Introduction

SOftEng
http://softeng.polito.it

Outline

- Definition and concepts
- Process and product properties
- Principles

Definitions and concepts

Software

- **Software is a collection of computer programs, procedures, rules, associated documentation and data.**
  - software development is more than merely the development of programs
  - software incorporates documents describing various views for various stakeholders (e.g. users, developers)
- **For a given problem, software is approximately 10 times more expensive to produce than a simple program [Brooks75: The Mythical Man Month]**
  - Average: 10 to 50 LoC per person day
  - About 7 LoC in critical systems

Software – types

- embedded
  - ABS, digital camera
- mass market (consumer software)
  - word processor, whatsapp, FB, ..
- process support (enterprise software)
  - production process (things): industrial automation
  - business process (information): management automation

Software – criticality

- safety critical
  - aerospace, military, medical, ..
- mission critical
  - 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

- cell phone, printer driver: 1M Lines of code
- cellular network, operating system: several Millions

- software systems are probably the most complex human artifacts

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]
- A bought a premium-class automobile recently, "it probably contains close to 100 million lines of software code," [4]

[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

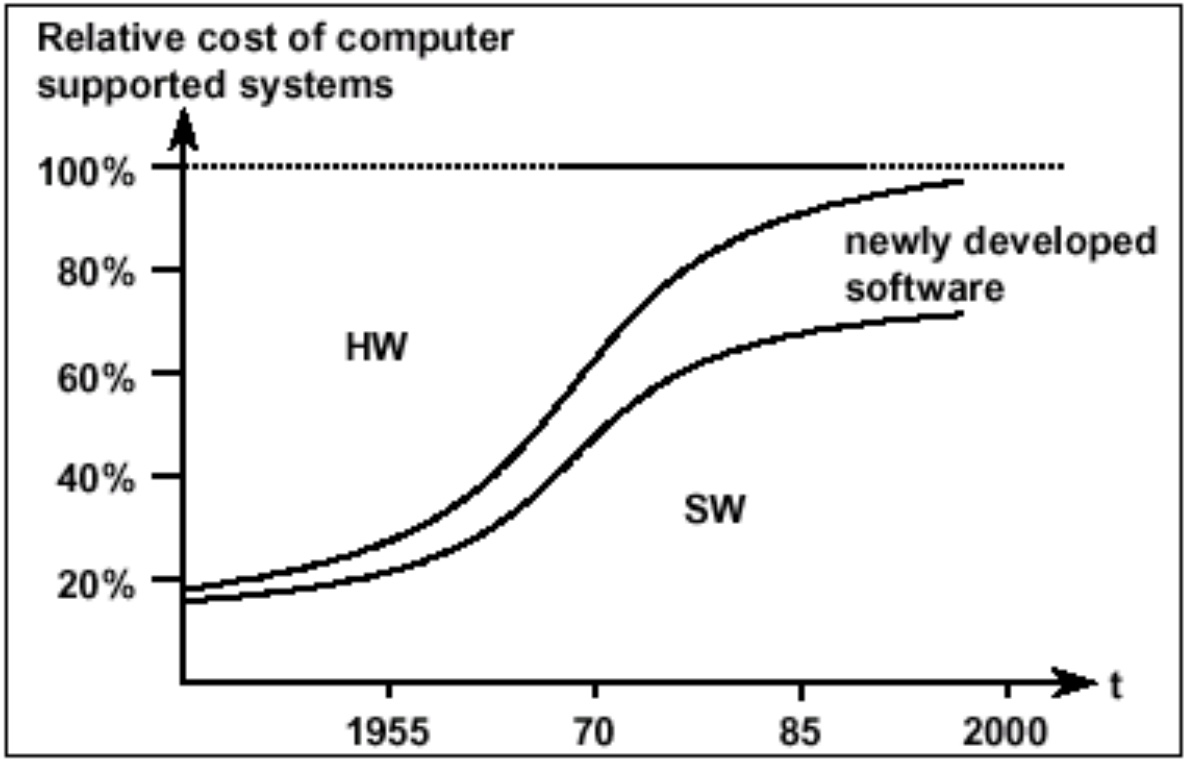
- Software is free
- Software is soft
- Software is produced
- Software ages

