# The software process

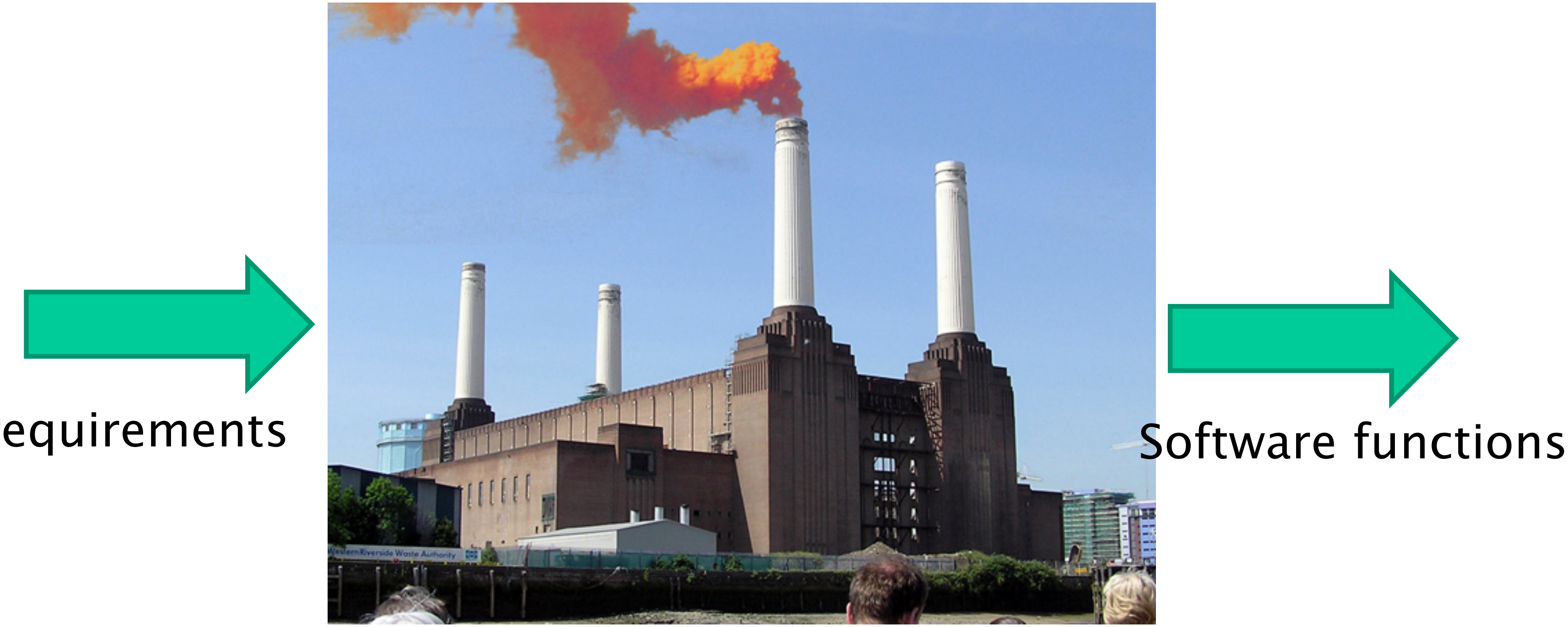
SoftEng

http://softeng.polito.it

# Outline

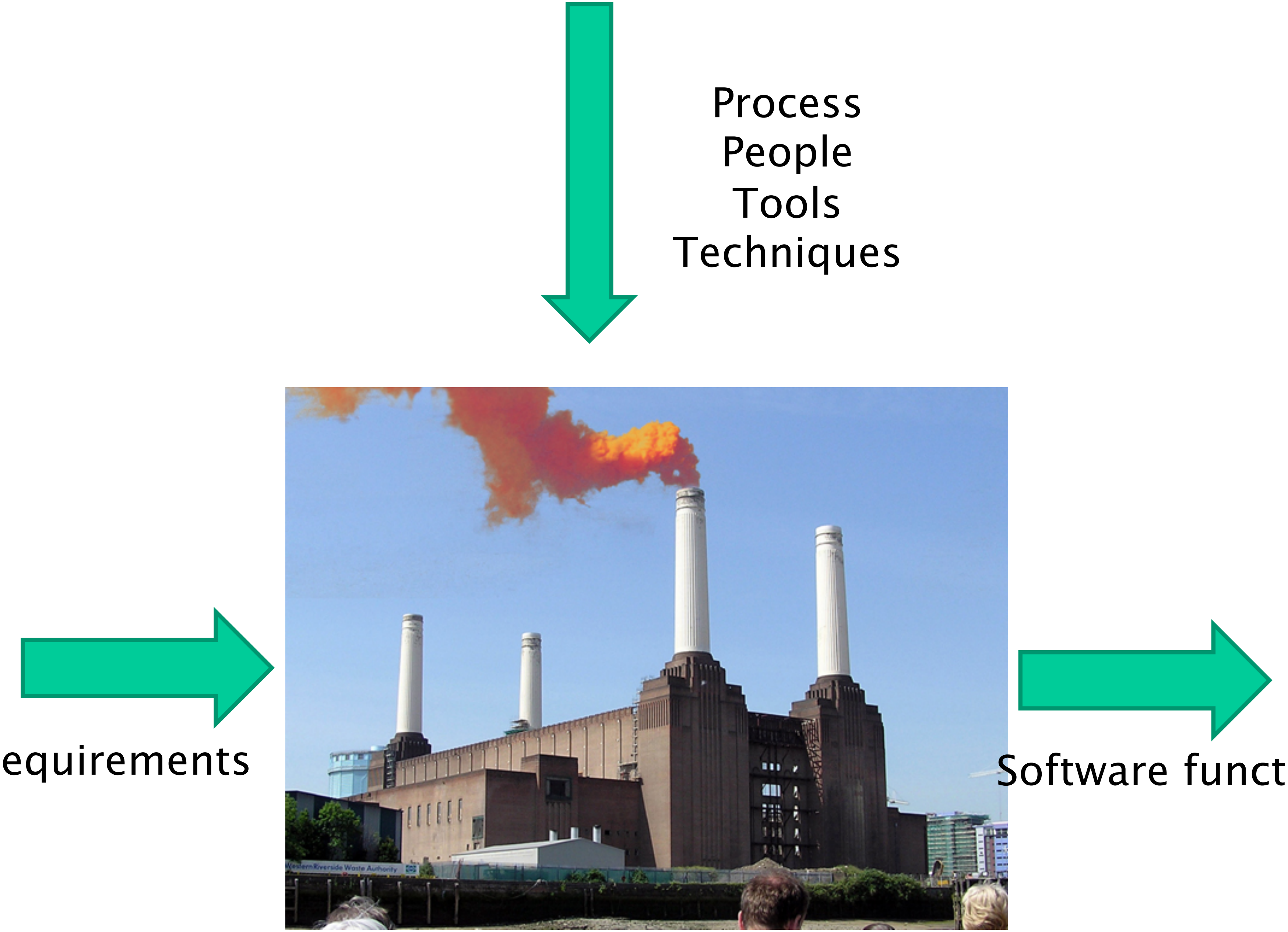
- Activities
  - Production (requirements, design, implementation), verification, management
- Phases
  - Development, operation, maintenance
- Comparison with traditional engineering
- System and Software process
- SE approaches
- Recent trends

