



SW Development Process Overview

Ciclo de Desarrollo de SW

Motivation -> **Quality** is a key factor

Quality is a key factor for competition

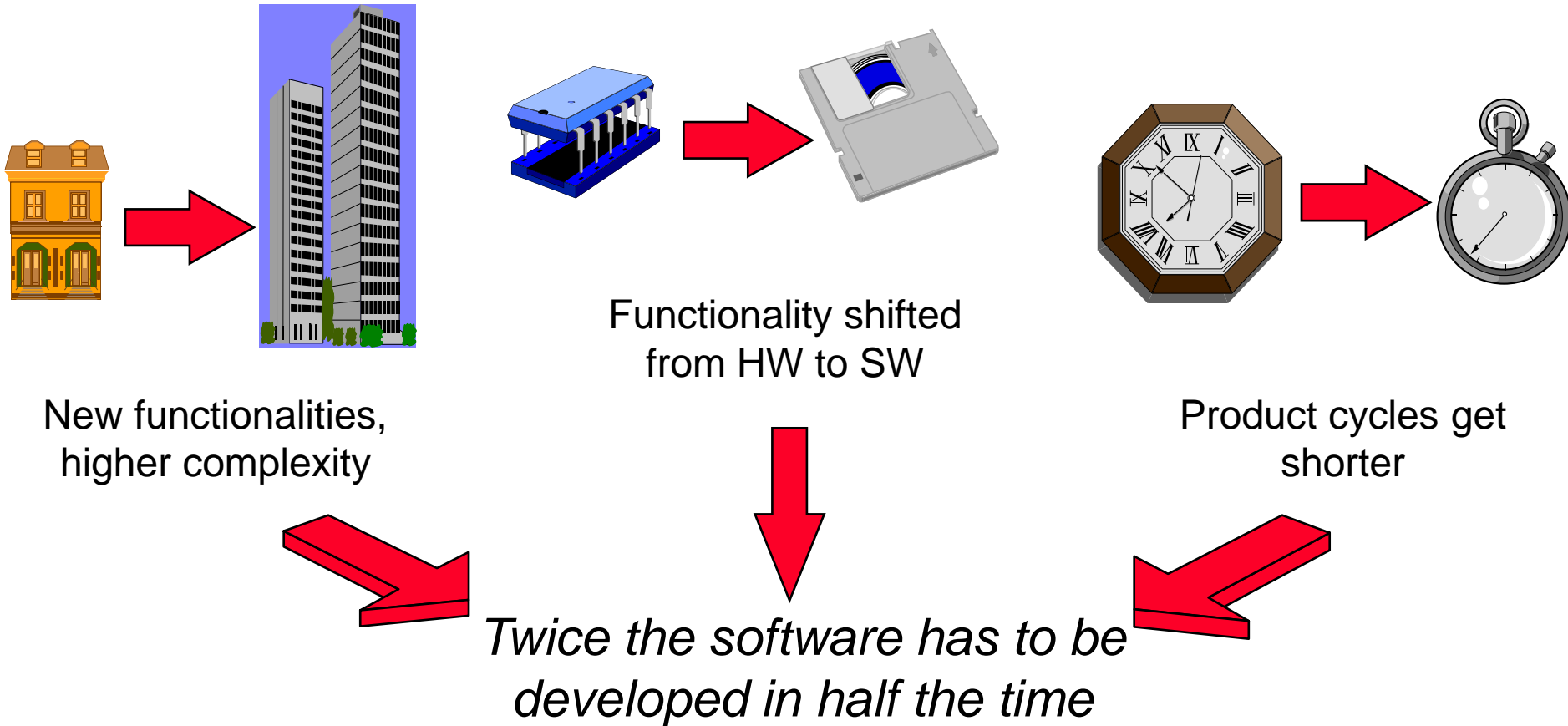
- Customer audits, assessments
- Standards (DIN, IEEE, ISO, CMMI, ...)
- Difference factor

Quality is a key factor for making profit

- fewer errors, errors detected earlier
- product liability
- faster creation of variants (easier reuse)

Good quality means the customer returns and not the product.

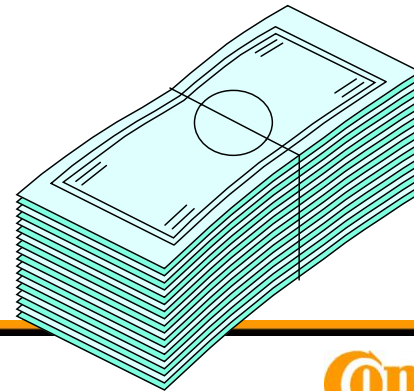
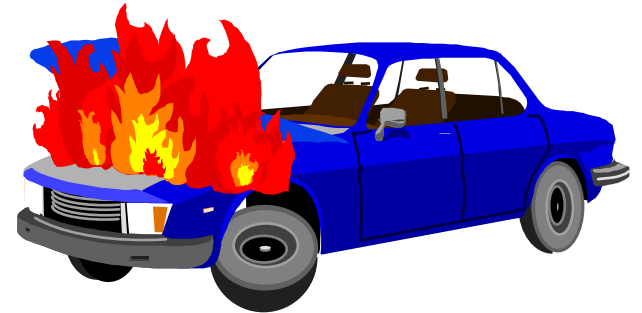
Motivation



Motivation

Risks

- Software Bugs may have dramatic consequences
- Software Bugs are systematic i.e. in every car!
- Supplier pays for recall costs
- Product Liability
- SW complexity often underestimated



Motivation

Famous SW Bugs

- NASA Apollo 11 landing problem (July 20, 1969).
- ESA Ariane 5 Flight 501 self-destruction 40 seconds after takeoff (June 4, 1996).
- NASA Mars Climate Orbiter destroyed due to entry of momentum data in imperial units instead of the metric system (September 23, 1999).
- NASA Mars Rover freezes due to too many open files in flash memory (January 21, 2004).
- NASA Mars Global Surveyor battery failure was the result of a series of events linked to a computer error made five months before (November 2, 2006).



