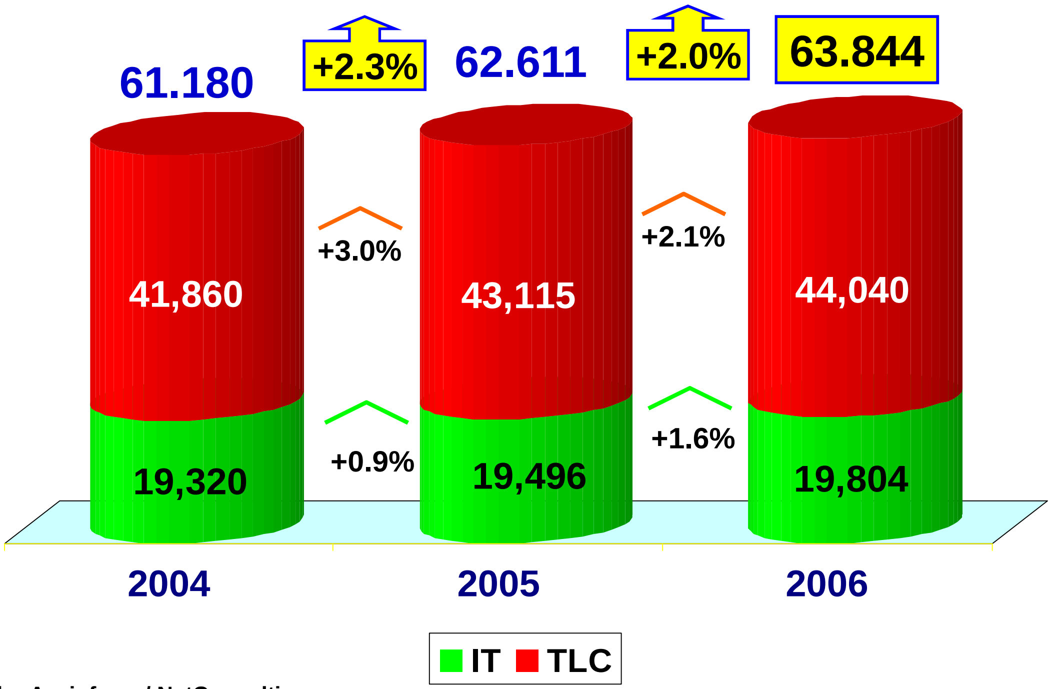
SoftEng http://softeng.polito.it

Italy

SoftEng http://softeng.polito.it

ICT Market

Valori in Milioni di Euro e in %

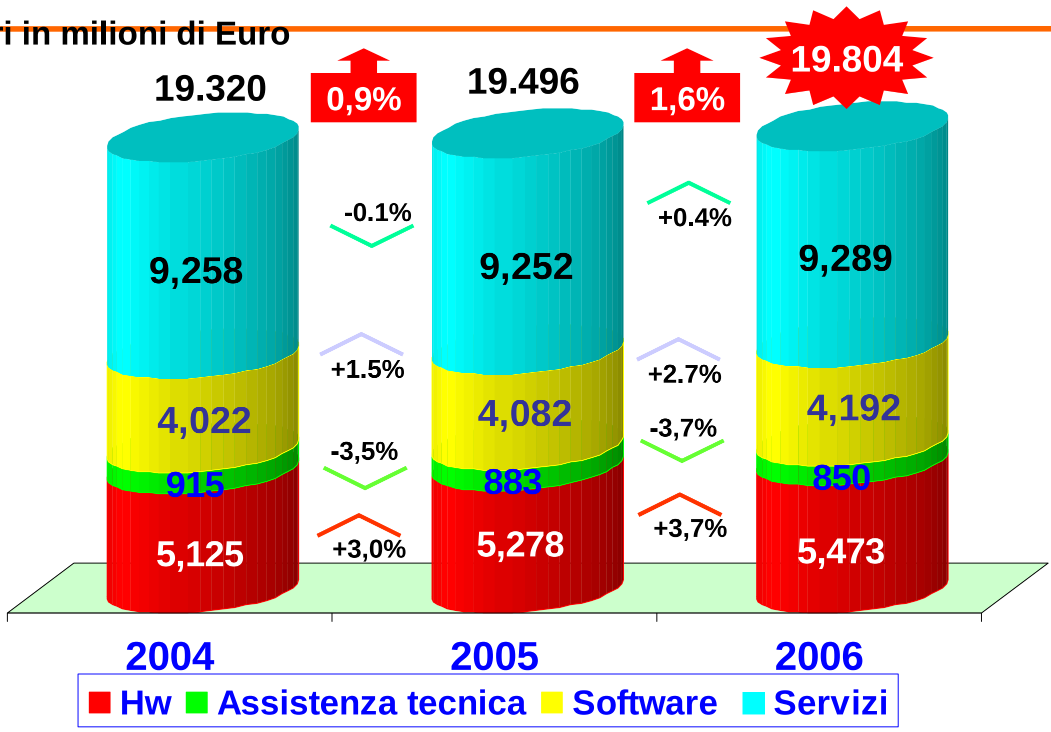


Fonte: AITech - Assinform / NetConsulting

SOftEng http://softeng.polito.it

IT Market

