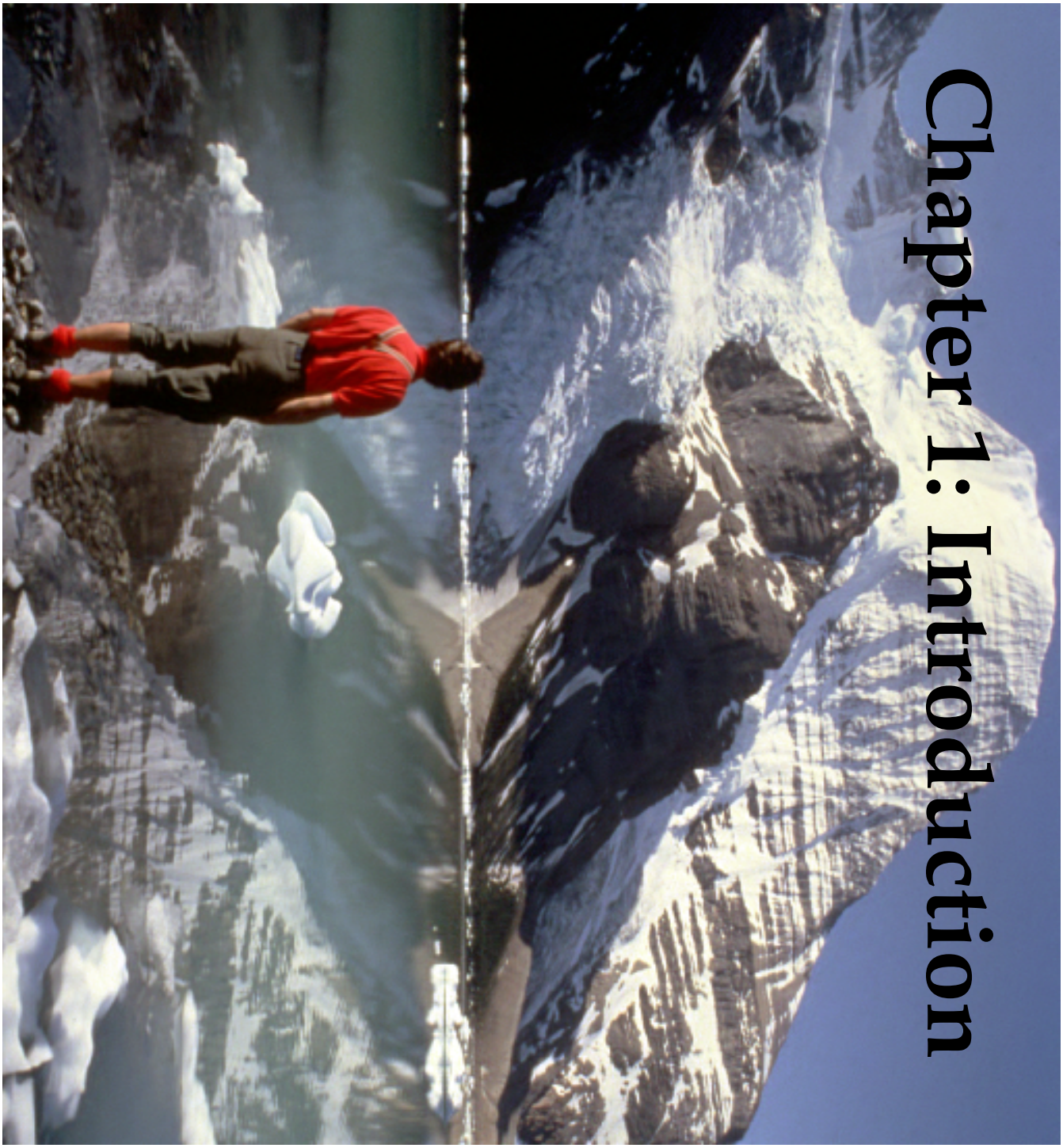
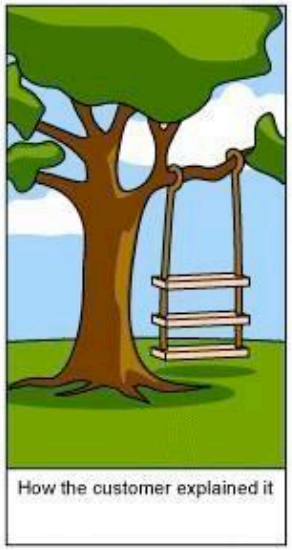
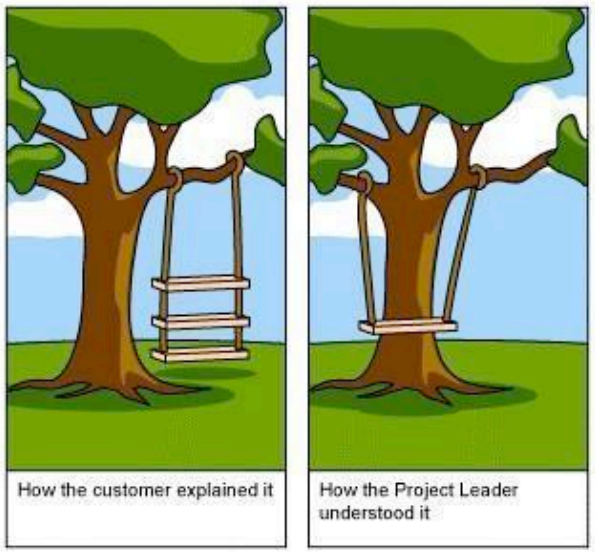
Object-Oriented Software Engineering
Using UML, Patterns, and Java



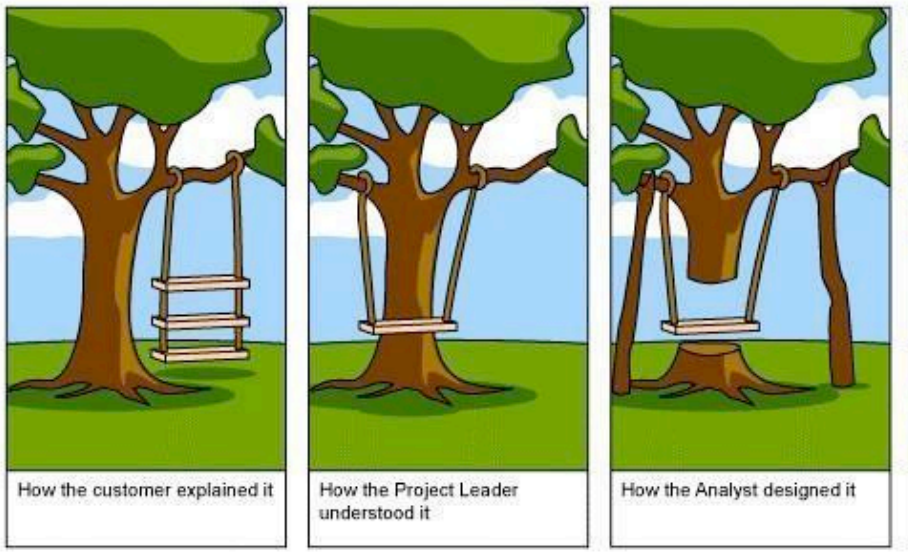
Ingegneria del software: scenario di riferimento