# Software engineering



requirements

Software functions



Process
People
Tools
Techniques

requirements

Software functions

# Activities

# Goal

Produce software

- documents, data, code

with defined, predictable process properties

- cost, duration

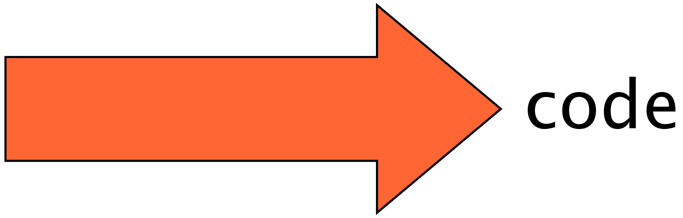
and product properties

- functionality, reliability, ..

# How to achieve the goal?

# From the bottom up

- We need the final thing
  - Executable code
- But we do not write the executable
  - Source code



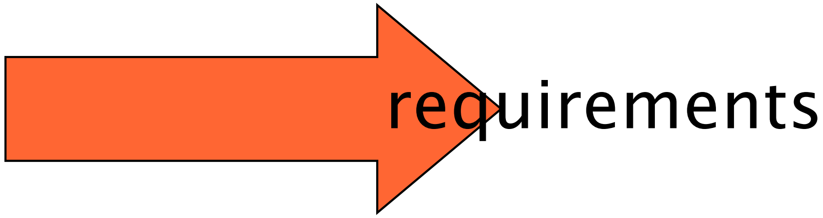
code

- But the source code is large
  - Several physical units
    - Files and directories
  - Several logical units
    - Functions
    - classes
    - Packages
    - Subsystems

→ design

- So, what units? How do we define and organize them?

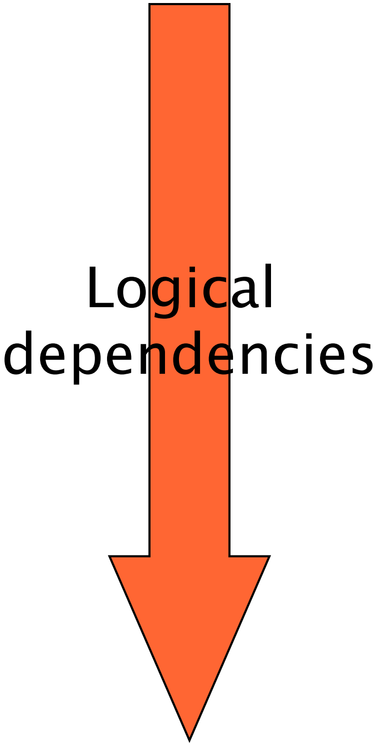
- But, exactly, what the software should do?
  - Add numbers, count cars, forecast weather, control mobile phone, support administration of company?



requirements

# The production activities

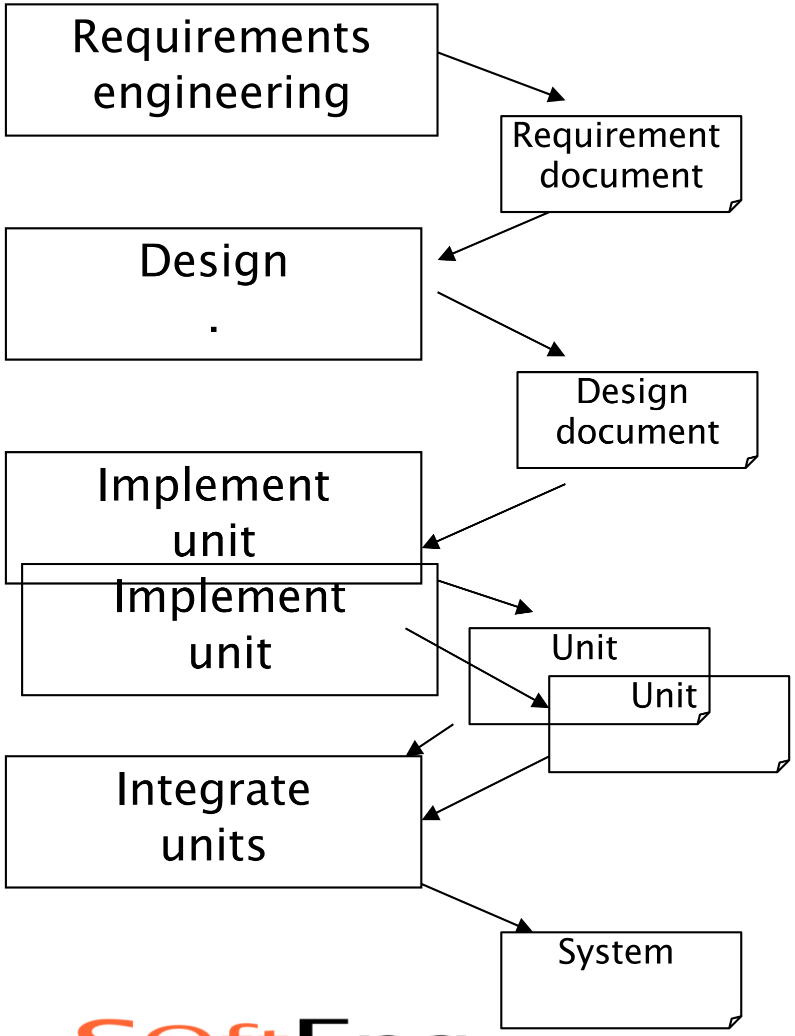
- Requirement engineering
  - What the software should do
- Architecture and design
  - What units and how organized
- Implementation
  - Write source code, (executable code)
  - Integrate units



# The production activities (2)

- Logically, each activity depends on the previous ones
  - To design, one must know the requirements
  - To implement, one must know the design and the requirements
- First approach is to do these activities in sequence
  - See waterfall model later
- In practice feedbacks and recycles must be provided
- Requirements and design are written down in documents