Motivation

Famous SW Bugs

- The Therac-25 accidents (1985-1987), which caused at least five deaths.
- A misuse of medical diagnosis software created by Multidata Systems International, at the National Cancer Society in Panama City, caused, by different estimates, between five and eight cancer patients to die of over-radiation. (2000)
- The year 2000 problem, popularly known as the "Y2K bug", spawned fears of worldwide economic collapse and an industry of consultants providing last-minute fixes.
- The Pentium FDIV bug.



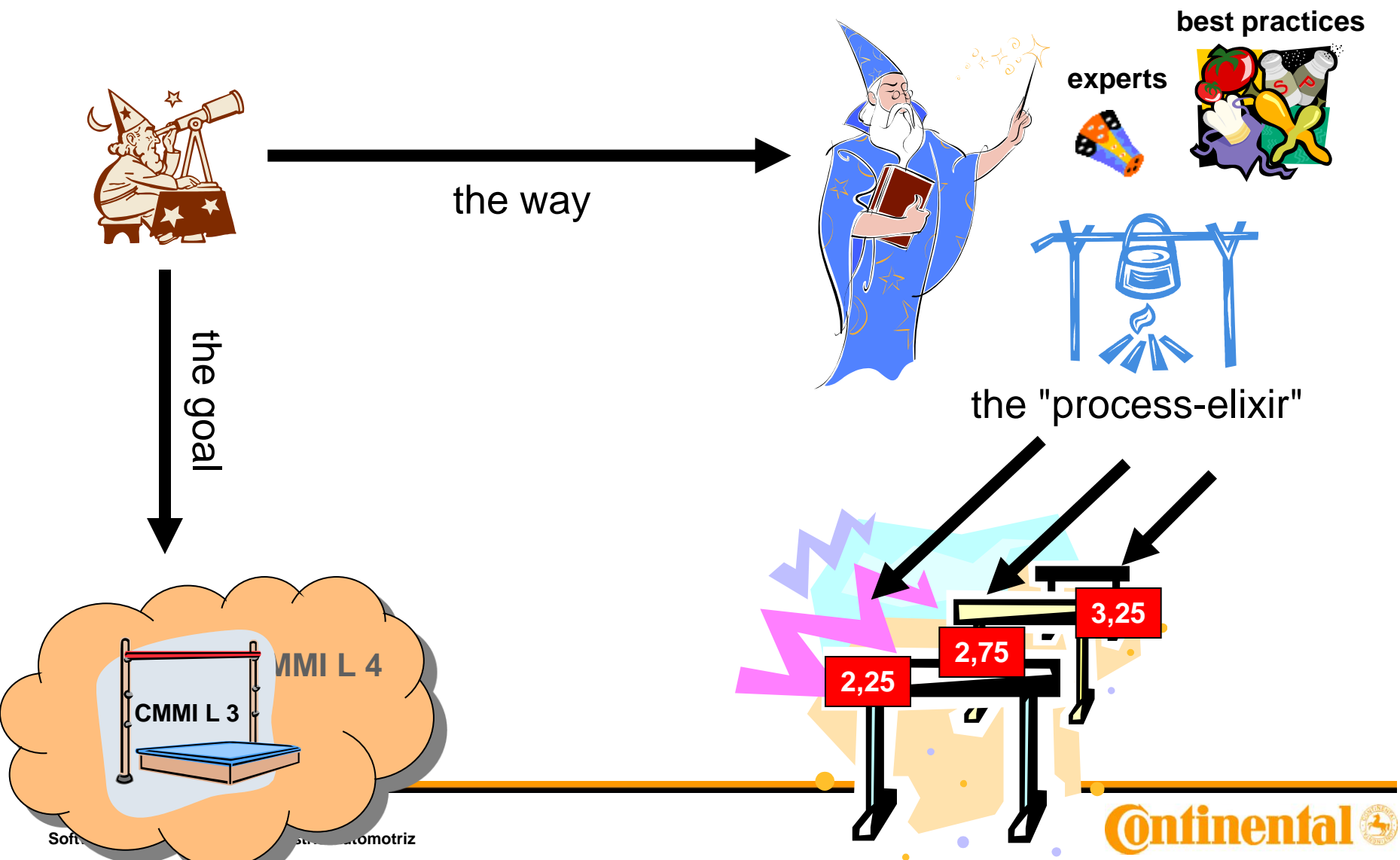
Motivation

Famous SW Bugs

- The 2003 North America blackout was triggered by a local outage that went undetected due to a race condition in General Electric Energy's XA/21 monitoring software.
- The software error of a MIM-104 Patriot, which ultimately contributed to the deaths of 28 Americans in Dhahran, Saudi Arabia (February 25, 1991).
- Chinook crash on Mull of Kintyre: the cause of this event remains a mystery, but strong suspicions have been raised that software problems were a contributory factor.



Prologue – how it started



Questions

- ▶ What is a process?
- ▶ What is a good process
- ▶ Why developing along a process?
- ▶ How to judge, if a process is a good process?