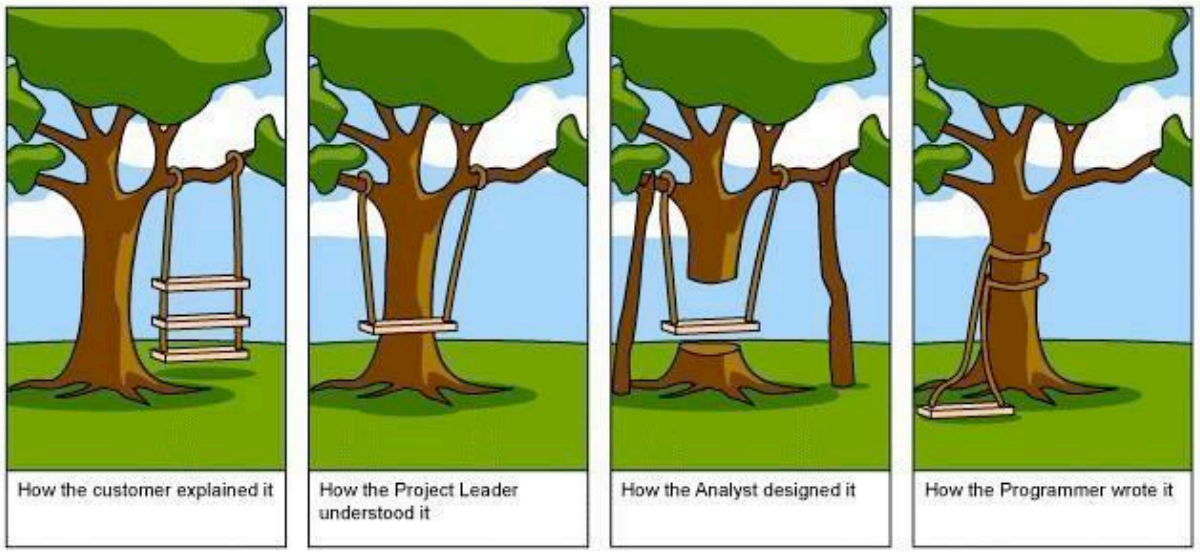
Ingegneria del software: scenario di riferimento



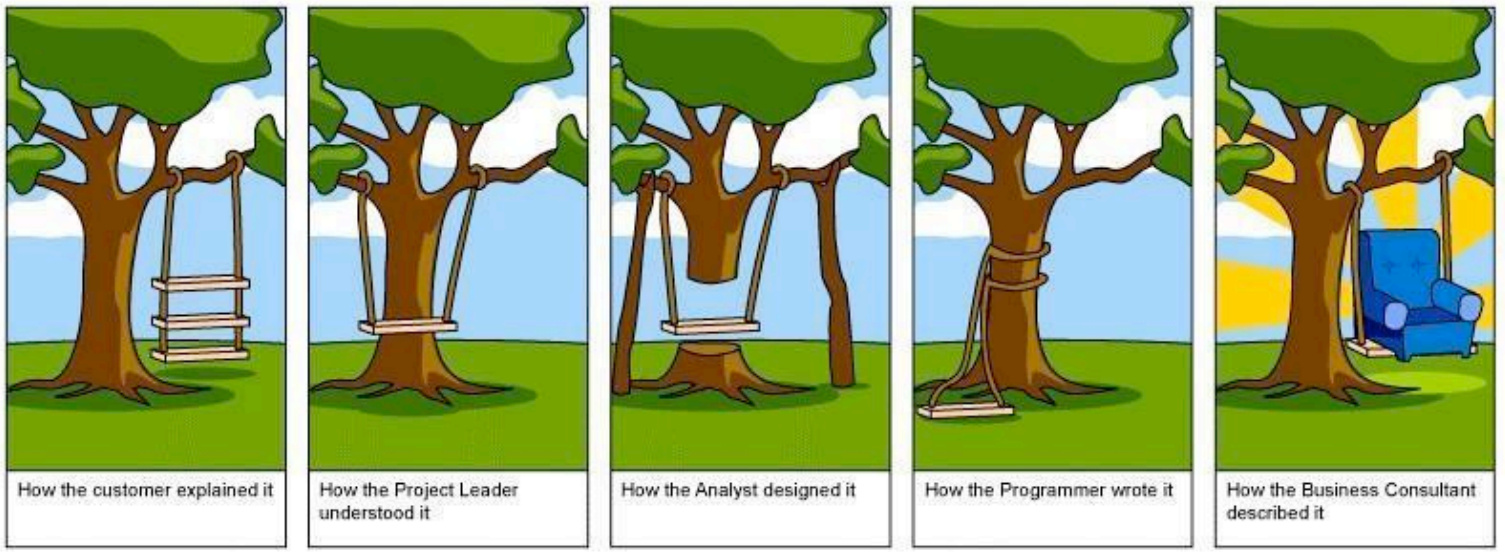
Ingegneria del software: scenario di riferimento



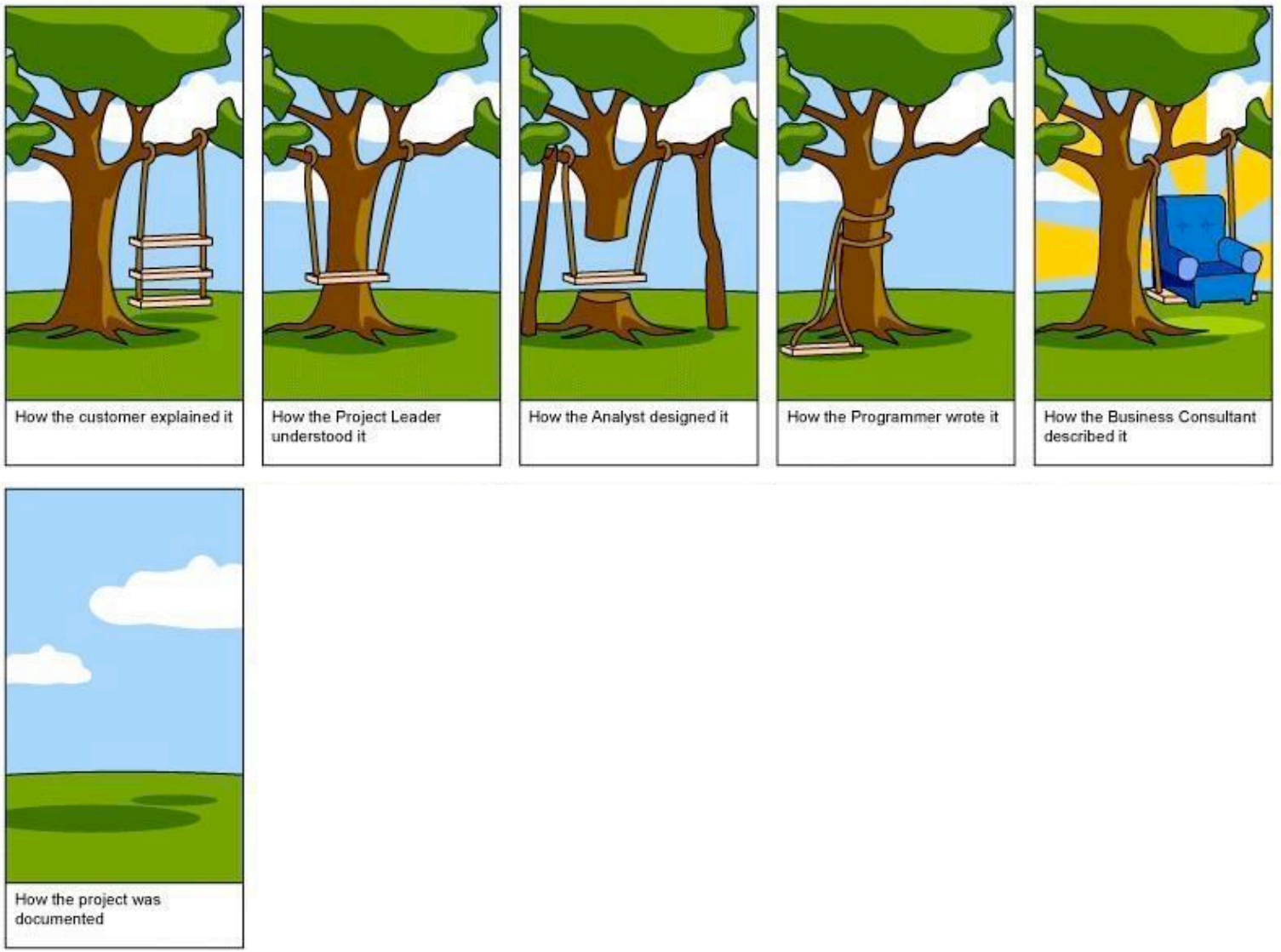
Ingegneria del software: scenario di riferimento



