Introducción a la Ingeniería del Software

Ingeniería del Software

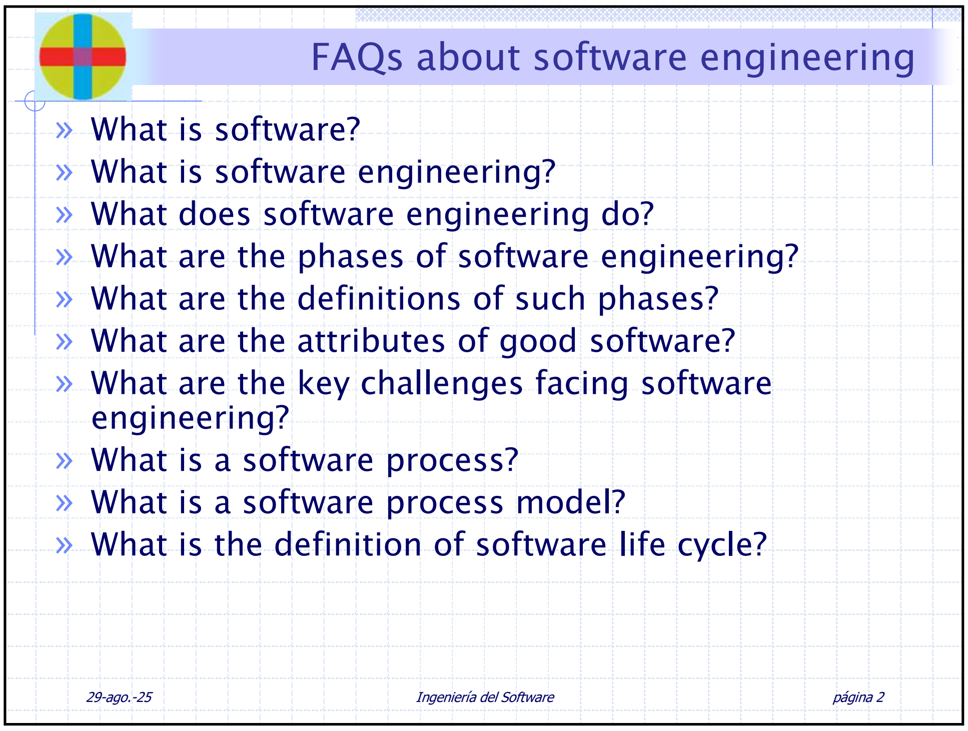
Curso 2025/2026

Universidad San Pablo-CEU

Escuela Politécnica Superior

Campus de Montepríncipe

1

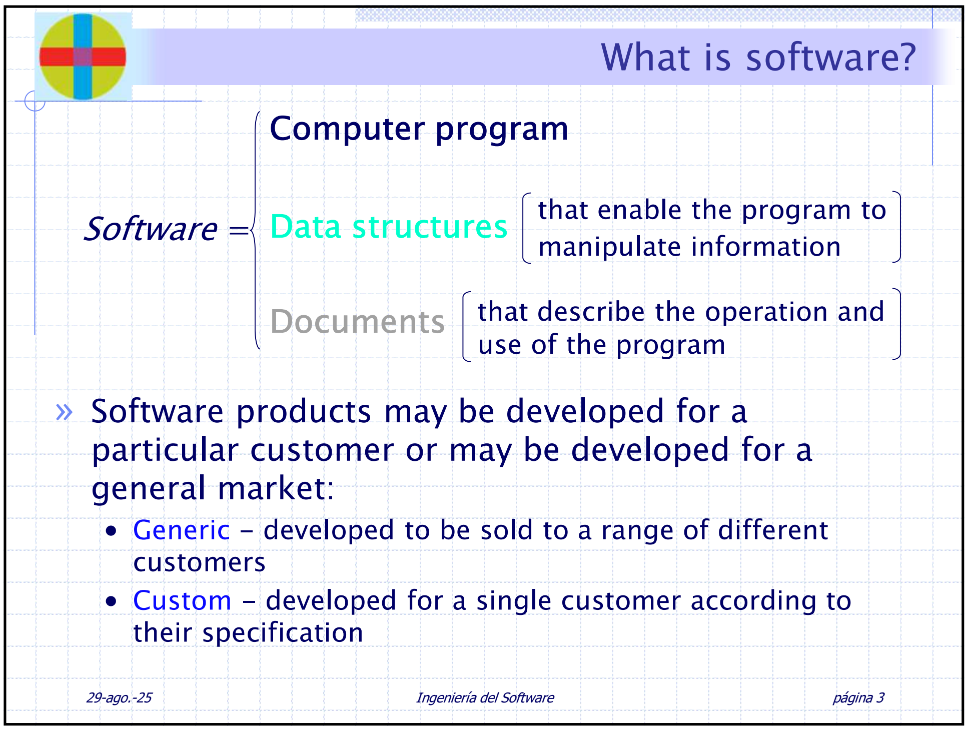


FAQs about software engineering

- What is software?
- What is software engineering?
- What does software engineering do?
- What are the phases of software engineering?
- What are the definitions of such phases?
- What are the attributes of good software?
- What are the key challenges facing software engineering?
- What is a software process?
- What is a software process model?
- What is the definition of software life cycle?

29-ago.-25 *Ingeniería del Software* *página 2*

2



What is software?

Software =
- Computer program
- Data structures — that enable the program to manipulate information
- Documents — that describe the operation and use of the program

» Software products may be developed for a particular customer or may be developed for a general market:
- Generic – developed to be sold to a range of different customers
- Custom – developed for a single customer according to their specification

3

Failure curve for hardware

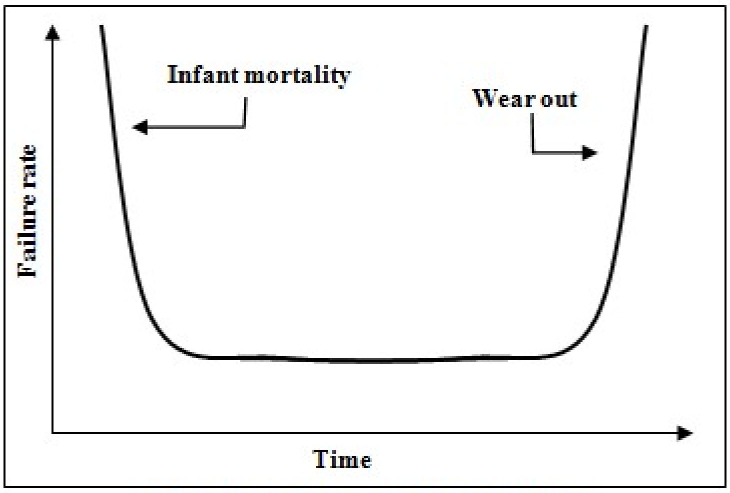


Figure 01: Failure curve for hardware

4

Failure curve for software

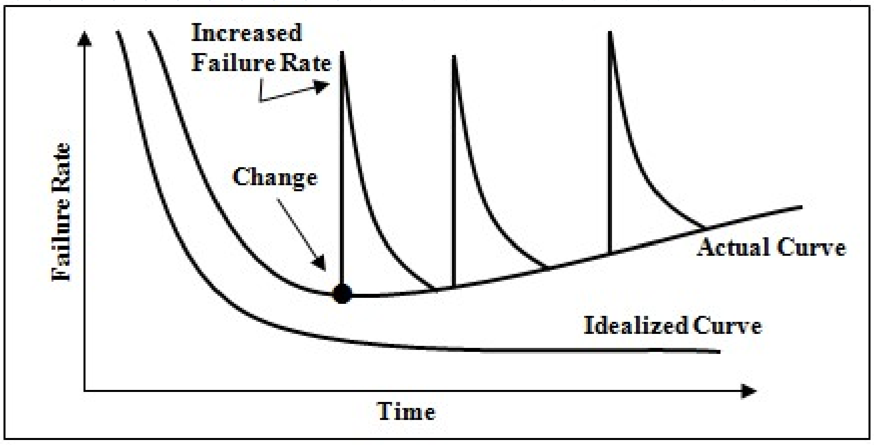
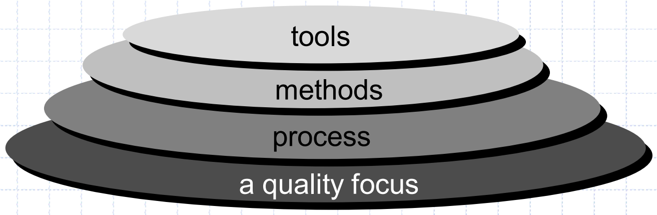


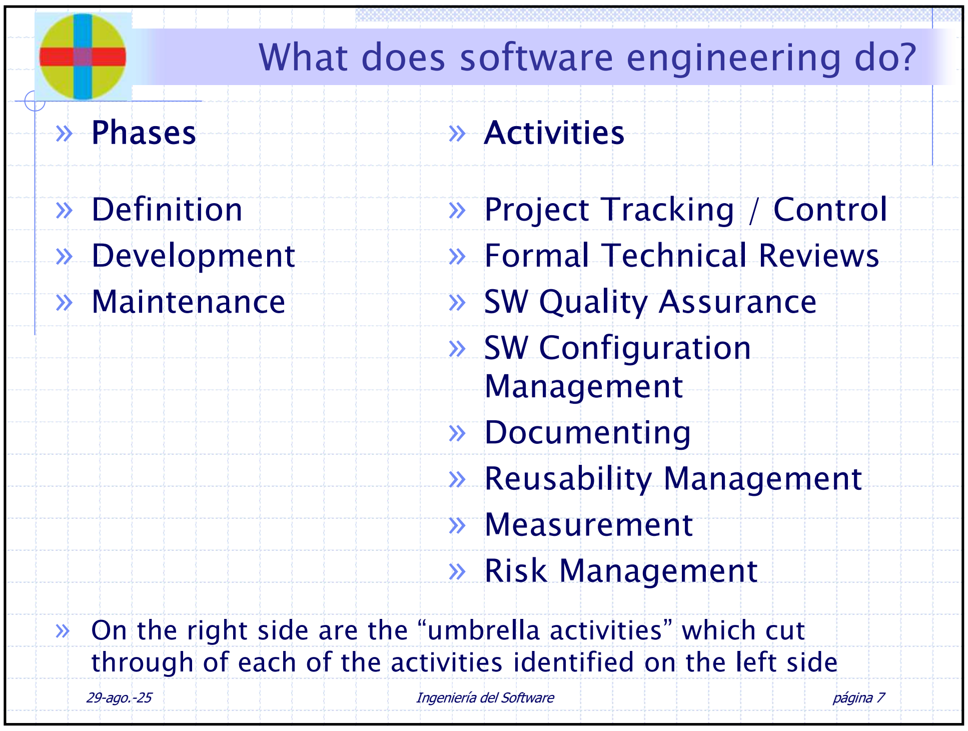
Figure 02: Failure Curves for software

What is software engineering?

