Politecnico
di Milano


# Software Engineering

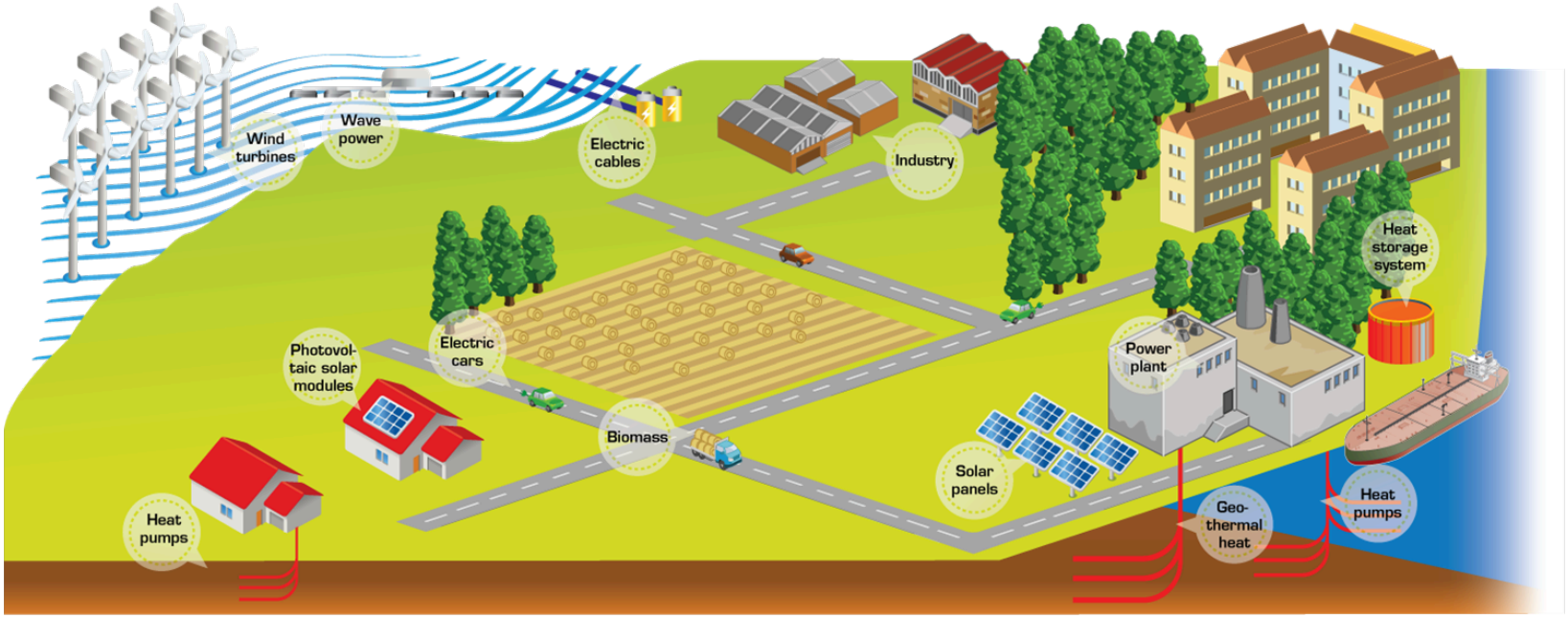
Definitions
History
The process and product
Phases of the development process

# Why software engineering is important


- Software is everywhere and our society is now totally dependent on software-intensive systems
- Society could not function without software

# Transport systems

# Energy systems



# Manufacturing systems

# Bahrain Grandprix April 2018

https://www.thenational.ae/sport/f1/injured-ferrari-mechanic-gives-thumbs-up-from-hospital-after-bahrain-grand-prix-accident-involving-kimi-raikkonen-1.720005

## Injured Ferrari mechanic gives thumbs up from hospital after Bahrain Grand Prix accident involving Kimi Raikkonen

Francesco Cigarini suffered broken tibia and fibula in his left leg in botched second pit-stop during Sunday's race

Agence France-Presse
April 9, 2018
*Updated: April 9, 2018 02:26 PM*

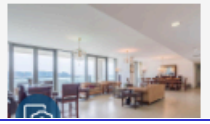
4 shares

**EDITOR'S PICKS**

*EDUCATION*
**10@10 podcast: Education in the UAE with Mark Leppard**

*MENA*
**Displaced Iraqis return to destruction and death ahead of elections**

*ENERGY*
**Saudi Arabia's first wind project receives four bids**



*LUXURY*
**Inside a Dh7.5m penthouse at Al Raha Beach - in pictures**

# The reason for the incident

https://www.motorsport.com/f1/news/ferrari-explains-error-that-injured-mechanic-in-bahrain-1025494/