General Definition of Process

- **How do you define process?**

- ▶ A process is a set of practices performed to achieve a given purpose; it may include tools, methods, materials, and/or people.
- ▶ While process is often described as a leg of the process-people-technology triad, it may also be considered the “glue” that unifies the other aspects.

What Is a Process Model?

- ▶ A model is a structured collection of elements that describe characteristics of effective processes.
- ▶ Processes included are those proven by experience to be effective.

How Is a Model Used?

A model is used

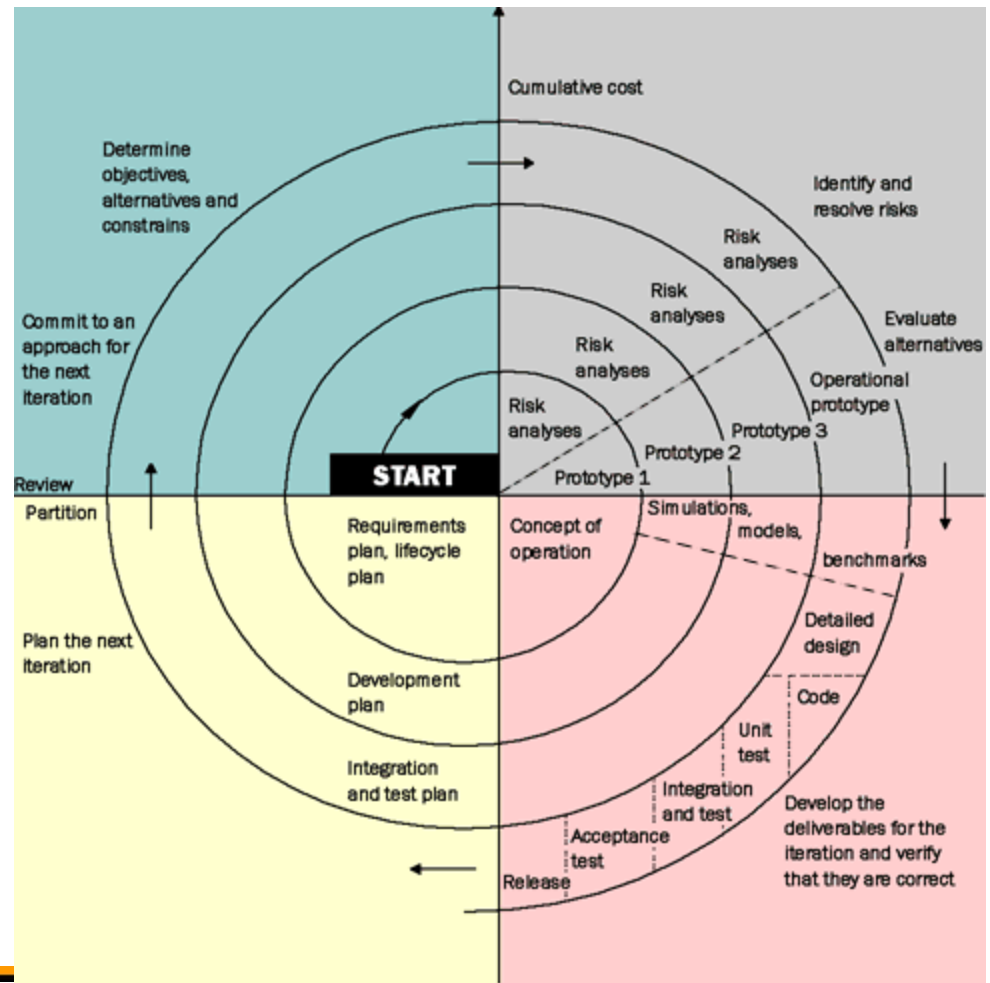
- ▶ to help set process improvement objectives and priorities, improve processes, and provide guidance for ensuring stable, capable, and mature processes
- ▶ as a guide for improvement of organizational processes

Why Is a Model Important?

- ▶ A model provides
 - ▶ a place to start
 - ▶ the benefit of a community's prior experiences
 - ▶ a common language and a shared vision
 - ▶ a framework for prioritizing actions