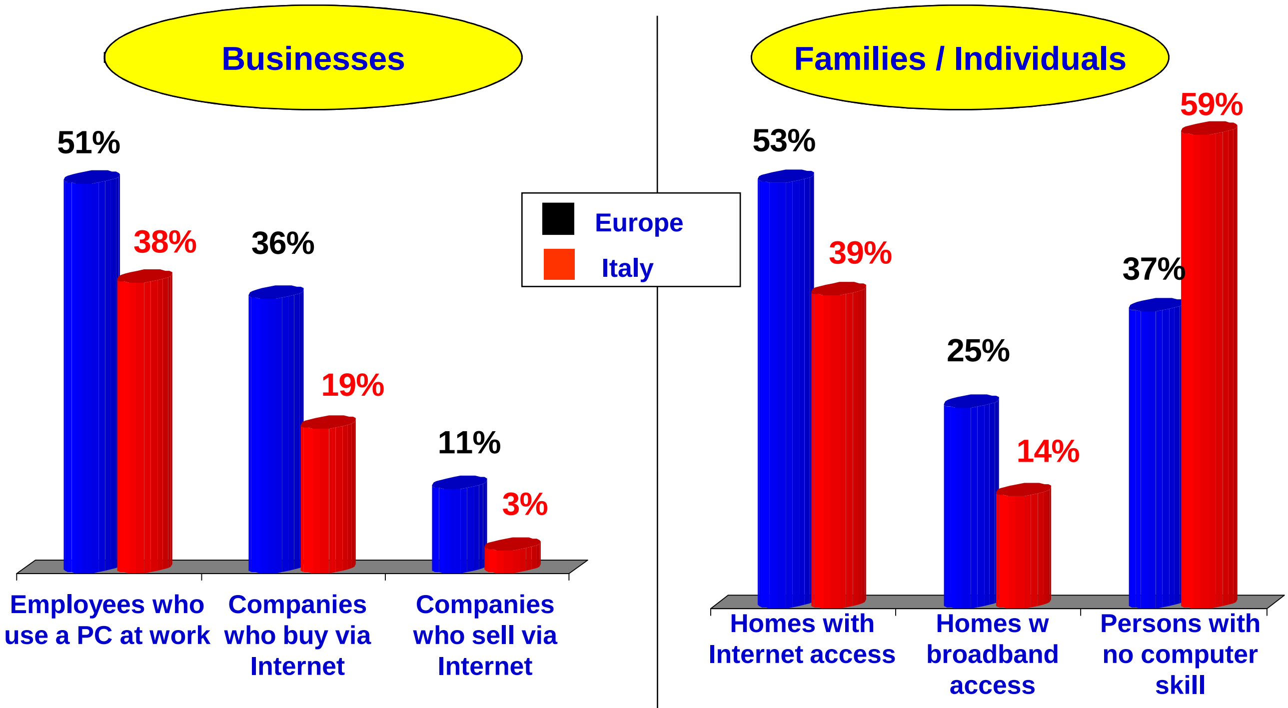
Valori in milioni di Euro



Fonte: AITech - Assinform / NetConsulting

SOftEng http://softeng.polito.it

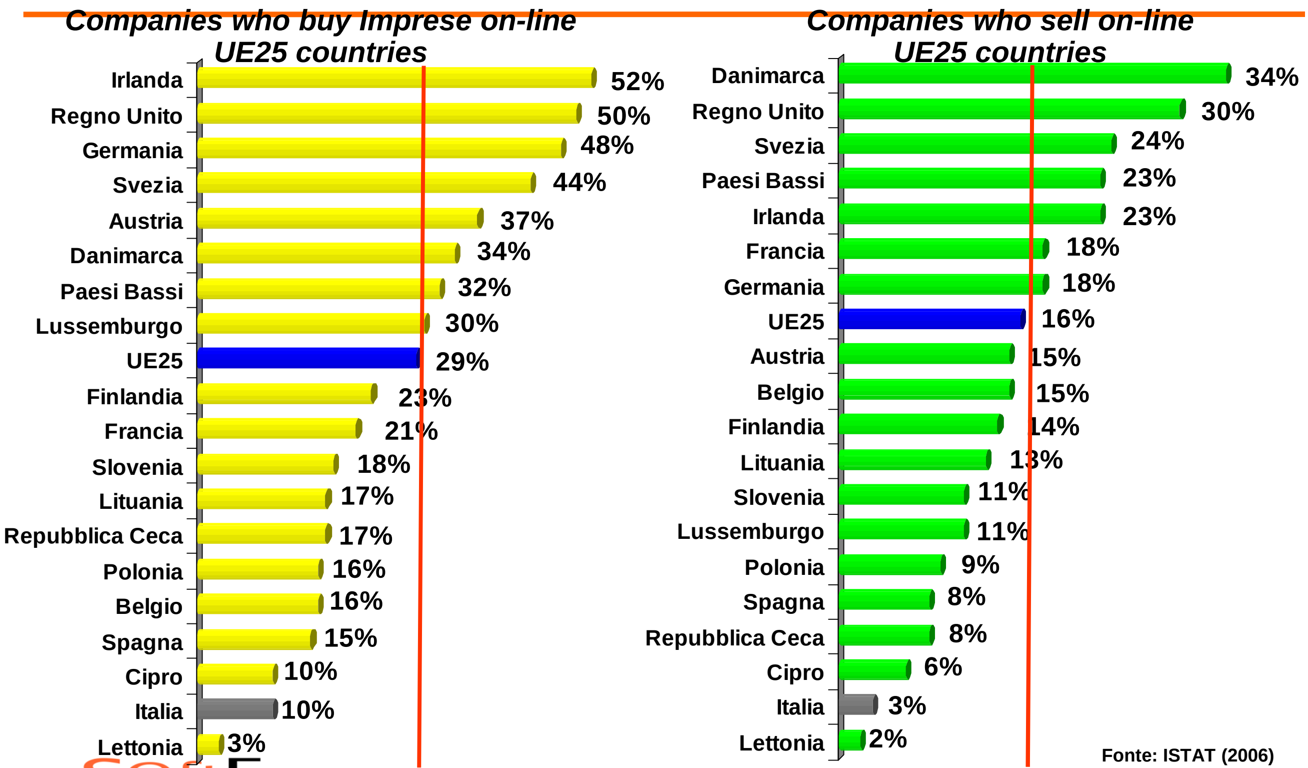
Diffusion



Fonte: Eurostat (2006)

SoftEng http://softeng.polito.it

Diffusion



Fonte: ISTAT (2006)

