Components Based Design and Development

Computer Engineering Studies
Universidad Carlos III de Madrid

Unit 2: Software Engineering Quick Overview

Juan Llorens

Högskolan på Åland – Finland / Universidad Carlos III de Madrid - Spain

Juan.llorens@uc3m.es

Software Engineering

- Engineering...

  "the ***profession*** in which
  a knowledge of the ***mathematical*** and ***natural sciences*** gained by study, experience and practice
  is applied with judgment
  to develop ways to ***utilize***, economically, the ***materials and forces of nature for the benefit of mankind***." (Accreditation Board for Engineering and Technology, 1996).

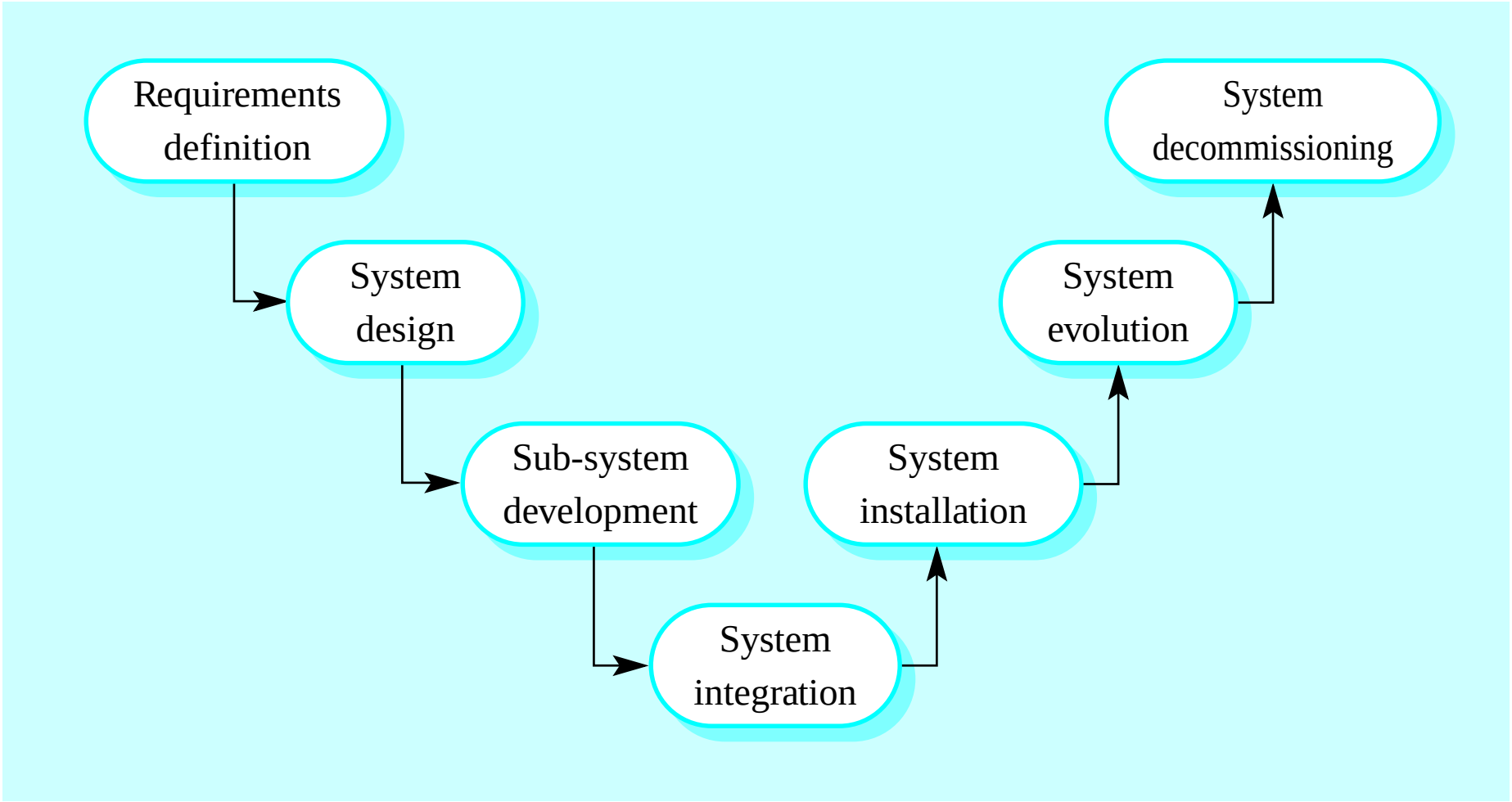
- Particularities of software engineering
  - The "product" (software)
  - A lot of development, little engineering discipline
  - Frequent changes in the product
  - It is not seen as necessary by the "so called" software engineers