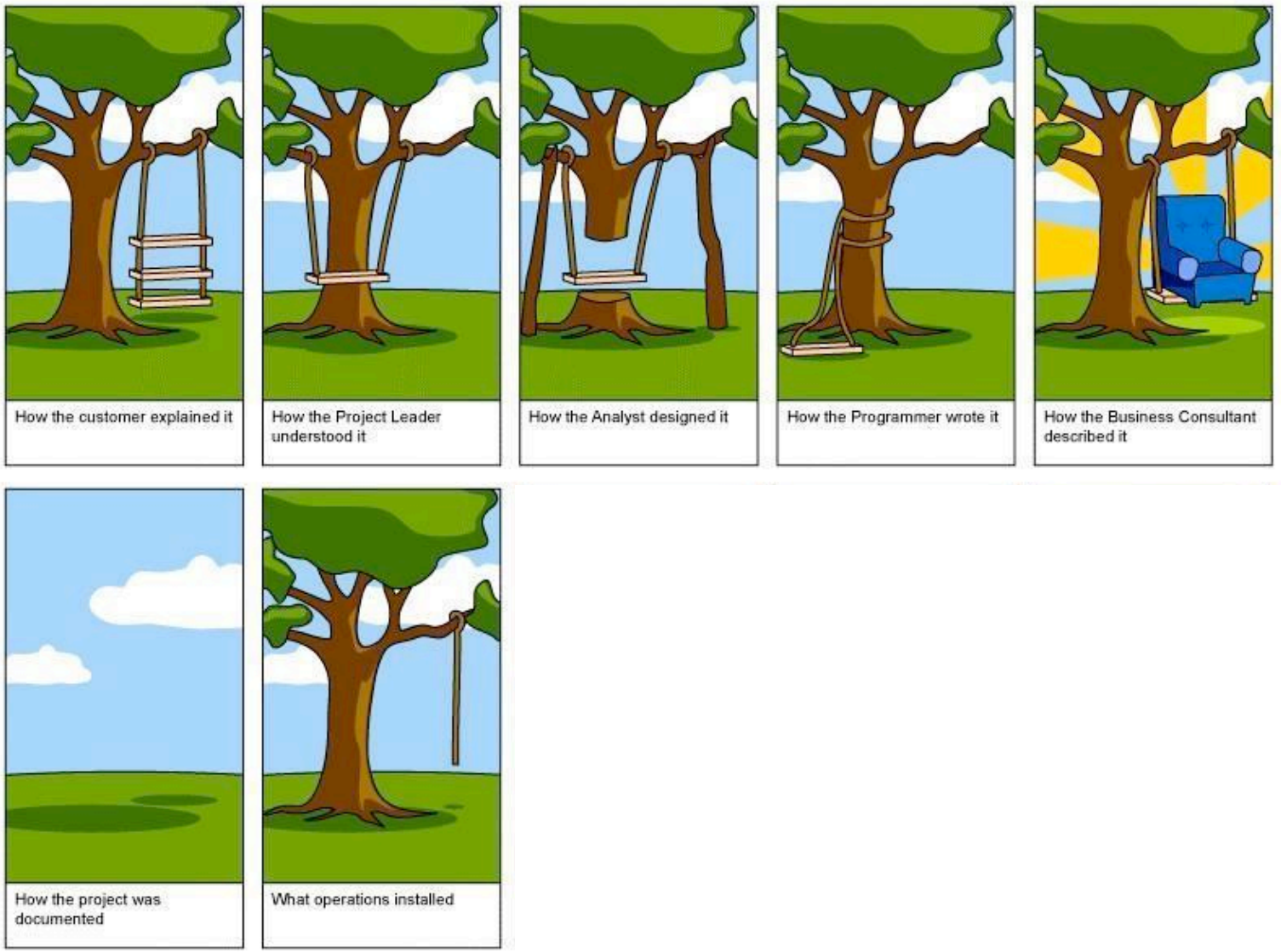
Ingegneria del software: scenario di riferimento