# Production activities



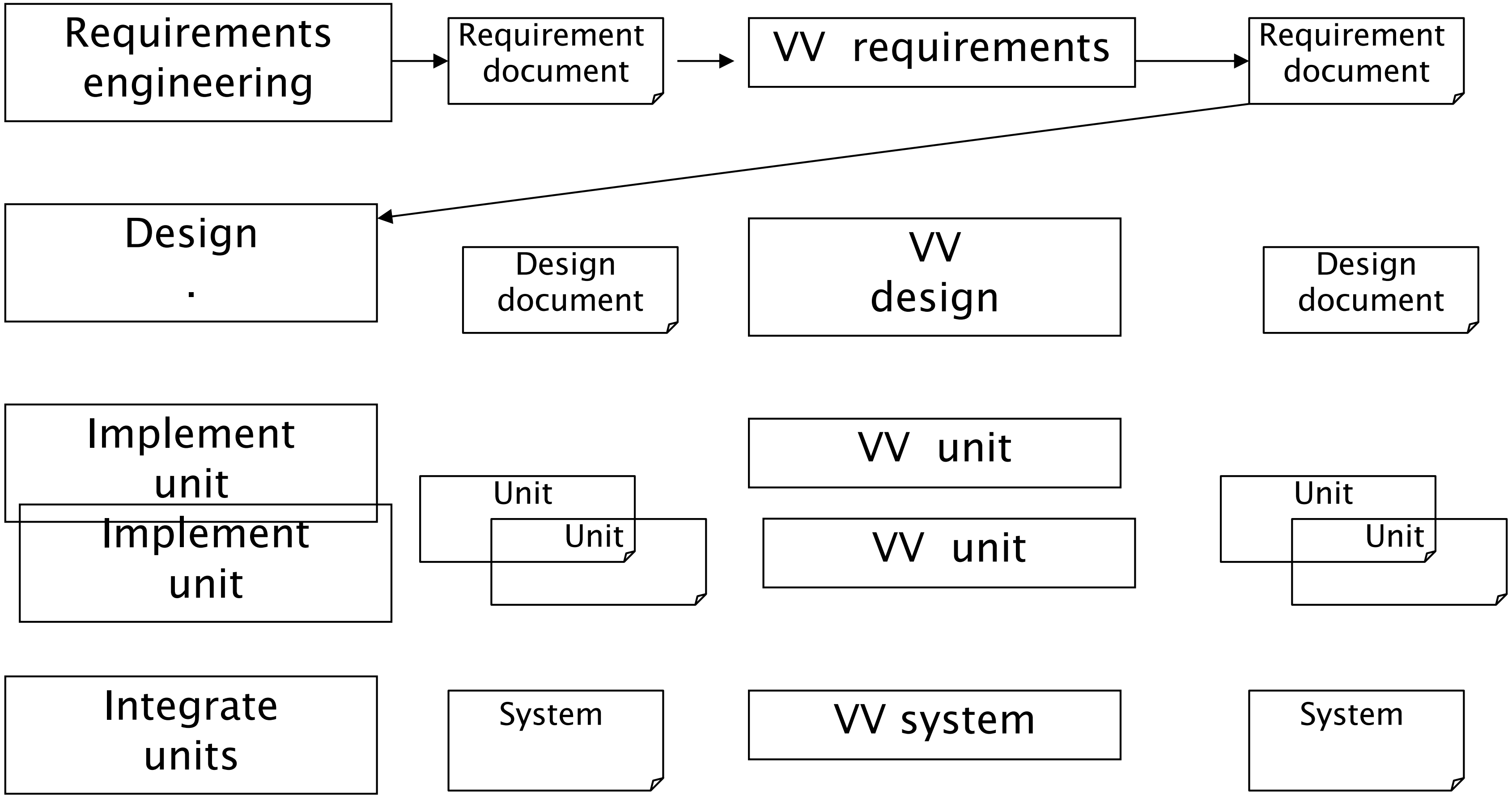
- Ok, we did it
  - Does it work?
  - Is it doing what it should do?

    - Or
  - Did we understand the requirements correctly?
  - Did we implement the requirements correctly?

# The V & V activities

- V & V = verification and validation
- Control that the requirements are correct
  - Externally: did we understand what the customer/user wants?
  - Internally: is the document consistent?
- Control that the design is correct
  - Externally: is the design capable of supporting the requirements
  - Internally: is the design consistent?
- Control that the code is correct
  - Externally: is the code capable of supporting the requirements and the design?
  - Internally: is the code consistent (syntactic checks)

# Production + VV activities

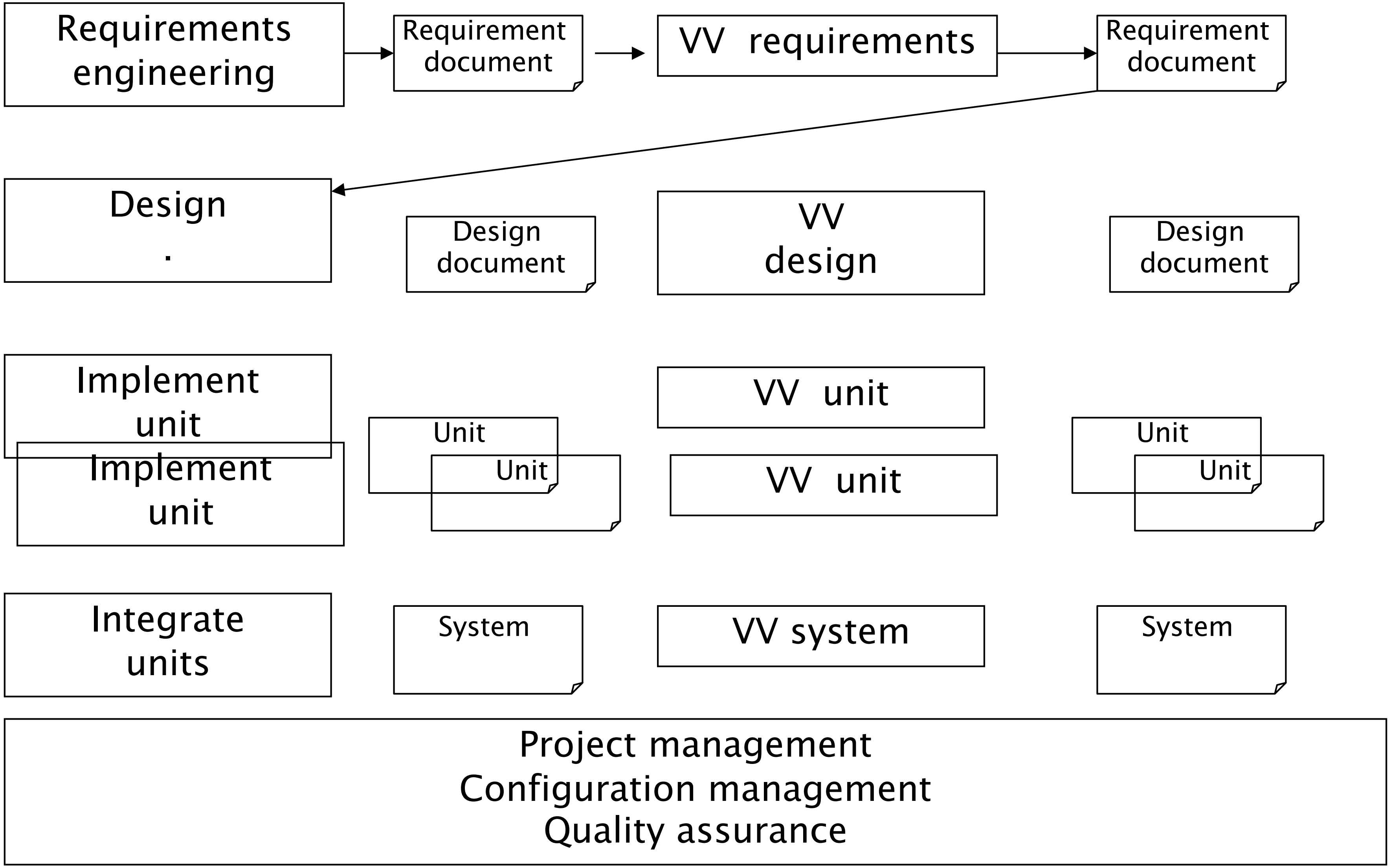


- Well, seems a lot of work
  - Who does what, when?
  - With what resources?
  - How much will it cost, when will we finish?
  - Where are the documents and units? Who can modify what?
  - Are we doing it state of the art?