From Gonzalo Genova

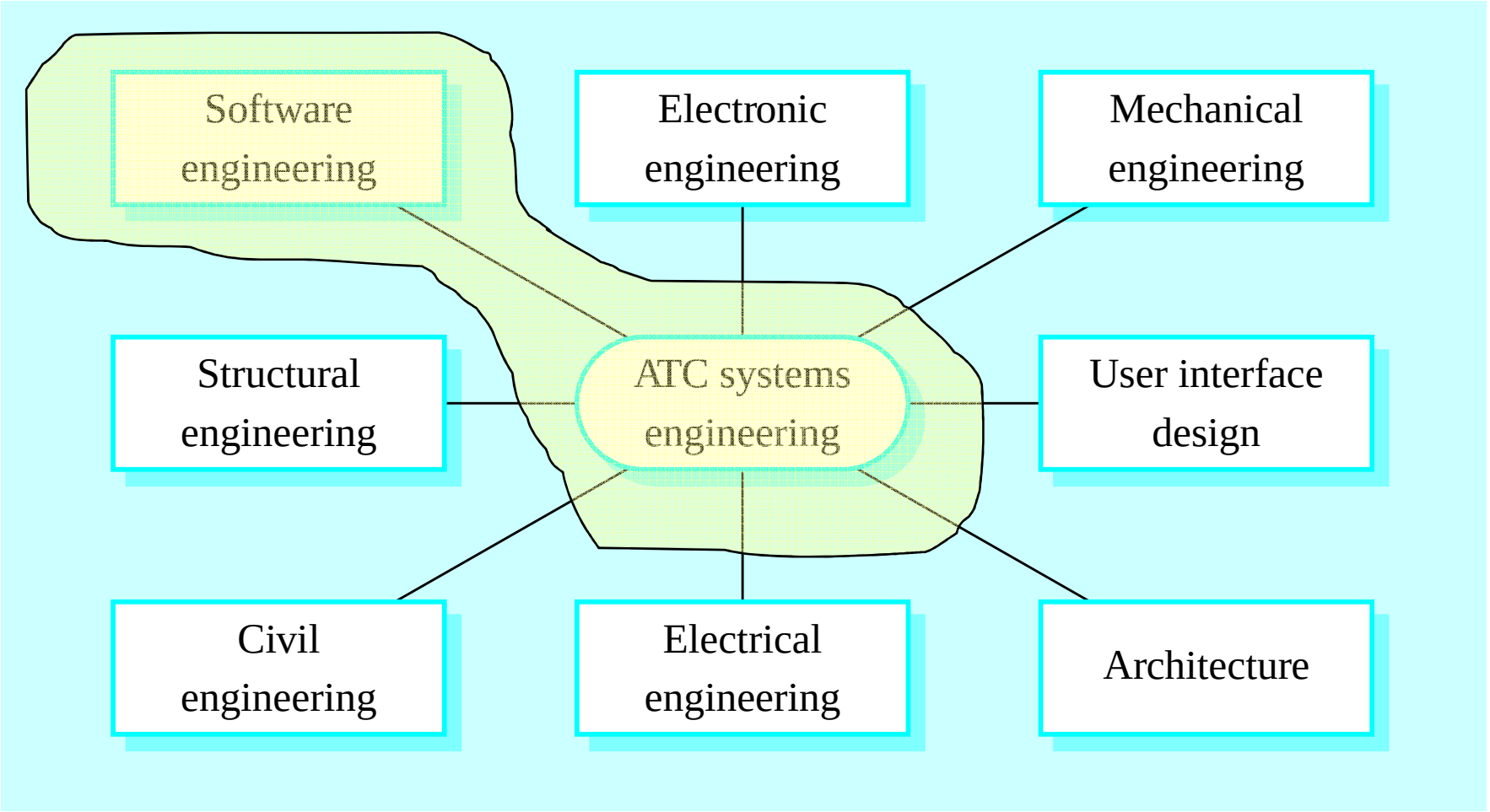
Software Engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP in all developed countries.

The systems engineering process



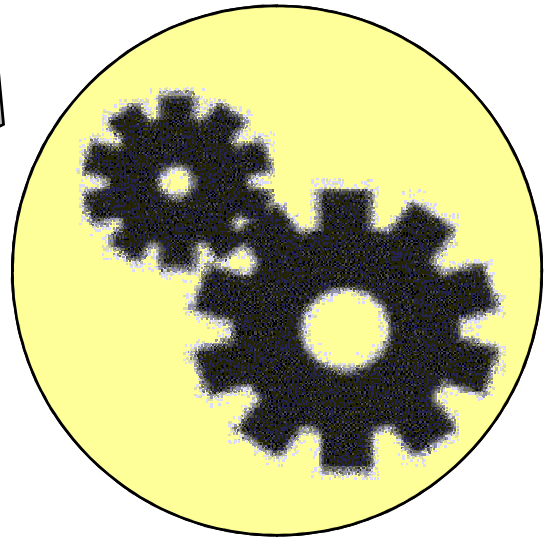
Inter-disciplinary involvement



What is a software process?

Objectives, needs, ideas

problems, failures,..



How is a software project done?

“Software application”

…Not a Software system

What is a software process?

- A set of activities whose goal is the development or evolution of software.
- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands.

What is a software process model?

- A simplified representation of a software process, presented from a specific perspective.
- Examples of process perspectives are
  - Workflow perspective - sequence of activities;
  - Data-flow perspective - information flow;
  - Role/action perspective - who does what.
- Generic process models
  - Waterfall;
  - Iterative development;
  - Component-based software engineering.