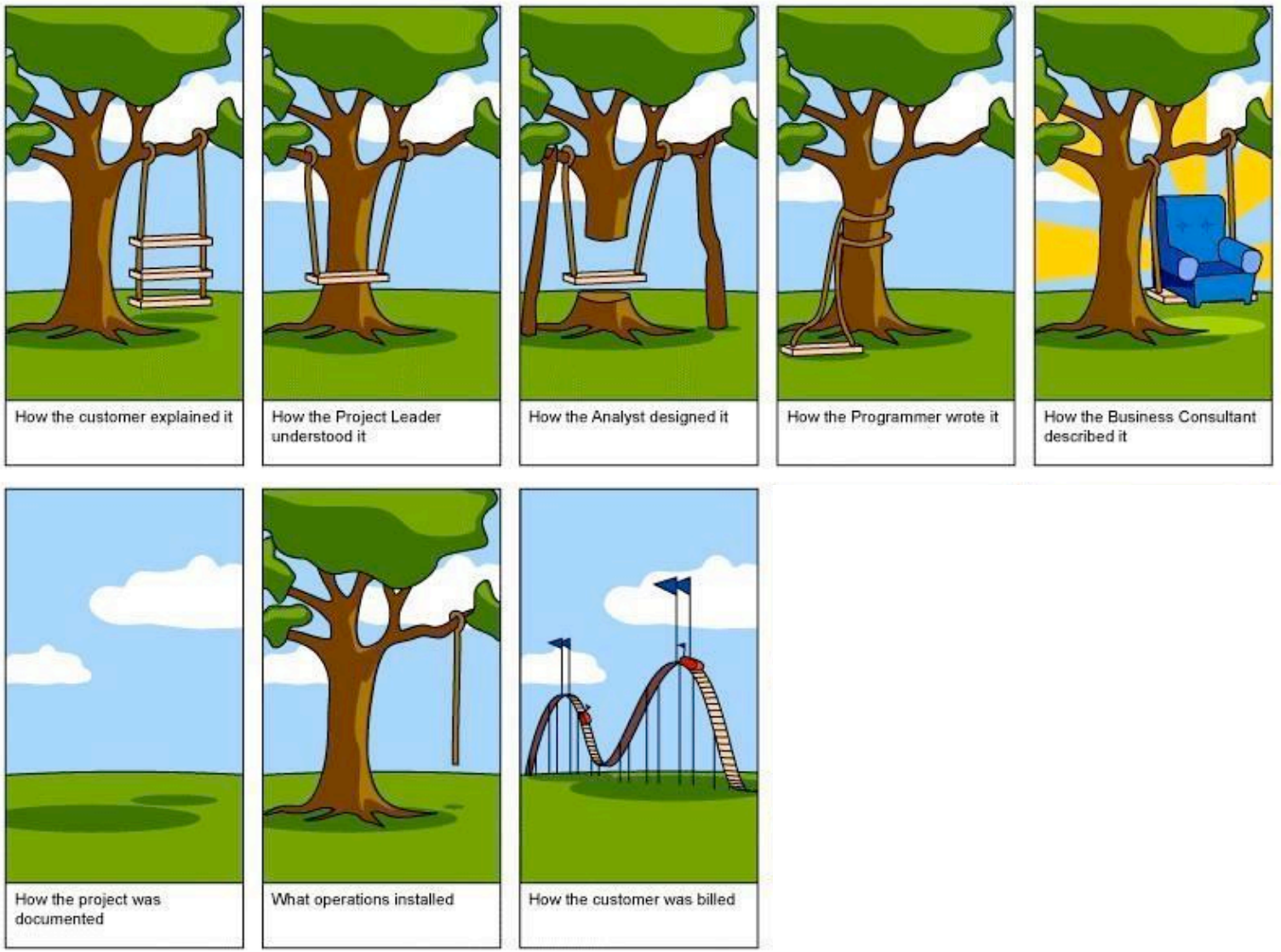
Ingegneria del software: scenario di riferimento



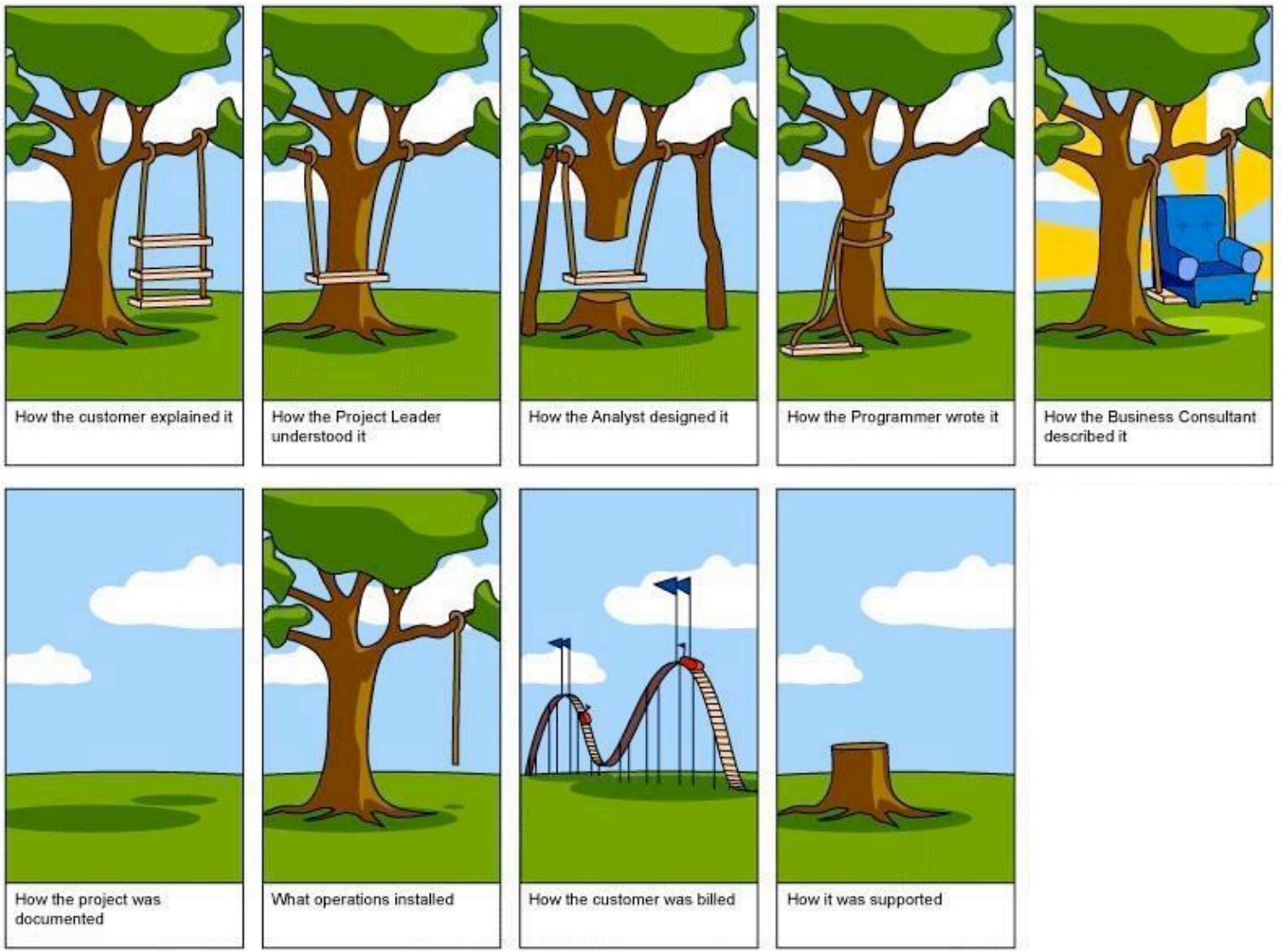
Ingegneria del software: scenario di riferimento