# The management activities

- Project management
  - Assign work and monitor progress
  - Estimate and control budget
- Configuration management
  - Identify, store documents and units
  - Keep track of relationships and history
- Quality assurance
  - Define quality goals
  - Define how work will be done
  - Control results

# The whole picture



Requirements engineering

Requirement document

VV requirements

Requirement document

Design
.

Design document

VV
design

Design document

Implement unit

Implement unit

Unit

Unit

VV unit

VV unit

Unit

Unit

Integrate units

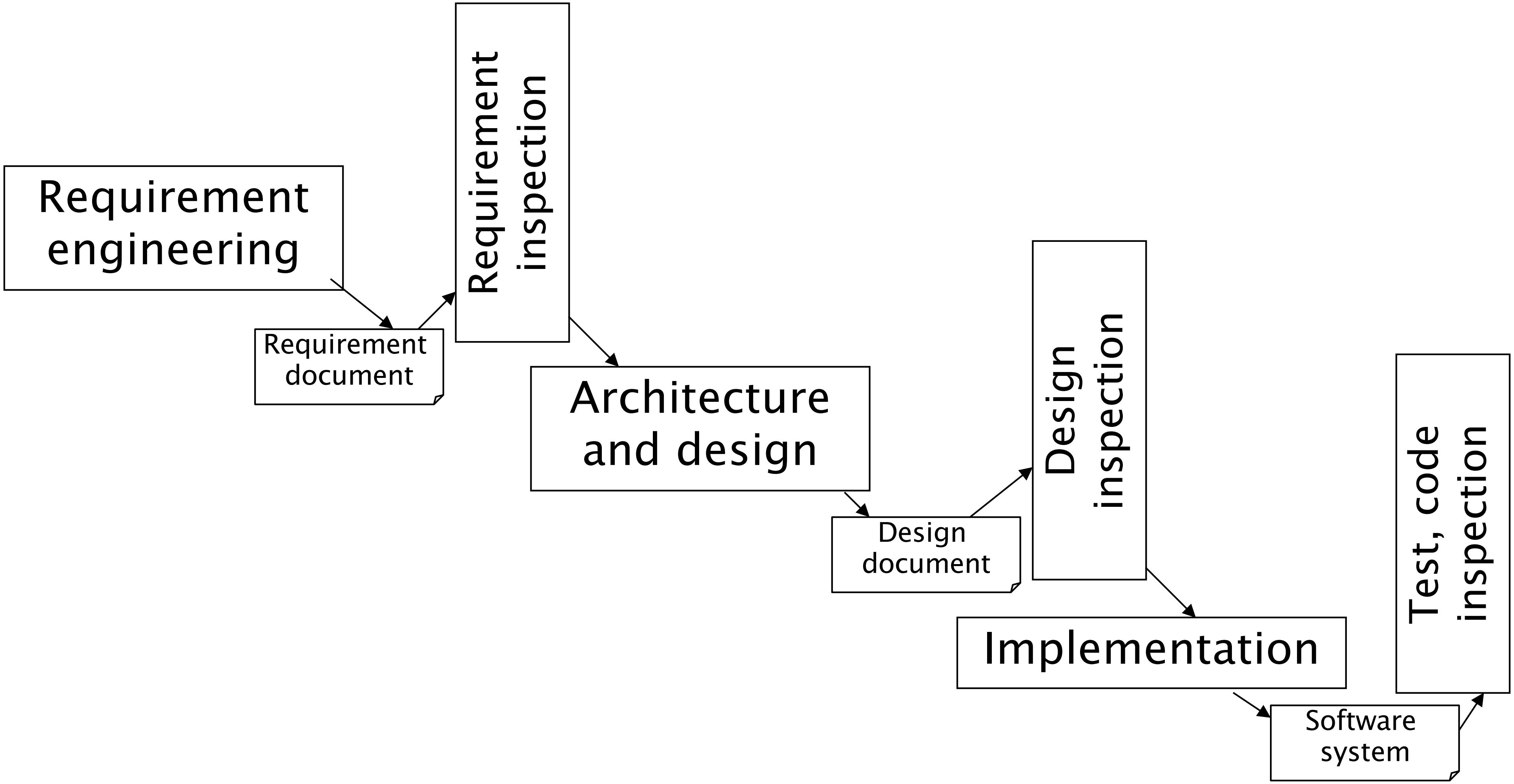
System

VV system

System

Project management
Configuration management