» The IEEE definition of **Software engineering**:
- (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software
- (2) The study of approaches as in (1)

Software engineering





What does software engineering do?

» **Phases**

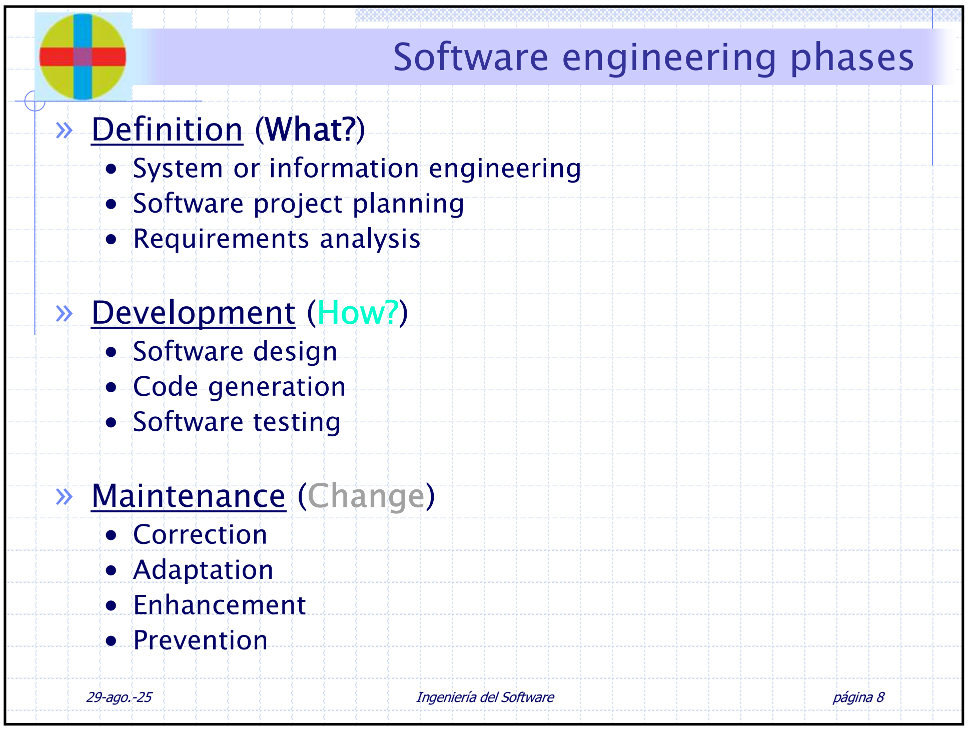
» Definition
» Development
» Maintenance

» **Activities**

» Project Tracking / Control
» Formal Technical Reviews
» SW Quality Assurance
» SW Configuration Management
» Documenting
» Reusability Management
» Measurement
» Risk Management

» On the right side are the “umbrella activities” which cut through of each of the activities identified on the left side

7



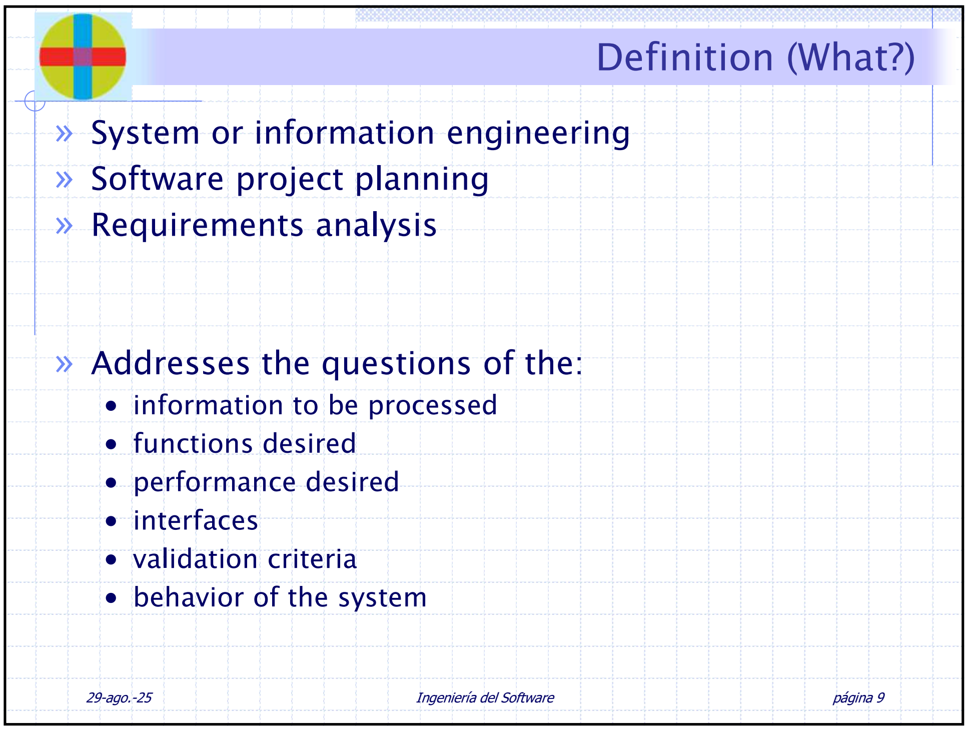
Software engineering phases

» Definition (**What?**)
- System or information engineering
- Software project planning
- Requirements analysis

» Development (How?)
- Software design
- Code generation
- Software testing

» Maintenance (Change)
- Correction
- Adaptation
- Enhancement
- Prevention

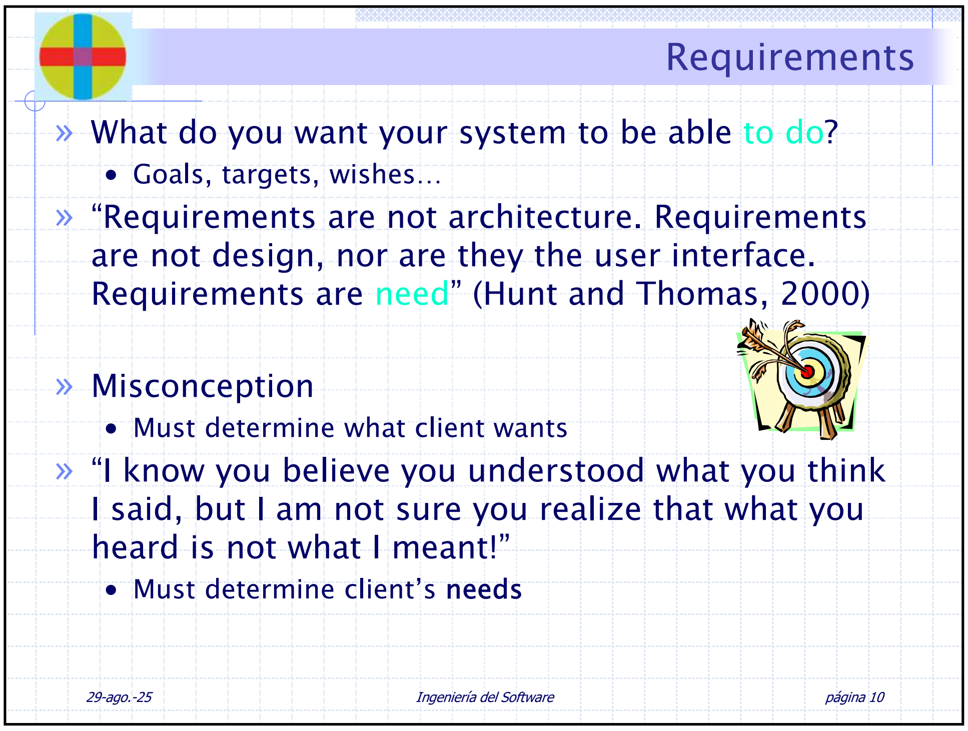
8



Definition (What?)

