Software Engineering

A Brief (and fast) Overview

ISEP / LETI / ESOFT

Topics

- Software Engineering Definition
- Software Development
- Software Development Process

Software Engineering Definition

What does Software stands for?

- Looking to some online definitions:
  - “[set of] instructions that tell a computer what to do” [1]
    - “A set of instructions […] is called a program, or software program” [1]
    - “A program is an executable code, which serves some computational purpose” [2]
  - “comprises the entire set of programs, procedures, and routines associated with the operation of a computer system” [1]
  - “the programs used to direct the operation of a computer, as well as documentation giving instructions on how to use them” [3]
  - “includes computer programs, libraries and related non-executable data, such as online documentation or digital media” [4]
  - “when made for a specific requirement is called software product” [2]

What does Engineering stands for?

- Looking to some online definitions:
  - “making practical application of the knowledge of pure sciences” [5]
  - “use of science and math to design or make things [… by] engineers” [6]
    - “[Engineers] are professionals who invent, design, analyze, build and test machines, complex systems, structures, gadgets and materials” [7]
    - “[fulfilling] functional objectives and requirements while considering the limitations imposed by practicality, regulation, safety and cost” [7]
  - “the creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination” [8]
  - “is all about developing products, using well-defined, scientific principles and methods” [2]

What does Software Engineering stands for?

- Looking to some online definitions:
  - “a branch of computer science that deals with the design, implementation, and maintenance of complex computer programs” [9]
  - “is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures” [2]
  - “the outcome [...] is an efficient and reliable software product” [2]

What does Software Engineering stands for?

- Looking to more formal definitions:
  - “the application of a systematic, disciplined [...] approach to the development, operation, and maintenance of software” [10]
  - “an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use” [11]
    - By applying “theories, methods and tools where these are appropriate” [11]
    - “is not just concerned with the technical processes of software development but also with activities such as software project management and with the development of tools, methods, and theories to support software production” [11]

As so, Software Engineering (SE)…

- (Engineering) implies
  - Adopting a systematic approach
  - Applying sciences and scientific principles, methods and tools
  - To develop/construct a software product
  - Meeting requirements/limitations
- Software as a product
  - Comprises executable and non-executable data
  - Satisfies specific requirements/limitations
- Concerned with project management too

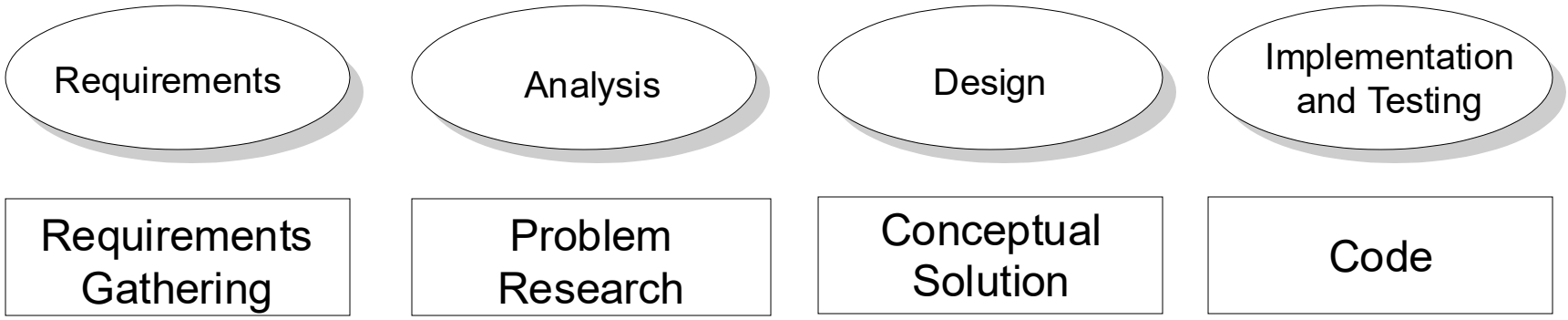
Relation between SE and other disciplines

- Computer Science
  - provides the scientific foundation for software development
- Systems Engineering
  - provides the foundation to integrate and operate the software as a component of much larger and complex system
- Management and Economic Science
  - provide the foundation to support, manage, control and allocate the human and financial resources needed

Software Development

SW Engineering is more than programming

- Knowing a programming language is necessary but not enough to create (good) software
- Between a good idea and good software there is much more than programming, namely:



But also…

- Documentation and information sharing during the process
- Good practices of requirements, analysis, design, coding, testing, …
- Human Factor/Skills:
  - Teamwork (soft skill)
  - Communication (soft skill)

Software as a Product

- It is intangible
- It is unusually flexible
- Many software projects are unique

But which are the qualities of a good software?