Quality assurance

# The whole picture (2)



Configuration management

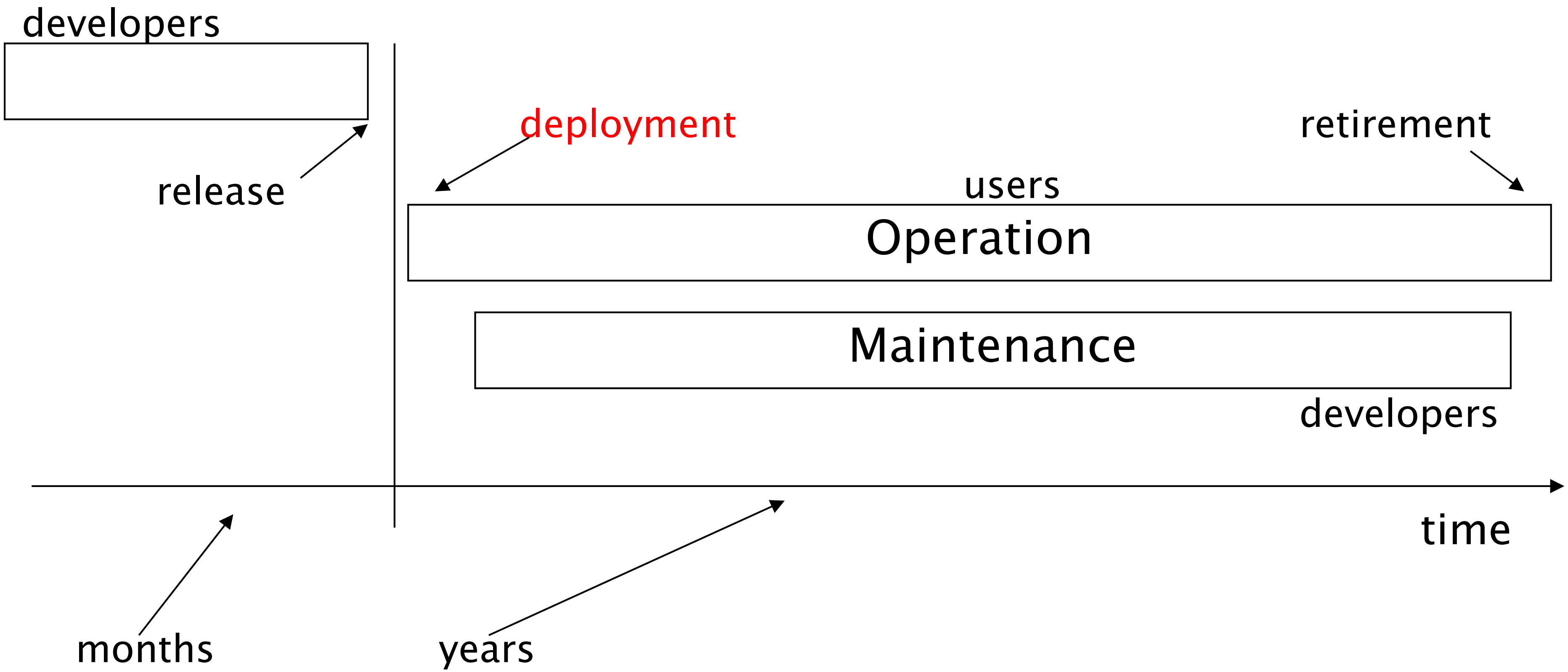
Project management

# Phases

# Beyond development

- Development is only the first part of the game
    - Operate the software
        - Deployment, operation
    - Modify the software
        - Maintenance
    - End up
        - retirement

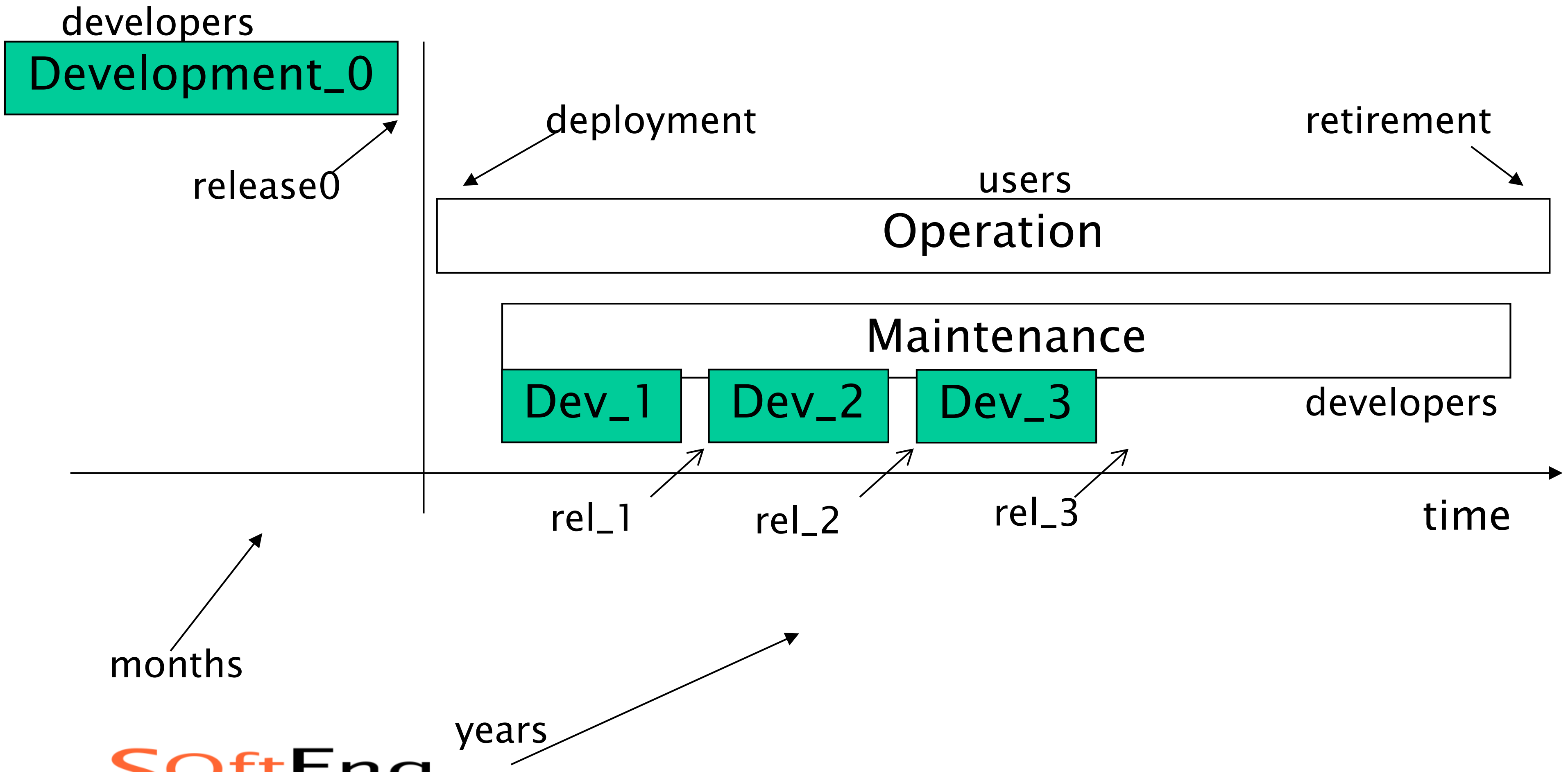
# The main phases



# Maintenance

- Can be seen as a sequence of developments
- First development usually longer
- Next developments constrained by previous ones and related choices
  - If dev_0 chooses java, next developments are in Java
  - If dev_0 chooses client server model, next developments keep C/S