By: Scott Mitchell, Journalist

2018-04-13

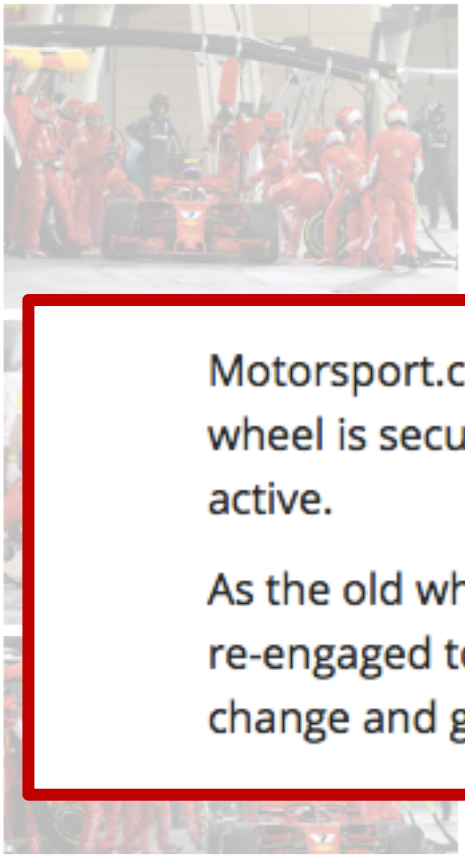
GET ALERTS

**Ferrari has revealed a sensor confusion triggered the unsafe release of Kimi Raikkonen's car that led to a pit crew member suffering a broken leg during the Bahrain Grand Prix.**

A problem removing the left-rear during Raikkonen's second pitstops delayed Francesco Cigarini from fitting a new tyre, and he was standing in front of the wheel when the Finn was given the green light to leave the pit box.

Cigarini was hit by Raikkonen's car as the Finn pulled away, and required surgery after sustaining a fractured shinbone and fibula.

# The reason for the incident



By: Scott Mitchell, Journalist

2018-04-13

GET ALERTS

Ferrari has revealed a sensor confusion triggered the unsafe release of Kimi Raikkonen's car that led to a pit

Motorsport.com understands the system is designed to check whether a wheel is securely fitted and whether the wheel gun has been sufficiently active.

As the old wheel was not removed, and the gun was disengaged and then re-engaged to try again, the system registered it as a completed wheel change and gave Raikkonen the green light.

# 911 outage on April 2014

https://www.theatlantic.com/technology/archive/2017/09/saving-the-world-from-code/540393/


- 911 is the phone number for emergency service in USA
  - Equivalent to 112 in Italy
- During the night of the 10th of April, the population of Washington State had no 911 service for six hours
  - People calling were getting the busy signal
  - A woman called more than 37 times while a stranger was breaking into her house
  - Servants at the central office were not aware of the problem
- Why did this happen?
  - The software dispatching the calls had a counter used to assign a unique identifier to each call.
  - The counter went over the threshold defined by developers…
  - All calls from that moment on were rejected