Qualities of a good software

Metric	Description
Correctness	the degree to which each software adheres to the specified requirements
Portability	the degree of how easy each software can be used in different computer configurations
Reusability	the degree of how easy each software can be reused in the development of other software
Reliability	the frequency and criticality of software failures - a failure is an unacceptable effect or behavior occurring under permitted operating conditions
Maintainability	the degree of how easy changes can be made to the software to meet new requirements and/or correct deficiencies
Efficiency	the degree to which the software fulfills its purpose without wasting resources
…	…

Emphasis on these metrics varies from one project to another

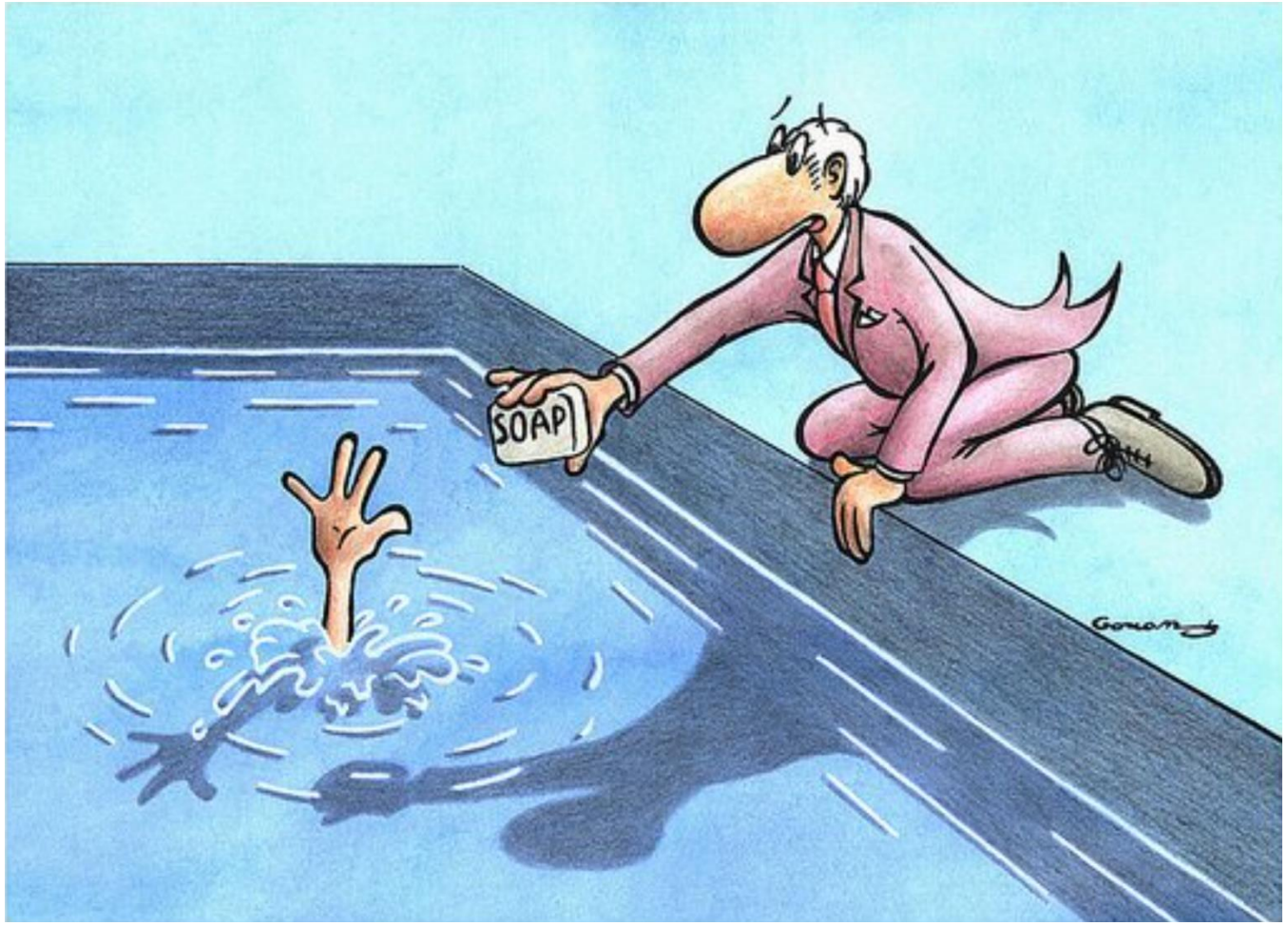
Some problems in SW development

- Misunderstood business
  - Without direct communication, the development team starts guessing what the customer wants/needs
- Business changes
  - New laws, market changes, business priorities
- High defect rates
  - Software is put into production, but the defects are so high that it is not used
- Outdated system
  - Software is deployed to production, but after a while, costs of changes and of repairing defects dictate that the system needs to be replaced

