Software Systems Verification and Validation

Lecture 11–12 - CMM

Lect. dr. Andreea Vescan

Babeș-Bolyai University
Cluj-Napoca

2015-2016

Five views of Software **Quality**

- Transcendental view
  - recognized through experience; viewed to be something ideal.
- User view
  - the extend to which a product meets user needs/expectations; a user is concerned with whether or not a product is fit for use.
- Manufacturing view
  - quality is seen as conforming to requirements;
  - concept of *process* - does conforming to process standards will lead to good products?
- Product view
  - hypothesis: *If a product is manufactured with good internal properties, then it will have good external qualities.*;
  - causal relationships between internal and external properties
- Value-based view
  - a merger of two independent concepts: excellence and worth
  - central idea: how much a customer is willing to pay for a certain level of quality.

Measuring **Quality**

- Measurement - a quantitative view of the quality concept.
- Reasons - for developing a quantitative view of a system:
  - Measurement allows us to establish baseline for qualities;
  - Organizations make continuous improvements in their process models - and an improvement has a cost associated with it;
  - The present level of quality of a product needs to be evaluated so the need for improvement can be investigated.
- Measurement of User's View
  - quality factors: functionality, reliability, usability;
  - Gilb's technique: The quality concept is broken down into component parts until each can be stated in terms of directly measurable attributes.
- Measurement of Manufacturer's View
  - Defect Count: How many defects have been detected?
  - Rework Cost: How much does it cost to fix the known defects?

Measuring **Quality** (cont.)

- Measurement of Manufacturer's View (cont.)
  - Defect Count: How many defects have been detected?
  - Rework Cost: How much does it cost to fix the known defects?
  - Analyze the defects
    - For each defect identify the development phase in which it was introduced and the phase in which it was discovered.
    - Categorize the defects based on modules.
    - To compare defects across modules and products in a meaningful way, normalize the defect count by product size.
    - Separate the defects found during operation from the ones found during development.
  - After defects are detected, the developers make an effort to fix them.
  - The rework cost:
    - Development Rework Cost
    - Operation Rework Cost

Quality factors

- Quality factor = a behavioral characteristic of a system.

Quality Factors	Definition
Correctness	Extent to which a program satisfies its specifications and fulfills the user's mission objectives
Reliability	Extent to which a program can be expected to perform its intended function with required precision
Efficiency	Amount of computing resources and code required by a program to perform a function
Integrity	Extent to which access to software or data by unauthorized persons can be controlled
Usability	Effort required to learn, operate, prepare input, and interpret output of a program
Maintainability	Effort required to locate and fix a defect in an operational program
Testability	Effort required to test a program to ensure that it performs its intended functions
Flexibility	Effort required to modify an operational program
Portability	Effort required to transfer a program from one hardware and/or software environment to another
Reusability	Extent to which parts of a software system can be reused in other applications
Interoperability	Effort required to couple one system with another

Figure: McCall's Quality Factors

Quality factors

- Quality factors - have been grouped into three broad categories:

Quality factors

- Quality factors - have been grouped into three broad categories:

Quality Categories	Quality Factors	Broad Objectives
Product operation	Correctness	Does it do what the customer wants?
	Reliability	Does it do it accurately all of the time?
	Efficiency	Does it quickly solve the intended problem?
	Integrity	Is it secure?
	Usability	Can I run it?
Product revision	Maintainability	Can it be fixed?
	Testability	Can it be tested?
	Flexibility	Can it be changed?
Product transition	Portability	Can it be used on another machine?
	Reusability	Can parts of it be reused?
	Interoperability	Can it interface with another system?

Figure: Categorizations of McCall's Quality Factors

Quality criteria

- Quality criterion - is an attribute of a quality factor that is related to software development.
- Relationship between Quality Factors and Criteria - characteristics
  - If an effort is made to improve one quality factor, another quality factor may be degraded.
  - Some quality factors positively impact others.