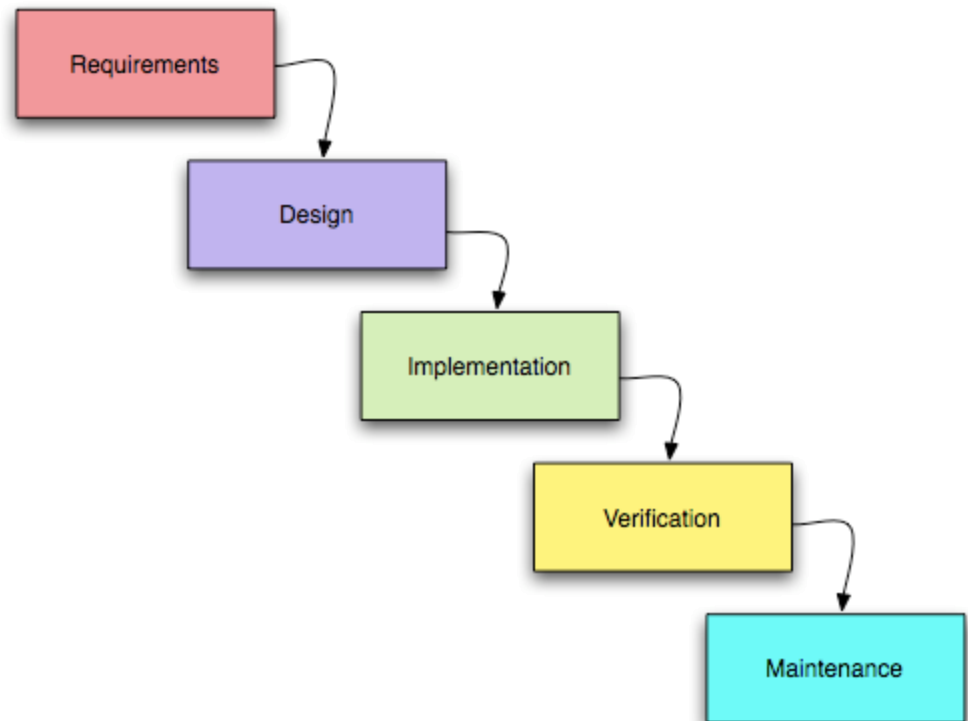
Development Models - Spiral Model

- ▶ prototypes get more details as project advance
- ▶ radial dimensions suggests project cost
- ▶ angular dimension represents progress within each iteration
- ▶ allows changes to specifications
- ▶ planning and estimations can get realistic due to experience gain in previous iterations
- ▶ people workload is more omogenous



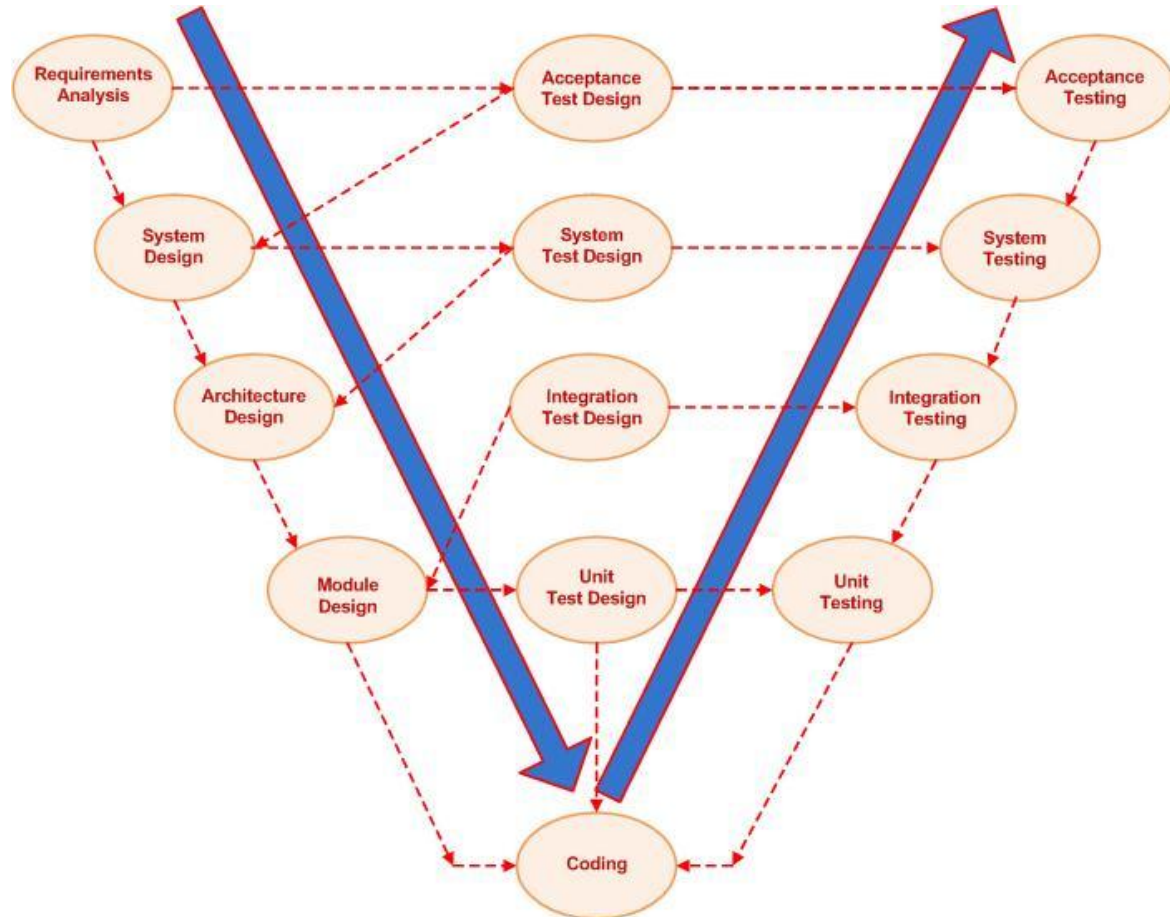
Waterfall model

- Development phases start after the previous phase ended
- documentation is very important
- requirements have to be frozen before design is started
- verifications are embedded in each phase
- no changes possible
- management control is difficult
- after one phase is over, the expert have not work and have to be assigned for other projects