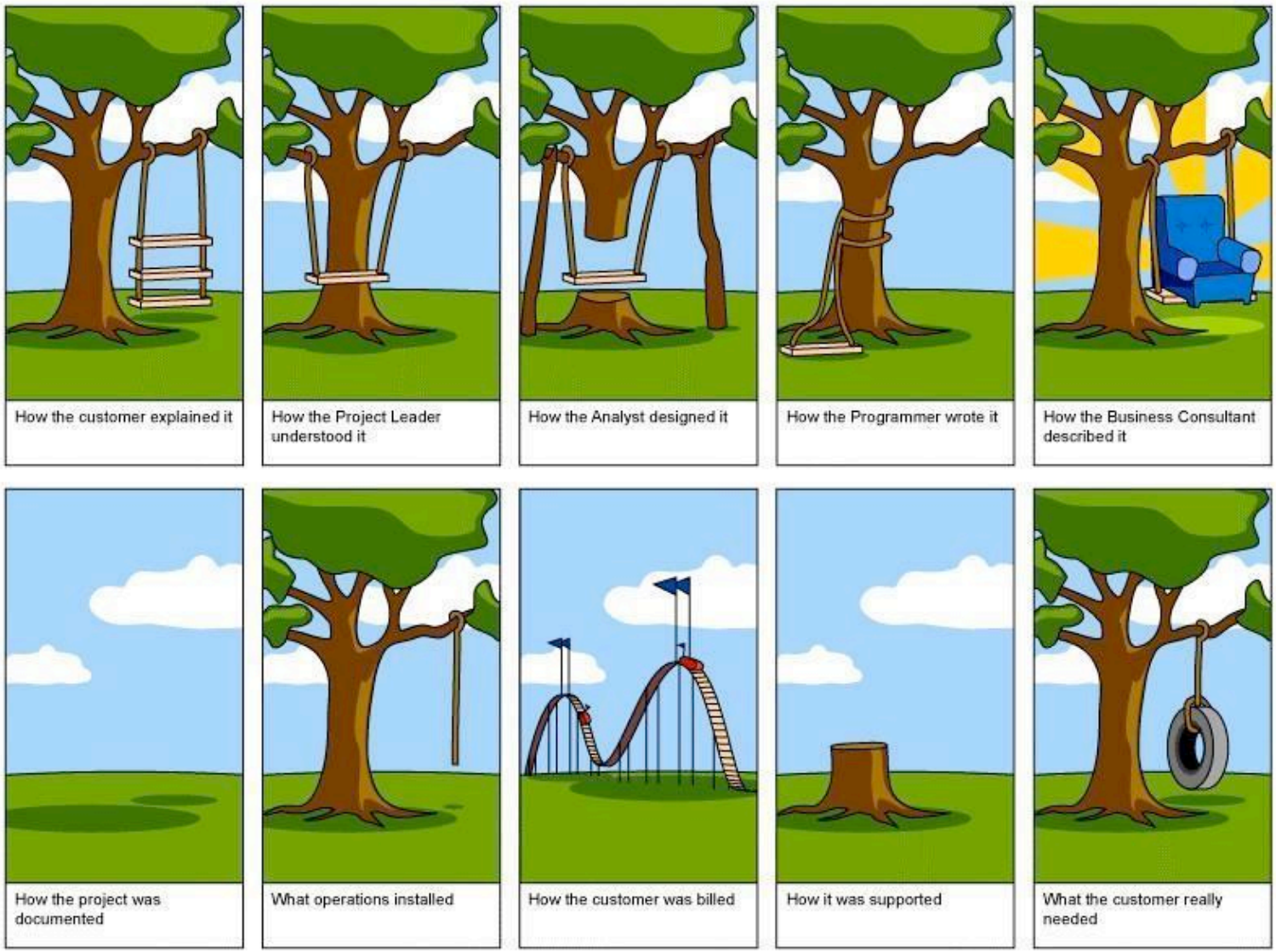
Ingegneria del software: scenario di riferimento



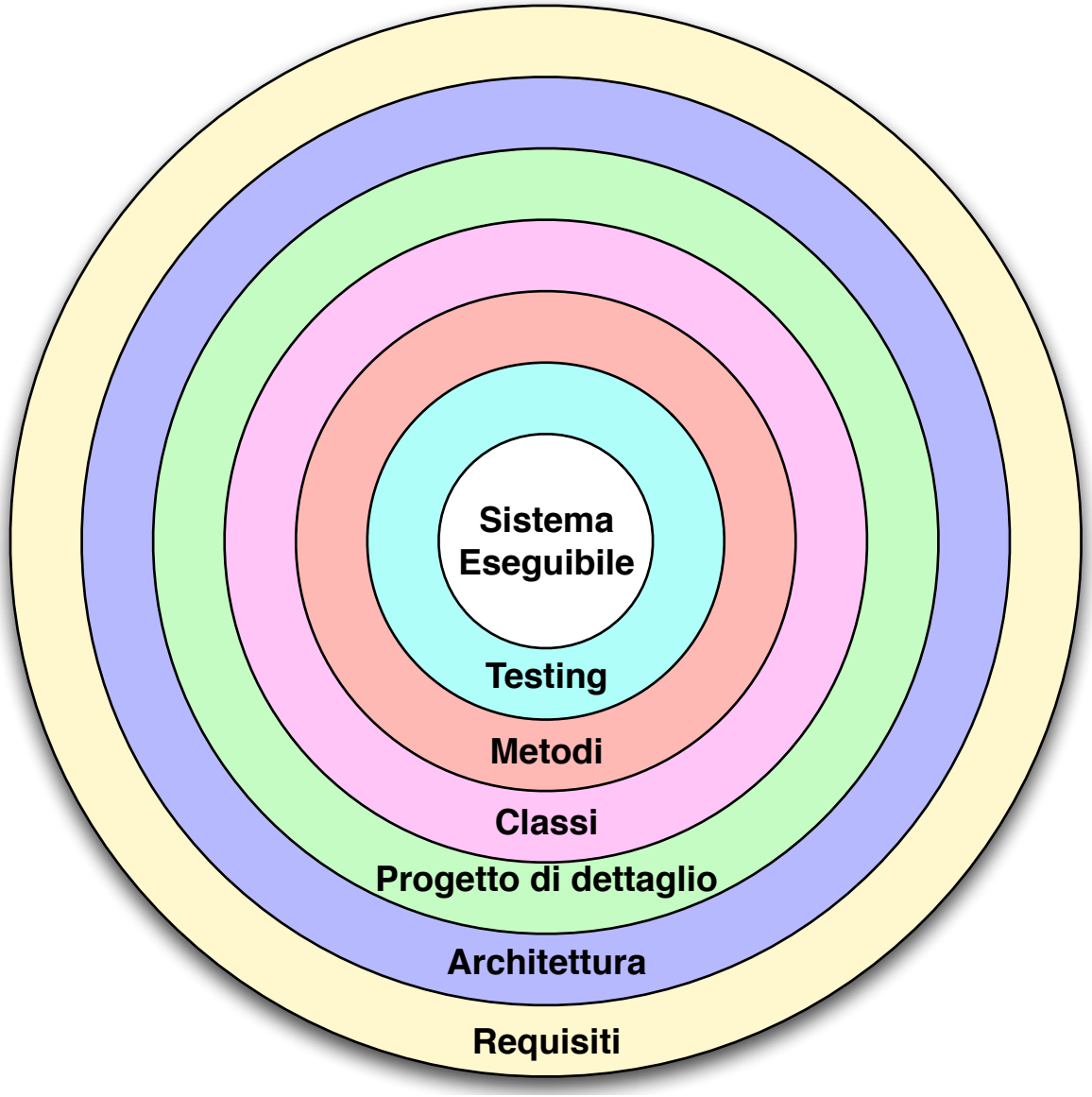
Ingegneria del software: scenario di riferimento