SoftEng http://softeng.polito.it

IT (2006)

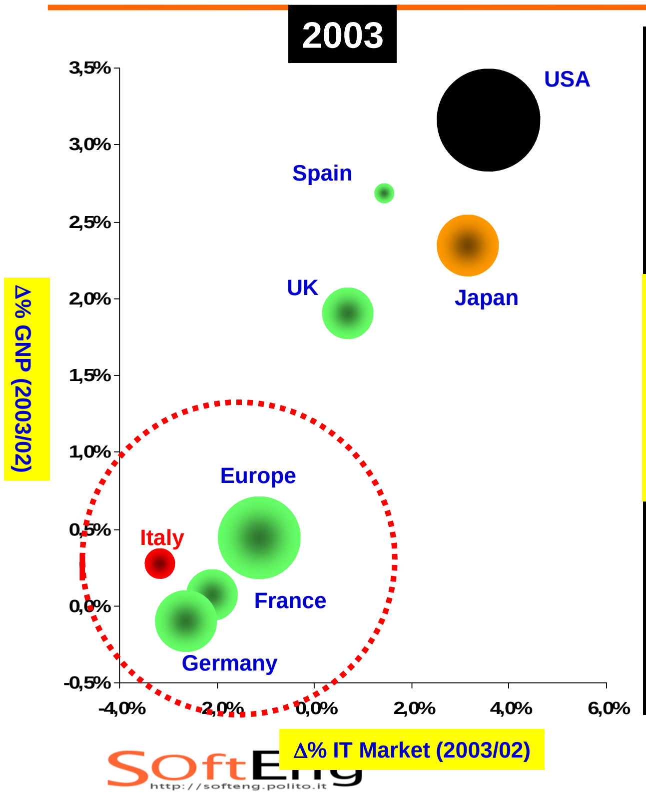
Euro

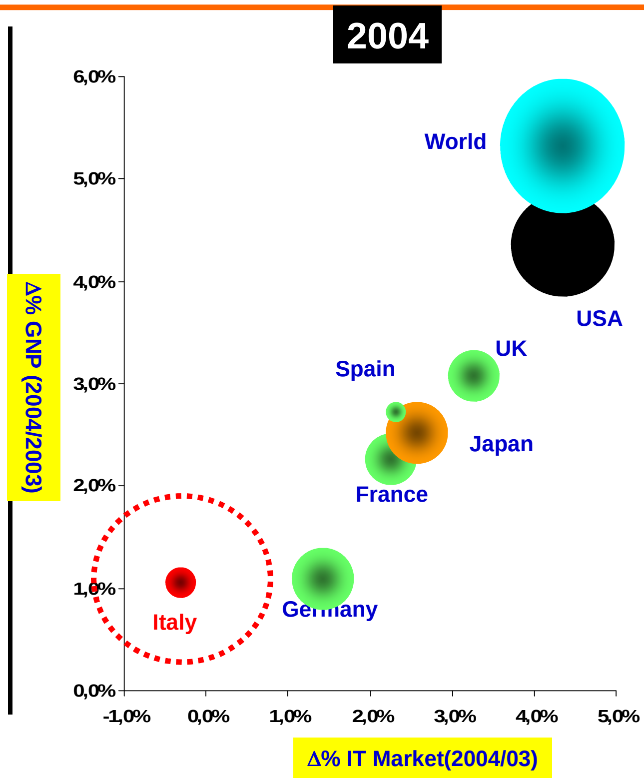
	IT expense/ GNP	per capita IT expense	per employee IT expense
USA	3.9%	1,408	2,945
Japan	2.3%	878	1,765
Germany	3.1%	812	1,837
UK	3.1%	983	2,095
France	3.2%	839	2,050
Italy	1.9%	341	878
Spain	1.9%	372	748

Software, innovation, development

- Evidence of correlation between ICT diffusion and wealth
  - Positive correlation IT usage and per capita GNP
  - Positive correlation productivity increase and ICT usage