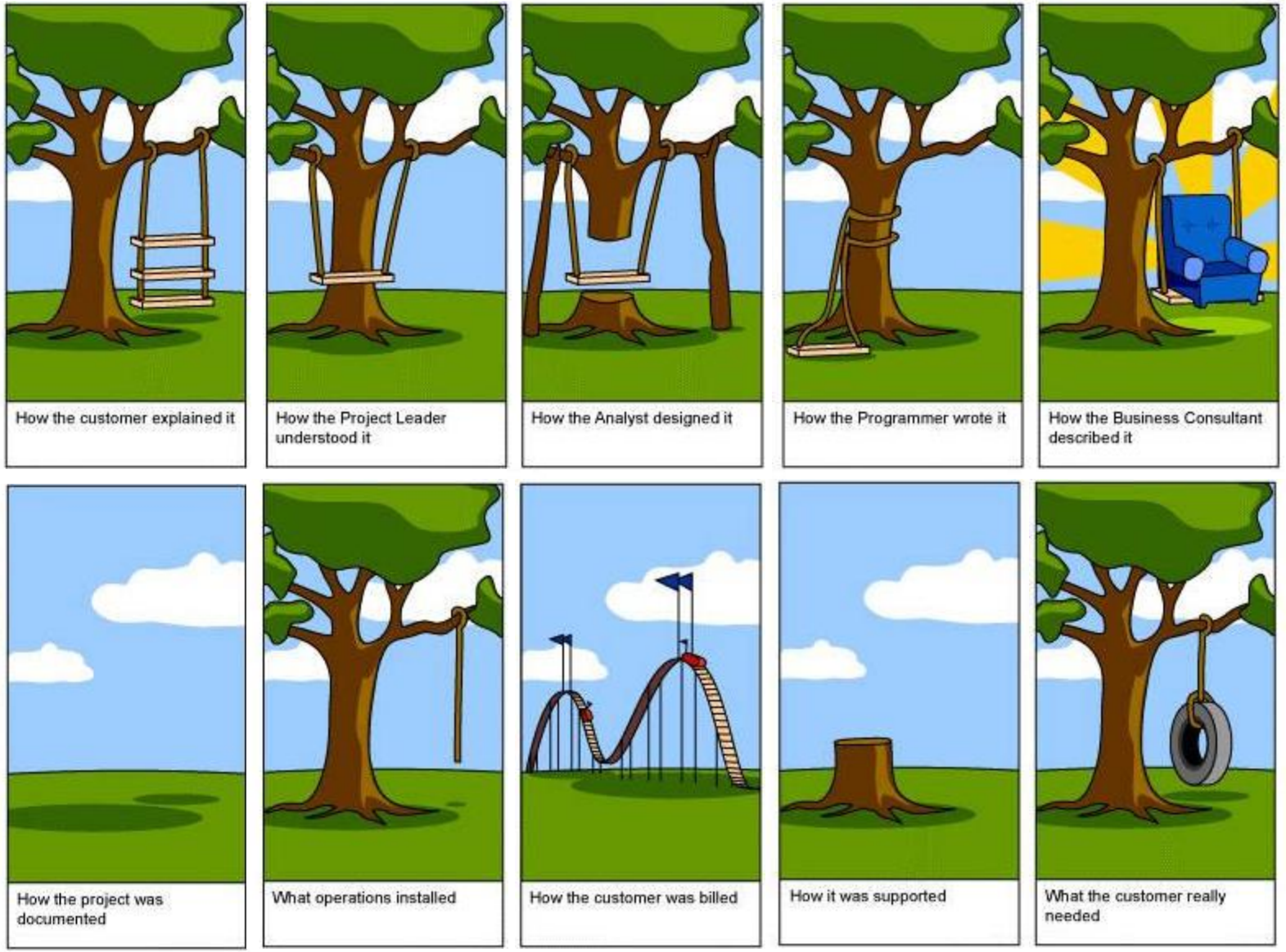
Difficulties and Traps of SW development