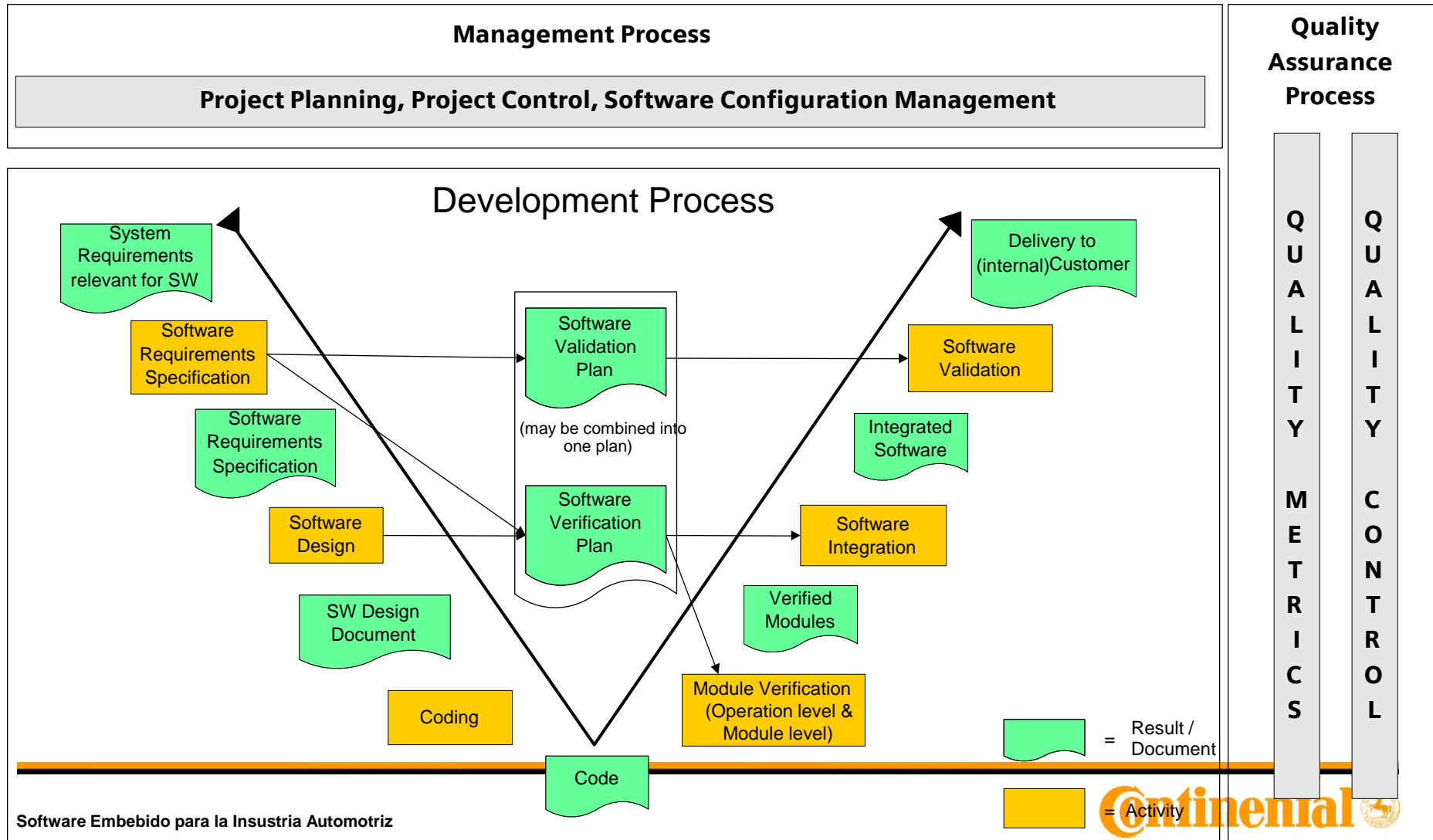


V model or "V cycle"

- V model is an extension of the Waterfall model
- V model emphasizes the relations between the development phases and associated verification phases

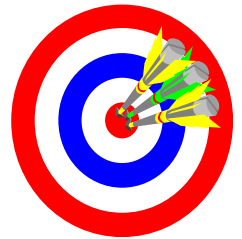


SW Development V-Cycle



Aims of the Procedure

Concentration on *what* has to be done to develop software of high quality in time, not on *how* it should be done.



Main objectives:

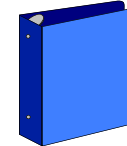
- Satisfy the **needs** and **requirements** expressed by the customer
- Create automotive software products in a **repeatable manner** based on a defined process
- **Reduce** the number of **defects** and the **costs** to correct them
- **Guarantee** the agreed **time** frame, **cost** framework and **results** via a controlled project progress



Development Process



Management Process

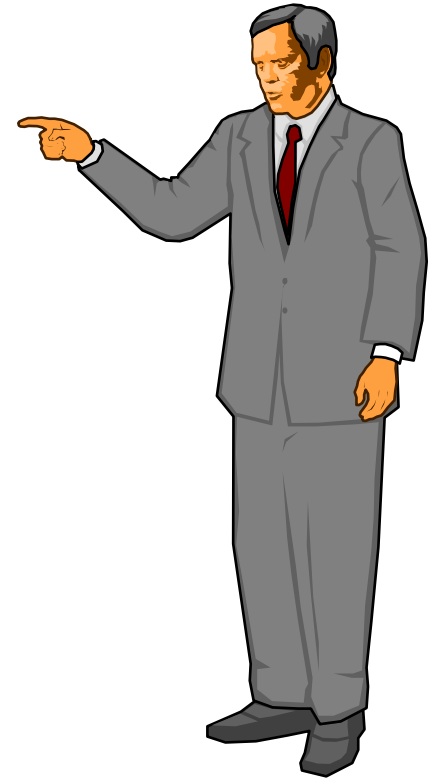


Quality Assurance Process



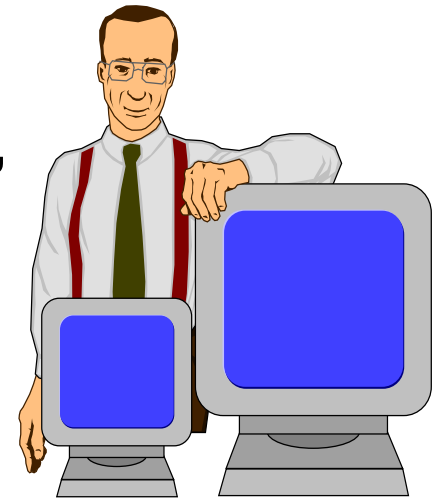
Management Process

- Software Team Leader (SWTL)
- Project Planning
- Project Control
- Configuration Management