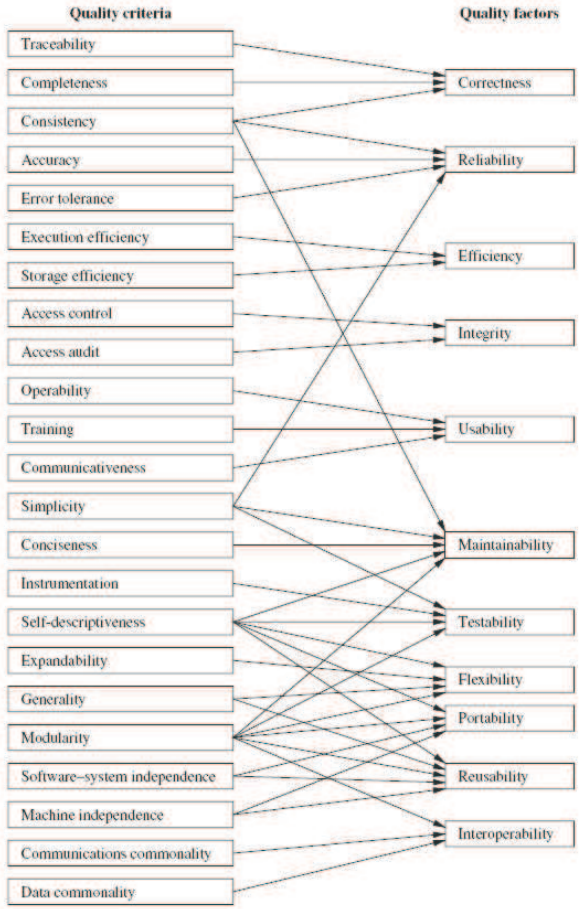
Quality criteria (cont.)

Quality Criteria	Definition
Access audit	Ease with which software and data can be checked for compliance with standards or other requirements
Access control	Provisions for control and protection of the software and data
Accuracy	Precision of computations and output
Communication commonality	Degree to which standard protocols and interfaces are used
Completeness	Degree to which a full implementation of the required functionalities has been achieved
Communicativeness	Ease with which inputs and outputs can be assimilated
Conciseness	Compactness of the source code, in terms of lines of code
Consistency	Use of uniform design and implementation techniques and notation throughout a project
Data commonality	Use of standard data representations
Error tolerance	Degree to which continuity of operation is ensured under adverse conditions
Execution efficiency	Run time efficiency of the software
Expandability	Degree to which storage requirements or software functions can be expanded

Quality Criteria	Definition
Generality	Breadth of the potential application of software components
Hardware independence	Degree to which the software is dependent on the underlying hardware
Instrumentation	Degree to which the software provides for measurement of its use or identification of errors
Modularity	Provision of highly independent modules
Operability	Ease of operation of the software
Self-documentation	Provision of in-line documentation that explains implementation of components
Simplicity	Ease with which the software can be understood.
Software system independence	Degree to which the software is independent of its software environment—nonstandard language constructs, operating system, libraries, database management system, etc.
Software efficiency	Run time storage requirements of the software
Traceability	Ability to link software components to requirements
Training	Ease with which new users can use the system

Figure: McCall's Quality Criteria

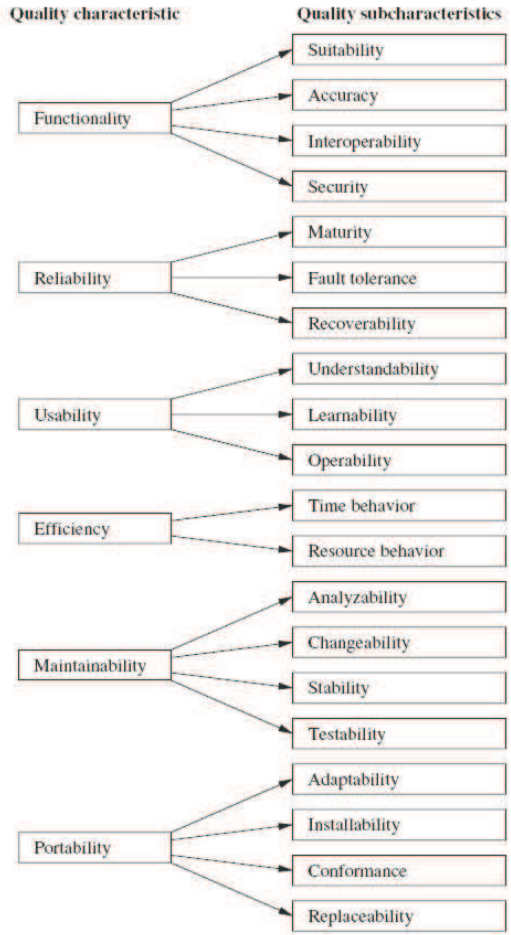
Quality criteria (cont.)