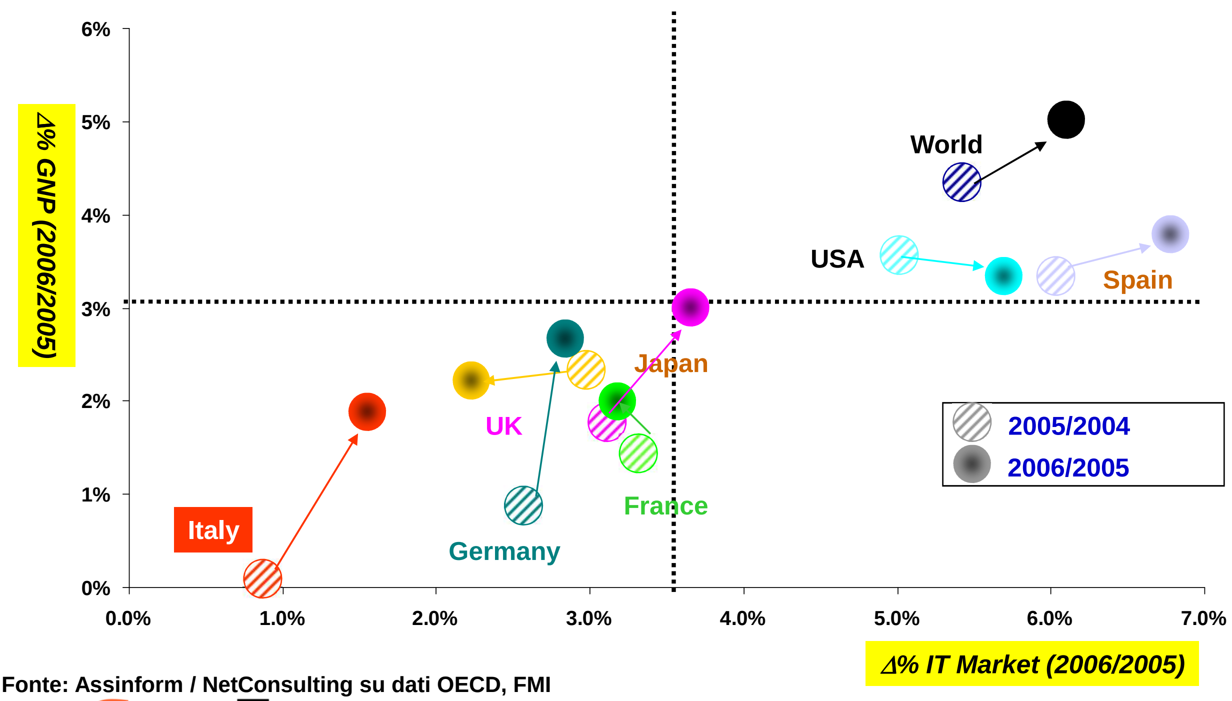
Development – IT investment





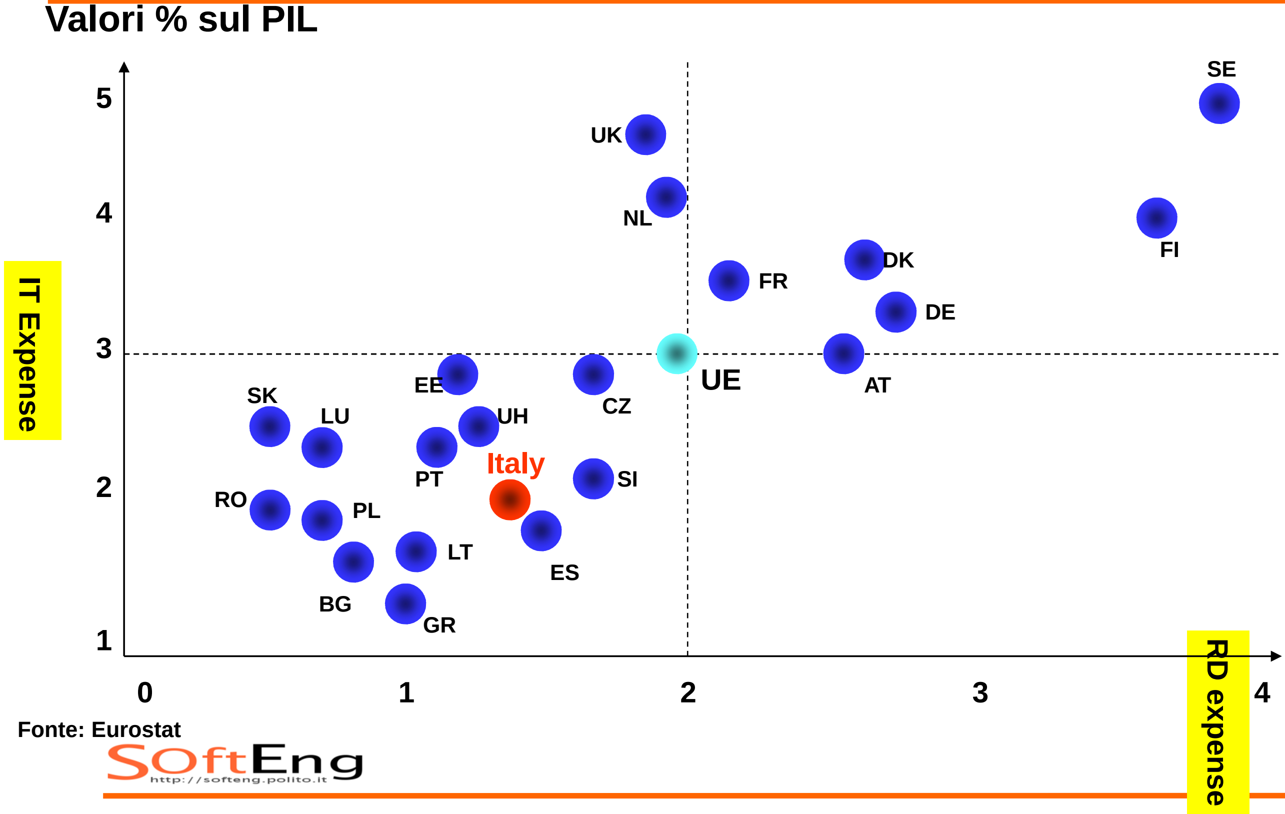
Fonte: Assinform / NetConsulting su dati OECD, FMI

Development – IT Investment



Fonte: Assinform / NetConsulting su dati OECD, FMI

R&D investment vs. IT investment



Fonte: Eurostat

SoftEng http://softeng.polito.it

Some data to think about

Computer use vs GDP

SoftEng http://softeng.polito.it

Other data

- Internet users growth

Internet users (as percentage of population)
ICT good exports (See also good imports and service exports)

SoftEng http://softeng.polito.it

WEF - ICT development of nations

Ranking WEF
(world economic forum)
Global IT report 2006-2007
www.weforum.org

1 – Denmark
Sweden
Singapore
Finland
Switzerland
Netherlands
United States
Iceland
United Kingdom
Norway
Canada
Hong Kong SAR
Taiwan, China
Japan
Australia
Germany
Austria
Israel
Korea, Rep.
Estonia
Ireland
New Zealand
France
Belgium
Luxembourg
Malaysia
Malta
Portugal
United Arab Emirates
Slovenia
Chile
Spain
Hungary
Czech Republic
Tunisia
Qatar
Thailand
40 - Italy

WEF – ICT development

- ICT conductive environment
  - Regulatory aspects, soft + hard infrastructure
- ICT readiness
  - Individuals, business, government
- ICT usage
  - Individuals, business, government