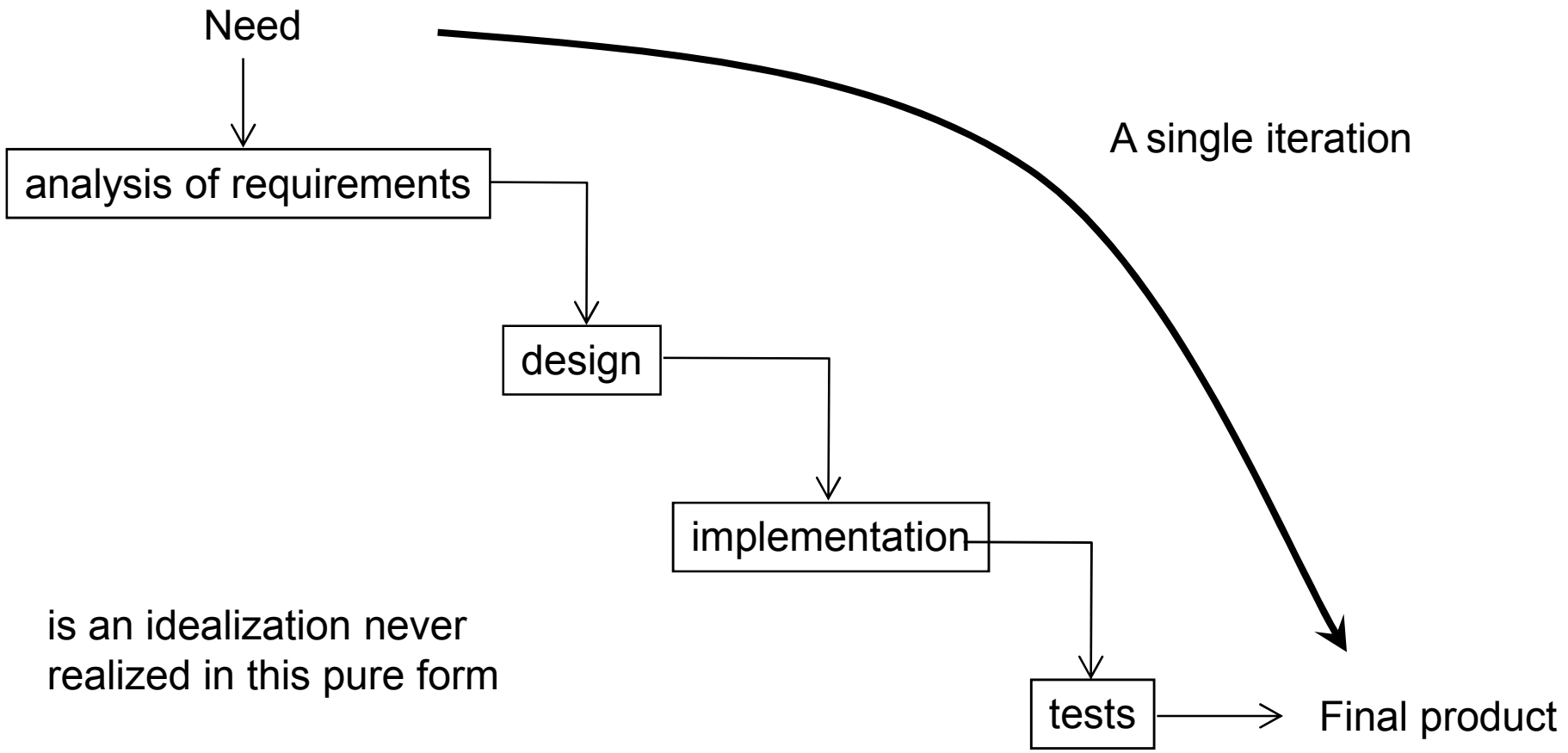
Usual workflow Activities

Req. analysis	design	implementation	tests

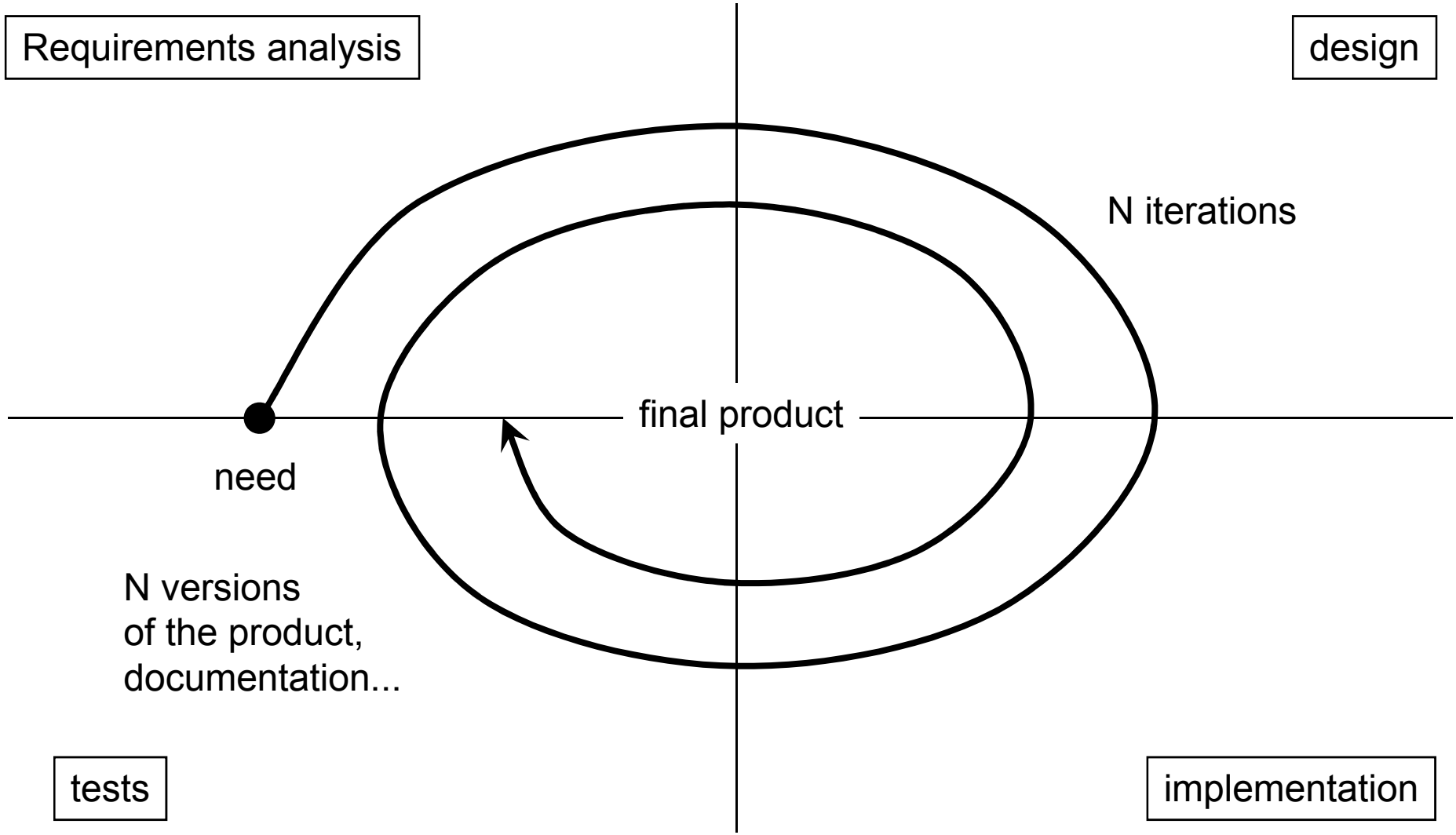
Not so Usual workflow Activities