# Why did these problems happen?

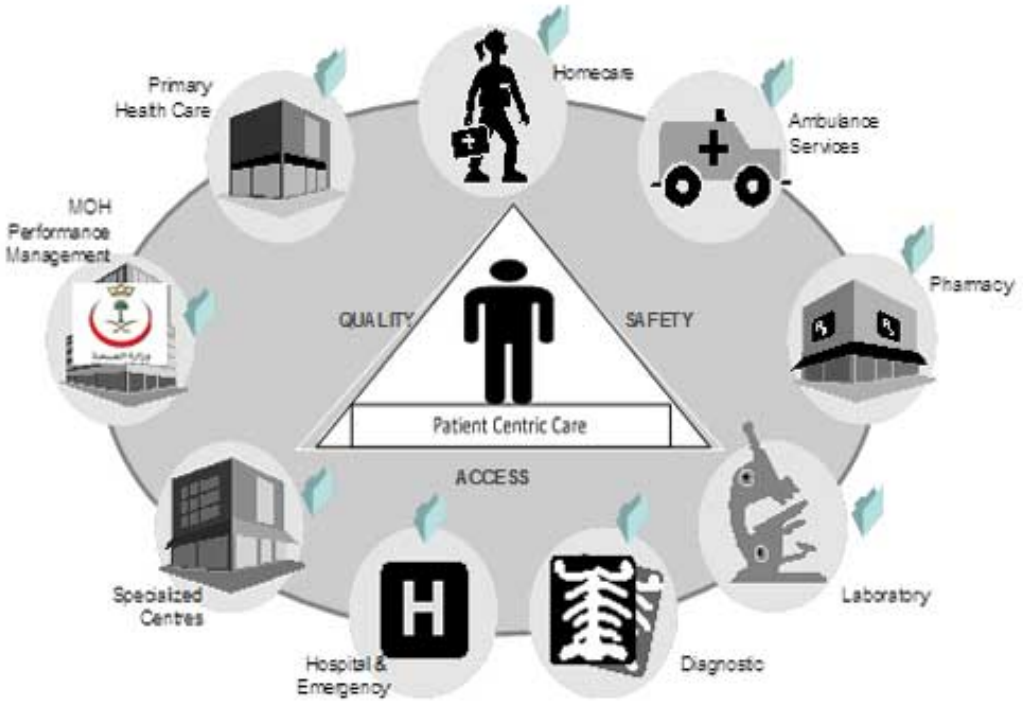

- <Intentionally left black for your reflection…>

# What do we need?

- <Intentionally left black for your reflection…>

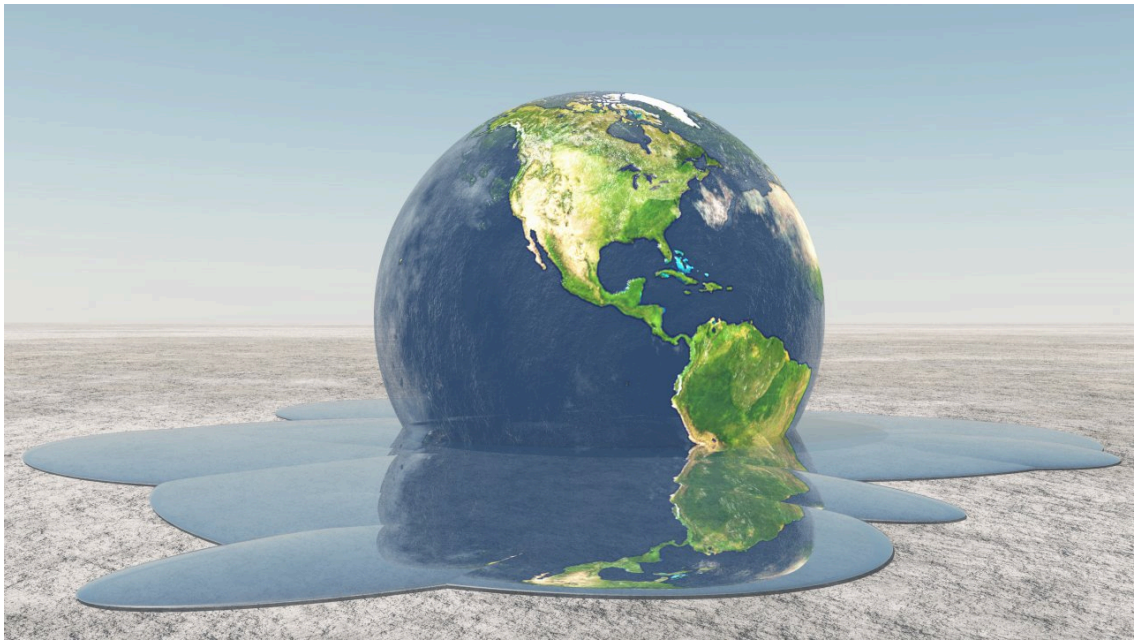
# New challenges

- More advanced software is required to address new challenges
  - Increasing number of elderly people



# New challenges

- More advanced software is required to address new challenges
  - Increasing number of elderly people
  - Climate change



# New challenges

- More advanced software is required to address new challenges
  - Increasing number of elderly people
  - Climate change
  - Combating international terrorism

# New challenges


- More advanced software is required to address new challenges
  - Increasing number of elderly people
  - Climate change
  - Combating international terrorism

- We need software engineering to help manage the complexity of new software-intensive systems, ensure that these systems are reliable and secure and meet the needs of their users
- Saving the World from Code (The Atlantic): https://www.theatlantic.com/technology/archive/2017/09/saving-the-world-from-code/540393/

# Software engineering: definitions


- Field of computer science dealing with software systems
  - large and complex
  - built by teams
  - exist in many versions
  - last many years
  - undergo changes
- Multi-person construction of multi-version software

# Software engineering: definitions


- Systematic approach to development, operation, maintenance, deployment, retirement of software
- Methodological and managerial discipline concerning the systematic production and maintenance of software products that are developed and maintained within anticipated and controlled time and cost limits

# Software engineering: definitions


- Deals with cost-effective solutions to practical problems by applying scientific knowledge in building software artifacts in the service of the human race
  - *cost-effective*
  - *practical problems*
  - *scientific knowledge*
  - *building things*
  - *service of the human race*

Engineering!

# Software engineer: required skills


- **Programming skills not enough**
  - programmer
    - develops a complete program
    - works on known specifications
    - works individually
  - software engineer
    - identifies requirements and develops specifications
    - designs a component to be combined with other components, developed, maintained, used by others; component can become part of several systems
    - works in a team

# Skills of software engineers


- Technical
- Project management
- Cognitive
- Enterprise organization
- Interaction with different cultures
- Domain knowledge

- The quality of human resources is of primary importance

# History: initial situation


- Software is art
- Computers used for “computing”
  - mathematical problems
  - designers = users
  - no extended lifetime
- The art of “programming”
  - low-level languages
  - resource constraints (speed and memory)

# From art to craft


- From computing to information management
- Requests for new (custom) software explode
  - users ≠ designers
  - EDP centers and software houses
- New high-level languages
- First large projects and first fiascos
  - time and budget, human cooperation failures
  - wrong specifications

# From craft to industrial development


- Term “software engineering” defined in a NATO conference in Garmisch, Oct. 1968
- Focuses of software engineering
  - Development methods and standards
  - Planning and management
  - Automation
  - Verifiable quality
  - Componentization
  - ….