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) but due to the execution of logical faults
  - hardware reliability concepts don't work

Software cannot be perfect

- All software faults may not be removed before execution

Software is not stable

- Software changes due to requirements changes, platform changes (and defect corrections)

Typical software problems

- Too expensive (up to a factor of 10).
- Delivered too late (up to a factor of 2).
- Does not live up to user expectations (e.g., reliability)

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

SE issues

- Large →team based development
  - communication and coordination between team members
- Long lifespan
  - Communication developers – maintainers
- Third party requirements
  - Communication non-computer specialist – computer specialist
  - Non functional properties essential

Functional vs. non functional

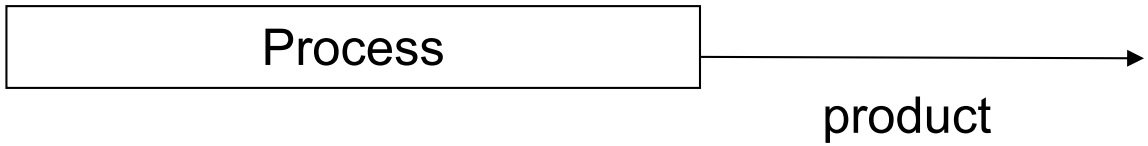
- Functional characteristics of software
  - “Add two integer numbers”
- Non functional properties
  - User interface usable by not computer expert
  - Precision
    - relative error $< 10^{-9}$
    - absolute error $< 10^{-8}$
  - Reliability
    - sum must be correct 99,999999% times
  - Performance, 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

Process & product properties

- Process properties
  - cost
  - effort
  - punctuality

Process & product properties

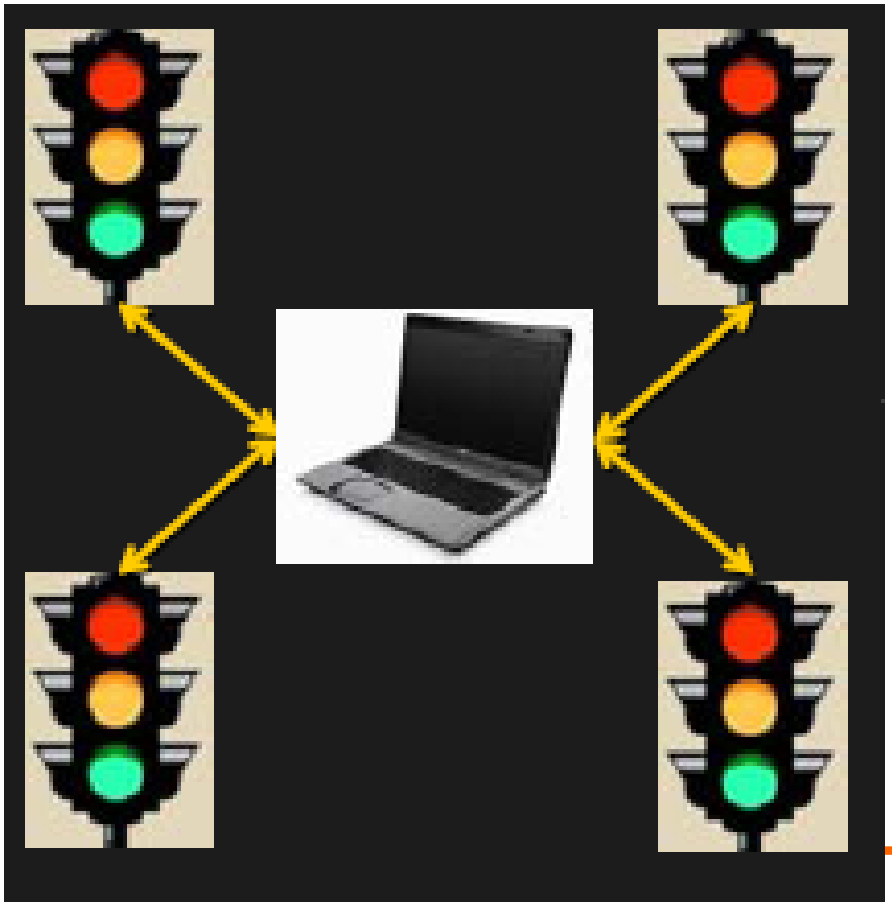
- Product properties
  - Functionality
  - Correctness
  - Reliability
  - Performance