Vocabulary Id.	Project Mgmt	Estimation	Debriefing
Sw For Reuse	Risk Mgmt.	Project Plan.	Indexing

Types of Software Processes : the waterfall process

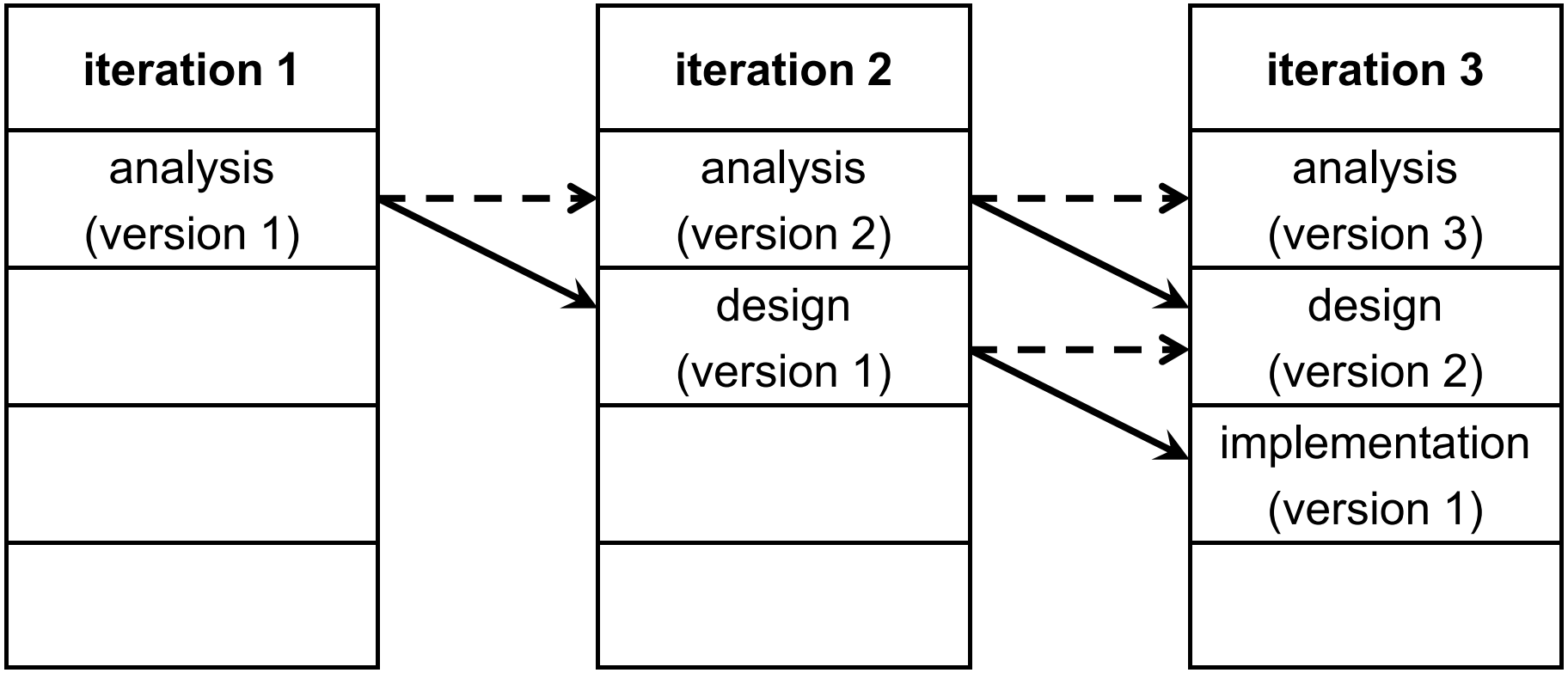


Iterative process [spiral]