Lack of context
Difficulty on identifying the users
Customer/users do not know what they really want/need
Failure on producing/getting value from the SW
Difficulties on prioritizing needs
Difficulties on estimating time and resources
If documents are too long... are not read
If documents are too technical... are not read
Difficulty on developing software incrementally
Feedback cycles are too long
Implementation criterion is “everything is implemented”
Long time-to-market
Difficulty on doing maintenance

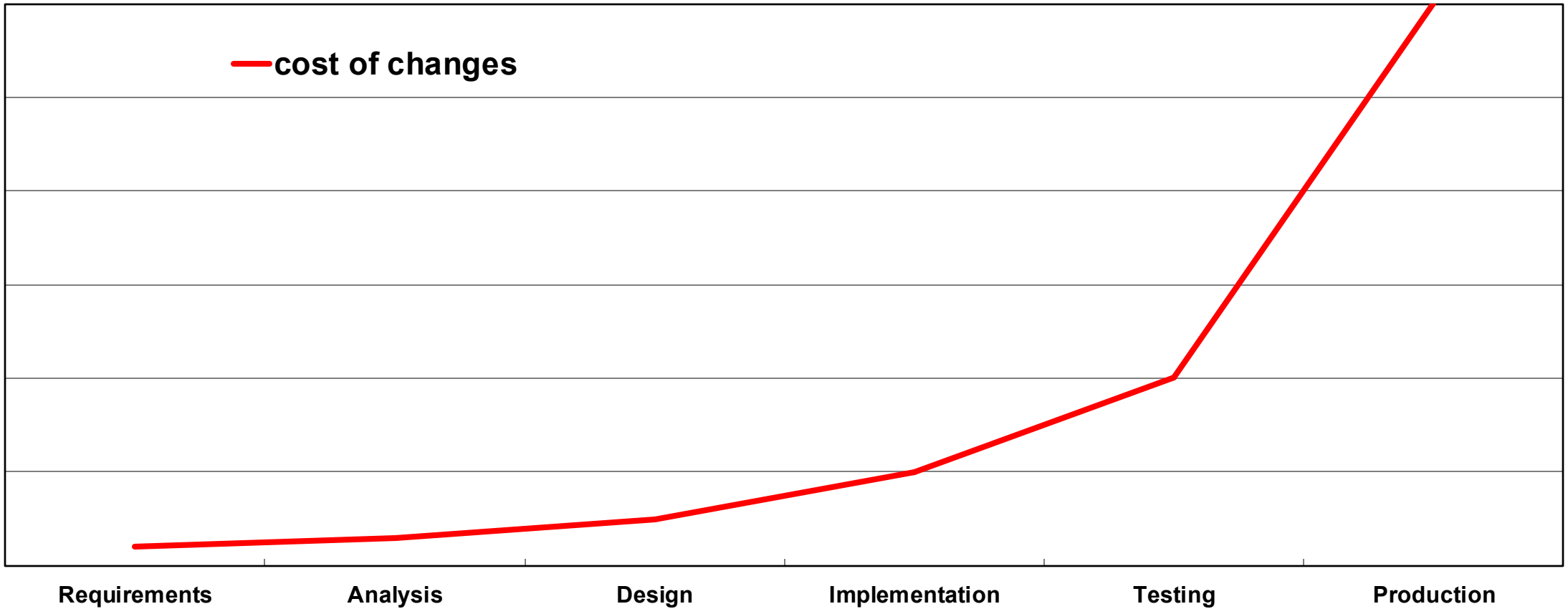
Communication (Failure)