# More on fiascos… Therac-25, 1985-87


- A computerized radiation therapy machine used in Canada (6 installations) e in USA (5 installations) for cancer therapy
- A software defect caused massive overdoses of radiations
- At least 4 persons died and others have been seriously injured
- http://sunnyday.mit.edu/therac-25.html

# More on fiascos… Therac-25, 1985-87


- Reasons for the problem
  - The software has been inherited by a previous model (Therac-20) where hardware control modules did not allow quantities of radiations above the threshold to be emitted
  - Hardware control modules have been eliminated in Therac-25
  - The control software has been developed by a single programmer
  - Task synchronization has been developed in an ad-hoc way
  - No test or formal verification, no software documentation


-00:00:17

# More on fiascos… Ariane 5, 1996


- The 4th of June 1996, 40 secs after take off, Ariane 5 broke up and exploded
  - The launcher contained a cluster of satellites for a value of 500M$ (of 1996)
  - The total cost for developing the launcher has been of 8000M$
- The explosion has been caused by a software failure
  - "The failure […] was caused by the complete loss of guidance and attitude information […]. This loss of information was due to specification and design errors in the software of the inertial reference system."
  - "The extensive reviews and tests carried out during the Ariane 5 Development Programme did not include adequate analysis and testing of the inertial reference system or of the complete flight control system, which could have detected the potential failure."
- http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html

# Dimensions of large systems

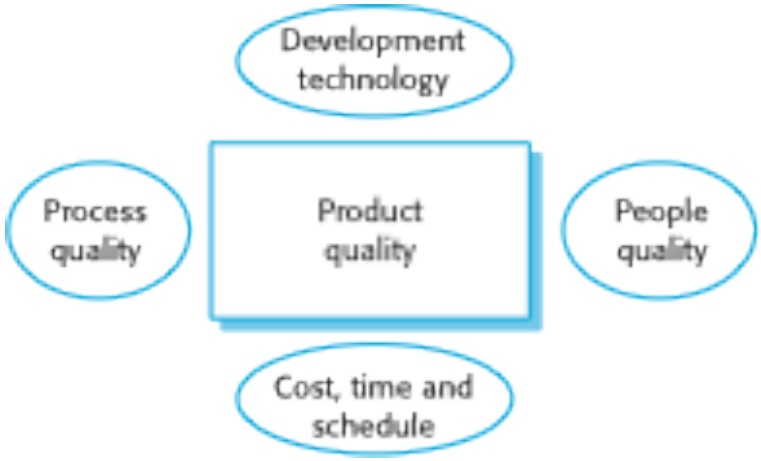

- Examples from Michiel van Genuchten, Towards a Software Factory, Kluwer 1992
  - Space shuttle project
    - 5.6 million code lines, total person year 22k, 1200 Million$
  - CityBank teller machine
    - 780000 code lines, total person year 150, 13.2 Million$
- Debian 5.0 distribution (2009) http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.464.5108&rep=rep1&type=pdf
  - 324 million code lines, estimated total person year 84,000 and cost 6,100 million Euros

# Process and product


- Our goal is to develop **software products**
- The **process** is how we do it
- Both are extremely important, due to the nature of the software product
- Both have qualities
  - in addition, quality of process affects quality of product
  - …even though, other aspects such as the quality of the development team are important as well

# The software product

- Different from traditional types of products
  - intangible
    - difficult to describe and evaluate
  - malleable
  - human intensive
    - does not involve any trivial manufacturing process
- Aspects affecting product quality



# Product quality attributes


- Correctness
  - software is correct if it satisfies the specifications
- Reliability
  - can be defined mathematically as “probability of absence of failures for a certain time period”
- Robustness
  - software behaves “reasonably” even in unforeseen circumstances (e.g., incorrect input, hardware failure)
- Performance
  - efficient use of resources
- Usability
  - expected users find the system easy to use

# Other qualities


- Maintainability
- Reusability
  - similar to maintainability, but applies to components
- Portability
  - similar to maintainability (adaptation to different target environments)
- Interoperability
  - coexist and cooperate with other applications