Process & product properties

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

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. Never allow green in both directions

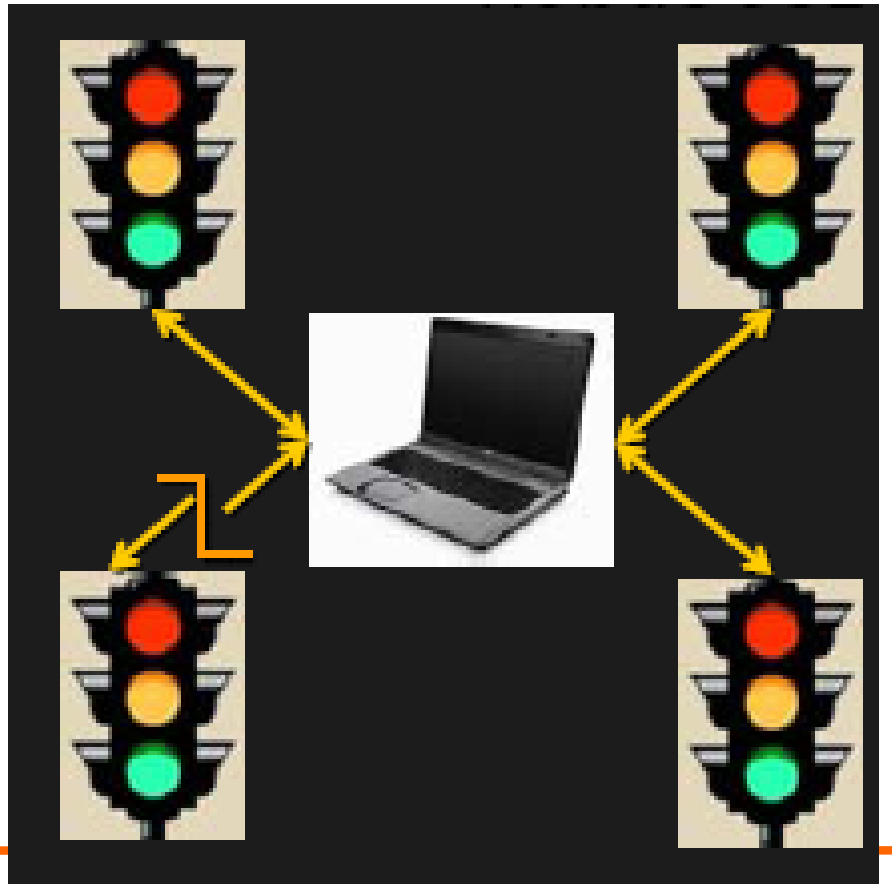
Performance

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

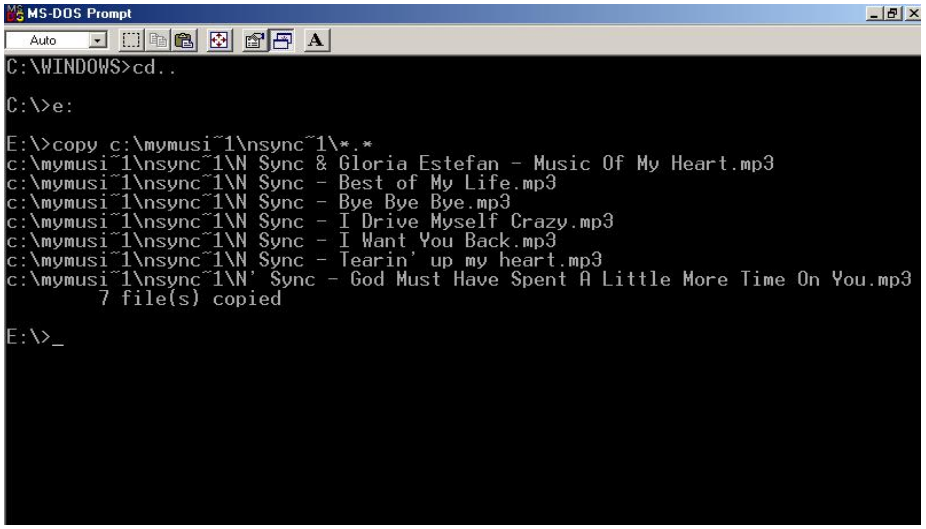
Robustness

- 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