ISO 9126

- To define general framework for software quality.
- ISO 9126 - is a standardized software quality document which defines six independent categories of quality characteristics:
  - Functionality
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability
- The quality characteristics are further decomposed into more concrete subcharacteristics.

ISO 9126 (cont.)



McCall's quality model and the ISO 9126 model

- The two models focus on the same abstract entity: software quality.
- Similarities:
  - quality factor (McCall) $\equiv$ quality characteristic (ISO 9126)
  - quality factors/characteristics are found in both models: reliability, usability, efficiency, maintainability, portability.
- Differences:
  - ISO 9126 model - characteristics visible to the users, whereas -McCall - internal qualities as well.
  - McCall: one quality criterion can impact several quality factors, whereas -ISO 9126 - one subcharacteristic impacts exactly one quality characteristics.
  - A high-level quality factor, such as testability, in the McCall model is a low-level subcharacteristic of maintainability in the ISO 9126 model.

ISO 9000:2000

- Standardization in the field of communication - positive
- Standardization in the field of software development - mixed reactions:
  - it curtails individual drive to be innovative;
  - reduce the activity of reinventing the same or similar processes for development and quality assurance.
- There are three components of the ISO 9000:2000 standard:
  - ISO 9000: Fundamentals and vocabulary
  - ISO 9001: Requirements
  - ISO 9004: Guidelines for performance improvements