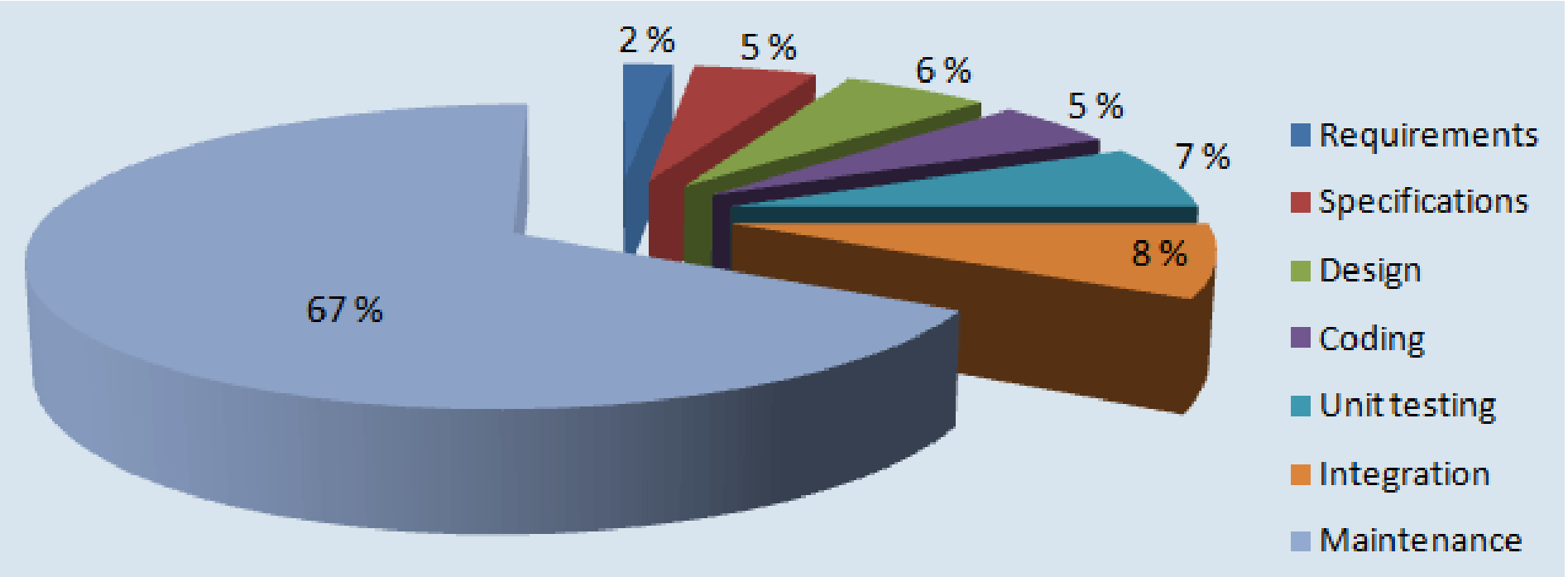
Metaphor / Perspectives



Costs of changing Software



Costs Per activity



Source: Digital Aggregates

…Considering these issues and…

- Software systems are increasingly important in our daily lives
- Software development failures can cost much more than the cost of the software and the time spent

- Effective software engineering is therefore fundamental and urgent
  - Introducing the notion of process / method / model
  - To impose a disciplined and well-defined process on SW development
  - More effective and efficient

Software Development Process

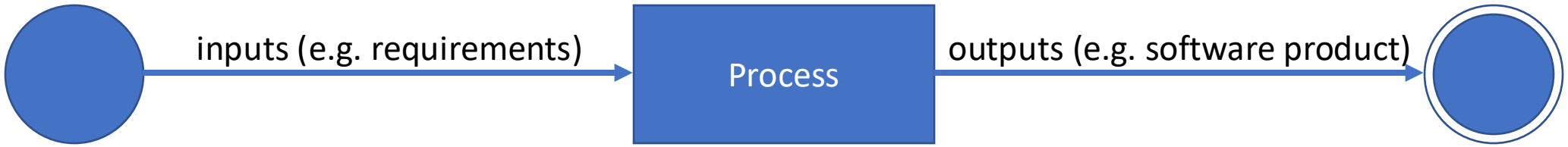
Why do we need SDP for SW development?

- Software development easily becomes a chaotic activity characterized by “coding and composing”
- Software is written without an underlying plan, becoming full of short-term decisions

- This (might) work while software is small and has little complexity
  - As software size and complexity grows, it becomes increasingly difficult to add new features
  - Bugs become increasingly relevant and more difficult to solve

What is a process?