Software Team Leader

- the main interface and partner of the Project Leader (PL) for the software development
- member of the software team & the project core team
- nominated as being responsible for the planning, tracking and managing of the software related activities
- should be involved right from the start



Software Team Leader (2)

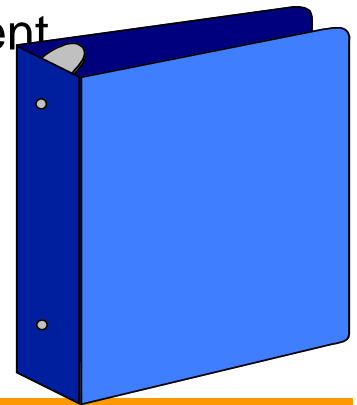
- approves the release of SW related documents and deliveries
- the PL must agree with his SWTL on
 - ◆ the rights and duties of the SWTL
 - ◆ the resources available for software
 - ◆ the milestones of the PL
 - ◆ *the handling of change requests and SW anomaly reports that affect the software*



Project Planning

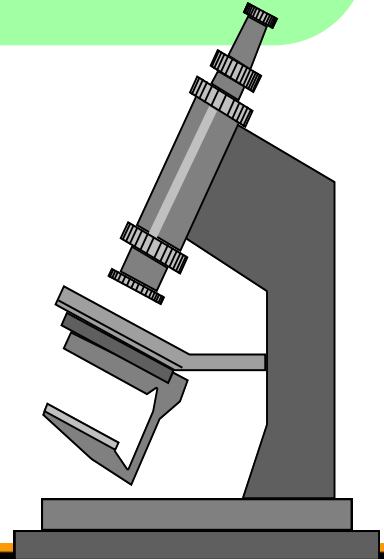
The main instrument for project planning is the *Software Development Plan*. It contains the following information:

- description of the embedded system project
- **Organization Diagram** defining all responsibilities (e.g. Software Team Leader, SCCB) along with the names assigned to them
- **schedule** and **resource planning** and the assumptions that led to it
- intended **tailoring** of the default software development process
- **risk management** which identifies the areas that represent the highest risks to the success of the project and measures against them
- guidelines and methods intended to be used during the project
- intervals for plan tracking



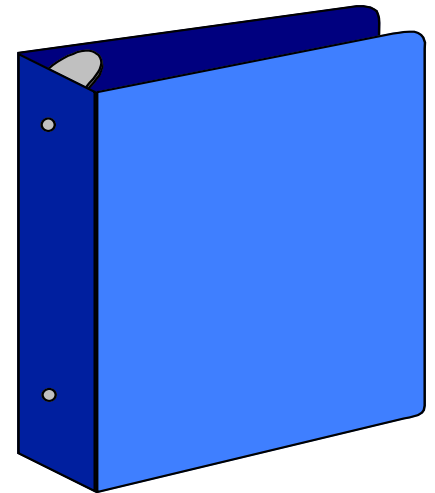
Project Control

- ▶ The purpose of *Software Project Control* is to
 - provide adequate visibility into actual progress
 - trigger management activities when the software project's performance deviates significantly from the software planning
 - check whether the current development process is still adequate
- ▶ Implementation of the process includes
 - ▶ the nomination of a **Software Team Leader** who regularly checks
 - ▶ actual results,
 - ▶ costs, and
 - ▶ performances,
 - ▶ regular **meetings** with the project team.
 - ▶ **milestones**



Software Project Folder

- ▶ the central repository for all documents related to the software part of a project
- ▶ must always contain the latest version of any plan, specification etc.
- ▶ should be accessible to all team members
- ▶ is managed by the SWTL



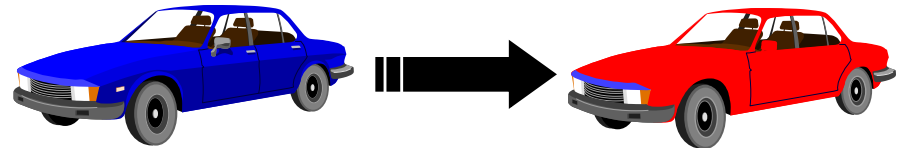
Configuration Management (1)

▶ The three parts of (Software) Configuration Management

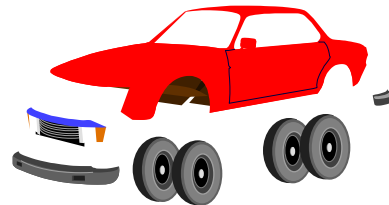
▶ Version control



▶ Change control



▶ Build control

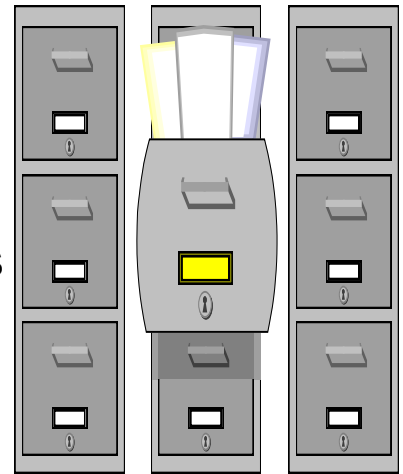


Configuration Management (2)

The purpose of *Software Configuration Management (SCM)* is to establish and maintain the integrity of the outputs of the software project throughout the life cycle.