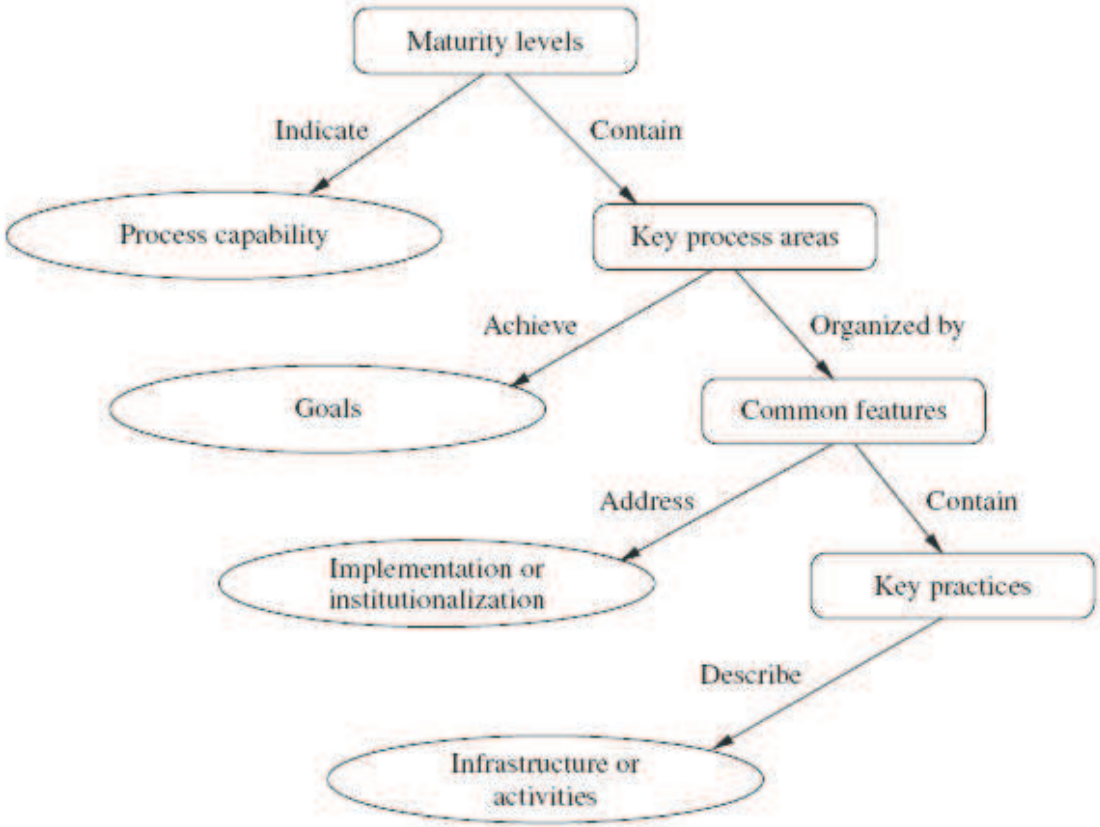
Software **process**

- Follow a defined process - benefits:
  - The process can be repeated in subsequent projects.
  - The process can be evaluated by using a variety of metrics (cost, quality, time to deliver the product).
  - Actions can be taken to improve the process to achieve better results.
- To be able to improve a defined process, organizations need to evaluate its capabilities and limitations.
- SEI developed the Capability Maturity Model (CMM)
- Testing Maturity Model (TMM) - to evaluate a testing process.
- The Test Process Improvement (TPI) model - for an organization to be able to improve its testing process.