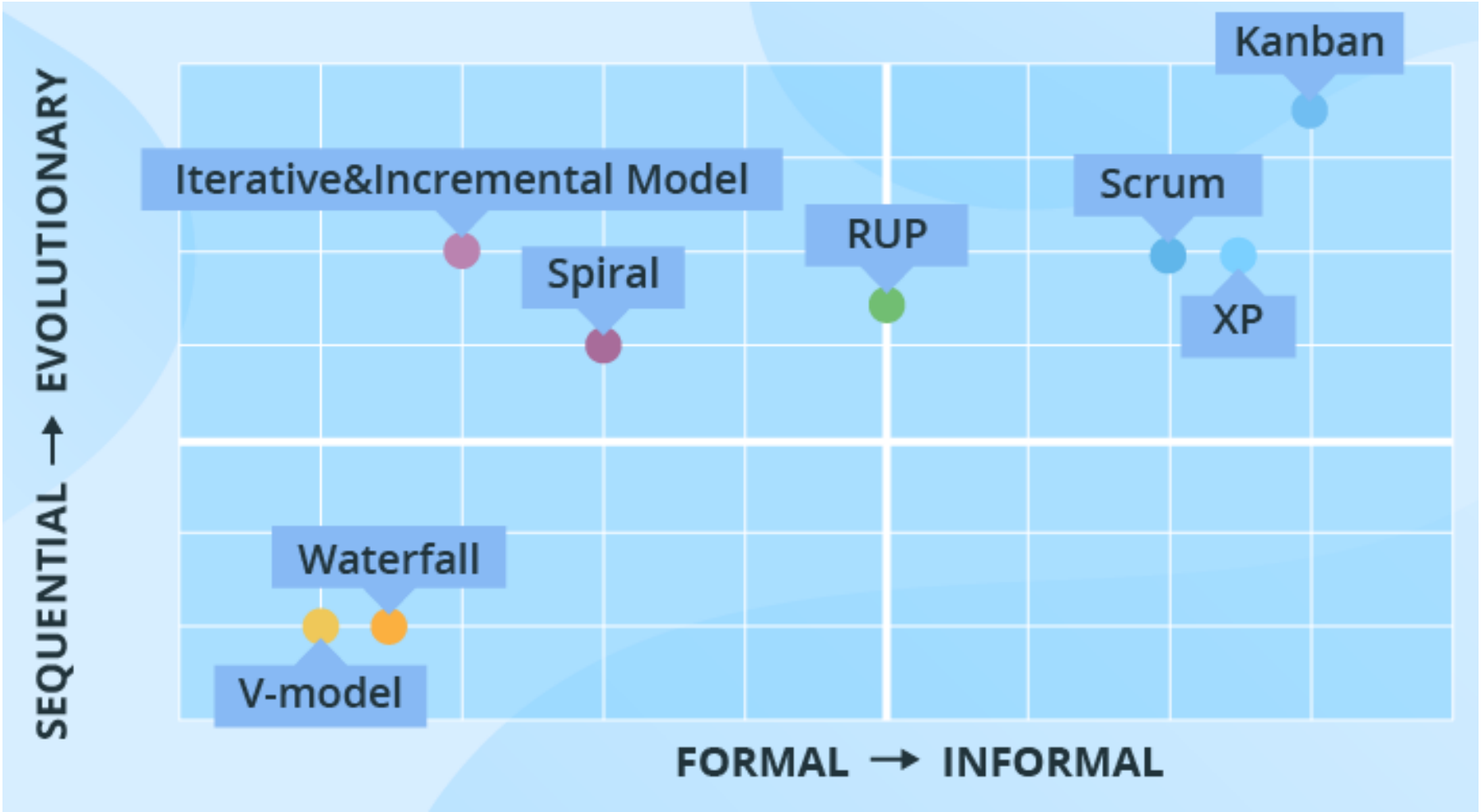
- A specification defining **who** does **what**, **when** and **how** in order to meet a given goal
- It comprehends a **set of interrelated activities** that transform a set of inputs in a set of outputs



Software Development Process

- Coherent sequence of practices systematizing the development or evolution of software systems
- Set of activities, restrictions and resources that produce a desired result
  - Each activity must have a clear input and a clear output criterion
- Structures the activities by establishing an order or sequence
  - Set of principles or criteria that explain the objectives of each activity
  - Decision points in planning
  - Synchronization points for the collaborative work of team

Some popular SW development processes



Extracted from [12]

Core Software Engineering Activities (1/6)

- Requirements gathering
- Feasibility study
- Analysis
- Design
- Implementation
- Testing
- Deployment
- Configuration management
- Operations and Maintenance
- Project management

Core Software Engineering Activities (2/6)

- Requirements gathering
- Feasibility study
- Analysis
- Design
- Implementation
- Testing
- Deployment
- Configuration management
- Operations and Maintenance
- Project management

+ Human Factors

(teamwork and communication)

Core Software Engineering Activities (3/6)

- **Requirements gathering**
- ~~Feasibility study~~
- **Analysis**
- **Design**
- **Implementation**
- **Testing**
- ~~Deployment~~
- ~~Configuration management~~
- ~~Operations and Maintenance~~
- ~~Project management~~

+ Human Factors

(teamwork and communication)

Core Software Engineering Activities (4/6)