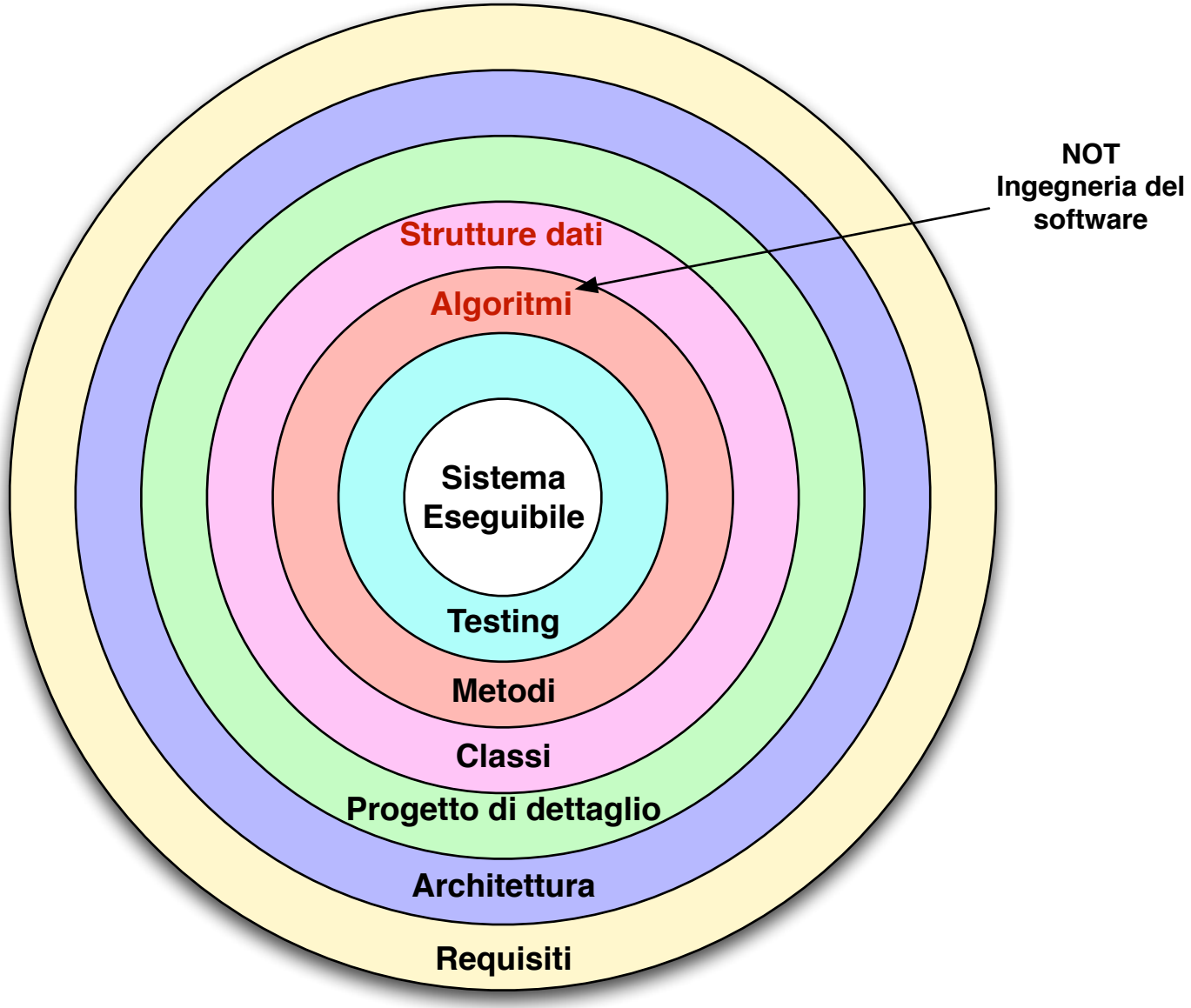
Ingegneria del software: scenario di riferimento