# Process qualities: productivity

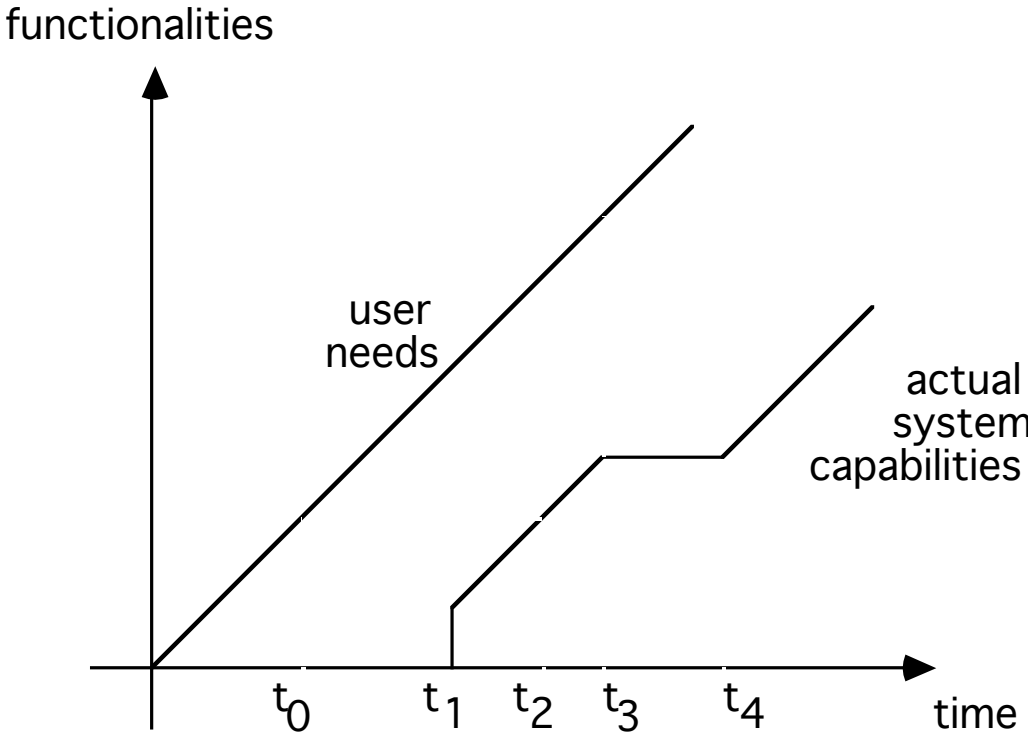

- Productivity
  - how can we measure it?
    - delivered item by a unity of effort
- Unity of effort
  - person month
    - WARNING: persons and months cannot be interchanged
- Delivered item
  - lines of code (and variations)
  - function points

# Productivity: folk data


- From the Brooks book “The Mythical Man-Month” (1975)
  - About 1000 lines of code per year per person
- Looking at numbers in slide 28
  - Space shuttle: 255 lines of code per year per person
  - Citybank: 5200 lines of code per year per person
  - Debian : 3860 lines of code per year per person
- Then
  - extreme variance among individuals
  - extreme variance with group dynamics
    - Brooks “law”: “adding people to a late project makes the project late(er)”

# Process qualities: timeliness

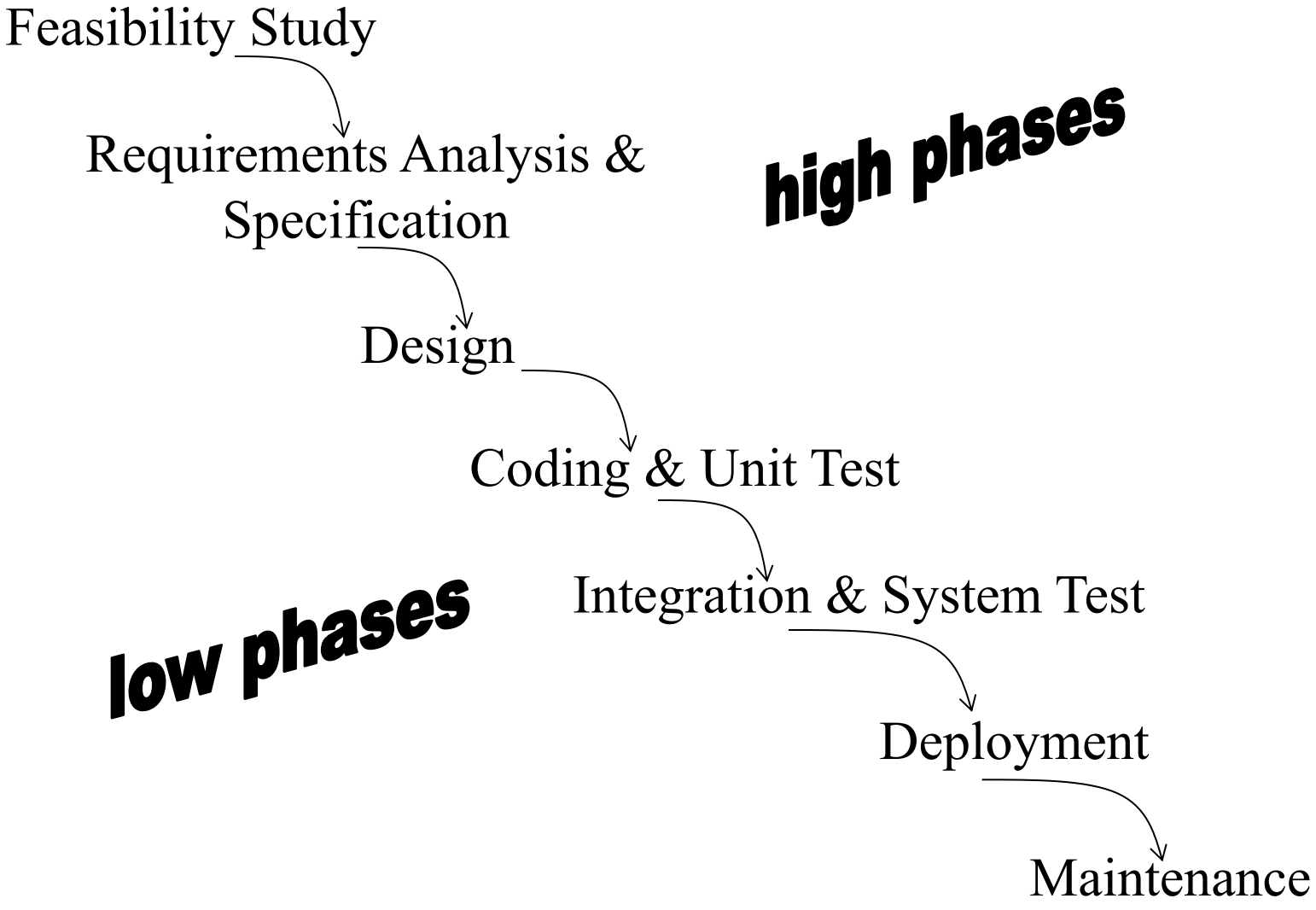
- Ability to respond to change requests in a timely fashion