- **Requirements gathering**
  - Aims at identifying and eliciting the requirements of the software product to be developed with the stakeholders
- Feasibility study
  - Aims at assessing if the intended software project is feasible in terms such as technologically, financially and practically
- **Analysis**
  - Aims at getting a deeper understanding of the scope of the project, related domain business and development constraints
- **Design**
  - Aims at outcoming a conceptual solution that enables to fulfill the software requirements, comprising artifacts from a coarser granularity (i.e., architectural level) to a finer granularity

Core Software Engineering Activities (5/6)

- **Implementation**
  - Aims at the programming/coding of the software program in a suitable language and in accordance with the design
- **Testing**
  - Aims at the verification and validation of the developed software by specifying and conducting different kinds of tests (e.g. unitary, integration, system (end-to-end), acceptance, …)
- Deployment
  - Aims at installing (and integrating) the software in a computational environment (e.g. personal computer) in order to be used/operated by its users

Core Software Engineering Activities (6/6)

- Configuration management
  - Aims at managing, organizing and controlling changes in items
    - e.g. requirements, government policy and rules, code, project schedule/resources) that may affect the final product
- Operations and Maintenance
  - Aims at ensuring and keeping the software operating properly throughout time by, for instance, carrying out users training, fixing bugs/errors, updating the software to fulfill environmental and/or technological changes
- Project management
  - Aims at planning, scheduling, doing resource allocation, checking the execution progress of the software

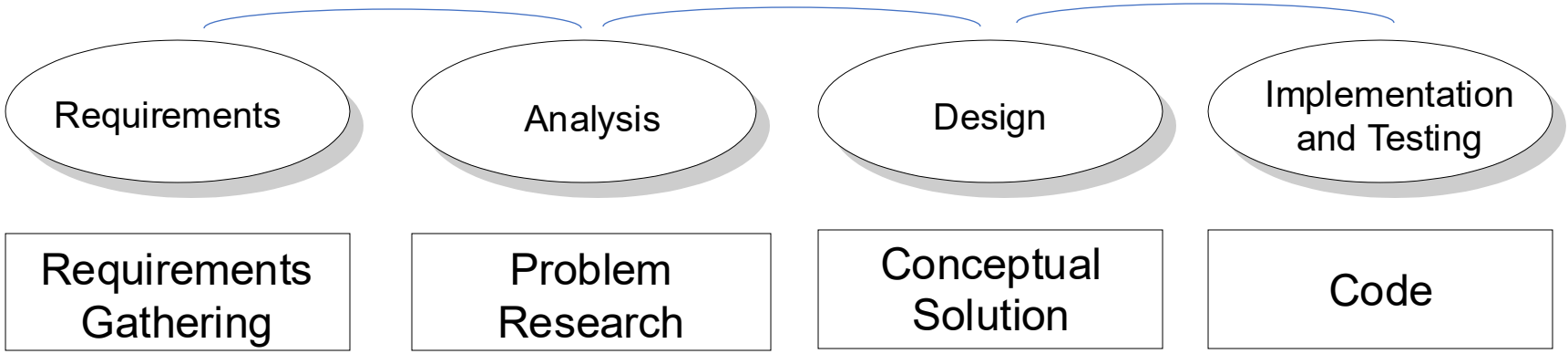
Human Factors (Soft Skills)

- Teamwork
  - Software development involves and requires an effective cooperation and communication between several and distinct participants
    - e.g. customer, stakeholders, users, development team, project manager