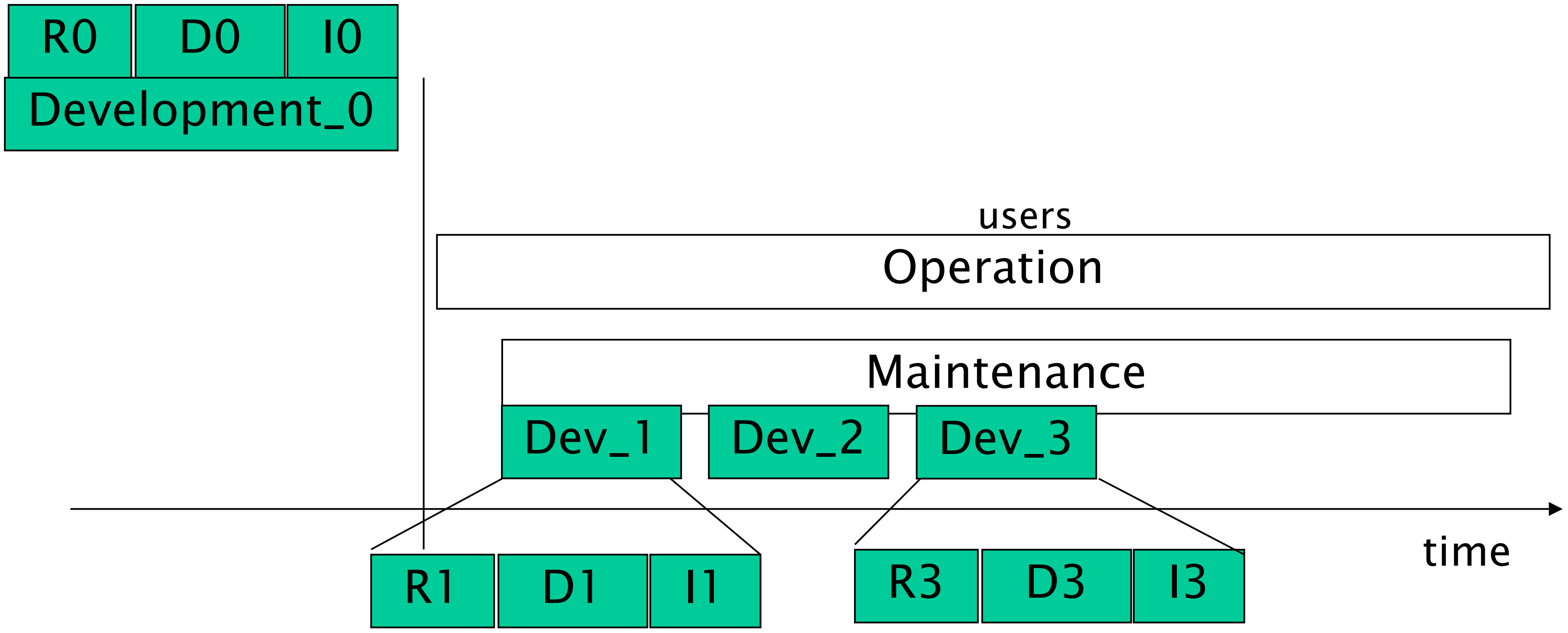
# Maintenance



# Maintenance

- Development and maintenance do the same activities (requirement, design, etc)
  - But in maintenance an activity is constrained by what has been done before
  - After years, the constraints are so many that changes become impossible

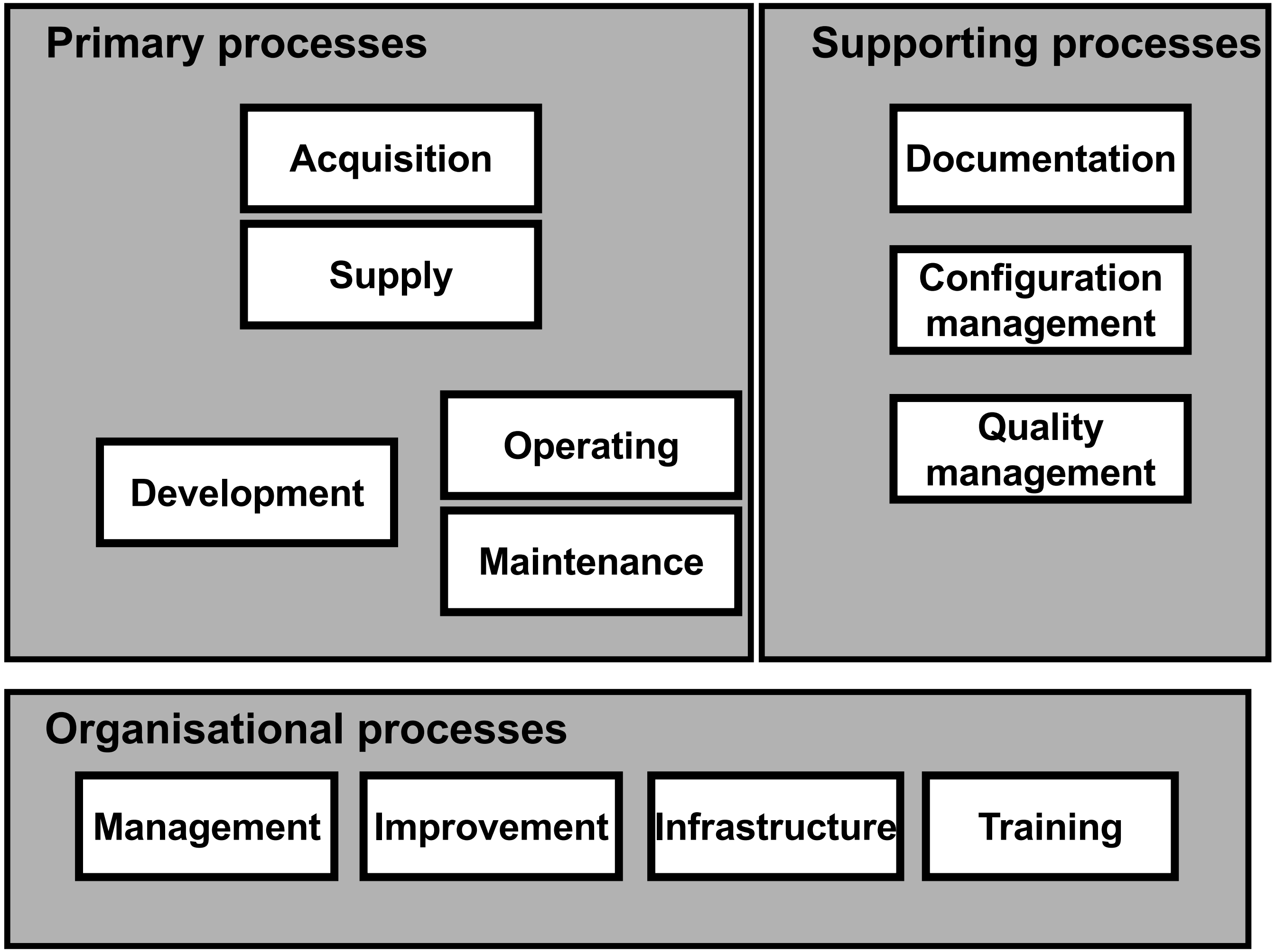
# Maintenance



- Development_0
  - Req_0 developed from scratch
  - Design_0 developed from req_0
  - Impl_0 developed from design_0
- Development_1
  - Req_1 from Req_0 (and Des_0, Impl_0)
  - Des_1 from Req_1
  - Impl_1 from Des_1