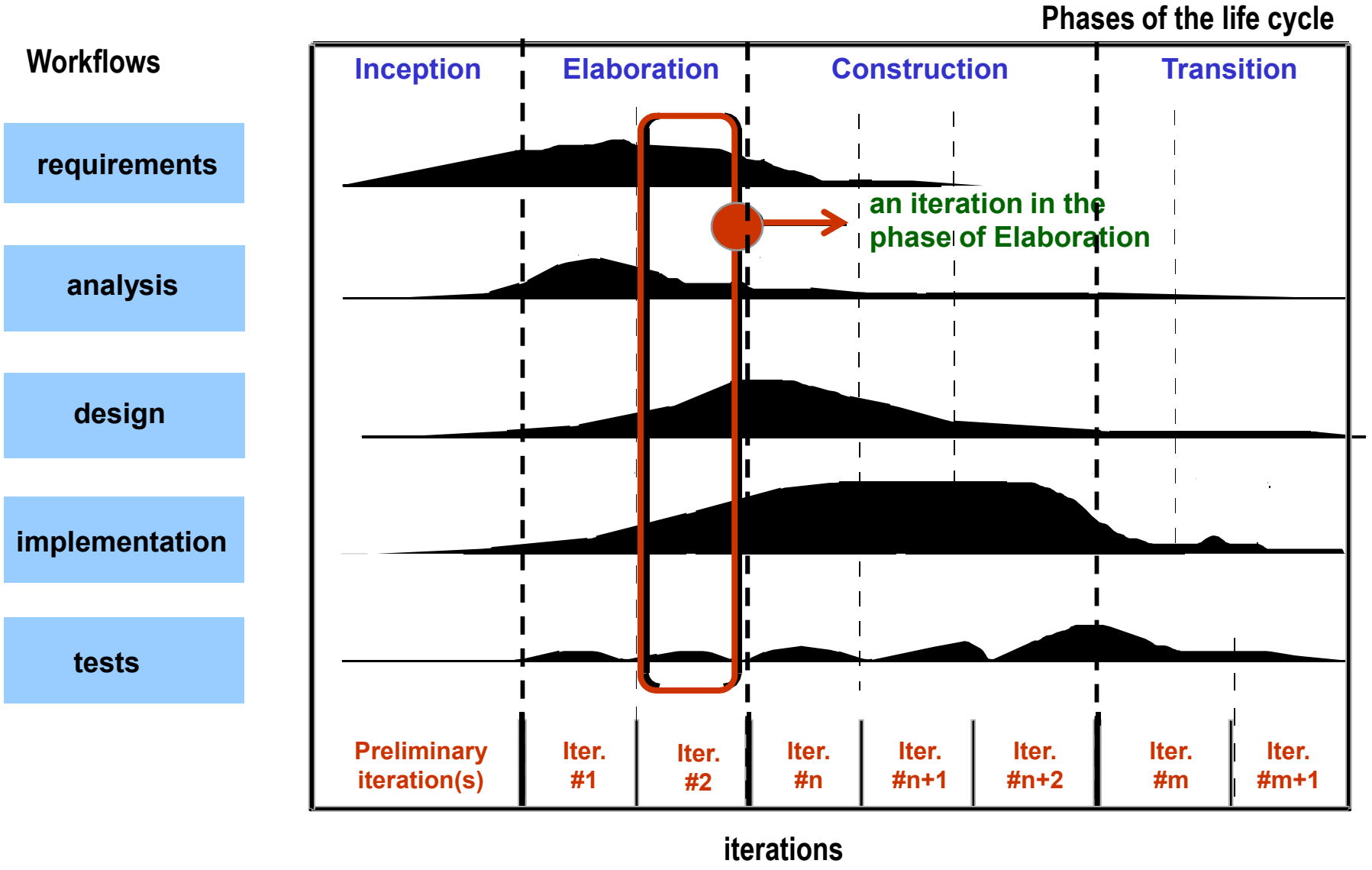


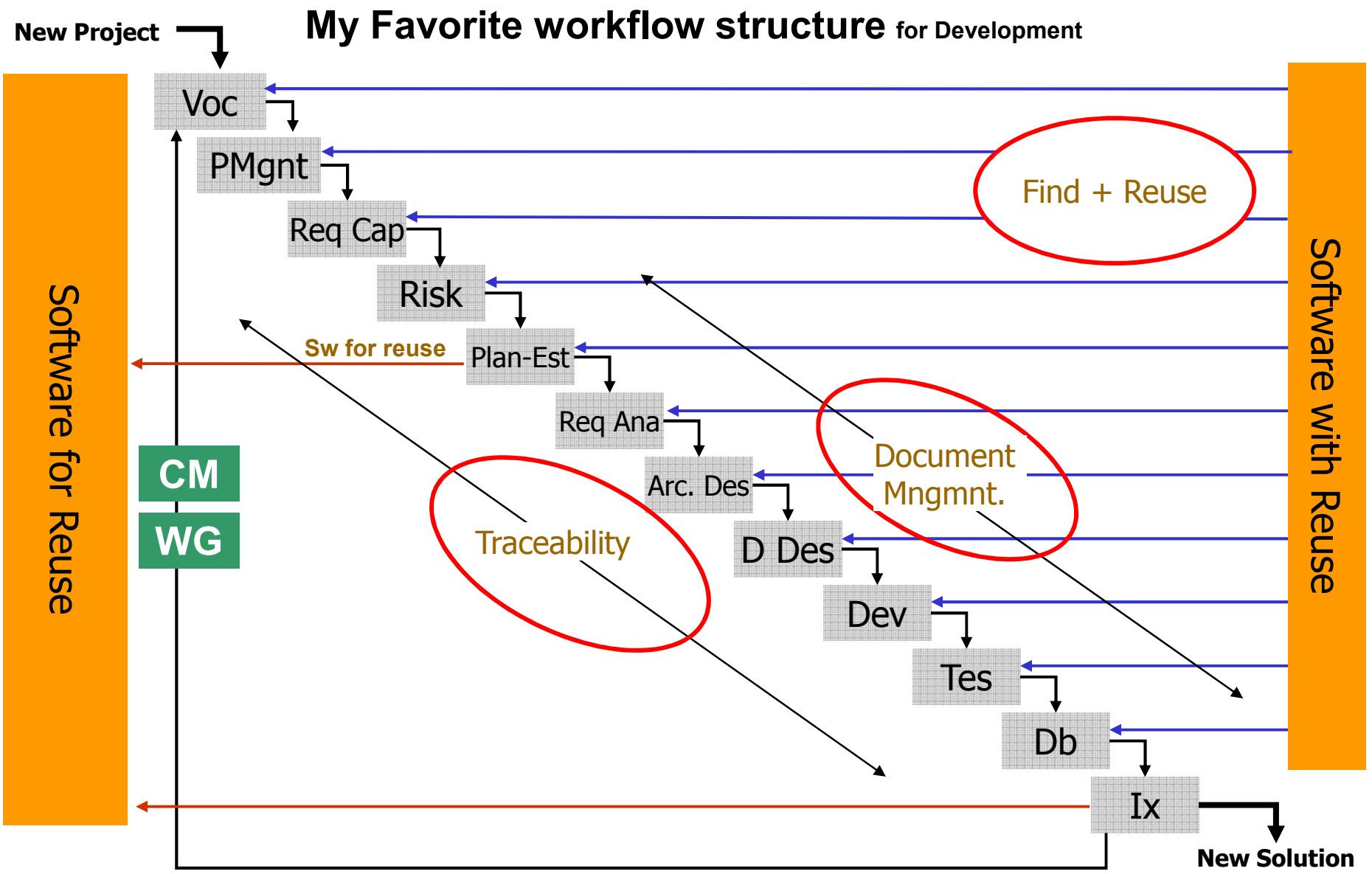
Iterative and incremental process

- Enables the **evolution in parallel** of several workflows, and by that the work in parallel of several teams of people
- the different versions of the documents produced in each iteration are not necessarily compatible between them: **organize well the documentation**

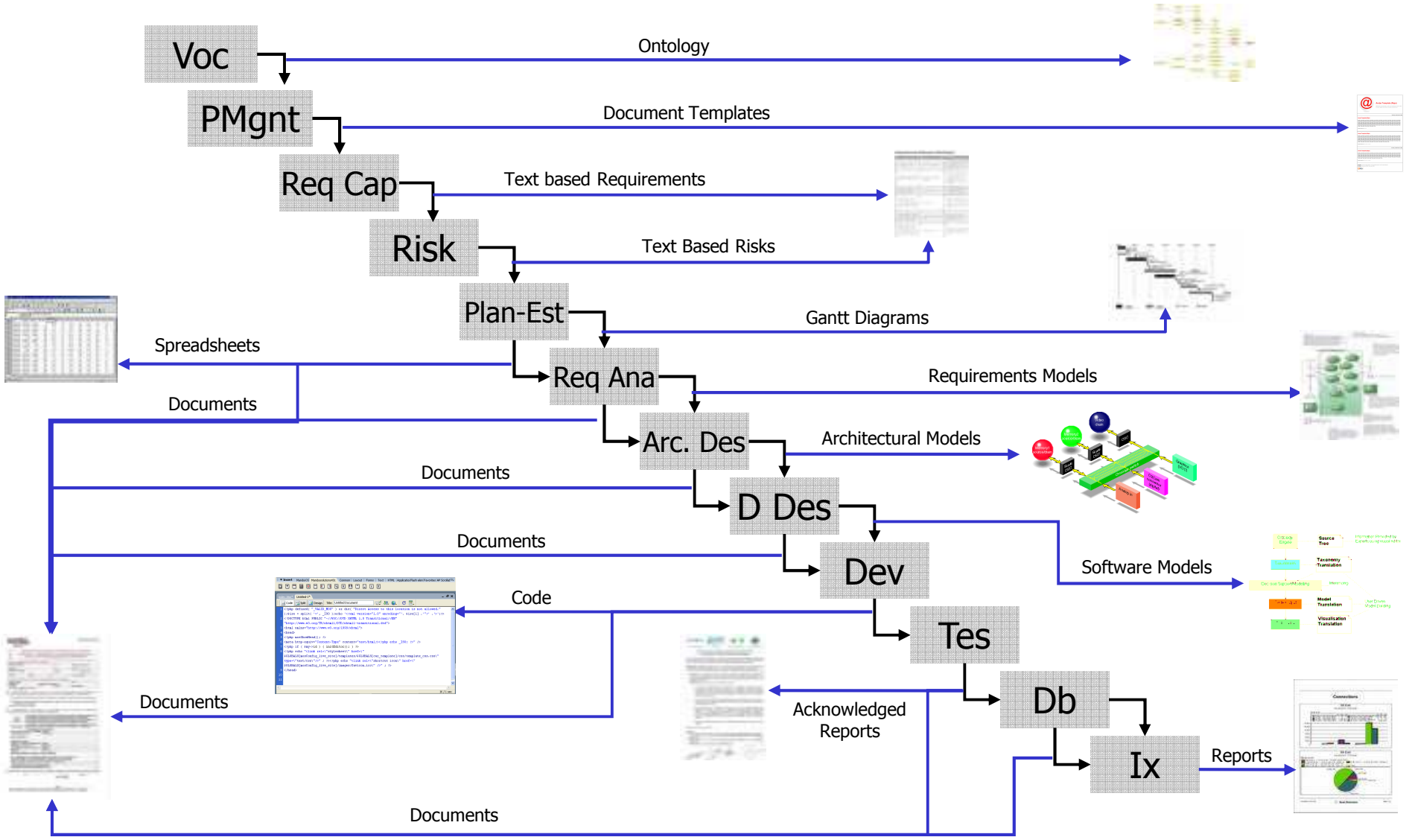


UP: Phases, iterations and workflows




New Project
My Favorite workflow structure for Development
Software for Reuse
Voc
PMgnt
Req Cap
Risk
Sw for reuse
Plan-Est
Req Ana
Arc. Des
D Des
Dev
Tes
Db
Ix
New Solution
CM
WG
Traceability
Find + Reuse
Document Mngmnt.
Software with Reuse

My Favorite dataflow structure for Development



Other frameworks and nomenclatures

Terminology of UP	Classical terminology
requirements	requirements analysis
analysis	
design	design
implementation	implementation
	integration
tests	tests

Eric Braude, *Software Engineering. An Object-Oriented Perspective*, John Wiley & Sons, 2001, p. 30.