SoftEng
http://softeng.polito.it

WEF – Global competitiveness

World Economic Forum- Global competitiveness report 2006-2007
www.weforum.org

1- Switzerland
Finland
Sweden
Denmark
Singapore
United States
Japan
Germany
Netherlands
United Kingdom
Hong Kong SAR
Norway
Taiwan, China
Iceland
Israel
Canada
Austria
France
Australia
Belgium
Ireland
Luxembourg
New Zealand
Korea, Rep.
Estonia
Malaysia
Chile
Spain
Czech Republic
Tunisia
Barbados
United Arab Emirates
Slovenia
Portugal
Thailand
Latvia
Slovak Republic
Qatar
Malta
Lithuania
Hungary
45 - Italy

Global competitiveness

- Institutions
- Infrastructure
- Macroeconomy
- Health + primary education
- Higher education
- Market efficiency
- Technological readiness
- Business sophistication
- Innovation

Failures

SoftEng http://softeng.polito.it

Ariane V (1996)

Ariane V (1996)

- The European launcher for earth satellites
- A software defect caused an error in computing the position and speed of the launcher some 30 seconds after launch. The wrong position data caused the controller to send signals to the engines to change the direction of the launcher so swiftly that the structure was subject to high tensions. The tension went over the acceptable thresholds and the safety controller ordered self destruction.

SoftEng http://softeng.polito.it

What happened

On june 4th 1996, the maiden flight of the Ariane 5 launcher ended in a failure;

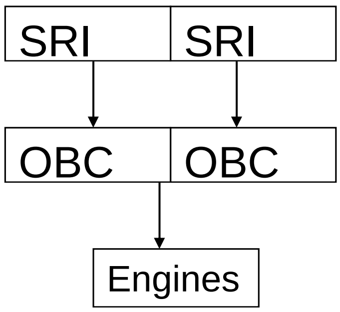
Only about 40 seconds after initiation of the flight sequence the launcher veered off its flight path, broke up and exploded;

The system failure was a direct result of a software failure.

The subsystem

SRI: computer-based inertial reference system, computes attitude and trajectory of the rocket and sends them to OBC. Redundant.

OBC (on board computer): executes flight program, controls engines. Redundant.



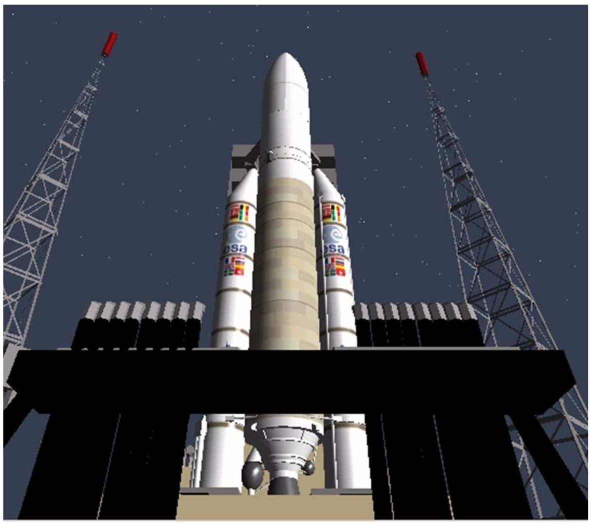
The problem

Software failure on SRI. Occurred when, in function F, an attempt to convert a 64-bit floating point number to a signed 16-bit integer caused the number to overflow.

There was no exception handler associated with the conversion so the system exception management facilities were invoked. These shut down the SRI.

The backup SRI had the same software, and behaved in exactly the same way.

The OBC received diagnostic commands from shutting down SRI, and interpreted them as normal data, commanding engines to extreme position, resulting in unforeseen stresses on the rocket, that caused separation of the boosters from the main stage, in turn triggering the self-destruct system of the launcher.



Why?

Why the overflow?

Why no exception handling?

SRI was reused from Ariane 4. The physical characteristics of Ariane 4 (A smaller vehicle) are such that it has a lower initial acceleration and build up of horizontal velocity than Ariane 5. The value of the variable on Ariane 4 could never reach a level that caused overflow in function F during the launch period.

Besides, function F was NOT needed in Ariane 5 (was in Ariane 4). Decisions were made:

- Not to remove F as this could introduce new faults;
- Not to catch overflow exceptions because the processor was heavily loaded. For dependability reasons, it was thought desirable to have some spare processor capacity.

Mars Polar Lander (2000)

A probe expected to land on Mars for scientific exploration

A measure of length had to be exchanged between two components developed by two different teams.

The two teams used two different unit of measures.

The difference was very small and went unnoticed until the probe crashed on Mars

The problem

Key information is

- is NOT in the code
- (or is implicit in the code)

Software ≥ code

Software engineering is about handling these information

SoftEng http://softeng.polito.it

Key information – Ariane V

Function F(float x) works correctly if range of input parameter x is between x1 and x2

Not written

```
F(float x) {
}
```

Written in code

```
F(float x) {
  if (x < x1 or x > x2)
    then signal error
}
```

Written as comment

```
F(float x) {
  // x in range [x1 , x2 ]

}
```


Key info Mars polar lander

```
Int G (float y){
// Y is in meters per second


}
```

Definitions and concepts

Software

- Software = computer programs (= code) + data + procedures + documentation
- Producing *software* is 10x more expensive than producing *code* [Brooks75, the mythical man month]

SoftEng
http://softeng.polito.it

Software – types

- stand alone
  - word processor, game
- embedded
  - ABS, washing machine, digital camera, mobile phone, ..
- process support
  - production process (things): industrial automation
  - business process (information): management automation

SoftEng http://softeng.polito.it

Software vs. system

Stand alone software →

'software'

'software development'

'software engineering'

Embedded software →

'system'

= software + sensors + actuators

'system development'

'system engineering'

SoftEng http://softeng.polito.it

Software vs. system

Software development
(cfr. ISO 12207)
System development
(cfr. ISO 15288)
software development
+ hardware (sensors, actuators) development

Software – criticality

- Criticality = damage in case of malfunction
- safety critical
  - Damage to human lives
    - aerospace, military, medical, ..
- mission critical
  - Disruption of key services, large money loss
    - banking, logistics, industrial production, ..
- other
    - games, ..