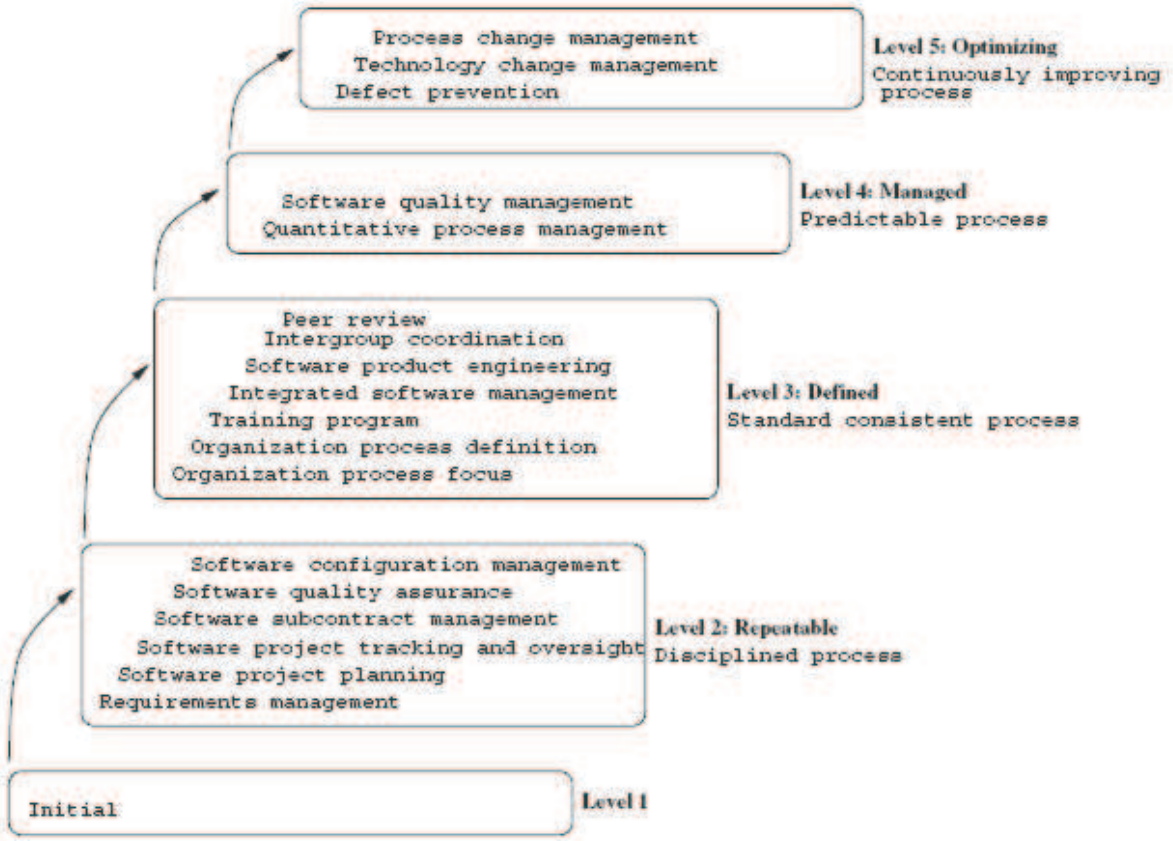
Capability Maturity Models

- Three desirable attributes in software development processes:
  - The products are of the highest quality.
  - Projects are completed according to their plans (schedules).
  - Projects are completed within the allocated budgets.
- The customers needs to gain confidence that the organization is capable of delivering the desired product, and such confidence can be gained by evaluating the capabilities of the organizations.
- The U.S. Department of Defence wanted to have a framework to evaluate the maturity of software processes used by organizations.
- The maturity level of a development process tells us to what extent the organization is capable of producing low-cost, high-quality software. Therefore, the evaluation framework is the CMM.

CMM Architecture



SW-CMM Maturity Levels



Five Levels of Maturity

- Level 1: Initial - no process model
- Level 2: Repeatable - concept of process; disciplined level - processes are used for repeatability
- Level 3: Defined - documentation; following an approved process
- Level 4: Managed - metrics - processes and products; correction actions are triggered
- Level 5: Optimizing - improve processes on a continual basis.

Test Process Improvement

- A test process - is a certain way of performing activities related to defect detection.
- A test process need to be improved for three reasons:
  - Quality - intention to have an improved test process - that gives us better insight into the quality of the system being tested.
  - Lead Time - a better test process saves testing time; example - prioritizing the execution of test cases so that difficult-to-fix defects are detected as early as possible.
  - Cost - a better test process is expected to be carried out with a lower cost and thereby reduces the overall cost of system development.

Levels of Maturity of Key Areas

- A test process is evaluated with respect to the 20 key areas. One is interested in knowing to what extend a certain key area has matured.
- The maturity levels of the key areas are denoted by A, B, C, and D (A-lowest level, D-highest level).

Levels of Maturity of Key Areas (cont)

Key area	Level A	Level B	Level C	Level D
Test strategy	Strategy for single high-level test	Combined strategy for high-level tests	Combined strategy for high-level tests plus low-level tests or evaluation	Combined strategy for all test and evaluation levels
Life-cycle model	Planning, specification, execution	Planning preparation, specification, execution, and completion		
Moment of involvement	Completion of test basis	Start of test basis	Start of requirements definition	Project initiation
Estimating and planning	Substantiated estimating and planning	Statistically substantiated estimating and planning		
Test specification technique	Informal techniques	Formal techniques		
Static test techniques	Inspection of test basis	Checklists		
Metrics	Project metrics (product)	Project metrics (process)	System metrics	Organization metrics (more than one system)
Test tools	Planning and control tools	Execution and analysis tools	Extensive automation of the test process	
Test environment	Managed and controlled test environment	Testing in most suitable environment	Environment on call	
Office environment	Adequate and timely office environment			
Commitment and motivation	Assignment of budget and time	Testing integrated in project organization	Test engineering	
Test functions and training	Test manager and testers	(Formal) methodical, technical, and functional support, management	Formal internal quality assurance	

Levels of Maturity of Key Areas (cont)

Key area	Level A	Level B	Level C	Level D
Scope of methodology	Project specific	Organization generic	Organization optimizing, R&D activities	
Communication	Internal communication	Project communication (defects, change control)	Communication within organization about the quality of the test process	
Reporting	Defects	Progress (status of tests and products), activities (costs and time, milestones), defects with priorities	Risks and recommendations, substantiated with metrics	Recommendations have a software process improvement character
Defect management	Internal defect management	Extensive defect management with flexible reporting facilities	Project defect management	
Testware management	Internal testware management	External management of test basis and test object	Reusable testware	Traceability system requirements to test cases
Test process management	Planning and execution	Planning, execution, monitoring, and adjusting	Monitoring and adjusting within organization	
Evaluation	Evaluation techniques	Evaluation strategy		
Low-level testing	Low-level test life cycle (planning, specification, and execution)	White-box techniques	Low-level test strategy	

Levels of Maturity of Key Areas (cont)

