Lesson 1: Introduction

Corso di Software Engineering

Laurea Magistrale in Ing. Informatica

Università degli Studi di Salerno

Outline

- Course presentation
- Software Engineering

Course Presentation

- Goals
  - Knowledge
    - Software Life-Cycle
    - Models, methods and tools for large-scale software systems
    - Design Patterns
  - Abilities
    - Take part in a software construction process
    - Communicate requisites and design choices
    - Use software tools for testing and version management

Course presentation

- Prerequisites
  - Object Oriented Programming
  - Java Language
- Textbook
  - F. Tsui, O. Karam, B. Bernal: “Essentials of Software Engineering” (3rd ed.), Jones & Bartlett
  - Gamma, Helm, Johnson, Vlissides: “Design Patterns”, Pearson Education Italia
- Additional reading
  - B. Bruegge, A.H. Dutoit: “Object Oriented Software Engineering Using UML, Patterns and Java” (3° ed.), Pearson
  - I. Sommerville, “Software Engineering” (10° ed.), Pearson

Course presentation

- Project work
- Exam
  - Discussion of the project
  - Oral interview

On-line material

- Moodle:
- https://elearning.unisa.it/course/view.php?id=1536

Software Engineering

The “Software Crisis”

- NATO Software Engineering Conference, (Germany, 1968):
  - Projects completed over-budget
  - Projects completed over-time (or never)
  - Ever decreasing software quality
  - Software unable to satisfy the needs for which it had been commissioned
  - Ever increasing maintenance costs

The “Software crisis”

- Conclusions of the conference
  - Software development was still a recent and immature fields
  - Software construction required a sound, systematic approach like other engineering disciplines

Failure causes

- Increasing complexity
  - “as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem” (E. Dijkstra, 1972)
- But it is not the only cause...

The Chaos Report

- The Standish Group, 1995
- Only 16% of software projects were completely successful
- Top failure cases:
  - Lack of user input
  - Incomplete requirements and specifications
  - Changing requirements and specifications