How to perform an Iterative Software Process

- divide a project in **mini-projects**, easier to manage and complete
- each mini-project is an **iteration**
- each iteration contains **all the elements** of a normal project:
  - planning
  - analysis and design
  - construction
  - integration and tests
  - generate a version of the product (internal or external)
- each iteration generates a **baseline** that includes a partially completed version of the final system, and all the associated documentation
- the successive iterations build on top of each other until the final system is finished
- the difference between two baselines is known as an **increment**

What are software engineering methods?

- Structured approaches to software development which include system models, notations, rules, design advice and process guidance.

- Model descriptions
  - Descriptions of graphical models which should be produced;
- Rules
  - Constraints applied to system models;
- Recommendations
  - Advice on good design practice;
- Process guidance
  - What activities to follow.

Based on Ian Sommerville 2004 - Software Engineering, 7th edition. Chapter 1

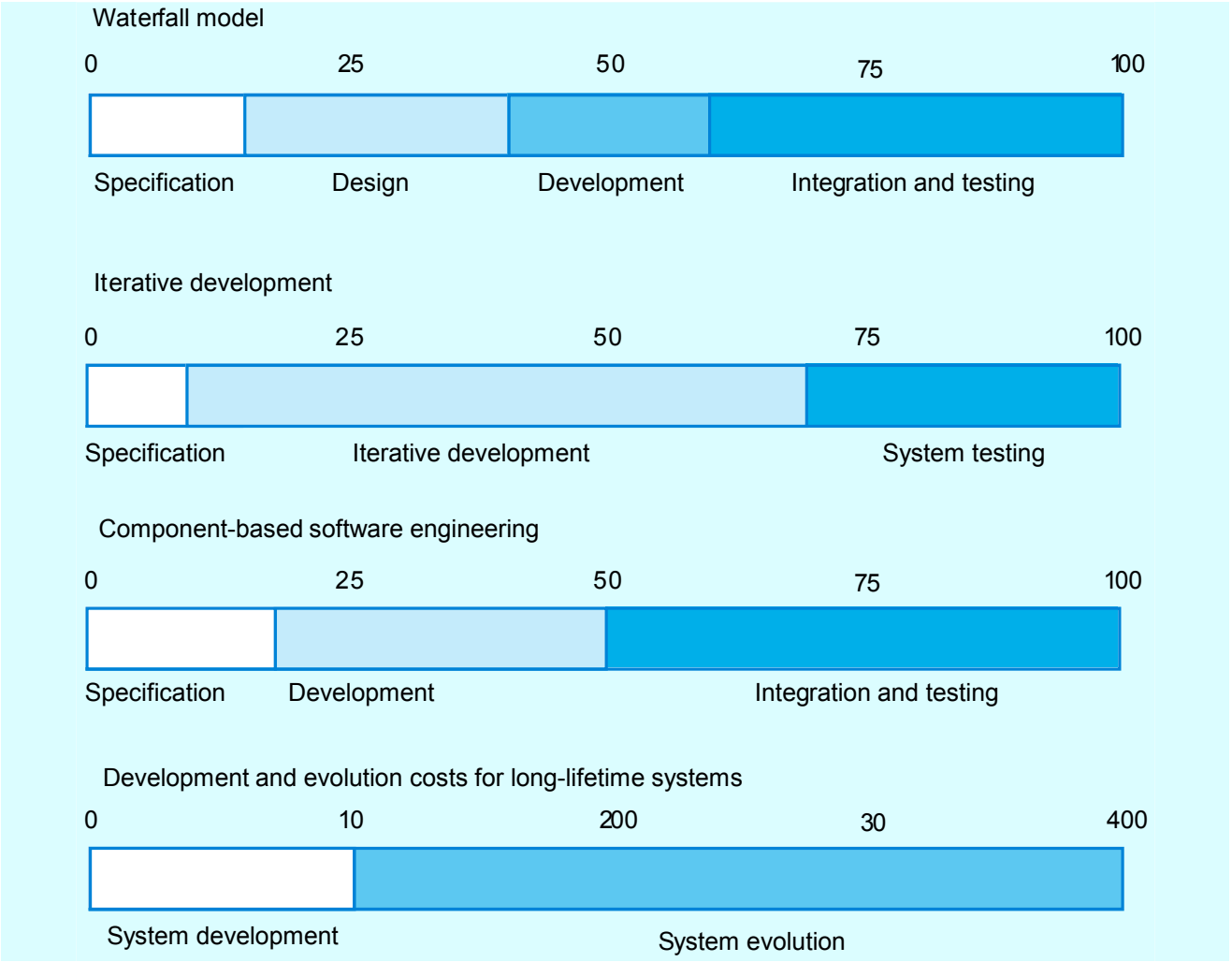
What are the costs of software engineering?

- Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

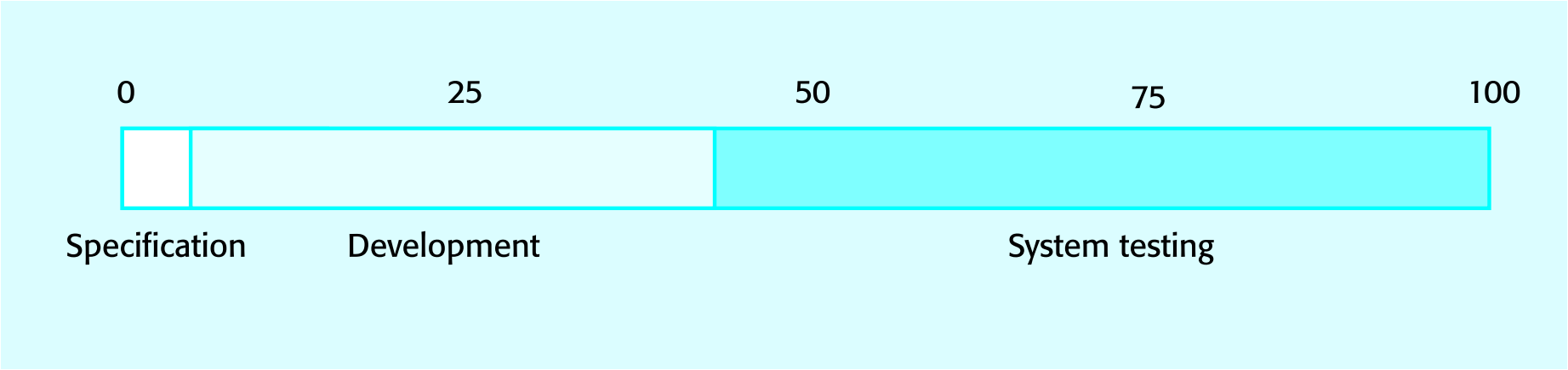
Software costs

- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.

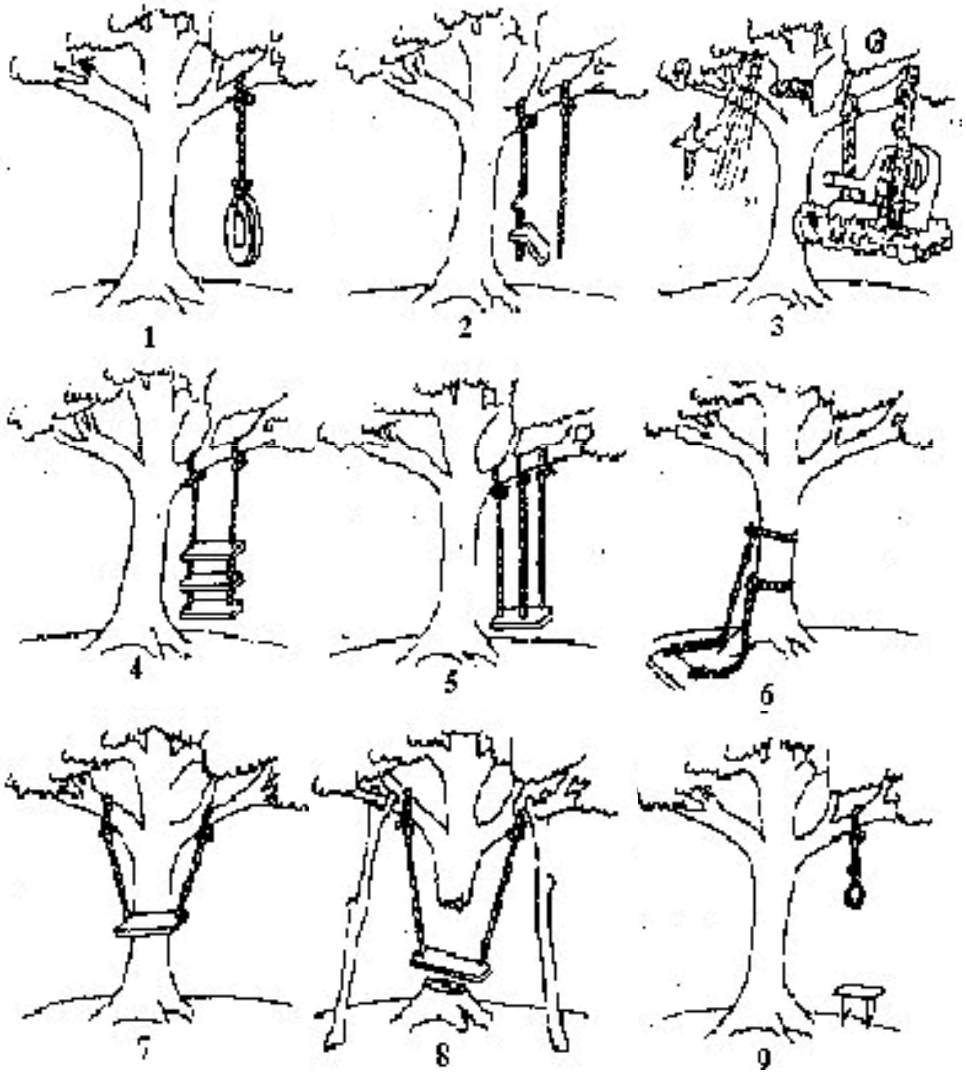
Activity cost distribution



Product development costs



The maturity of the Software Engineering discipline (swing)



1. **Real need**: what the customer really wanted.
2. **Client side**: what the client was able to describe as his/her clear need.
3. **Sale process**: what the software manufacturer promised the client.
4. **Requirements**: the final understanding of what the customer described as requirements.
5. **Analysis**: how was planned that the system should finally work at the client side.
6. **Design**: how the system should perform the analysed functionality.
7. **Coding**: what the programmer finally produced.
8. **On-site installation**: what was really installed in the client site.
9. **Test**: what the responsibles saw in the system.