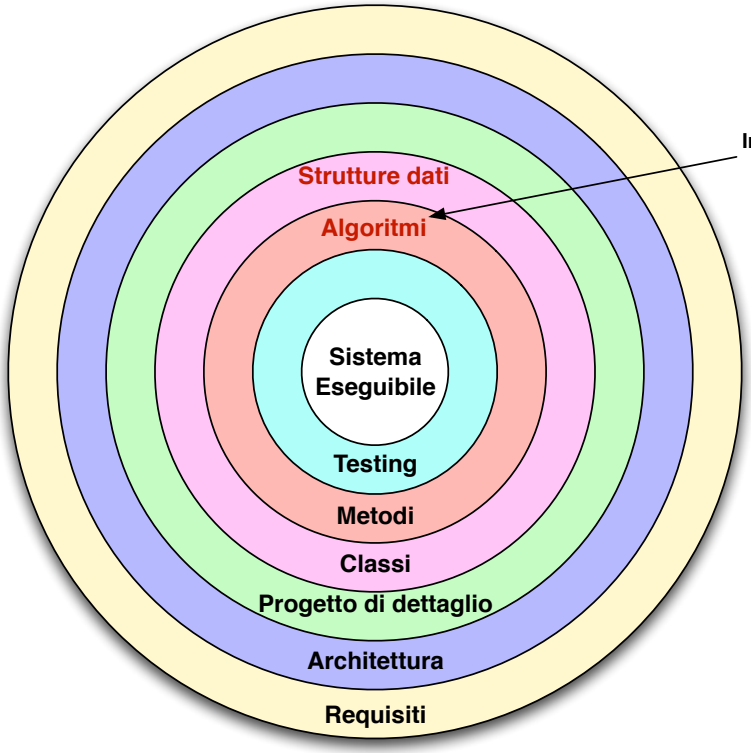
Software: il modello a cipolla



Il modello a cipolla/2

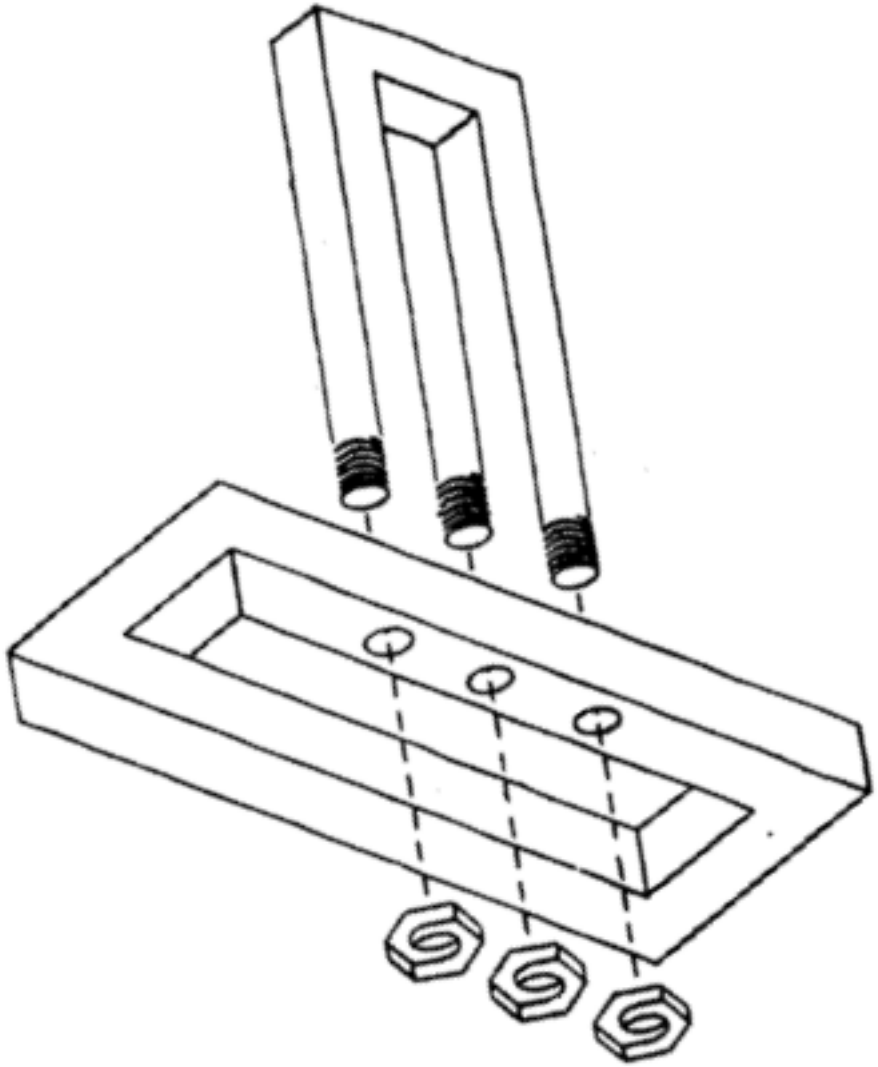


Il modello a cipolla/2

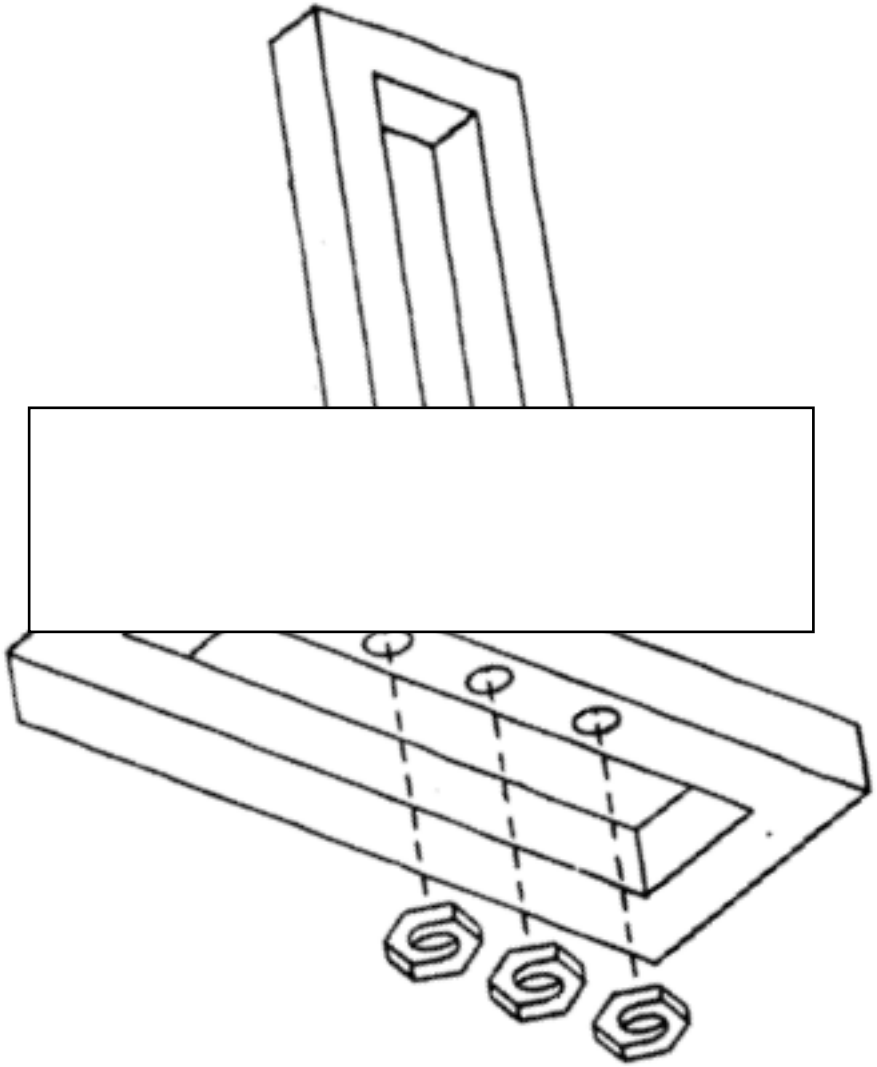


- Fino ad oggi avete studiato:
  - Algoritmi
  - Strutture dati
- Agli inizi dell'informatica bastavano a fare i 'programmi'
  - Wirth: Algoritmi +strutture dati = programmi
- L'ingegneria del software aggiunge altri 4 FONDAMENTALI strati e completa il modello

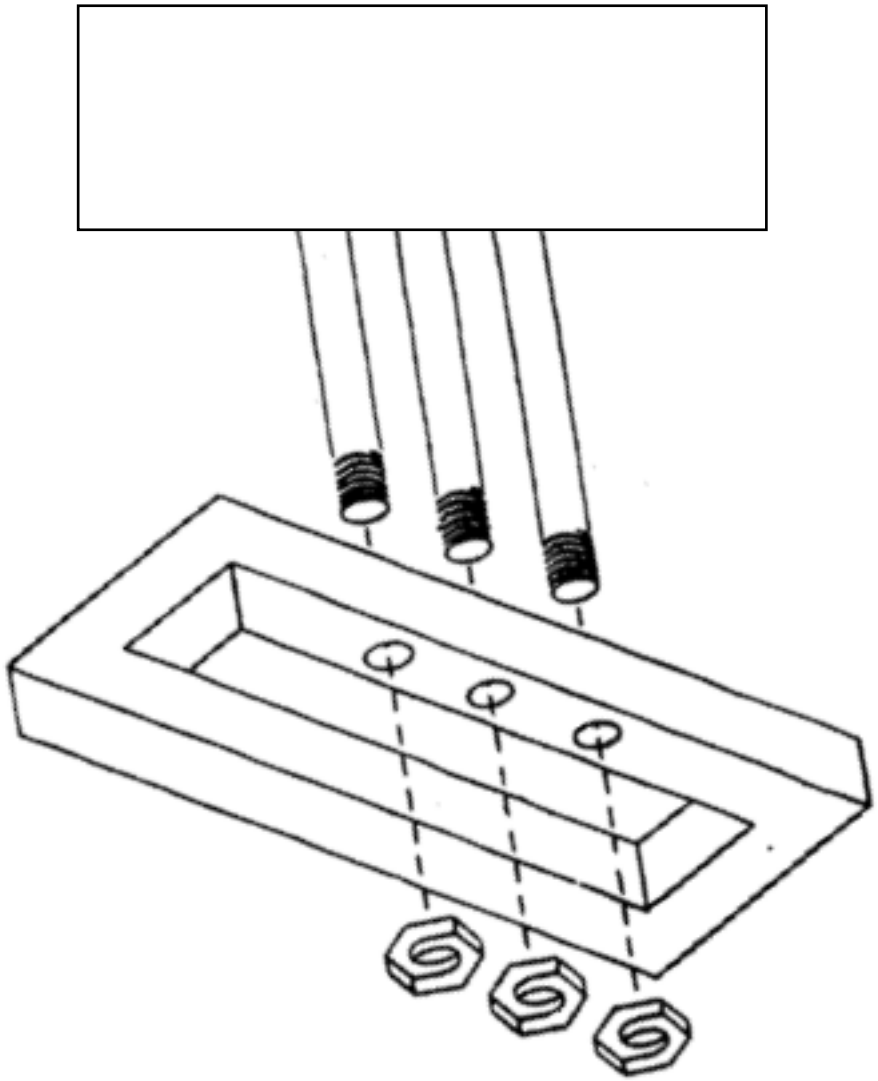
Can you develop this system?