# ISO/IEC 12207

International standard for software lifecycle processes



Primary processes

Acquisition

Supply

Development

Operating

Maintenance

Supporting processes

Documentation

Configuration management

Quality management

Organisational processes

Management

Improvement

Infrastructure

Training

# Scenarios in dev / maint / op

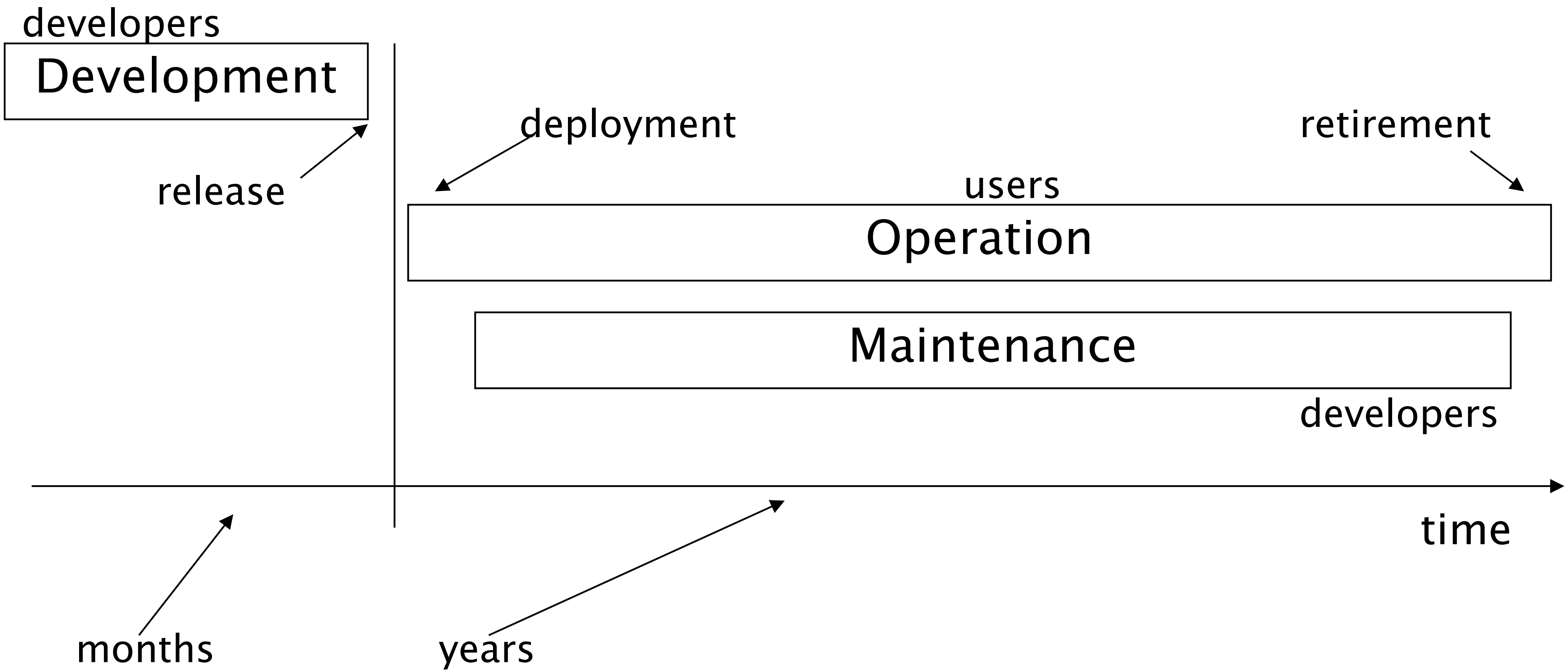
- Scenario 1: IT to support businesses
  - Development: several months
  - Operation: years
  - Maintenance: years, up to 60% of overall costs
- Scenario 2: consumer software (games)
  - Development: months
  - Operation: months (weeks)
  - Virtually no maintenance

# Scenarios in dev / maint / op

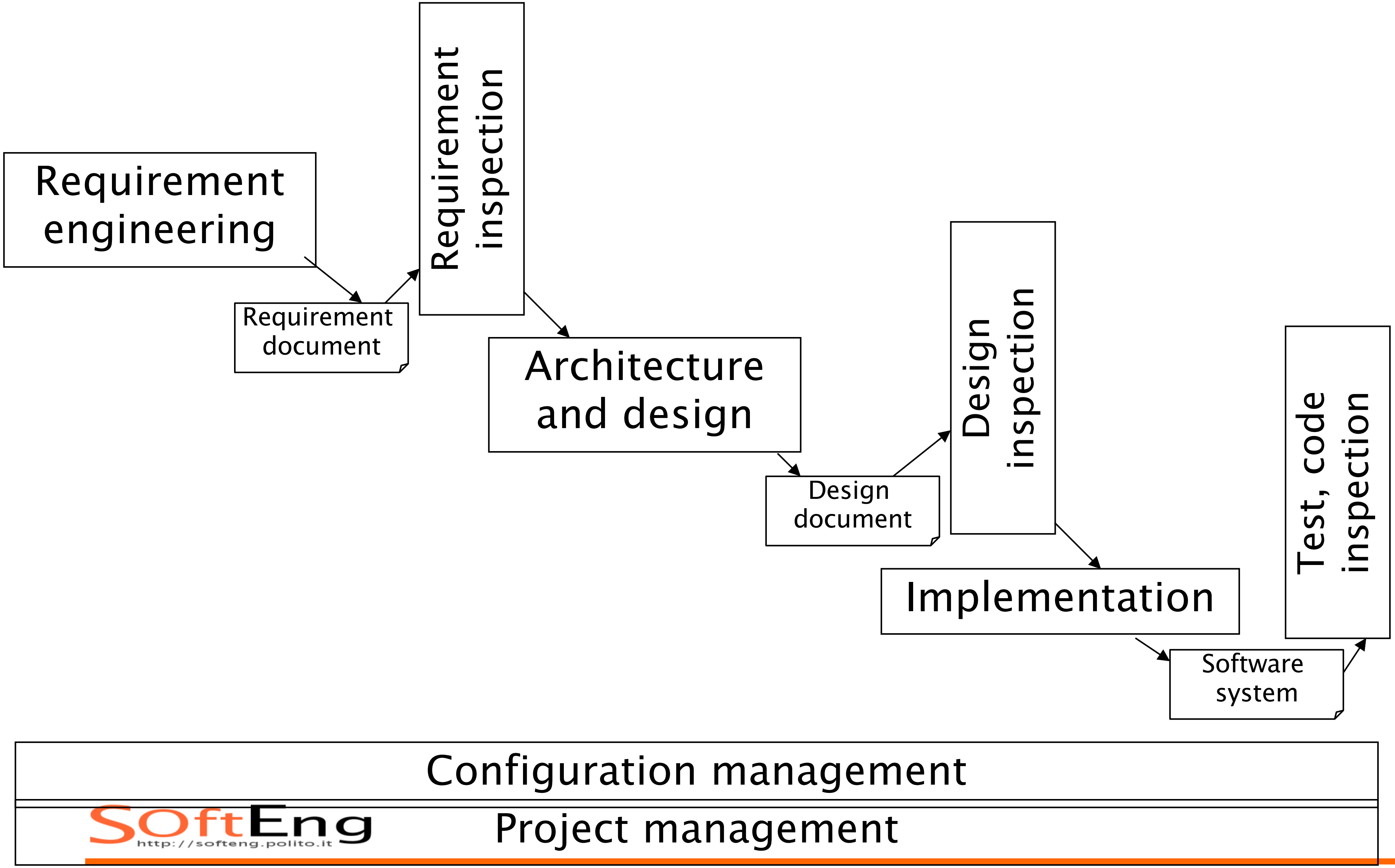
- Scenario 3: Operating System
  - Development: years
  - Operation: years
  - Maintenance: years, up to 60% of overall costs
- Scenario 31: Commercial OS (MS)
  - 2, 3 years to develop
  - Several years maintenance
    - Patches issued every day
    - Major releases (Service Pack) at long intervals
  - In parallel development of a new release
    - Cfr W3.1, 95, NT, 2000, XP, Vista, 7, …