- System or information engineering
- Software project planning
- Requirements analysis

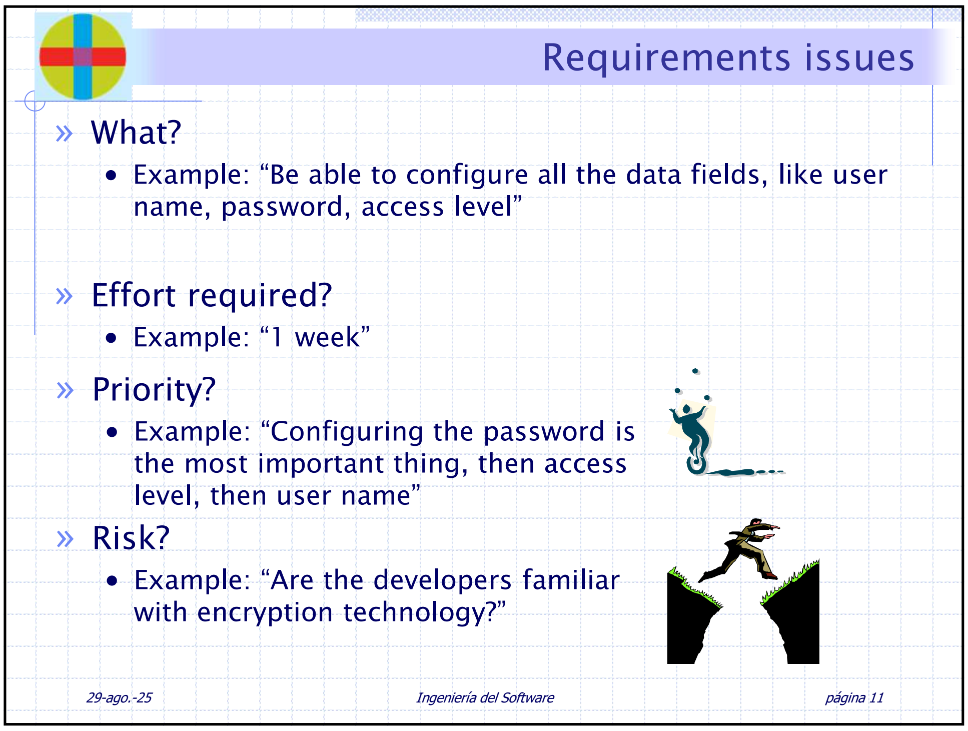
- Addresses the questions of the:
  - information to be processed
  - functions desired
  - performance desired
  - interfaces
  - validation criteria
  - behavior of the system



Requirements

- What do you want your system to be able to do?
  - Goals, targets, wishes…
- “Requirements are not architecture. Requirements are not design, nor are they the user interface. Requirements are need” (Hunt and Thomas, 2000)

- Misconception
  - Must determine what client wants
- “I know you believe you understood what you think I said, but I am not sure you realize that what you heard is not what I meant!”
  - Must determine client’s **needs**

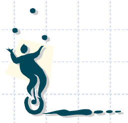


Requirements issues

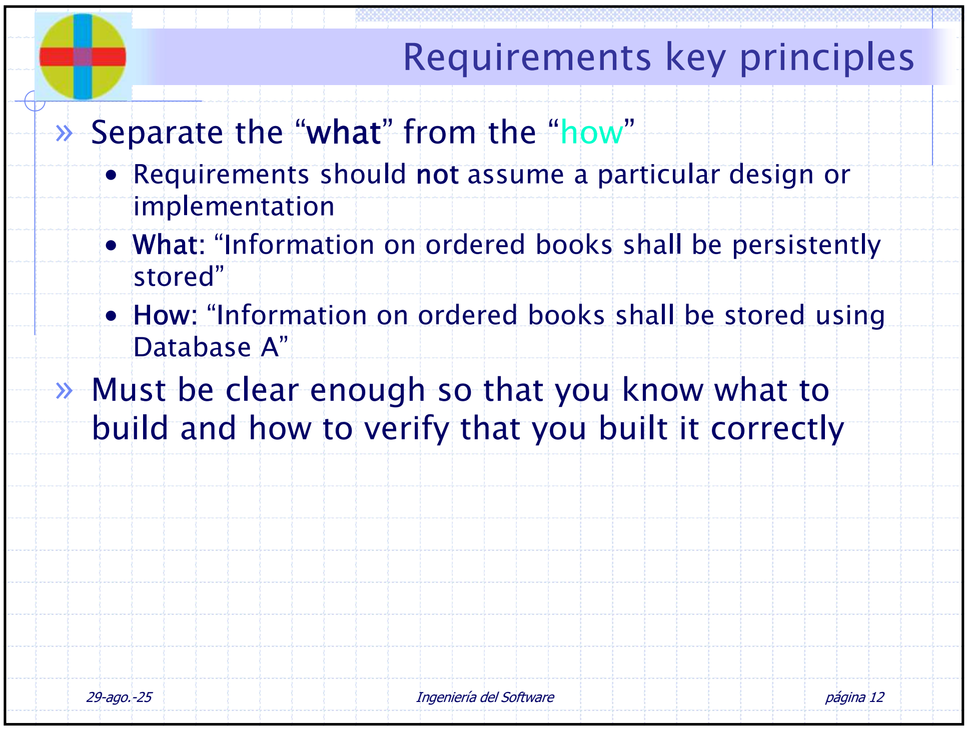
» What?
- Example: "Be able to configure all the data fields, like user name, password, access level"

» Effort required?
- Example: "1 week"

» Priority?
- Example: "Configuring the password is the most important thing, then access level, then user name"



» Risk?
- Example: "Are the developers familiar with encryption technology?"



Requirements key principles

» Separate the "**what**" from the "how"
- Requirements should **not** assume a particular design or implementation
- **What**: "Information on ordered books shall be persistently stored"
- **How**: "Information on ordered books shall be stored using Database A"

» Must be clear enough so that you know what to build and how to verify that you built it correctly

Implementing requirement analysis

» Verbal
- Interviewing (primary technique)
- Structured versus unstructured interviews

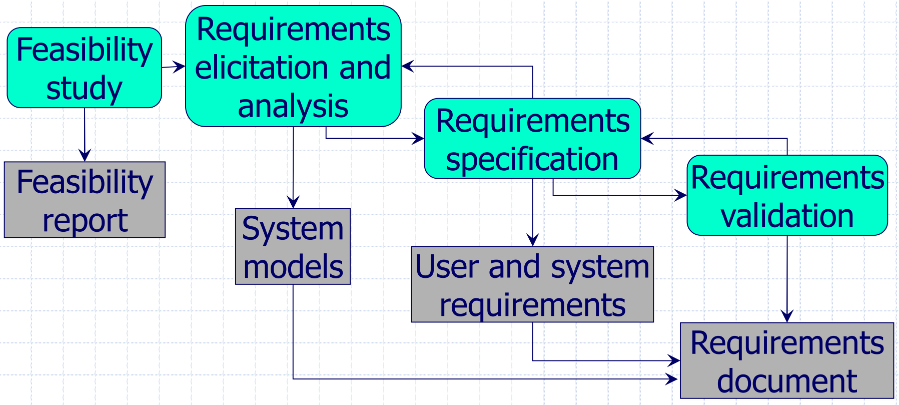
» Written
- Notes on paper, index cards, sticky notes, questionnaires, formal documents
- Web sites

» Prototyping
- But remember, “Prototyping is a learning experience. Its value lies not in the code produced, but in the lessons learned” (Hunt and Thomas, 2000)

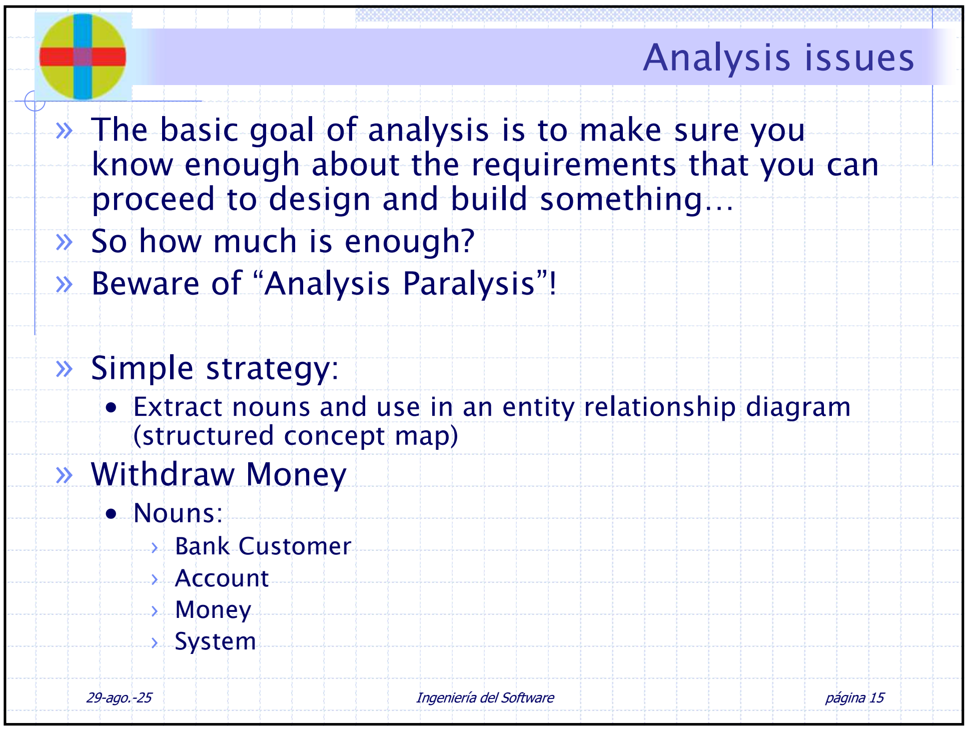
13

The requirements engineering process

- Feasibility study
- Requirements elicitation and analysis
- Requirements specification
- Requirements validation