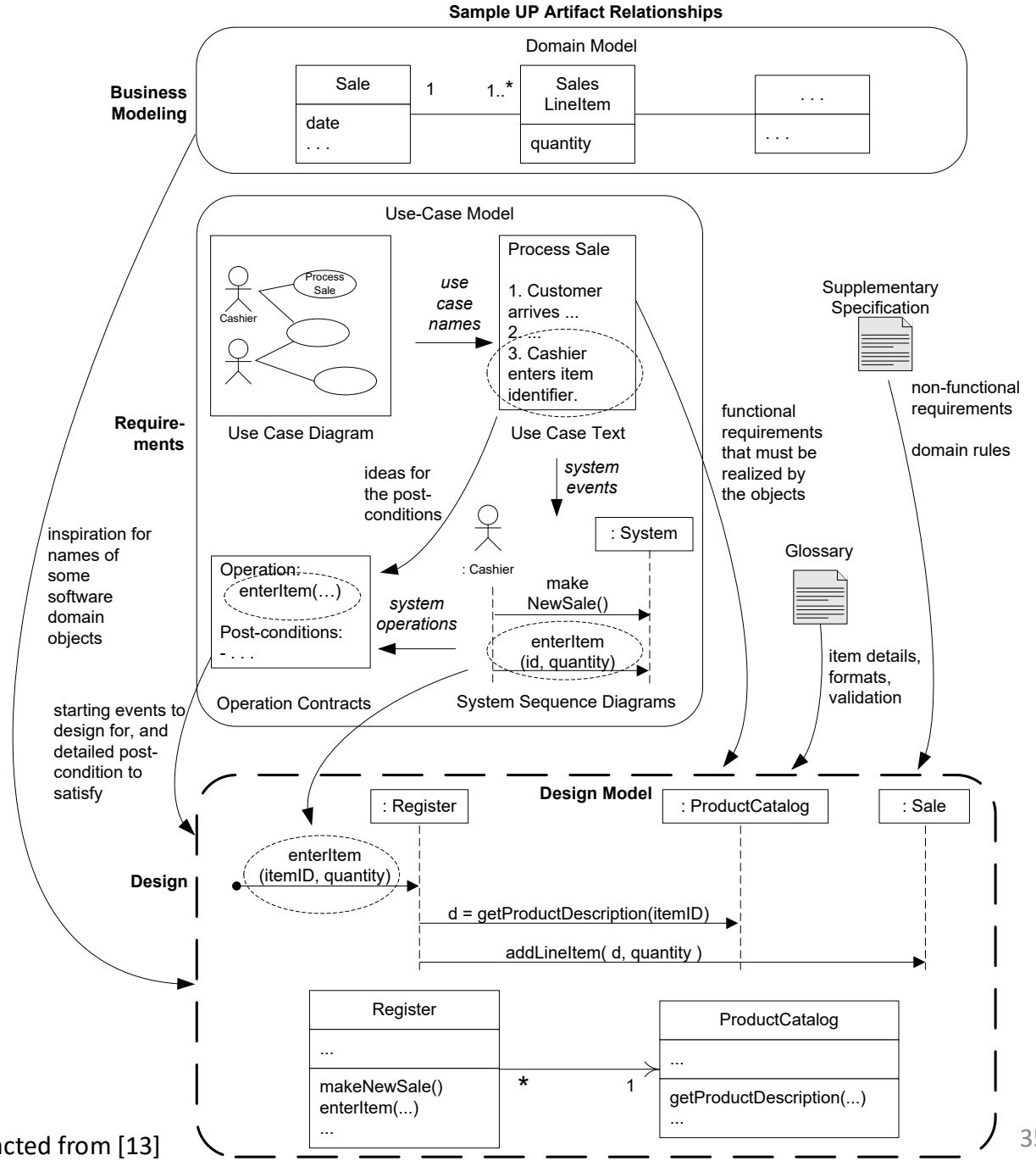
- Communication
  - Essential and transversal to all activities aiming that the different participants communicate with each other by using the proper means
    - to promote the correct / intended interpretation of what is being communicated
    - e.g. natural/informal, semi-formal and formal languages

The Process to Adopt

1. **Requirements**
   - Identify what the stakeholders want
   - Clarify and record restrictions and requirements
2. **Analysis**
   - Do the right thing
   - Problem and requirements research
   - Exploring requirements details (e.g. OO analysis)
3. **Design**
   - Do the thing right
   - Conceptual solution that aims to fulfill the requirements
   - Lead to implementation, but it is not implementation
4. **Implementation and Testing**
   - Build and test the thing



The Artifacts



Extracted from [13]

Summary

- Mindless-Driven Development leads to decrease software quality, ultimately resulting in higher costs and extended project timelines
- The adoption of a Software Development Process helps on solving most of these issues
- Next, we are going to debate about some of the activities of a typical Software Development Process: Requirements

References

[1] https://www.britannica.com/technology/software

[2] https://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm

[3] https://www.dictionary.com/browse/software

[4] https://psu.pb.unizin.org/ist110/chapter/2-2-computer-software

[5] https://www.dictionary.com/browse/engineering

[6] https://simple.wikipedia.org/wiki/Engineering

[7] https://en.wikipedia.org/wiki/Engineer

[8] https://www.britannica.com/technology/engineering

[9] https://www.merriam-webster.com/dictionary/software%20engineering

References

[10] 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology

[11] Sommerville, I. (2010). Software Engineering (9th ed.). Addison-Wesley. ISBN: 978-0-137-03515-1

[12] https://www.scnsoft.com/blog/software-development-models

[13] Larman, C. (2004). Applying UML and Patterns (3rd ed.). Prentice Hall. ISBN: 978-0-131-48906-6