# In summary, top down

# Phases


developers
Development
release
deployment
retirement
users
Operation
Maintenance
developers
time
months
years

# Development, activities



# Comparison with traditional engineering

# The software process

- Not new
- Just applying engineering approach to software production
- What do aeronautics engineers do?

# Production + test activities

- Requirement definition (“what”)
  - airplane, civil usage
  - capacity > 400 people
  - range > 12000km,
  - Noise level < xdB, consumption < .., acquisition cost < y$, operation cost < w $/year
- high level design (“how”)
  - Blueprints of the airplane
  - Definition of subsystems
    - Avionics, structure, engines
  - Mathematical models
    - Structural (wings and frame)
    - Thermodinamic (engines)

- low level design
  - Further definition of subsystems
  - In several cases subcontracted or acquired (engine)
- implementation
  - Implementation of each subsystem
- unit test
  - Verification that subsystem complies to its specification

- Integration
  - Put subsystems together (ex. wing + frame)
- Integration test
  - Test the assemblies
- Acceptance test
  - Does it fly?
- Certification
  - FAA or other tests that it flies and issues a certificate
  - (a defined and long list of checks)

# Management activities

- project management
  - project planning
  - project tracking
  - budgeting, accounting
- configuration management
  - Parts and assemblies
  - change control
- Quality management
  - Quality handbook
  - Quality plan
  - roles

# Is there a difference?

Traditional engineering

- Hundreds year old
- Theory from physics or other hard science, laws and mathematical models
- Maturity of customers and managers

Software engineering

- 50 years old
- Limited theories and laws. More a social science?
- Variable maturity of customers and managers

# System and software process

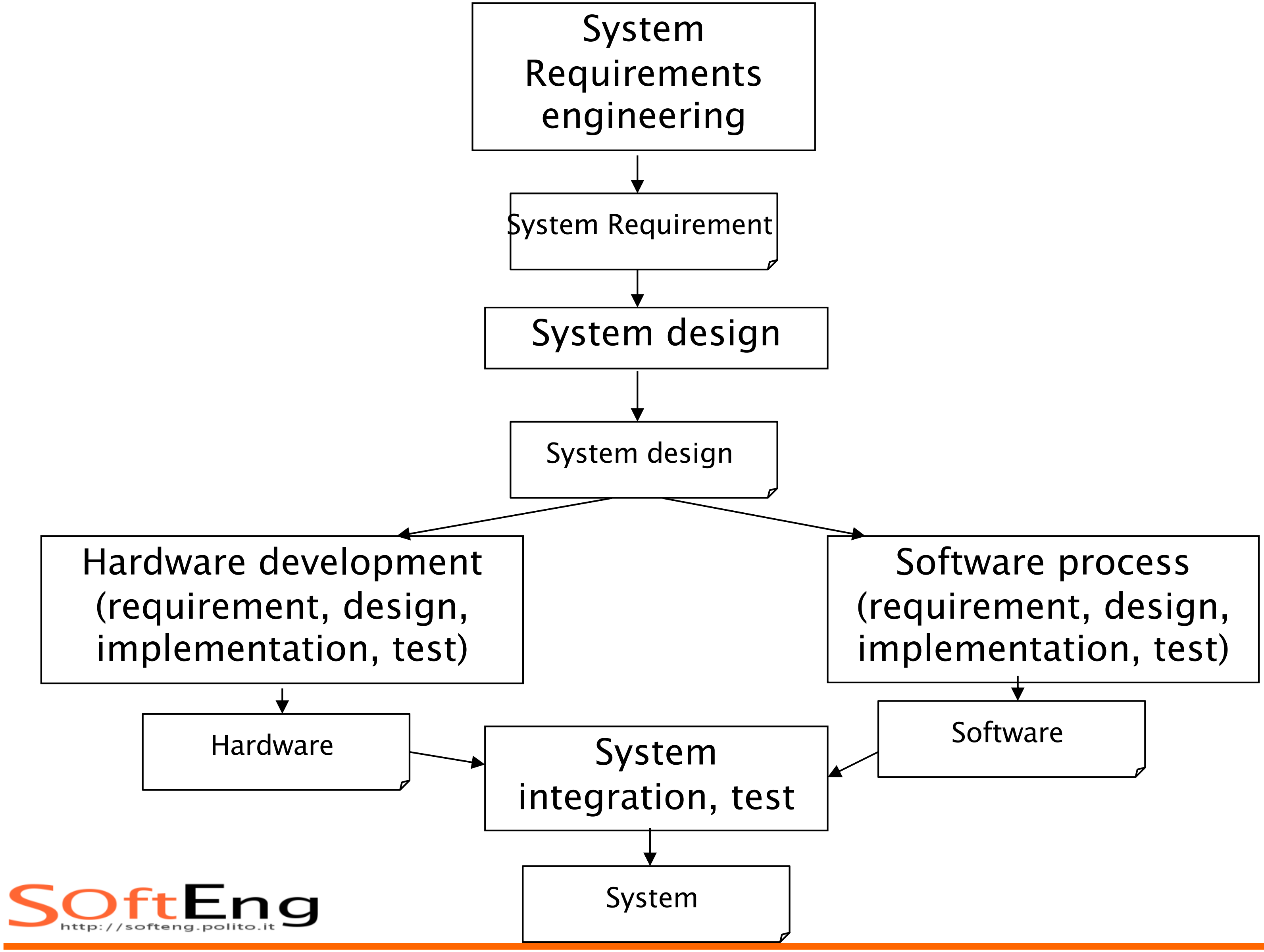
# System vs. software

- Different types of software require different processes
  - Stand alone software → software process
  - Embedded software → system process

# The system process

- System requirements
- System design
- Software development
  - Requirements, design, implementation, test, integration
- System integration and test

# The system process


System Requirements engineering
System Requirement
System design
System design
Hardware development (requirement, design, implementation, test)
Software process (requirement, design, implementation, test)
Hardware
Software
System integration, test
System

# 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
- The course ‘Software Engineering II’ will be based on approach 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

- Main phases are development, operation, maintenance
- Development has production, control and management activities
- The software process is the reference framework for techniques and tools
- For embedded software the software process is part of the system process
- Different categories of processes organize these activities in different ways