	Scale												
	Controlled						Efficient					Optimizing		
Key area	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Test strategy		A	—	—	—	—	B	—	—	—	C	—	D	
Life-cycle model		A	—	—	B									
Moment of involvement			A	—	—	—	B	—	—	—	C	—	D	
Estimating and planning				A	—	—	—	—	—	—	B			
Test specification techniques		A	—	B										
Static test techniques					A	—	B							
Metrics						A	—	—	B	—	—	C	—	D
Test tools					A	—	—	B	—	—	C			
Test environment				A	—	—	—	B	—	—	—	—	—	C
Office environment				A										
Commitment and motivation		A	—	—	—	B	—	—	—	—	—	C		
Test functions and training				A	—	—	B	—	—	—	C			
Scope of methodology					A	—	—	—	—	—	B	—	—	C
Communication			A	—	B	—	—	—	—	—	—	C		
Reporting		A	—	—	B	—	C	—	—	—	—	D		
Defect management		A	—	—	—	B	—	C						
Testware management			A	—	—	B	—	—	—	C	—	—	—	D
Test process management		A	—	B	—	—	—	—	—	—	—	C		
Evaluation							A	—	—	B				
Low-level testing					A	—	B	—	C					

Maturity Levels of Test Processes

- The 13 scales of maturity are divided into three qualitative groups:
  - controlled - a test process executed in a controlled manner means that all component activities are planned, and those are executed in phases according to a planned strategy.
  - efficient - more effort needs to be made to achive efficienty in testing.
  - optimizing - optimizing a test process means performing testing tasks in the best possible manner from the standpoints of quality, time and cost.

Testing Maturity Model

- There is a need for a framework to asses and improve testing processes.
- The TMM - describes an evolutionary path of test process maturity in five levels (stages).
- The TMM - gives guidance concerning how to improve a test process.
- Each state is characterized by the concepts of:
  - Maturity Goals
  - Maturity Subgoals
  - Activities, Task, and Responsibilities

Levels of Testing Maturity Model

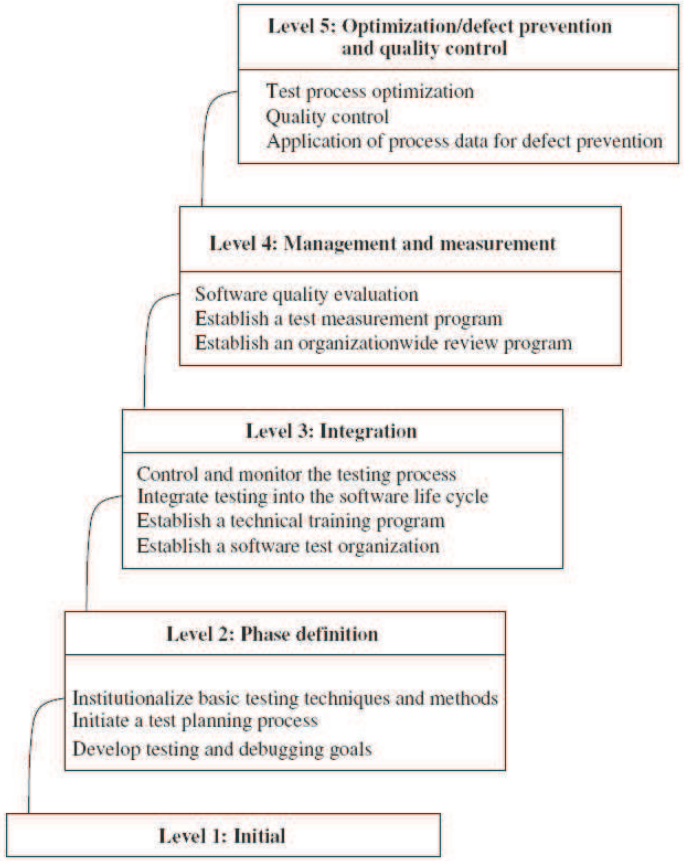


Figure: Five levels of TMM

Next lecture - Final Exam

- Exam - dates
  - UBB - homepage
    `https://www.cs.ubbcluj.ro/programarea-sesiunii-de-examen`
  - SSVV - homepage
    `https://www.cs.ubbcluj.ro/~avescan/?q=node/219`
- Grades for laboratories + seminars + practical