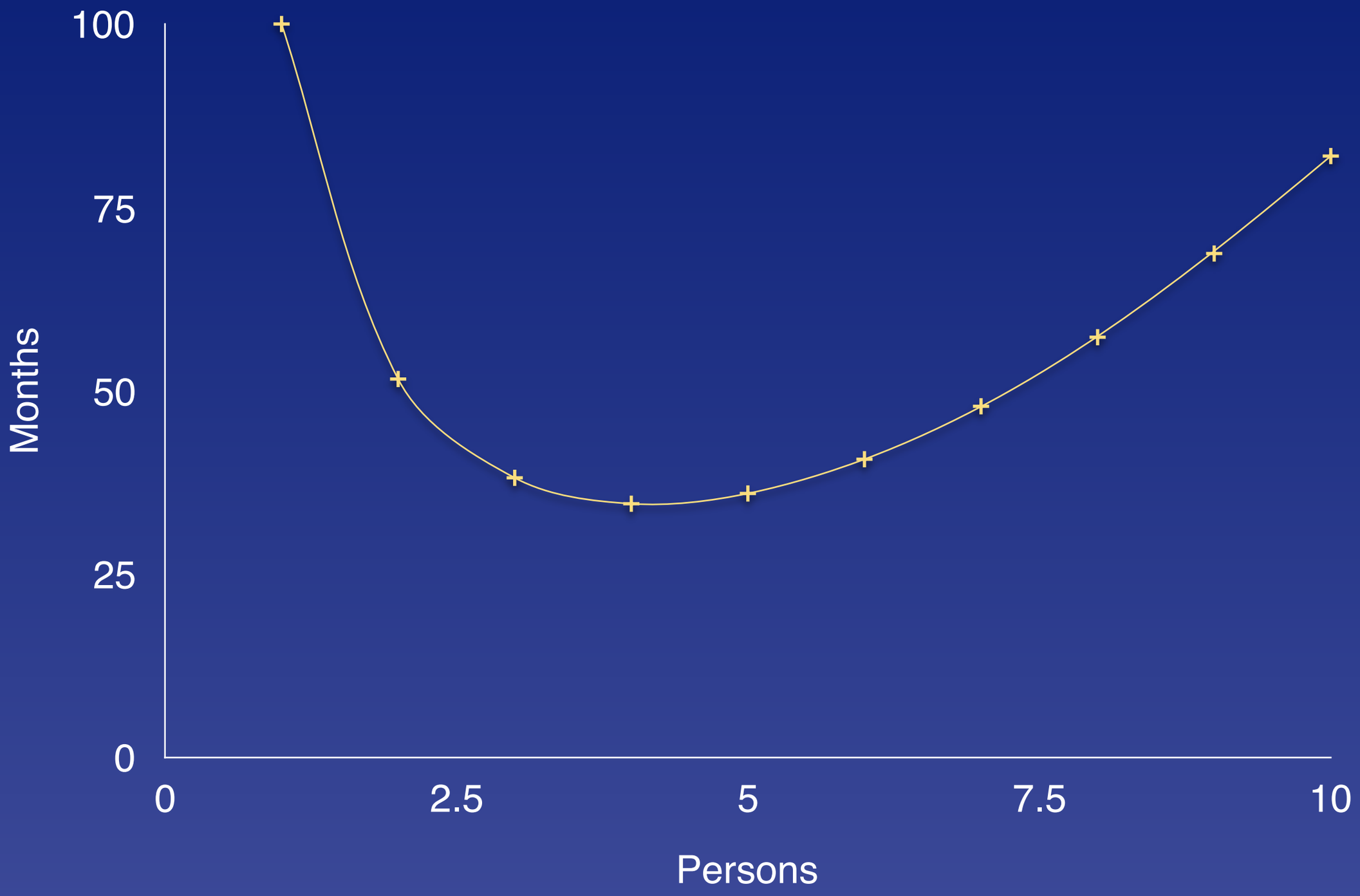
Brooks' Law

- Adding manpower to a late software project makes it later (F. Brooks, 1975)
  - Companies measure software cost in man-months
  - Implicit assumption: time and people are interchangeable
    - a 100 man-months projects can be completed by 10 persons in 10 months, or 25 persons in 4 months etc.
  - … but this assumption is wrong!

Brooks' Law

- Not every task can be evenly divided among several persons
- Increasing the number of persons working on a project increases the need of communications
  - communications proportional to:
    $N*(N-1)/2 = O(N^2)$

Brooks' Law



Brooks' Law

- Conclusions:
  - Software production depends on technical aspects, but also on social and organizative factors
  - Communication is a key factor
  - Often, project failure is due to errors in the management of non-technical factors

Common problems

- Cowboy programmers
  - If you need comments to understand my code, you are too stupid to read it.
  - My code is perfect, so why should test it?
  - My code is perfect, so I won't update it.
  - If the program crashes, it is a user fault.
  - etc...

May happen in small companies; this kind of guy cannot w in a team!

Common problems

- Lack of discipline is not the only problem:
  - difficult to communicate across hierarchy levels of the organization (e.g. shooting the messenger)
  - managers try to keep their subjects isolated from each other (e.g. mushroom management)
  - too much emphasis on formal, rather than substantial communication (e.g. see the CIA “Simple Sabotage Field Manual”)

Software Engineering

- An engineering discipline concerned with all aspects of software production
  - engineering:
    - appropriate usage of models, methods and tools
    - innovative solutions where existing ones are not appropriate
    - respect of constraints and effective usage of resources
  - production:
    - coding is just one of the technical activities involved
    - there are also non-technical activities

No Silver Bullet

- Periodically, new “magical” methodologies or tools are proposed as solution for the software problems
  - Structured Programming
  - Object-Oriented Programming
  - Visual IDE
  - Component-based Systems
  - Java
  - XML
  - .NET
  - Web Services and SOA
  - . . .

No Silver Bullet

- But software projects still fail...
- According to Brooks (1986) software complexity is due to:
  - Accidental factors (e.g. the available technology)
  - Essential factors (e.g. the inherent complexity of the problem)
- A new technology can only improve accidental factors
  - Never trust miracle solutions…

Conclusions

- Software production is a complex and (economically) risky endeavor
- Key factors are not only technical, but also human
- A systematic engineering approach is needed
- No simple and universal solution
- No substitute for the knowledge and experience of a good engineer