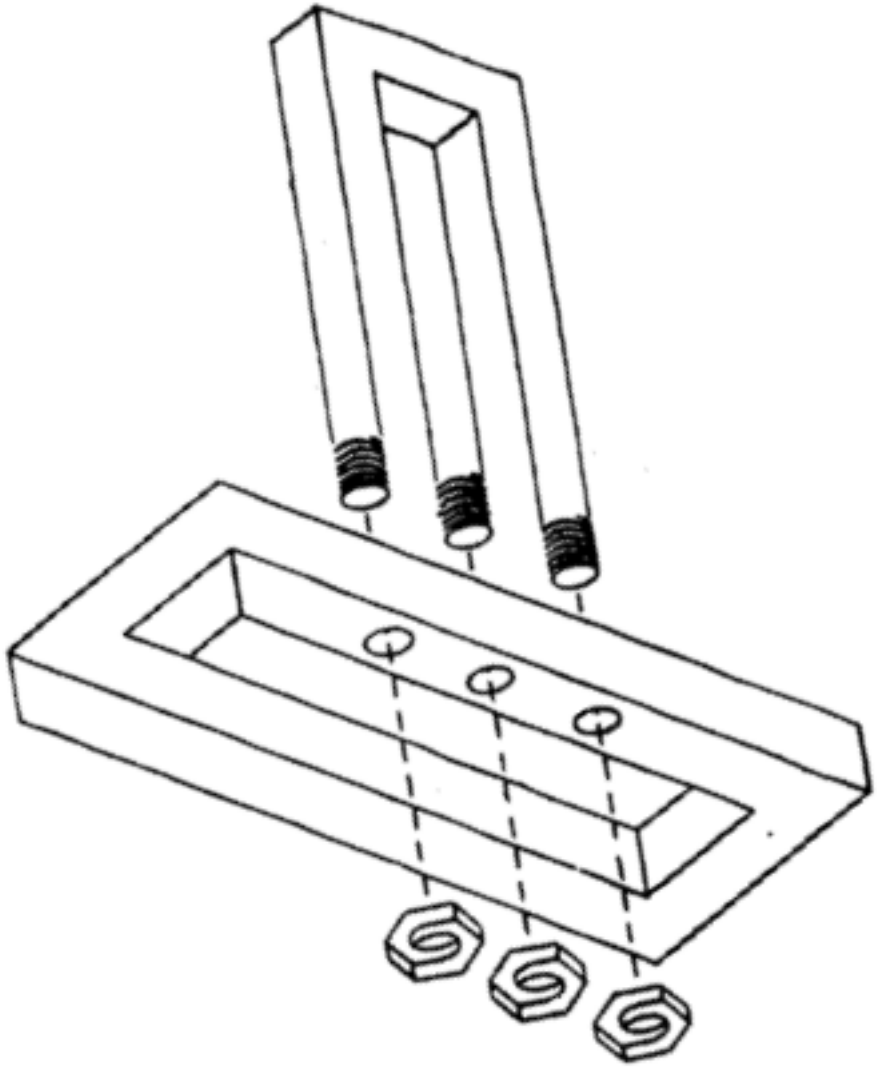
Can you develop this system?



Can you develop this system?



Can you develop this system?



The impossible Fork

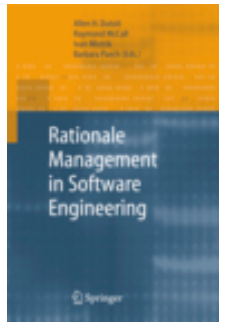
Why is Software Development difficult?

- The problem is usually ambiguous
- The requirements are usually unclear and changing when they become clearer
- The problem domain (called application domain) is complex, and so is the solution domain
- The development process is difficult to manage
- Software offers extreme flexibility
- Software is a discrete system
  - Continuous systems have no hidden surprises
  - Discrete systems can have hidden surprises!

David Lorge Parnas - an early pioneer in software engineering who developed the concepts of modularity and information hiding in systems which are the foundation of object oriented methodologies.

Software Development is more than just Writing Code

- It is problem solving
  - Understanding a problem
  - Proposing a solution and plan
  - Engineering a system based on the proposed solution using a ***good*** design
- It is about dealing with complexity
  - Creating abstractions and models
  - Notations for abstractions
- It is knowledge management
  - Elicitation, analysis, design, validation of the system and the solution process
- It is rationale management
  - Making the design and development decisions explicit to all stakeholders involved.



Computer Science vs. Engineering

- Computer Scientist
  - Assumes techniques and tools have to be developed.
  - Proves theorems about algorithms, designs languages, defines knowledge representation schemes
  - Has infinite time…
- Engineer
  - Develops a solution for a problem formulated by a client
  - Uses computers & languages, techniques and tools
- Software Engineer
  - Works in multiple application domains
  - Has only 3 months...
  - …while changes occurs in the problem formulation (requirements) and also in the available technology.

Software Engineering: A Working Definition

Software Engineering is a collection of techniques, methodologies and tools that help with the production of

A high quality *software* system developed with a given *budget* before a given *deadline* while *change* occurs

Challenge: Dealing with complexity and change

Software Engineering: A Problem Solving Activity

- **Analysis:**
  - Understand the nature of the problem and break the problem into pieces
- **Synthesis:**
  - Put the pieces together into a large structure

For problem solving we use techniques, methodologies and tools.

Course Outline

Dealing with Complexity

- Notations (UML, OCL)
- Requirements Engineering, Analysis and Design
  - OOSE, SA/SD, scenario-based design, formal specifications
- Testing
  - Vertical and horizontal testing

Dealing with Change

- Rationale Management
  - Knowledge Management
  - ~~Patterns~~
- Release Management
  - Configuration Management, Continuous Integration
- Software Life Cycle
  - Linear models
  - Iterative models
  - Activity-vs Entity-based views
- ~~Project Management~~