Activities of SCM:

- **identifying** and defining the **configuration items** of a system
- **controlling** the **release** and **change** of all configuration items
- **recording** and reporting the **status** of configuration items and **change requests**
- **verifying** the **completeness** and correctness of configuration items

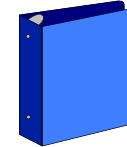




Development Process



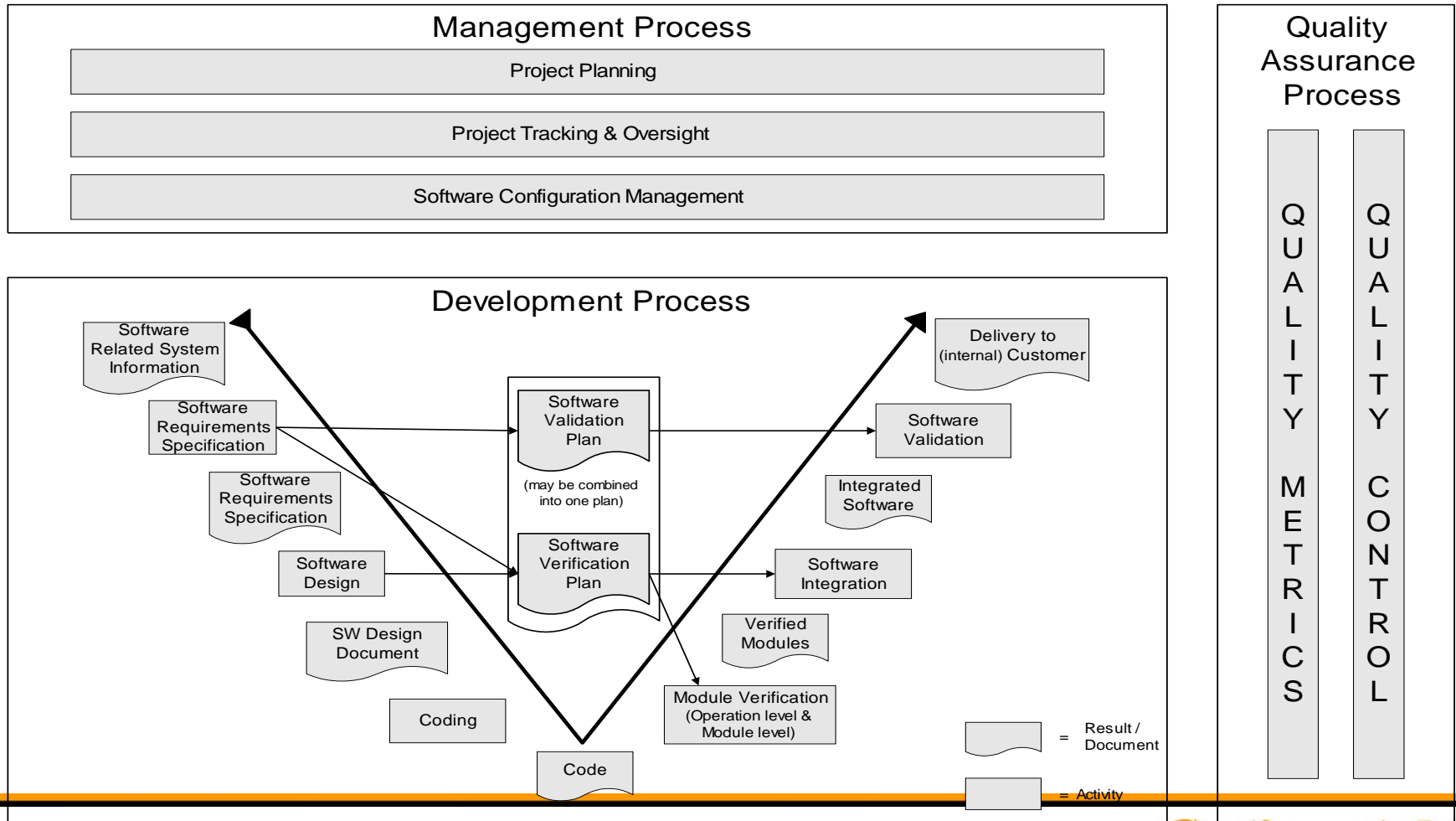
Management Process



Quality Assurance Process



SOFTWARE V-CYCLE



Software Requirements Specification

Input

- software related **system information** from customer, system department, hardware department

Output

- **software requirements specification**
- software validation plan & verification plan (partly)
- (software validation specification)

Activities

- ☐ check for completeness of software related system information
- ☐ complete software function description
- ☐ specify software validation tests
- ☐ specify tests for high level verification
- ☐ reach agreement with customer
- ☐ analyse impact of change requests / new features



Software Design

Input

- **software requirements specification**
- **change requests** (partly)
- software verification plan

Output

- **software architecture description**
 - ♦ modules, interfaces, data flow, control flow
 - ♦ real-time behaviour, interrupt system, OS
- test specification for integration
- **module description**
- test specification for module verification
- description of private/exported data/operations, algorithms etc.