Software – complexity

- Complexity: parts and interactions among parts
  - [H Simon, The sciences of the artificial 1969]
- IKEA table: 5- 10 components
- bicycle: 20 - 100
- car: 30.000
- airplane: 100.000

Software complexity

- As of 2012, the Linux 3.2 release had 14,998,651 lines of code.[1]
- Windows 7 about 50 millions lines of code [2]
- An Android operating system in a smart phone consists of 12 million lines of code [3]
- The F-22 Raptor, the current U.S. Air Force frontline jet fighter, consists of about 1.7 million lines of software code. [4]
- The F-35 Joint Strike Fighter requires about 5.7 million lines of code to operate its onboard systems. [4]
- Boeing’s new 787 Dreamliner requires about 6.5 million lines of software code to operate its avionics and onboard support systems. [4]
- Recent premium-class automobile ” probably contains close to 100 million lines of software code,” [4]

Software – complexity

- software systems are probably the most complex human artifacts
- One step ahead
  - Human brain
    - 86-100 G Neurons, ? synapses

Software – lifespan

Short: few months

many apps, games

Long: dozen of years

business support, process support, automotive

Software product scenarios

	type	criticality	complexity	lifespan
MS Office, MS windows	Stand alone	mission	high	5-10 yrs
Business support (bank)	Stand alone, embedded	mission	high	5-15 yrs
Automotive (ABS)	embedded	safety	Medium high	10 yrs
Airplane control	embedded	safety	high	10-20yrs
Computer game	Stand alone	no	Low to high	Weeks to years

Scenarios and process

The process must be adapted to the product scenario

[1] Thorsten Leemhuis (5 January 2012). "Summary and statistics - The H Open Source: News and Features". The H. Heinz Heise. Retrieved 11 Feb 2012.

[2] http://answers.yahoo.com/question/index?qid=20080712132328AAwyert

[3] https://docs.google.com/viewer?url=http%3A%2F%2Fwww.rttonline.com%2Ftt%2FTT2011_010.pdf

[4] http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code

Diffusion

- local
  - 1945 - 1980: scientific community, military, banks, large private organizations
- global
  - 1985 - today: ‘free’ hardware, huge diffusion of computing, impact on everyday’s life

Misconceptions

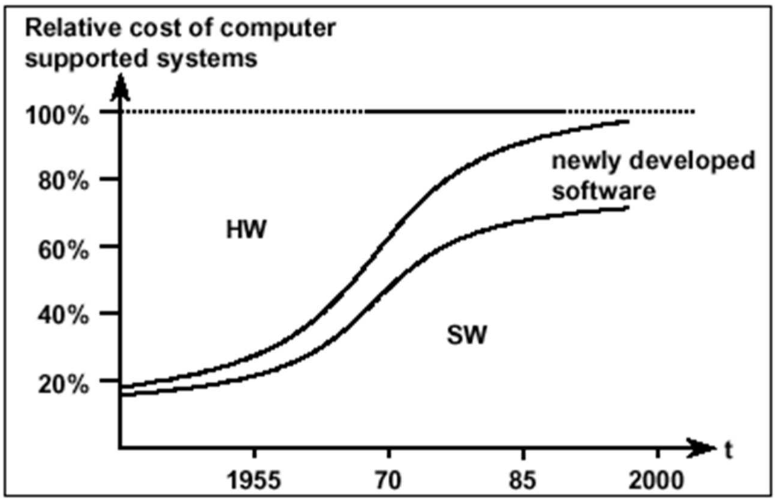
- Software is free
- Software is soft
- Software is produced
- Software ages

Software is free

- Very labor intensive --
  - assuming
  - Productivity = 200 – 1000 LOC per person month
  - Personal cost = : $ 8.000 per person month
  - $8 to $40 per LOC
- a medium sized project with 50.000 LOC costs between $400.000 to $1.600.000 in personnel

Software is free

- Cost of software is dominant



(Boehm 1976)

Software is soft

- Yes, softer than hardware but changing it is difficult and costly
  - Cost of maintenance > cost of development (if lifespan is long)
  - Maintenance becomes impossible at a certain point (architecture erosion)
- And change always happens

Software is produced

- Software is not mass produced (like machines)
  - replication (manufacturing) is almost effortless
- Software is developed
  - the description of the solution is the product
  - Non-deterministic, creative process due to human involvement
  - Controllable in a probabilistic manner only
  - Defects come from development (not from production)

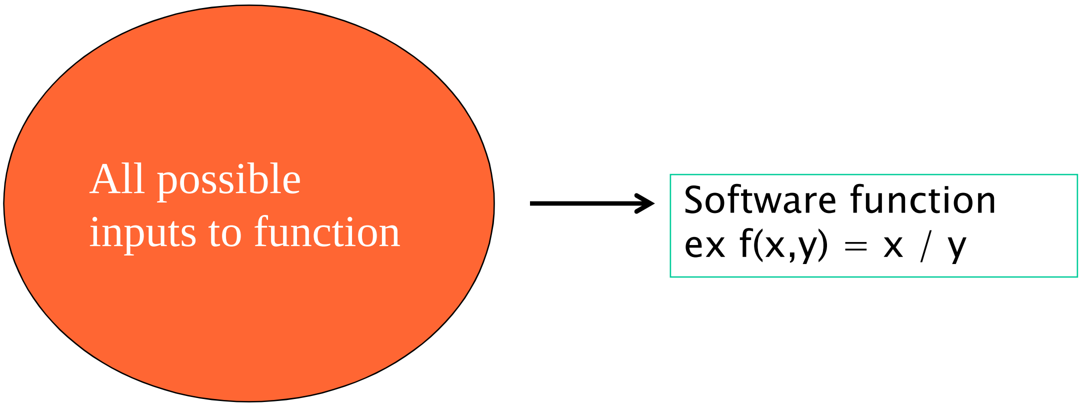
Software ages

Software does not break as it ages

Failures do not occur due to material fatigue (as with hardware)

hardware reliability concepts don't work

but due to the execution of logical faults, and these faults may appear with time