# Software lifecycles


- Initially, no reference model:
  - code&fix
- As a reaction to the many problems: traditional "waterfall" model
  - identify phases and activities
  - force linear progression from a phase to the next
  - no returns (they are harmful)
    - better planning and control
  - standardize outputs (artifacts) from each phase
  - Software like manufacturing
- Then, flexible processes: iterative models, agile movement, DevOps

# A waterfall organization



# Feasibility study & project estimation


- Cost/benefits analysis
- Determines whether the project should be started (e.g., buy vs make), possible alternatives, needed resources
- Produces a **Feasibility Study Document**
  - Preliminary problem description
  - Scenarios describing possible solutions
  - Costs and schedule for the different alternatives

# Req. analysis and specification


- Analyze the domain in which the application takes place
- Identify requirements
- Derive specifications for the software
  - Requires an interaction with the user
  - Requires an understanding of the properties of the domain
- Produces a **Requirements Analysis and Specification Document (RASD)**

# Design


- Defines the software architecture
  - Components (modules)
  - Relations among components
  - Interactions among components
- Goal
  - Support concurrent development, separate responsibilities
- Produces the **Design Document**

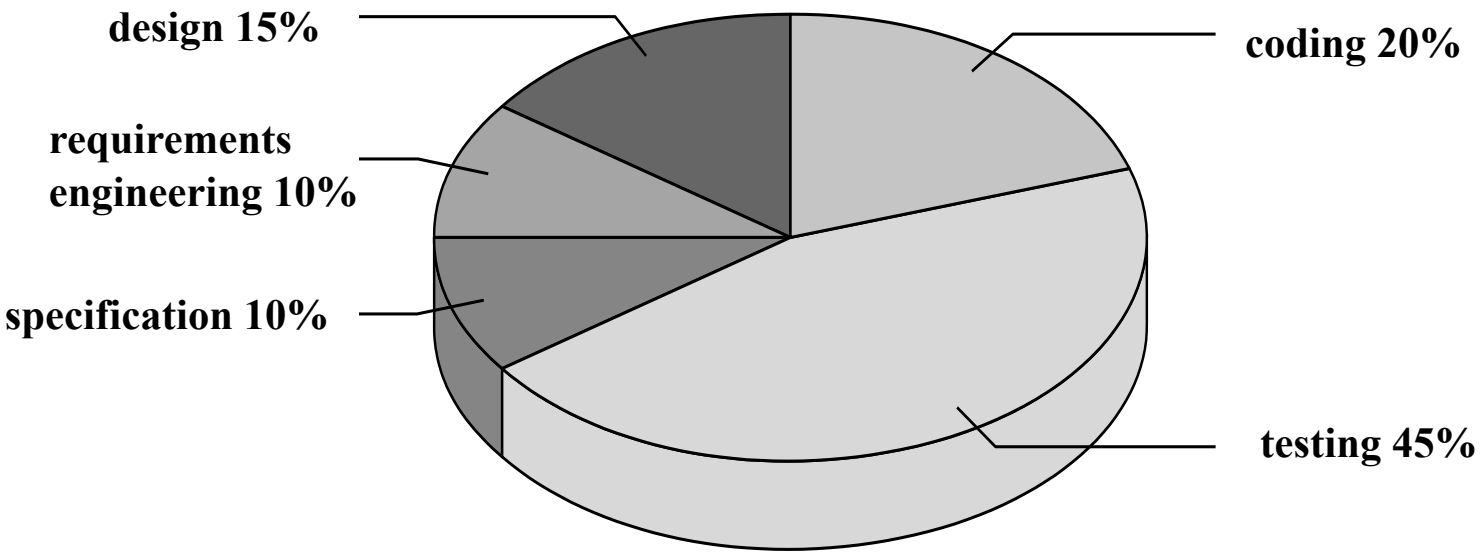
# Coding&Unit level quality assurance


- Each module is implemented using the chosen programming language
- Each module is tested in isolation by the module developer
- Inspection can be used as an additional quality assurance approach
- Programs include their documentation