14



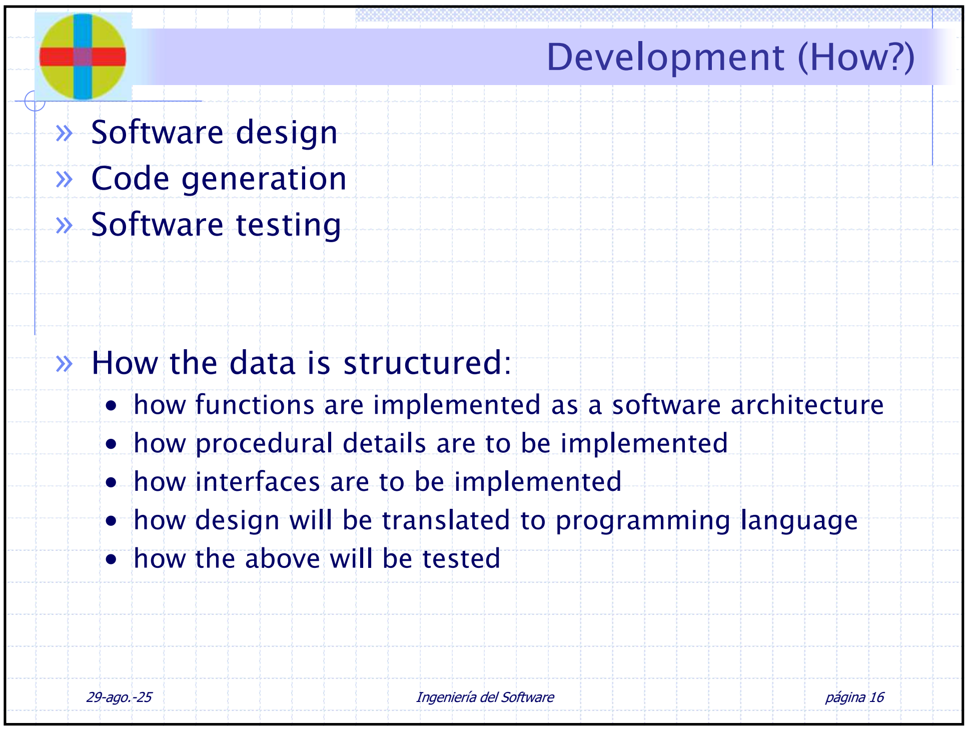
Analysis issues

» The basic goal of analysis is to make sure you know enough about the requirements that you can proceed to design and build something…
» So how much is enough?
» Beware of "Analysis Paralysis"!

» Simple strategy:
- Extract nouns and use in an entity relationship diagram (structured concept map)

» Withdraw Money
- Nouns:
  - Bank Customer
  - Account
  - Money
  - System

15

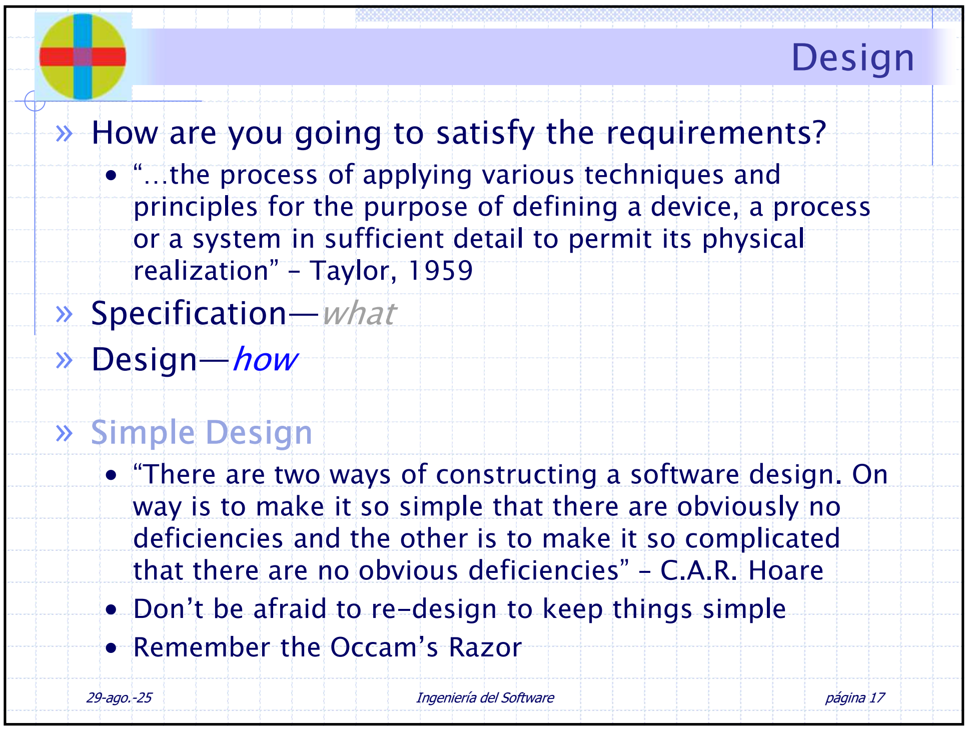


Development (How?)

» Software design
» Code generation
» Software testing

» How the data is structured:
- how functions are implemented as a software architecture
- how procedural details are to be implemented
- how interfaces are to be implemented
- how design will be translated to programming language
- how the above will be tested

16



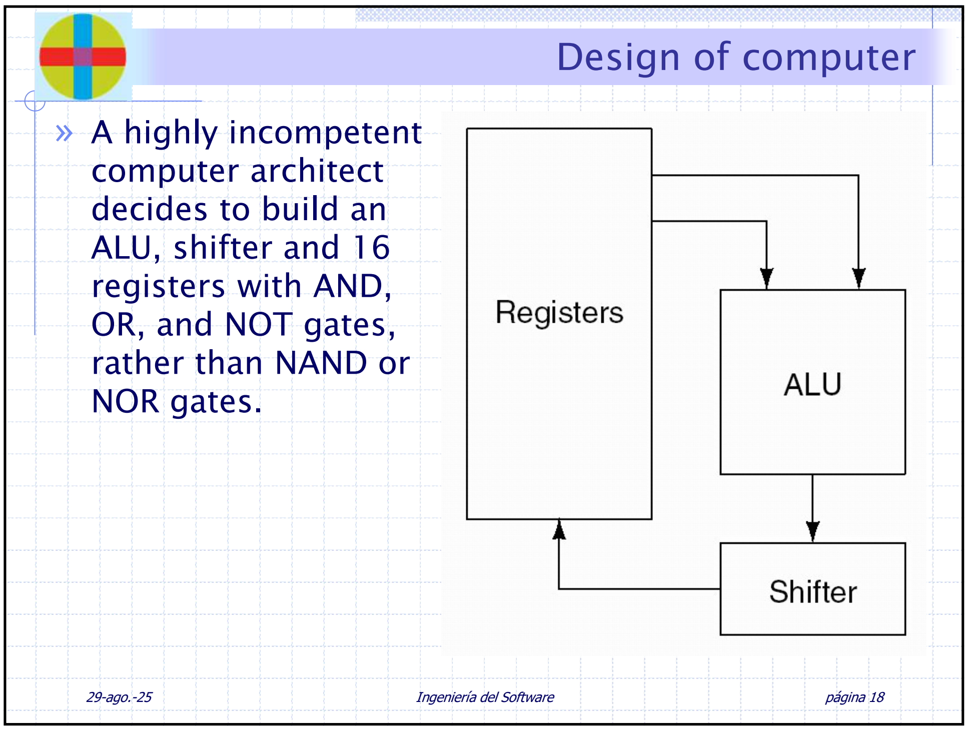
Design

» How are you going to satisfy the requirements?
- "...the process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization" – Taylor, 1959

» Specification—*what*

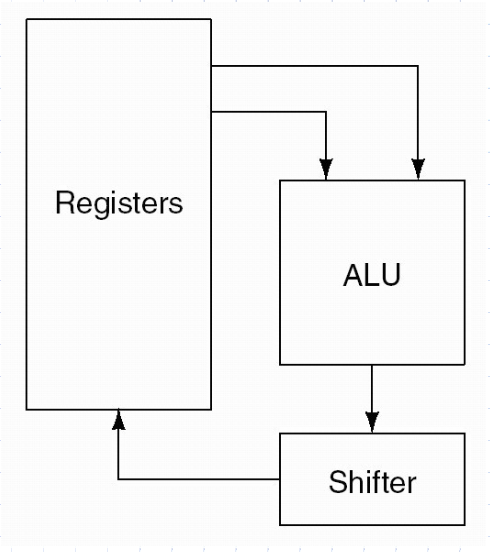
» Design—*how*

» Simple Design
- "There are two ways of constructing a software design. On way is to make it so simple that there are obviously no deficiencies and the other is to make it so complicated that there are no obvious deficiencies" – C.A.R. Hoare
- Don't be afraid to re-design to keep things simple
- Remember the Occam's Razor



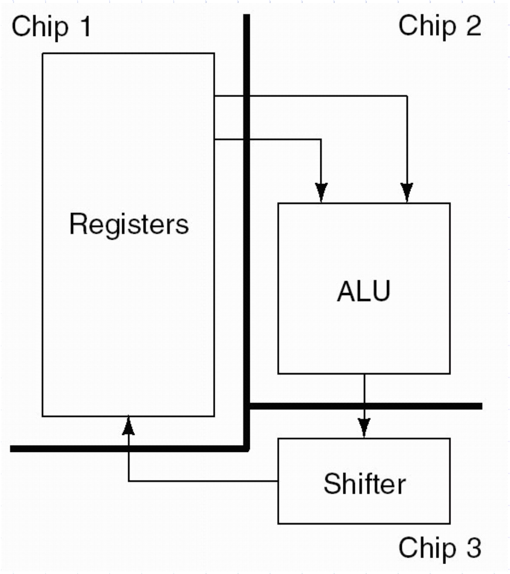
Design of computer

» A highly incompetent computer architect decides to build an ALU, shifter and 16 registers with AND, OR, and NOT gates, rather than NAND or NOR gates.