Principles

Principles

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

Separation of concerns

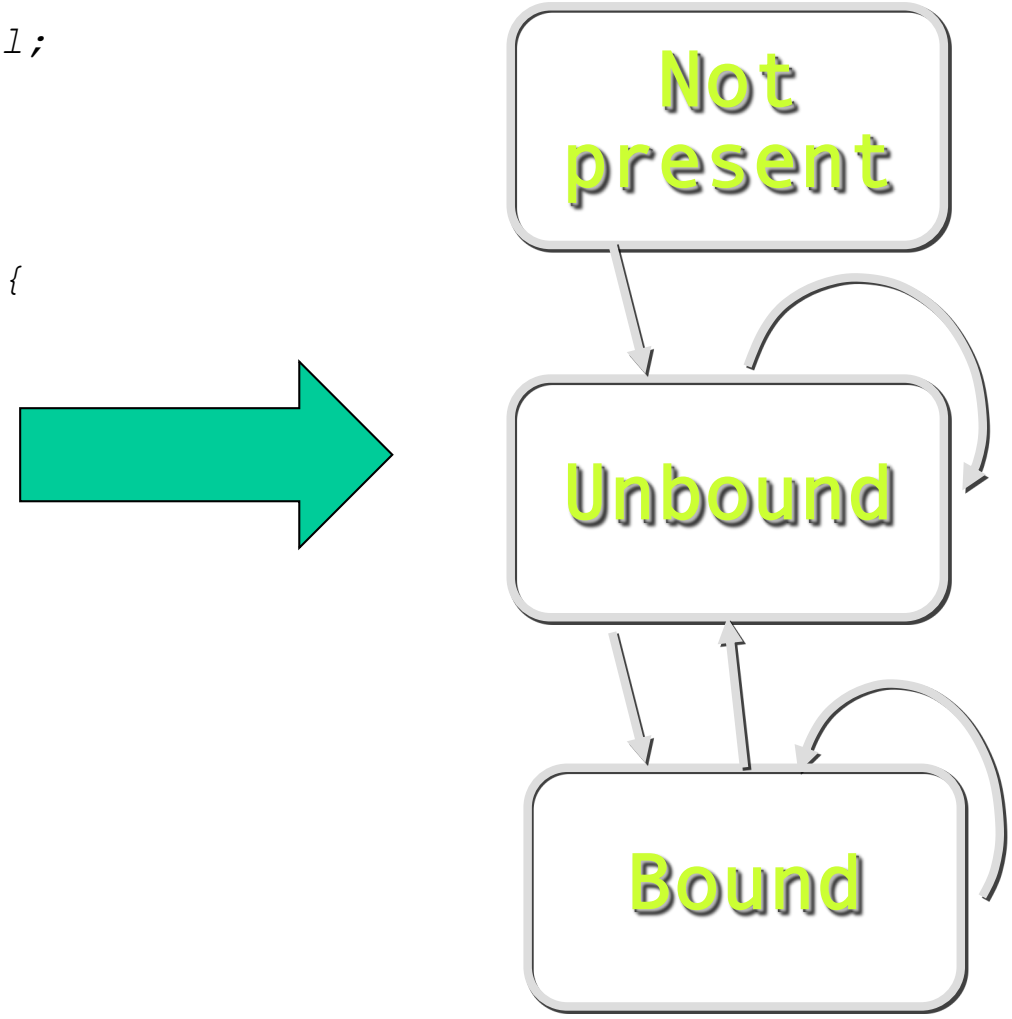
- Given a large, difficult problem, try to split it in many (independent) parts, consider a part at a time
  - In war: divide and conquer
  - In SE: software process, concentrate on what the system should do, then on how, then do it