# Integration&System test


- Modules are integrated into (sub)systems and integrated (sub)systems are tested
- This phase and the previous one may be integrated in an incremental implementation scheme
- Complete system test needed to verify overall properties
- Sometimes we have ***alpha test*** and ***beta test***

# Effort distribution (van Vliet 2008)



# Maintenance


- **Corrective** maintenance: deals with the repair of faults or defects found ≈ 20%
- Evolution
  - **adaptive** maintenance: consists of adapting software to changes in the environment (the hardware or the operating system, business rules, government policies…) ≈ 20%
  - **perfective** maintenance: mainly deals with accommodating to new or changed user requirements ≈ 50%
  - **preventive** maintenance: concerns activities aimed at increasing the system's maintainability ≈ 5%

# Correction vs evolution


- Distinction can be unclear, because specifications are often incomplete and ambiguous
- This causes problems because specs are often part of a contract between developer and customer
  - early frozen specs can be problematic, because they are more likely to be wrong

# Problems of software evolution

- It is (almost) never anticipated and planned; this causes disasters
- Software is very easy to change
  - often, under emergency, changes are applied directly to code
  - inconsistent state of project documents

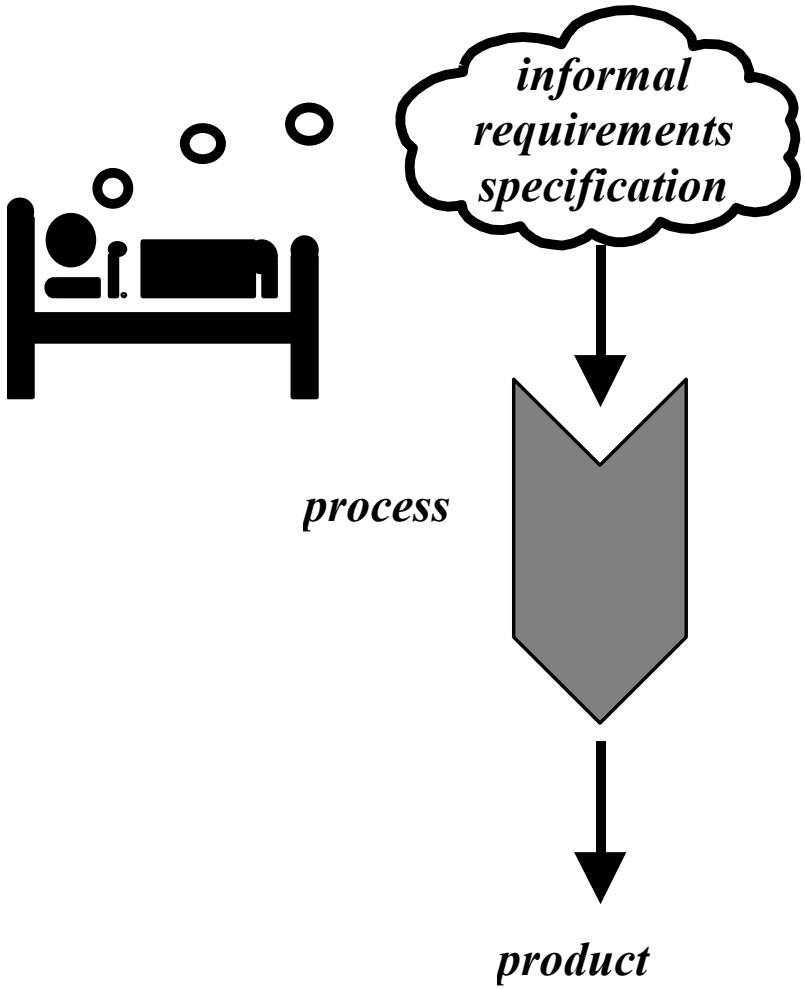
# How to face evolution


- Good engineering practice
  - first modify design, then change implementation
  - apply changes consistently in all documents