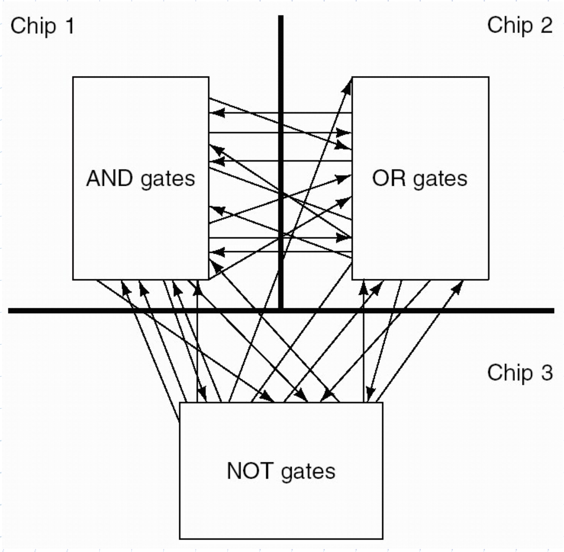
Design of computer (cont)

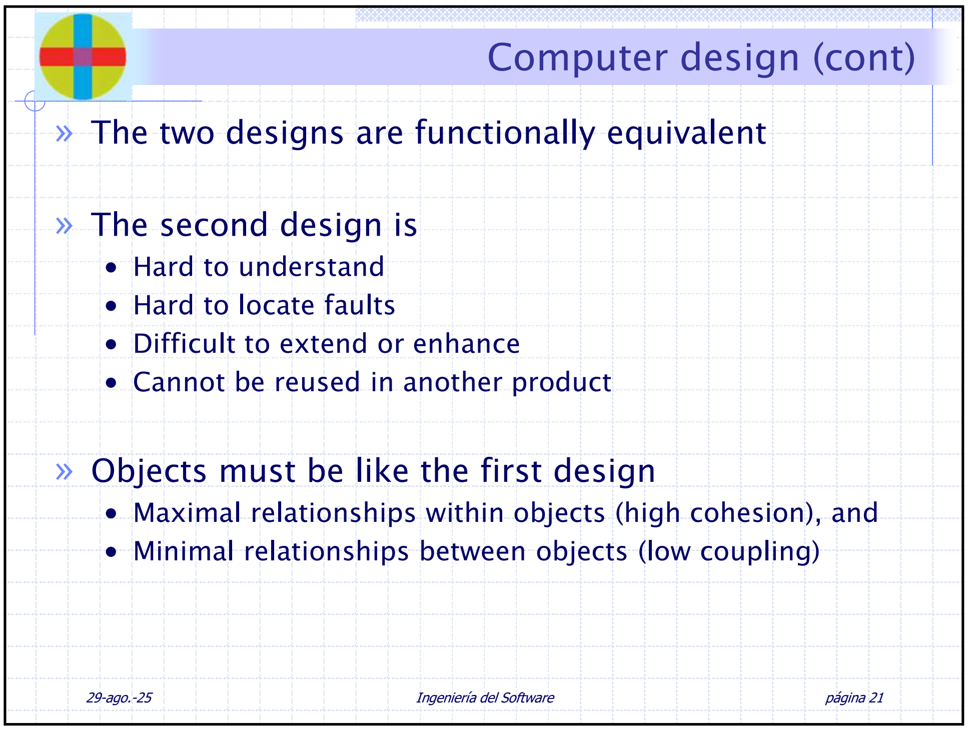
- Architect designs 3 silicon chips



Design of computer (cont)

- Redesign with one gate type per chip
- Resulting "masterpiece"





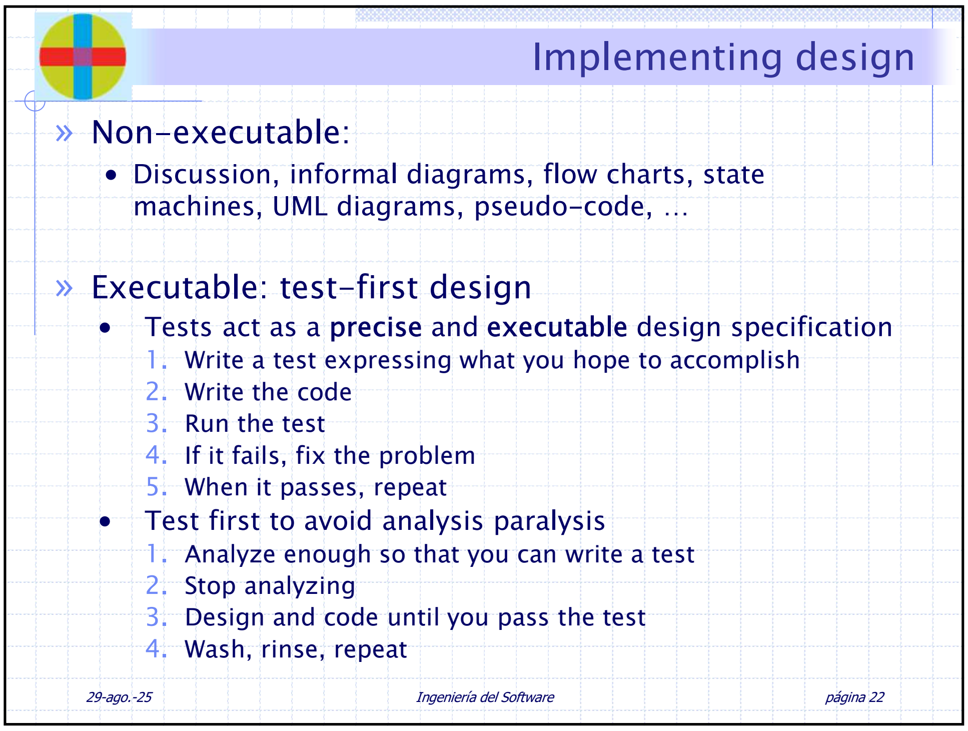
Computer design (cont)

» The two designs are functionally equivalent

» The second design is
- Hard to understand
- Hard to locate faults
- Difficult to extend or enhance
- Cannot be reused in another product

» Objects must be like the first design
- Maximal relationships within objects (high cohesion), and
- Minimal relationships between objects (low coupling)

21

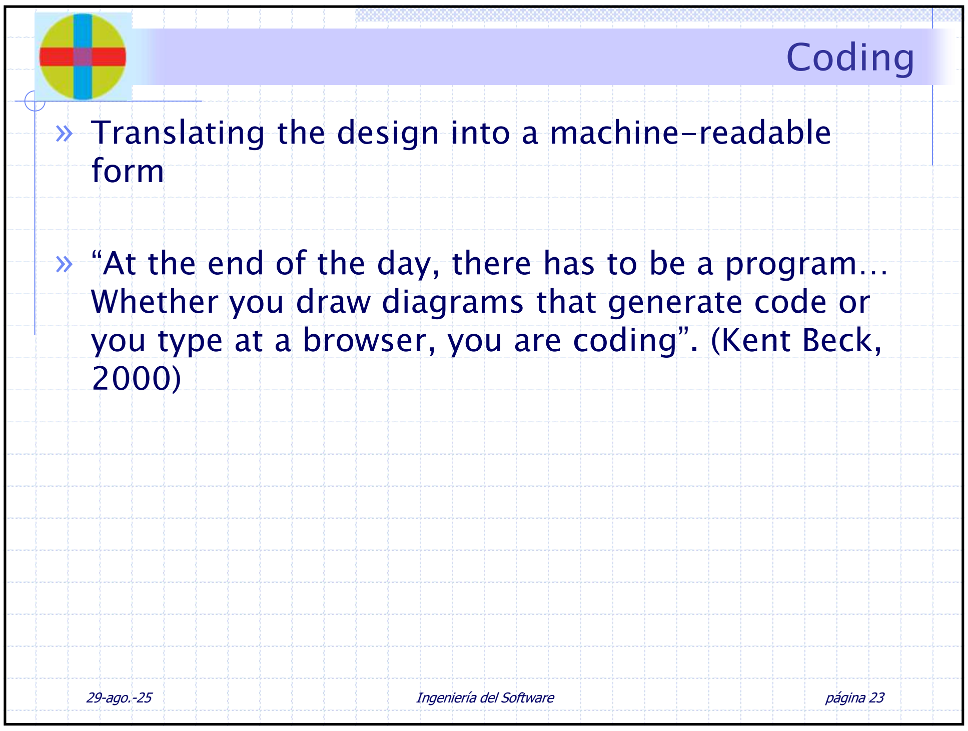


Implementing design

» Non-executable:
- Discussion, informal diagrams, flow charts, state machines, UML diagrams, pseudo-code, …

» Executable: test-first design
- Tests act as a **precise** and **executable** design specification
  1. Write a test expressing what you hope to accomplish
  2. Write the code
  3. Run the test
  4. If it fails, fix the problem
  5. When it passes, repeat
- Test first to avoid analysis paralysis
  1. Analyze enough so that you can write a test
  2. Stop analyzing
  3. Design and code until you pass the test
  4. Wash, rinse, repeat

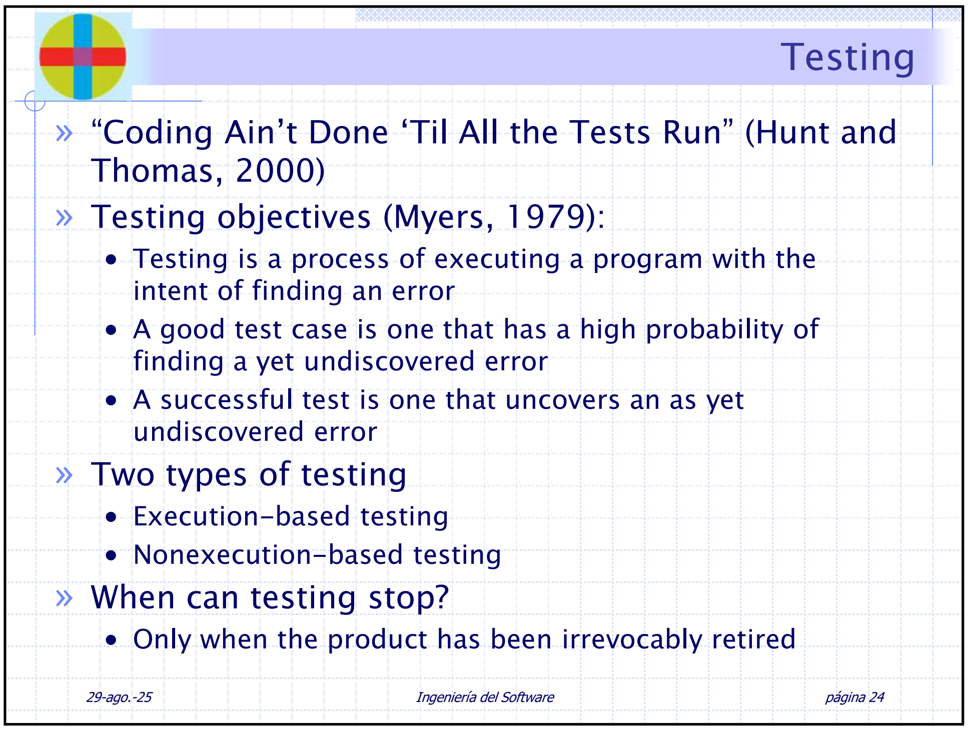
22



Coding

» Translating the design into a machine-readable form

» "At the end of the day, there has to be a program… Whether you draw diagrams that generate code or you type at a browser, you are coding". (Kent Beck, 2000)



