Programming Project: Introduction to Software Development

Guillermo Román Díez
groman@fi.upm.es

E.T.S. Ingenieros en Informática
Universidad Politécnica de Madrid

2019-2020

Norms

- The main goal of this subject is to familiarize the students with professional software development
- Though the main activity in the course is the development of a *Programming Project*, the final grade is composed by three parts:
    - A practical exercise about development tools and testing (in pairs)
        - Around the 6th week
        - 10% of the final grade
    - A multiple-choice test (January)
        - On January
        - 20% of the final grade
    - The development of a *programming project* in groups of 3-4 students
        - Starting around the 5th week and finishing in January
        - 70% of the final grade

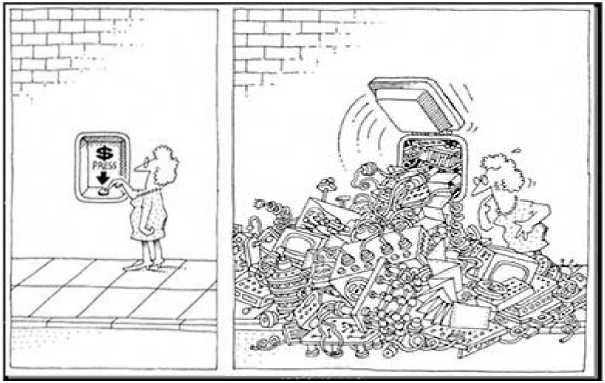
Illusion of Simplicity

The task of the software development team is to engineer the illusion of simplicity [1]

[1] Booch, 1993. *Object-Oriented Analysis and Design with Applications*

Illusion of Simplicity

The task of the software development team is to engineer the illusion of simplicity [1]



[1]Booch, 1993. *Object-Oriented Analysis and Design with Applications*

Software Engineering

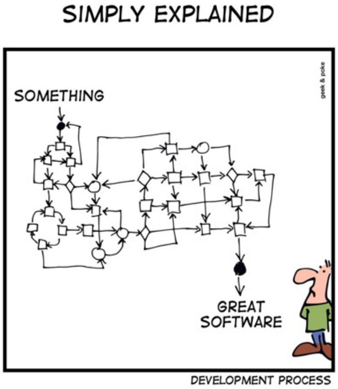
Definition (Software Engineering)

The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software

Software Engineering

Definition (Software Engineering)

The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software



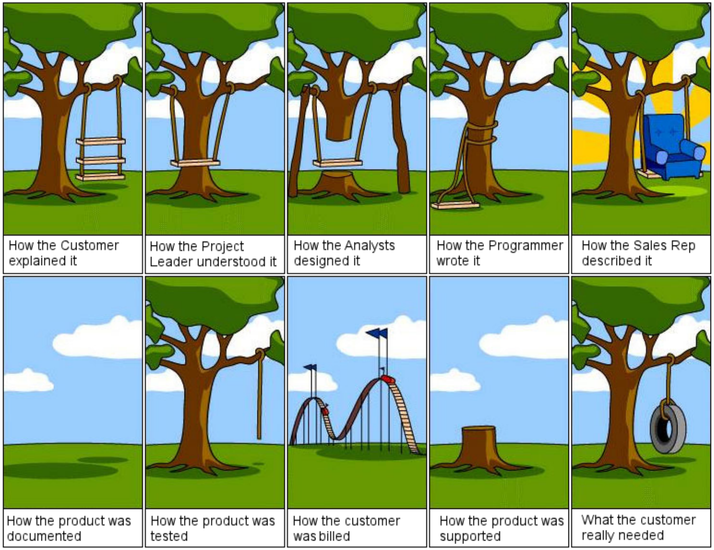
Software Engineering

- An engineering process implies the use of well understood techniques in a systematic way
  - Computer science is a young discipline and these techniques are evolving very fast
- Software projects always have multiple constraints
  - Time, budget, knowledge, human resources, customer requirements, . . .
  - Do not hold these constraints might led to project delays or cancellation
- Large software systems cannot be understood by one person
  - Teamwork is crucial for the success of the project
  - Forget the idea of having a *guru* who perfectly knows the whole system

Some Typical Problems

- Poor end-user description of to the project or inaccurate understanding of the customer needs
- Difficulties to deal with changing requirements
- Low performance of the software running in production
- Software hard to maintain or extend
  - Bad designed, poorly documented, . . .
- Software not properly tested (when it is tested. . .)
  - Some inputs are not covered by the test suites
  - It contains serious flaws (i.e. parts badly integrated)
- Collaboration problems
  - Teams not well organized with communication problems
  - Impossible to reconstruct who did something (what, when, why . . .)

Problems: Graphically


How the Customer explained it
How the Project Leader understood it
How the Analysts designed it
How the Programmer wrote it
How the Sales Rep described it
How the product was documented
How the product was tested
How the customer was billed
How the product was supported
What the customer really needed

Some software development good practices

- **Testing must be a first-class citizen**
  - Test continuously your project from end to end

Some software development good practices

- **Testing must be a first-class citizen**
  - Test continuously your project from end to end
- **Strive to keep your code simple**
  - Reduce unnecessary complexity in your software

Some software development good practices

- **Testing must be a first-class citizen**
  - Test continuously your project from end to end
- **Strive to keep your code simple**
  - Reduce unnecessary complexity in your software
- **Check the quality of the code**
  - Allow other people to check your code and use static analysis tools

Some software development good practices

- **Testing must be a first-class citizen**
  - Test continuously your project from end to end
- **Strive to keep your code simple**
  - Reduce unnecessary complexity in your software
- **Check the quality of the code**
  - Allow other people to check your code and use static analysis tools
- **Be realistic with the project estimation**
  - Bad estimations cause problems in quality, morale and output

Some software development good practices

- **Testing must be a first-class citizen**
  - Test continuously your project from end to end
- **Strive to keep your code simple**
  - Reduce unnecessary complexity in your software
- **Check the quality of the code**
  - Allow other people to check your code and use static analysis tools
- **Be realistic with the project estimation**
  - Bad estimations cause problems in quality, morale and output
- **Control the changes**
  - Use version control systems to track the project

Some software development good practices

- **Testing must be a first-class citizen**
  - Test continuously your project from end to end
- **Strive to keep your code simple**
  - Reduce unnecessary complexity in your software
- **Check the quality of the code**
  - Allow other people to check your code and use static analysis tools
- **Be realistic with the project estimation**
  - Bad estimations cause problems in quality, morale and output
- **Control the changes**
  - Use version control systems to track the project
- **Integrate continuously**
  - It helps to early detect and correct defects

Some software development good practices

- **Testing must be a first-class citizen**
  - Test continuously your project from end to end
- **Strive to keep your code simple**
  - Reduce unnecessary complexity in your software
- **Check the quality of the code**
  - Allow other people to check your code and use static analysis tools
- **Be realistic with the project estimation**
  - Bad estimations cause problems in quality, morale and output
- **Control the changes**
  - Use version control systems to track the project
- **Integrate continuously**
  - It helps to early detect and correct defects
- **Document your code**
  - Think in the rest of the world and in your future self

Outline of the Subject

1. Development Tools
   - `GIT (GitLab)`, `Maven`
2. Software Quality
   - `SonarQube`
3. Testing
   - Testing introduction
   - Automated Testing (`JUnit`)
   - Test Driven Development
4. Software Development
   - Continuous Integration
   - Agile and eXtreme Programming Ideas
5. Software Design
   - Design principles
   - Design Patterns
6. Project development