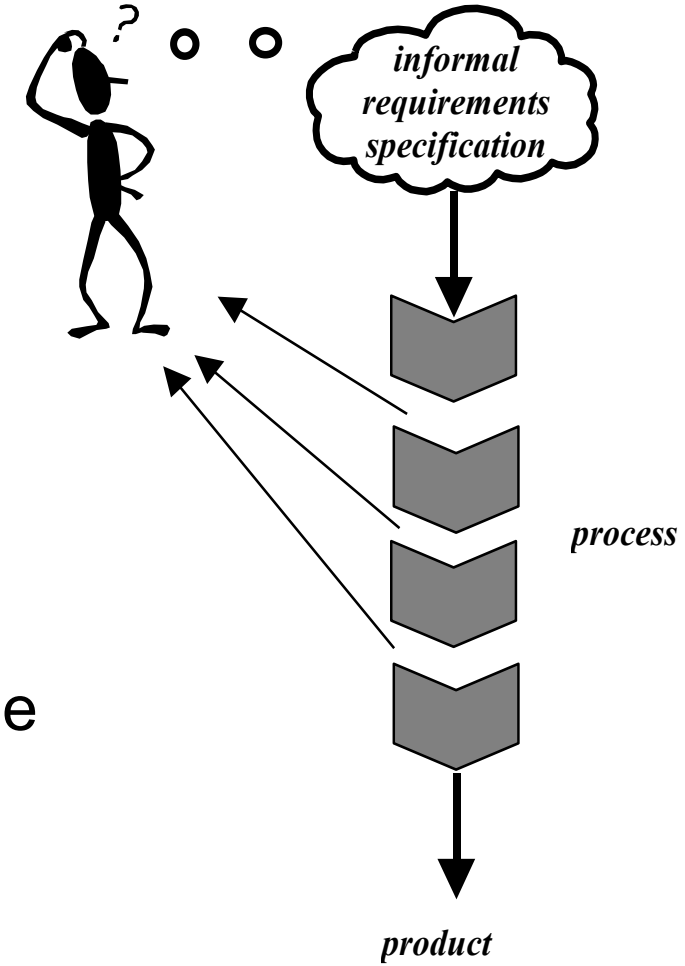
- Likely changes must be anticipated
- Software must be designed to accommodate future changes reliably and cheaply

*This is one of the main goals of software engineering*

# Waterfall is “black box”

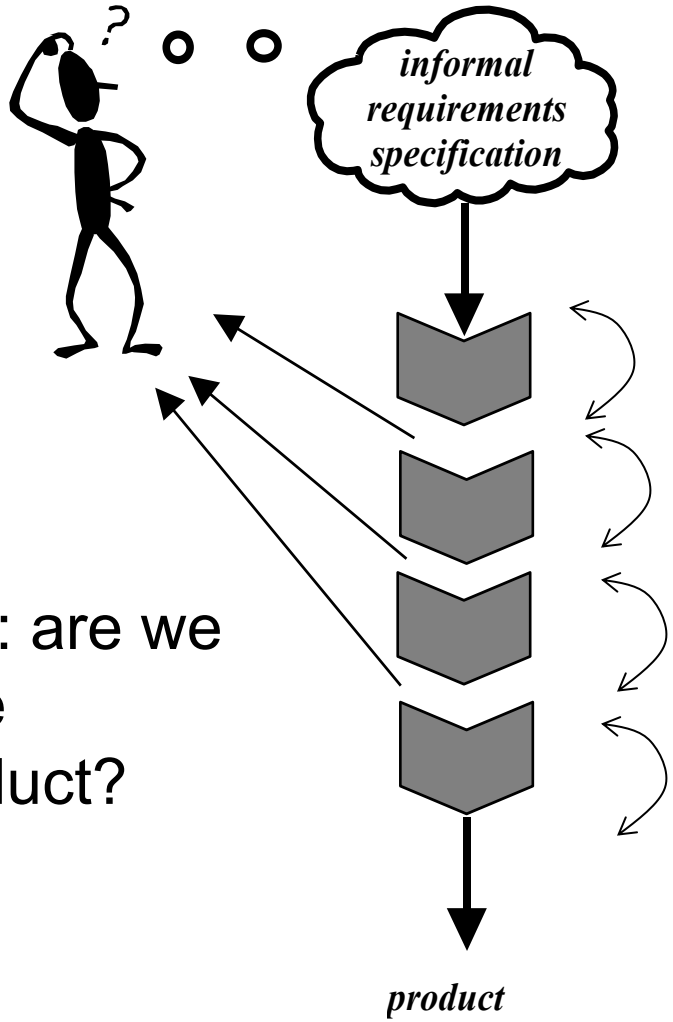


# Need for transparency




- Transparency allows early check and change via feedback
- It supports flexibility

# Verification and validation




- **Validation**: are we making the ”right” product?
- **Verification:** are we doing the product right?

# Flexible processes


- Adapt to changes, in particular in the requirements and specification
- The idea is to have incremental processes and be able to get feedback on increments
- Exist in many forms
  - SCRUM
  - Extreme Programming
  - Incremental releases and rapid prototyping
  - DevOps
  - …
- Also because of new deployment scenarios
  - Web-based applications
  - Mobile “apps”
  - ...


POLITECNICO
MILANO