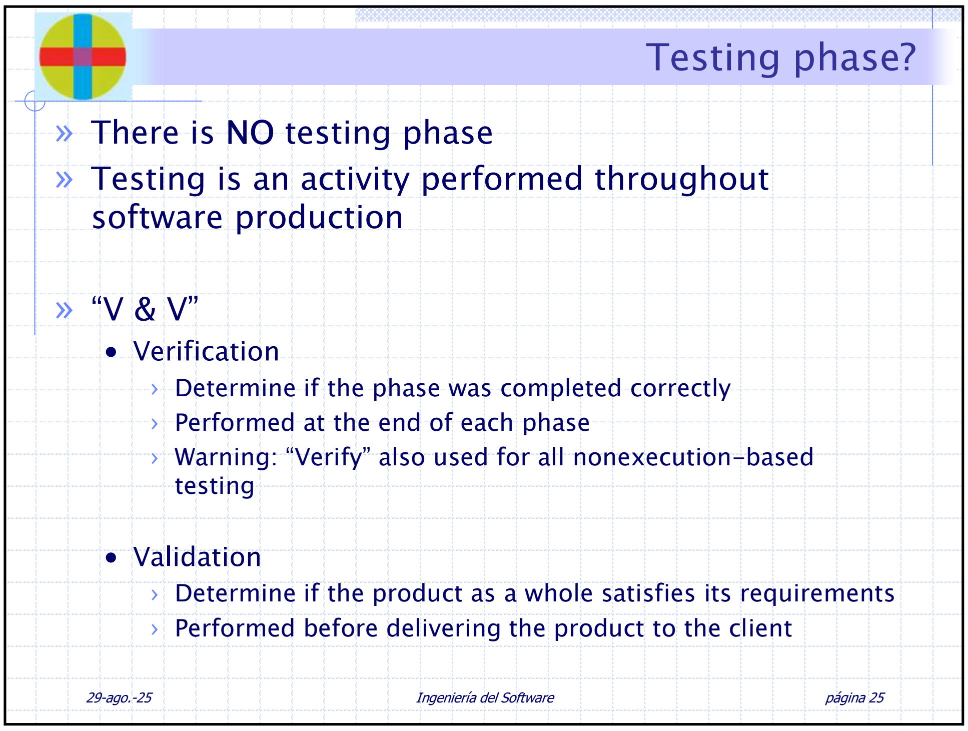
Testing

» "Coding Ain't Done 'Til All the Tests Run" (Hunt and Thomas, 2000)

» Testing objectives (Myers, 1979):
- Testing is a process of executing a program with the intent of finding an error
- A good test case is one that has a high probability of finding a yet undiscovered error
- A successful test is one that uncovers an as yet undiscovered error

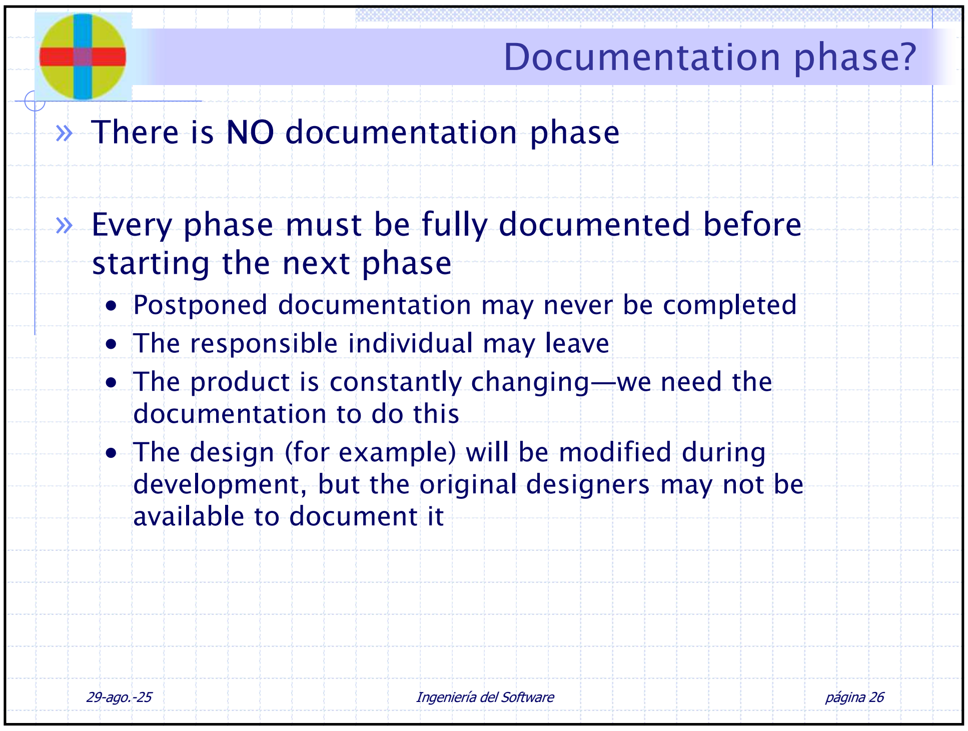
» Two types of testing
- Execution-based testing
- Nonexecution-based testing

» When can testing stop?
- Only when the product has been irrevocably retired



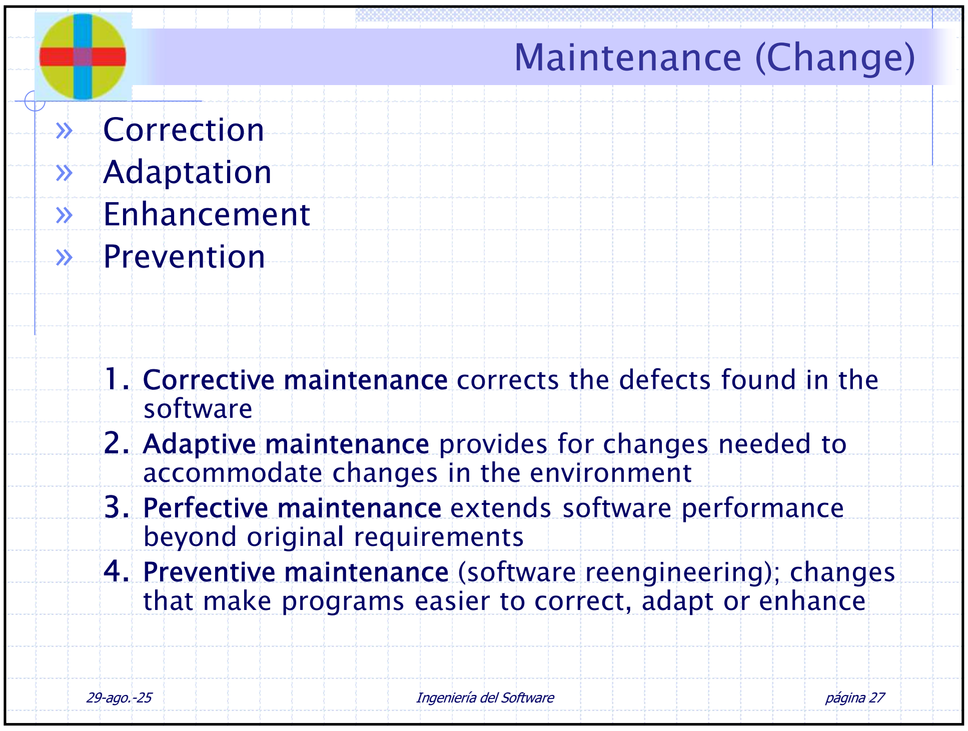
Testing phase?

- There is NO testing phase
- Testing is an activity performed throughout software production
- "V & V"
  - Verification
    - Determine if the phase was completed correctly
    - Performed at the end of each phase
    - Warning: "Verify" also used for all nonexecution-based testing
  - Validation
    - Determine if the product as a whole satisfies its requirements
    - Performed before delivering the product to the client



Documentation phase?