1 data input, out of millions, causes defect in function

Defects are in function since time 0, but they will appear only when the data input will be used, giving ‘ageing’ feeling

The Intel Pentium case

- Pentium P5, 1994
  - Defect in division algorithm of floating point unit (missing elements in look up table)
  - Only few sequences of input revealed the defect (one out of 9 billion)

Software ages (2)

Software cannot be perfect at the beginning

All software faults may not be removed before release

Besides, changes to software (requirements changes, platform changes, defect corrections) may introduce other defects

Software engineering

- Software engineering
  - Multi person construction of multi version software [Parnas]
  - Not 'solo programming'

Solo programming

- Size: small
  - One person can do it
- Developer is the user
  - No communication problems
- Lifespan: short
- Cost: limited (free)
- Properties: functional

Software engineering

- Size: large
  - Teams, documentation, communication and coordination problems
  - Modules and structure
- User is not the developer
  - 3rd party requirements, communication problems
- Lifespan: long (no ageing)
- Cost: development + operation/maintenance
- Properties: functional and not functional

Functional vs. non functional

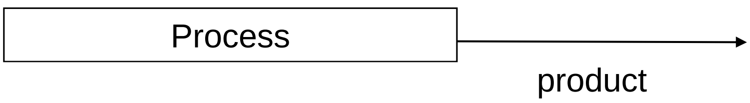
- Functional characteristics of software
  - "Add two integer numbers"
- Non functional properties
  - User interface usable by not computer expert
  - Precision
    - relative error < 10^{-9}
    - absolute error < 10^{-8}
  - Reliability
    - sum must be correct 99,999999% times
  - Performance, efficiency
    - Sum must be performed < 0,01 millisec
    - Sum must use <10 kbytes ram memory

Functional vs. non functional

- Non functional properties sometimes harder to express
- Harder to design into software
  - They are *emerging* properties
    - Depend on the whole system, i.e. reliability, performance

Process and product

Process and product



- Process: activities, people, tools
- Products: documents, data, code
- The quality of the product depends on the quality of the process
- The process depends on the product scenario

Process & product properties

- Process properties
  - Cost
  - Effort
    - Hours worked
  - Punctuality

Process & product properties

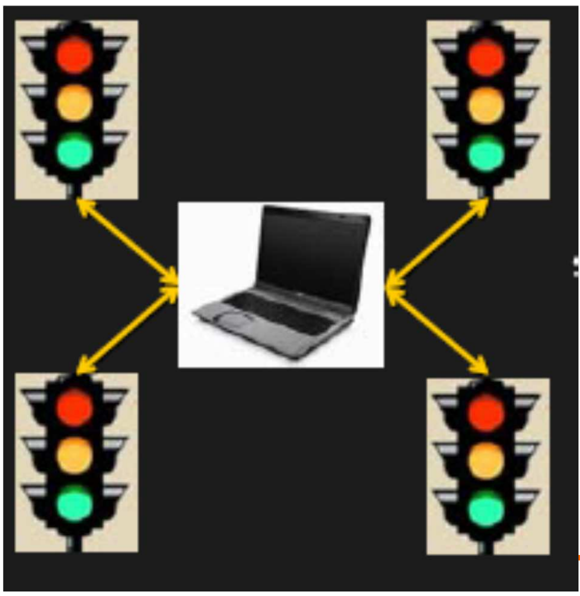
- Product properties (ISO 25010 – ex ISO 9126)
  - Functionality
  - Correctness
  - Reliability
  - Performance

Process & product properties

- Product properties
  - Safety
  - Robustness
  - Usability
  - Security
  - ..

Functionality

- Set of functions that satisfy stated or implied needs



- Ex. control 4 traffic lights in a road crossing so that ..
  - Green in one direction, red in other direction during x sec
  - Flashing yellow in one direction during y sec, red in other direction
  - Red in one direction during z sec, green in other direction

Correctness

- Capability of the product to provide the intended functionality in all cases
  - Ex. the intended sequence of signals is always satisfied

Reliability

- The ability of a system or component to perform its required functions under stated conditions for a specified period of time.
  - The intended sequence of signals is satisfied with high probability (ex P = 99.9%) during a year
    - Or, there is 1 failure every year

Safety

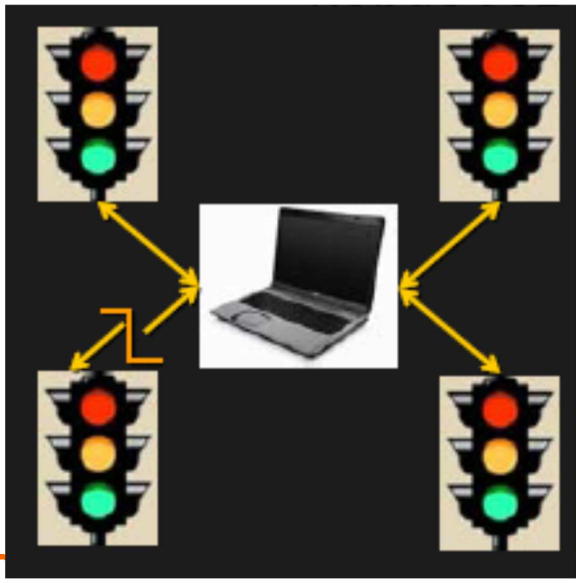
- Capability of avoiding hazards
  - Ex. f1 Never allow green in both directions
  - Ex. F2 Red light broken
  - F3 Red in all directions