Activities

- ☐ define software architecture
- ☐ define modules (**note**: there is no distinct detailed design phase)
- ☐ ensure traceability to software requirements specification
- ☐ use a design language / tools (if available) to avoid ambiguous wording



Coding

Input

- software requirements specification
- **software design documents**

Output

- translatable and linkable **source code**
 - ♦ structured
 - ♦ commented
 - ♦ according to coding standards

Activities

- ☐ write code (use tools like XTools if available)



Module Verification

Input

- software **verification plan**
- translatable source code
- **test specification** from design
- **test cases** (input/output data pairs from previous versions)

Output

- **verification report**
- test cases (for future versions)
- verified/reviewed modules

Activities

- ☐ create/refine test cases from test specification
- ☐ perform specified tests
- ☐ (automated) testing
- ☐ verification by code review



Software Integration

Input

- software **verification plan**
- verified modules
- **test specifications** from SW design and SW requirements specification
- **test cases** (from previous versions)

Activities

- ☐ create/refine test cases from test specification
- ☐ integrate modules step by step
- ☐ test all functions
- ☐ test all interfaces

Output

- verified **integrated software**
- **integration report**
- refined test cases (for future versions)



Software Validation

Input

- SW related system information
- software **requirements specification**
- software **validation plan**
- integrated software

Output

- **deliverable software**
- **release document**
- validation report

Activities

- ☐ check software performance and behavior against SW requirements spec.
- ☐ validate that the overall system behaviour fulfills all requirements (in target environment)



Note:

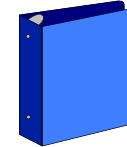
- The distinction between software validation and system validation often is not clearly defined yet. P04903 describes only validation activities directly related to SW development.



Development Process



Management Process



Quality Assurance Process



Quality Assurance Process (1)

The implementation of the *software quality assurance* depends on the project.

The set of actions ensuring that the software will effectively meet the formalized needs of the software quality must be planned in the *Software Quality Assurance Plan*.

Contents of the Software Quality Assurance Plan:

- **methods, rules** and **standards** used at each phase of the V-cycle
- **planned activities** of software quality control (i.e. reviews)
- **data** collected for **quality assessment**



Quality Assurance Process (2)

The software **quality** must be **verified** against the user **requirements** (ISO definition of quality). Also documents and code produced during each V-cycle phase must meet the quality requirements of the development process.

These activities are implemented by two sorts of *reviews*:

- **End-of-phase reviews (EoPR)**: a critical evaluation of the current project status.
- **Document reviews (DocR)**: a verification of a document and its technical examination and evaluation.



Reviews are described in detail in P04907

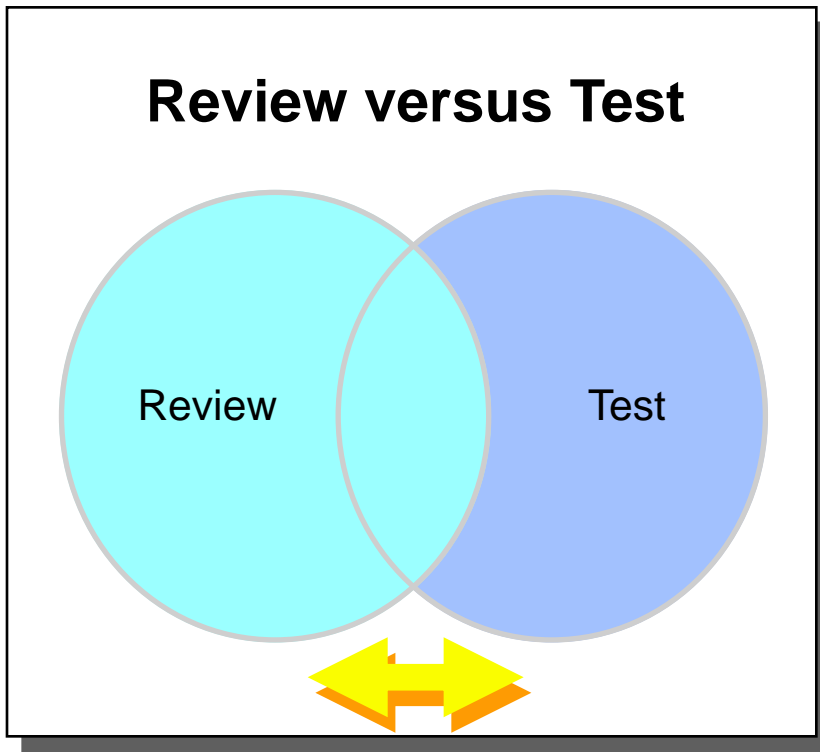
Quality Assurance Process (3)

The following reviews **must** be held:


- **document reviews** for
 - ♦ SW development plan (at project start and after major changes)
 - ♦ SW requirements specification
 - ♦ SW design document
- **end-of-phase reviews** for the
 - ♦ SW validation phase



Quality Assurance Process (3a)



- neither review nor test can achieve 100 % coverage
- both methods **complete** each other, do not replace each other
- reviews **disburden** test



reviews are an important means to improve the **software quality**

Quality Assurance Process (3b)

- ▶ Some general remarks on reviews:
- ▶ the object is judged, not the author!
- ▶ the participants are prepared!
- ▶ no solution finding during the meeting!
- ▶ open and objective discussion!
- ▶ no longer than two hours (divide in logical units if object too large)!
- ▶ reviews disburden test
- ▶ corrective actions after EoP-review if necessary



FIN

O como dicen en las películas: "The End"