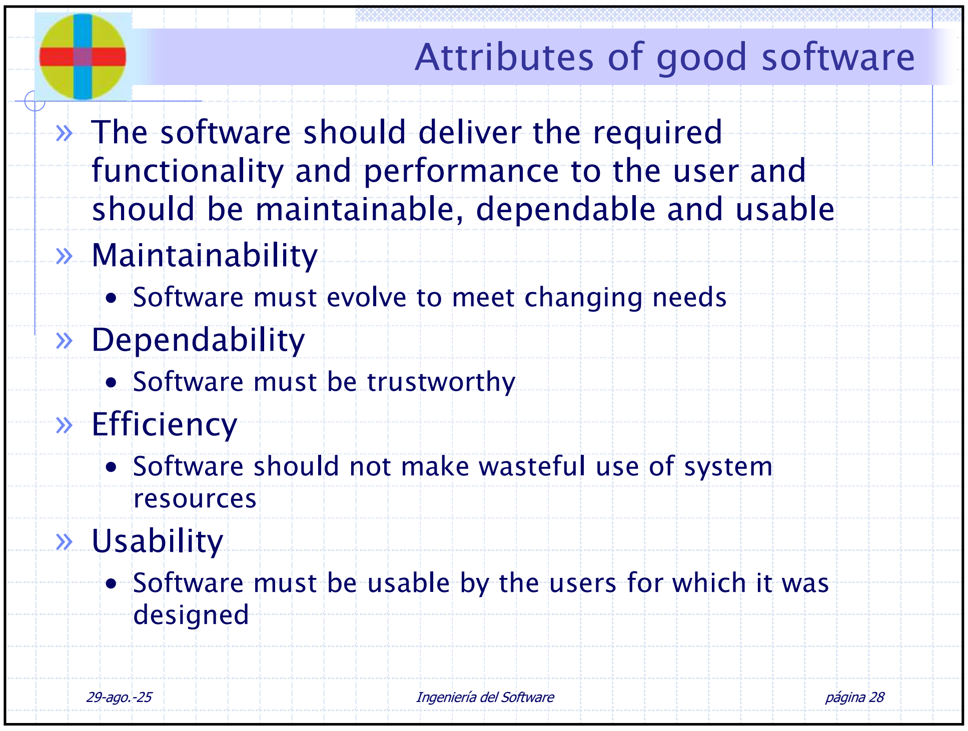
- There is NO documentation phase
- Every phase must be fully documented before starting the next phase
  - Postponed documentation may never be completed
  - The responsible individual may leave
  - The product is constantly changing—we need the documentation to do this
  - The design (for example) will be modified during development, but the original designers may not be available to document it



Maintenance (Change)

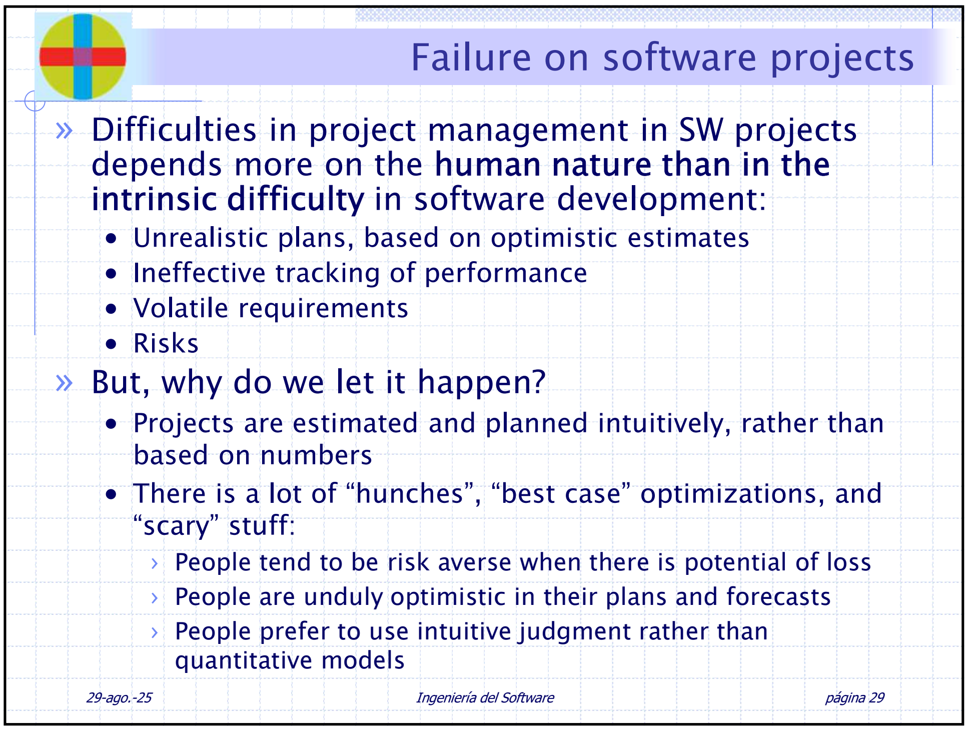
- Correction
- Adaptation
- Enhancement
- Prevention

1. **Corrective maintenance** corrects the defects found in the software
2. **Adaptive maintenance** provides for changes needed to accommodate changes in the environment
3. **Perfective maintenance** extends software performance beyond original requirements
4. **Preventive maintenance** (software reengineering); changes that make programs easier to correct, adapt or enhance



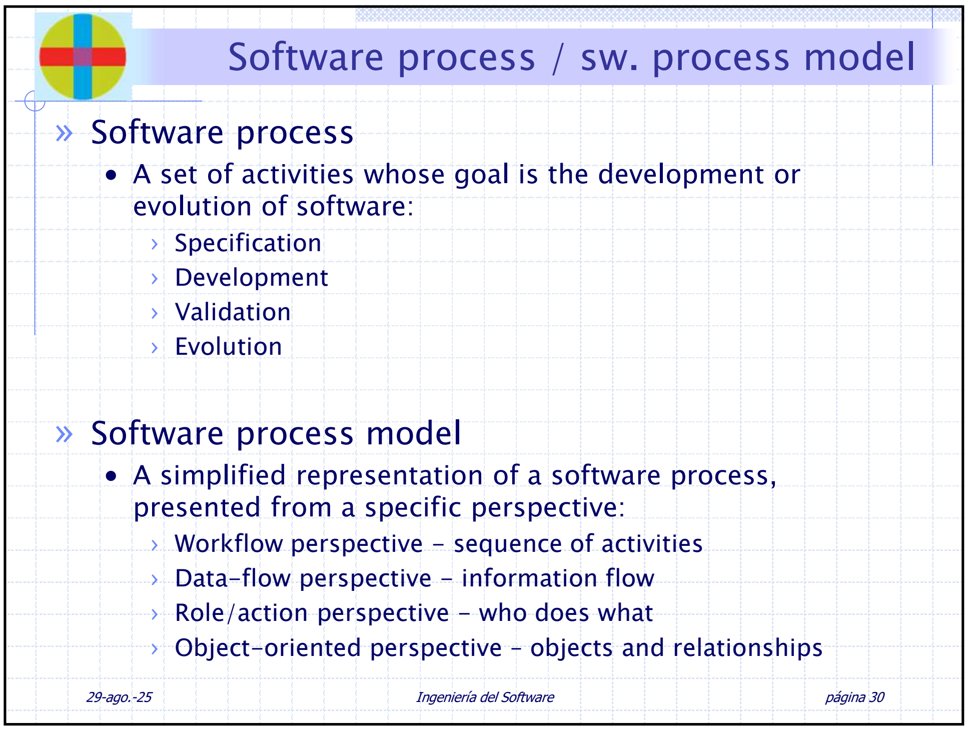
Attributes of good software

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable
- Maintainability
  - Software must evolve to meet changing needs
- Dependability
  - Software must be trustworthy
- Efficiency
  - Software should not make wasteful use of system resources
- Usability
  - Software must be usable by the users for which it was designed



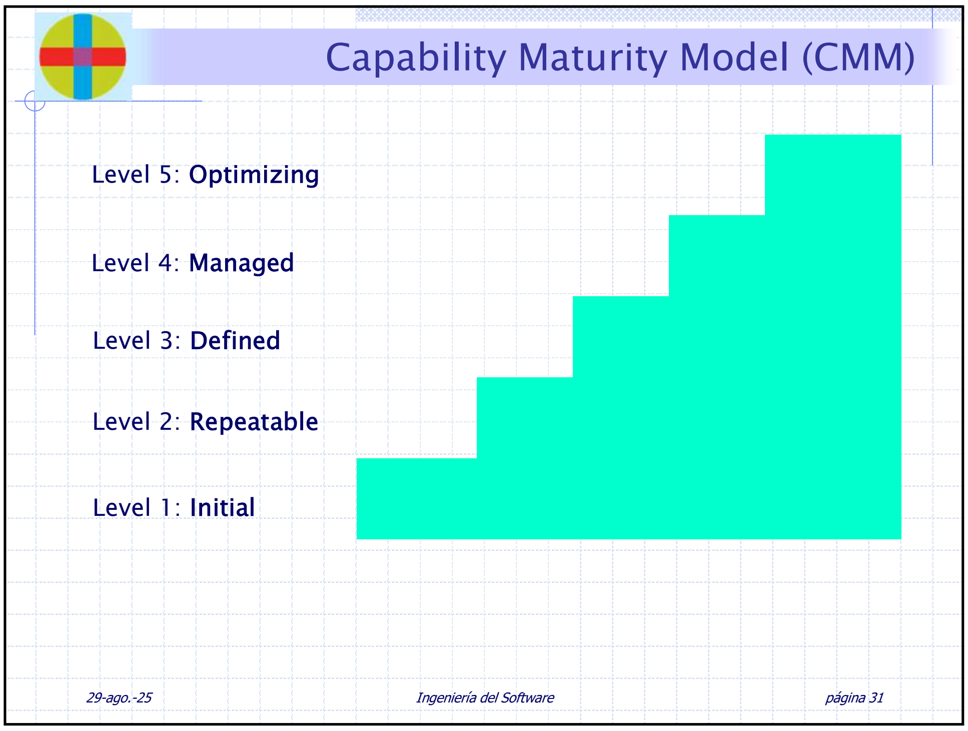
Failure on software projects

- Difficulties in project management in SW projects depends more on the **human nature than in the intrinsic difficulty** in software development:
  - Unrealistic plans, based on optimistic estimates
  - Ineffective tracking of performance
  - Volatile requirements
  - Risks
- But, why do we let it happen?
  - Projects are estimated and planned intuitively, rather than based on numbers
  - There is a lot of "hunches", "best case" optimizations, and "scary" stuff:
    - People tend to be risk averse when there is potential of loss
    - People are unduly optimistic in their plans and forecasts
    - People prefer to use intuitive judgment rather than quantitative models