Abstraction

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

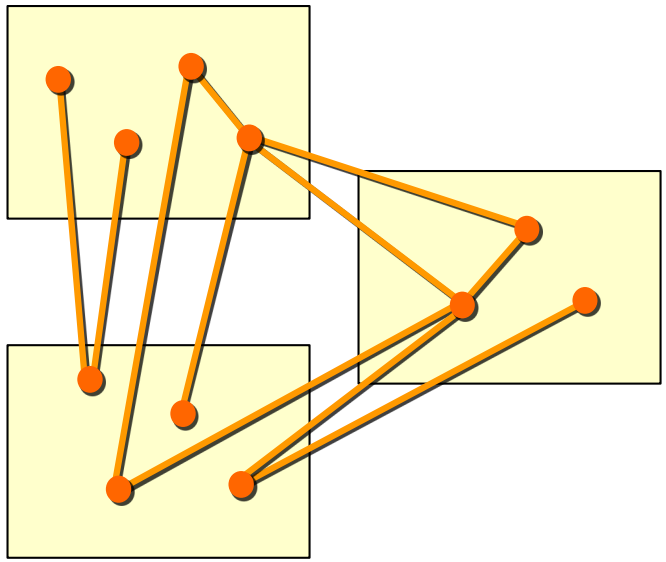
Abstraction

```
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(Component c) {
        component = c;
    }
    public void unbind() {
        component = null;
    }
    public booleanisBound() {
        return (component != null);
    }
}
```

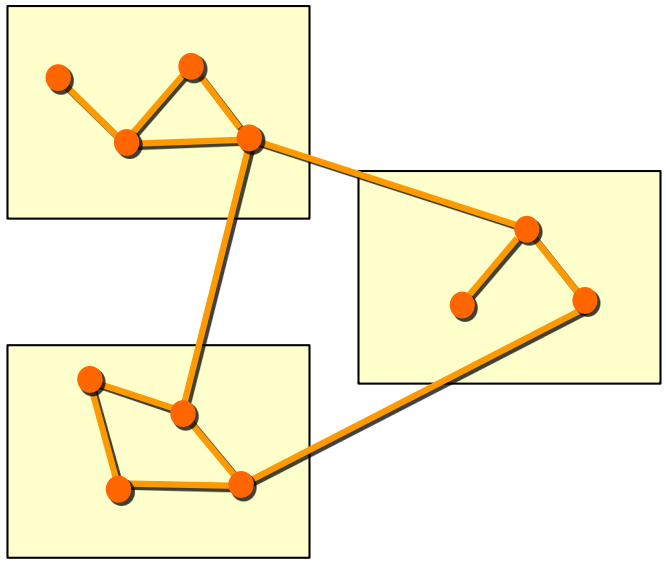


Modularity

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



high coupling



low coupling

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’

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