Performance

- Time: speed/delay to perform a function
- Space: memory required to perform a function

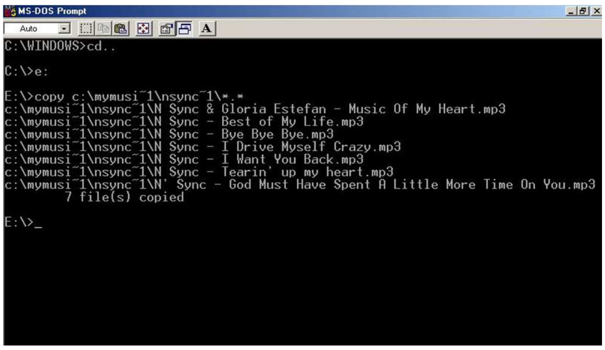
Robustness

- Capability of providing a reduced functionality in adverse conditions



- In case of broken cable the system provides a safe behavior
  - All red
  - All flashing yellow

Usability



- Ease of use of a function
  - Effort needed to use the product
  - Assessment by the user about using the product

Software engineering

- Principles, techniques, methods
- To guide the development and maintenance of software
- With defined process and product attributes

Process

See chapter

Principles

Principles

- Fundamental, broad coverage ideas, capable of producing positive, useful effects
  - Separation of concerns
  - Abstraction
  - Modularity

Separation of concerns

- Given a large, difficult problem, try to split it in many (independent) parts, consider a part at a time
  - In war: divide and conquer
  - In SE: software process, concentrate on what the system should do, then on how, then do it

Abstraction

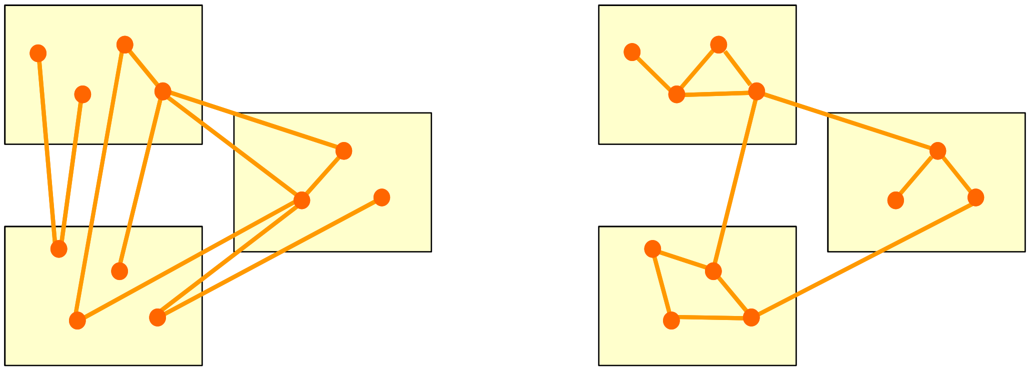
- Given a difficult problem/system, extract a simpler view of it, avoiding unneeded details
- Then reason on the abstract view (model)

Abstraction

```
package Computer;
public class Slot {
public String slotID;
private Component component = null;
    public Slot
(String _slotID,
boolean _installed,
boolean _required,
 Component _component
 ) {
slotID = _slotID;
installed = _installed;
required = _required;
component = _component;
    }
public void bind(Componentc) {
component = c;
    }
    public void unbind() {
component = null;
    }
    public booleanisBound() {
return (component != null);
    }}
```

Modularity

- Divide a complex system in modules, with high cohesion and low coupling



high coupling　　　　low coupling

SoftEng http://softeng.polito.it

Information hiding

- In complex systems, each module should hide to others as many details about its internal mechanisms/design choices, as possible
  - Another form of ‘high cohesion low coupling’

SoftEng http://softeng.polito.it

SE approaches

SE in one slide

- Activities
  - Production, VV, management
- Documents (and code)
  - To share and control information, decisions
- Techniques
  - To support activities
- Languages
  - To write documents (UML), code
- Models
  - To guide, support activities and the whole
  - CMM and CMM-I, ISO 9000-3, ISO 15504, ISO 12207, ISO 9126, IEEE, ..

Approaches

There are many different ways of putting everything together

But at least 3 approaches can be recognized

Three basic approaches to SE

Cow boy programming

Just code, all the rest is time lost and real programmers don't do it

1. Document based, semiformal, UML

   Semiformal language for documents (UML), hand (human) based transformations and controls

2. Formal/model based

   Formal languages for documents, automatic transformations and controls

3. Agile

   Limited use of documents, emphasis on code and tests

Approaches, diffusion

Cow boy programming

Not un-applied ..

1. Document based, semiformal, UML

   Standard industrial practice, especially on large projects and mature companies/domains

2. Formal

   Limited application in critical domains, small part of projects, does not scale up in large projects

3. Agile

   Latest approach, debated, limited but increasing usage

Approaches

This course is focused on approach 1

Specific lectures on approach 2 and 3

Recent trends in SE

Trends – development

- Component based SE
  - Buy + integrate vs. build
  - Open source or commercial
- Offshoring
- Outsourcing
- Agile

Trends – business models

ASP – pay per use

software is run on the provider's machines. Users use it through a network (Internet or Extranet). Users pay for using the software rather than purchasing it. E.g., mySAP.com.

Freeware and pro versions

a light version of the software is distributed free of charge. The professional version is charged. E.g., RealPlayer.

Shareware: software is distributed freely to facilitate trial use. Users pay for it if they decide to keep it and use it. E.g.,WinZIP.

Adware: the software is free. The interface show advertisement banners refreshed via Intenet. E.g., Eudora

Summary

- Software development is an important part of the economy, software is pervasive and a key factor in innovation and growth
- Software is not only computer programs
- Software engineering considers techniques and methods to develop large, long lived software, with many users

Summary

- Software is characterized by its function, its correctness, reliability, usability
- Key guiding principles are separation of concerns, abstraction, modularity