Software process / sw. process model

- Software process
  - A set of activities whose goal is the development or evolution of software:
    - Specification
    - Development
    - Validation
    - Evolution
- Software process model
  - A simplified representation of a software process, presented from a specific perspective:
    - Workflow perspective - sequence of activities
    - Data-flow perspective - information flow
    - Role/action perspective - who does what
    - Object-oriented perspective - objects and relationships



Capability Maturity Model (CMM)

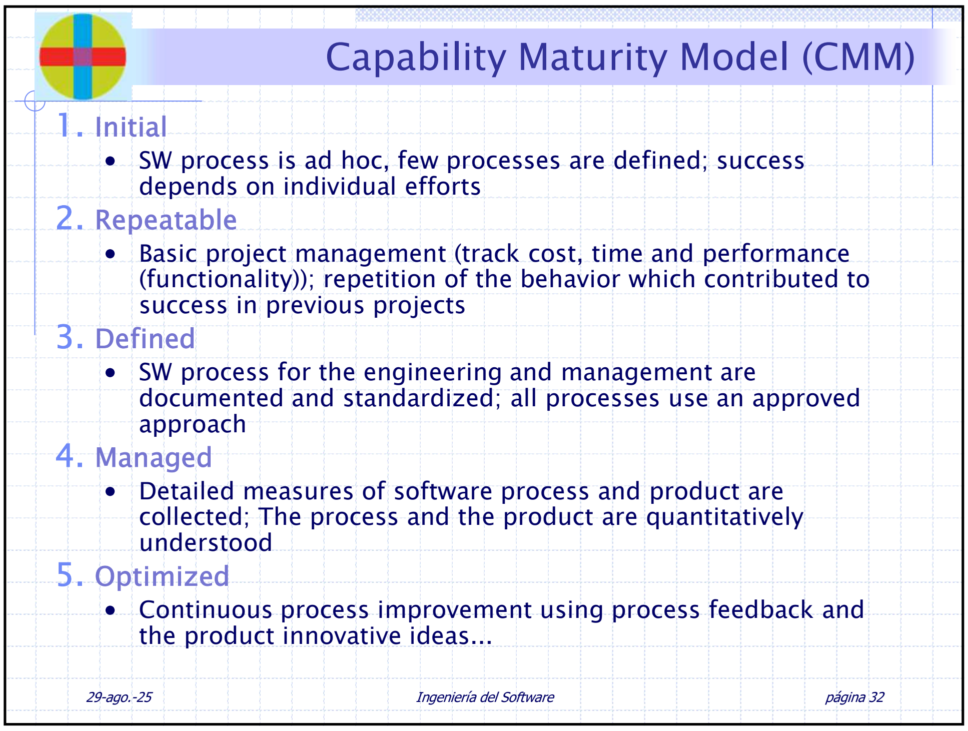
Level 5: **Optimizing**

Level 4: **Managed**

Level 3: **Defined**

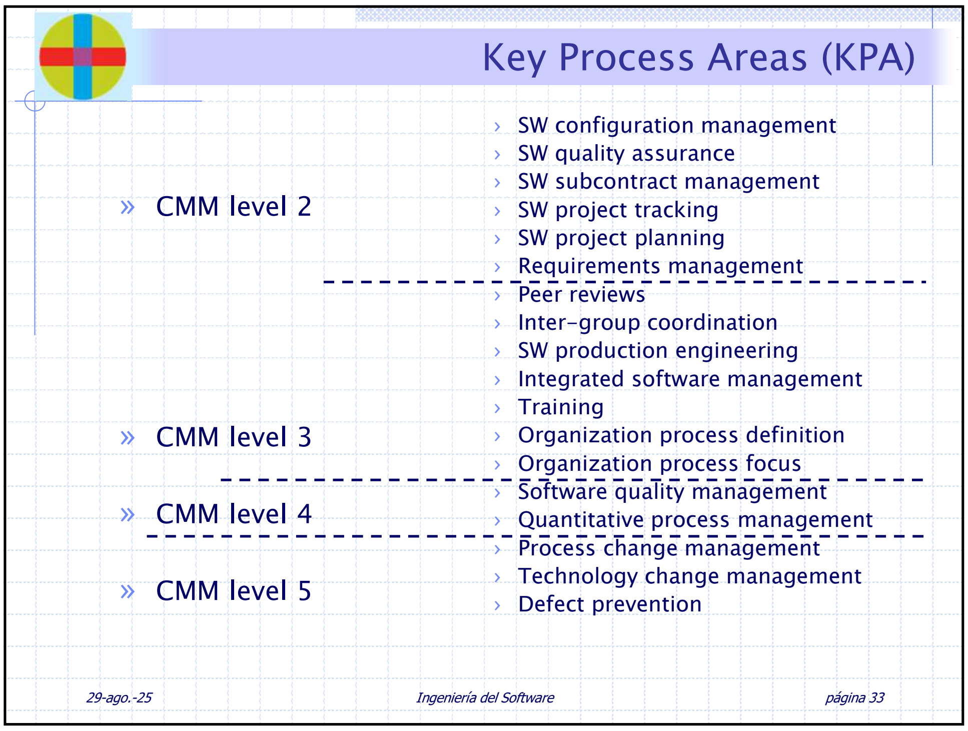
Level 2: **Repeatable**

Level 1: **Initial**



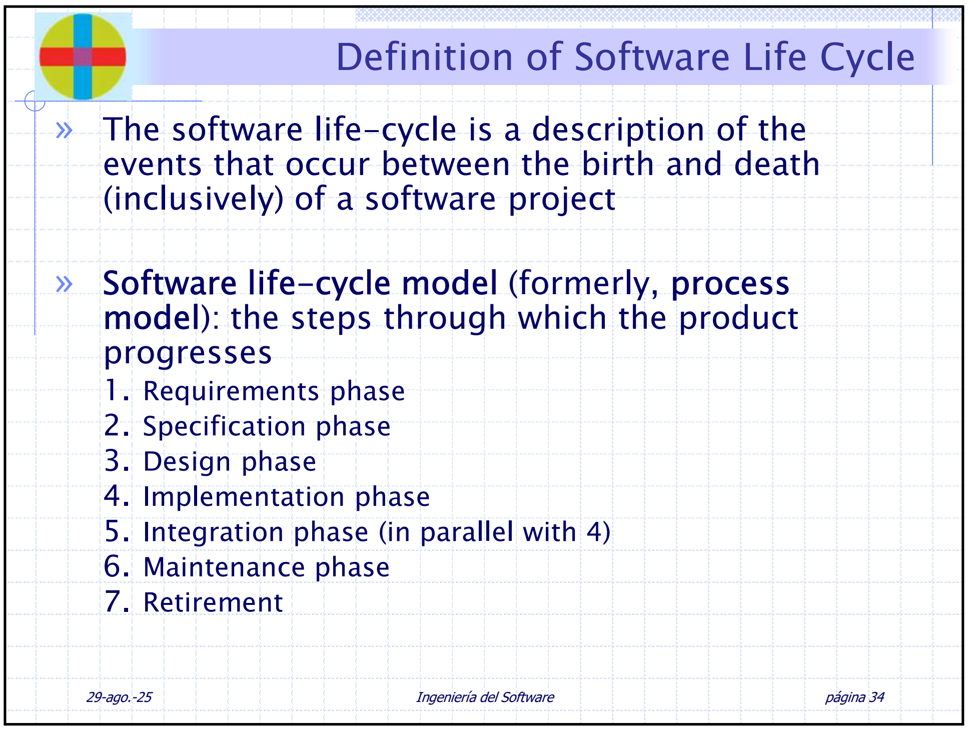
Capability Maturity Model (CMM)

1. Initial
   - SW process is ad hoc, few processes are defined; success depends on individual efforts
2. Repeatable
   - Basic project management (track cost, time and performance (functionality)); repetition of the behavior which contributed to success in previous projects
3. Defined
   - SW process for the engineering and management are documented and standardized; all processes use an approved approach
4. Managed
   - Detailed measures of software process and product are collected; The process and the product are quantitatively understood
5. Optimized
   - Continuous process improvement using process feedback and the product innovative ideas...



Key Process Areas (KPA)

» CMM level 2	› SW configuration management › SW quality assurance › SW subcontract management › SW project tracking › SW project planning › Requirements management
» CMM level 3	› Peer reviews › Inter-group coordination › SW production engineering › Integrated software management › Training › Organization process definition › Organization process focus
» CMM level 4	› Software quality management › Quantitative process management
» CMM level 5	› Process change management › Technology change management › Defect prevention



Definition of Software Life Cycle

» The software life-cycle is a description of the events that occur between the birth and death (inclusively) of a software project

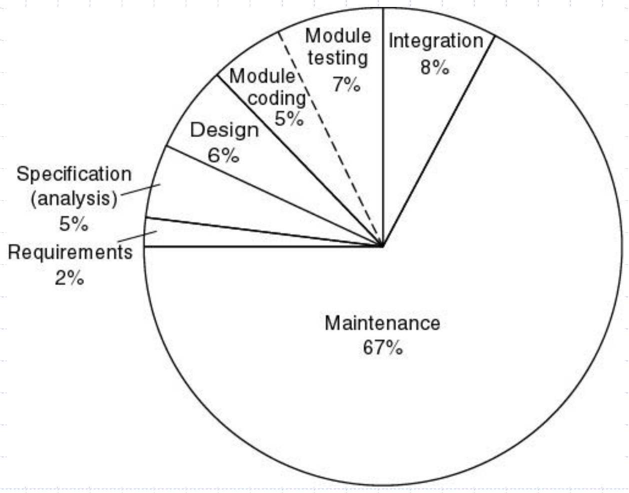
» **Software life-cycle model** (formerly, **process model**): the steps through which the product progresses

1. Requirements phase
2. Specification phase
3. Design phase
4. Implementation phase
5. Integration phase (in parallel with 4)
6. Maintenance phase
7. Retirement

Approximate relative cost of each phase

- 1976–1981 data
- Maintenance constitutes 67% of total cost

"Traditionally"



¿Preguntas?