UNIVERSITÄT
BERN

Introduction to Software Engineering

11. Software Quality

Roadmap

- What is quality?
- Quality Attributes
- Quality Assurance: Planning and Reviewing
- Quality System and Standards

Sources

- *Software Engineering*, I. Sommerville, 7th Edn., 2004.
- *Software Engineering — A Practitioner's Approach*, R. Pressman, Mc-Graw Hill, 5th Edn., 2001.
- *Fundamentals of Software Engineering*, C. Ghezzi, M. Jazayeri, D. Mandroli, Prentice-Hall 1991

Roadmap

- **What is quality?**
- Quality Attributes
- Quality Assurance: Planning and Reviewing
- Quality System and Standards

What is Quality?

Software Quality is *conformance to*:

- explicitly stated *functional and performance requirements*,
- explicitly documented *development standards*,
- *implicit characteristics* that are expected of all professionally developed software.

Problems with Software Quality

- Software specifications are usually *incomplete and often inconsistent*

- There is *tension* between:
  - customer quality requirements (efficiency, reliability, etc.)
  - developer quality requirements (maintainability, reusability, etc.)

- Some quality requirements are *hard to specify* in an unambiguous way
  - directly measurable qualities (e.g., errors/KLOC),
  - indirectly measurable qualities (e.g., usability).

Quality management is not just about reducing defects!

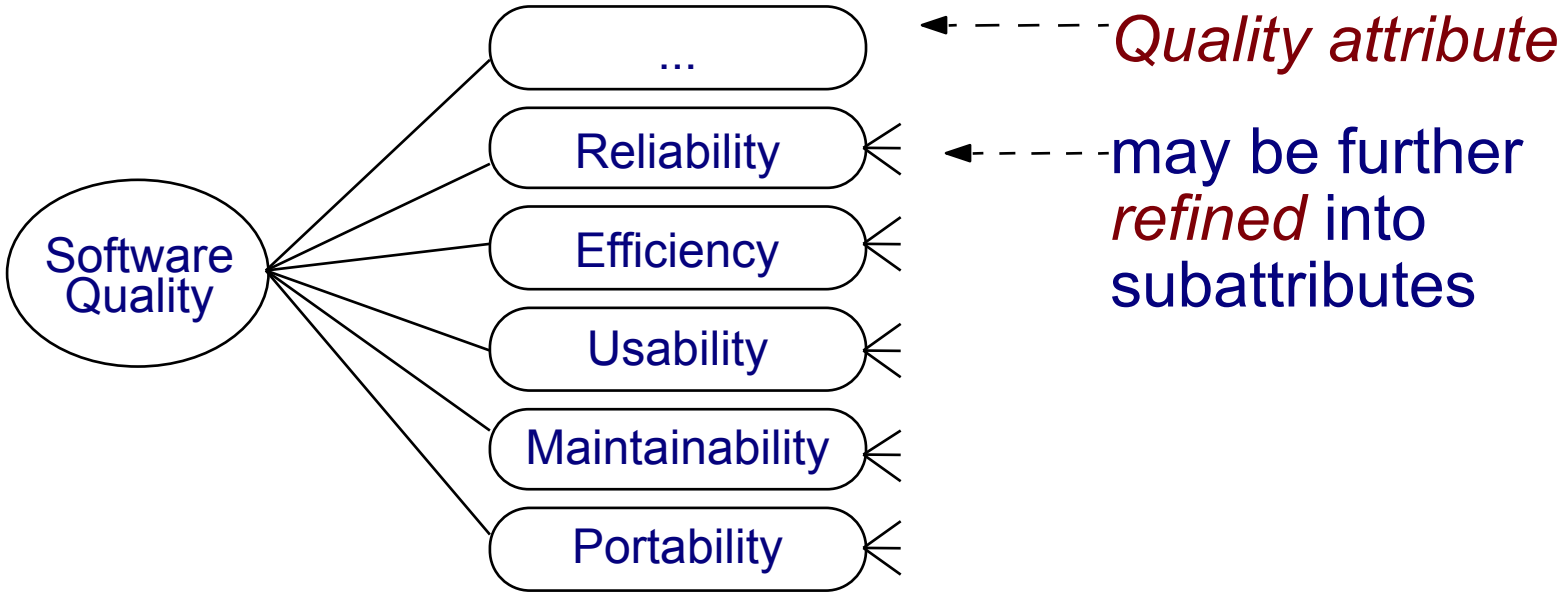
Roadmap

- What is quality?
- **Quality Attributes**
- Quality Assurance: Planning and Reviewing
- Quality System and Standards

Hierarchical Quality Model

Define quality via hierarchical quality model, i.e. a number of *quality attributes* (a.k.a. quality factors, quality aspects, ...)

Choose quality attributes (and weights) depending on the project context



Quality Attributes

Quality attributes apply both to the product and the process.

- ***product***: delivered to the customer
- ***process:*** produces the software product
- ***resources:*** (both the product and the process require resources)
  - Underlying assumption: a quality process leads to a quality product (cf. metaphor of manufacturing lines)

Quality Attributes ...

Quality attributes can be external or internal.

- ***External:*** Derived from the relationship between the environment and the system (or the process). (To derive, the system or process must run)
  - e.g. Reliability, Robustness
- ***Internal:*** Derived immediately from the product or process description (To derive, it is sufficient to have the description)
  - Underlying assumption: internal quality leads to external quality (cfr. metaphor manufacturing lines)
  - e.g. Efficiency

Correctness, Reliability, Robustness

Correctness

- A system is correct if it *behaves according to its specification*
  - An *absolute property* (i.e., a system cannot be “almost correct”)
  - ... in theory and practice *undecidable*

Reliability

- The user may rely on the system behaving properly
- Reliability is the *probability* that the system will operate as expected over a specified interval
  - A *relative property* (a system has a mean time between failure of 3 weeks)

Robustness

- A system is robust if it behaves reasonably *even in circumstances that were not specified*
- A *vague property* (once you specify the abnormal circumstances they become part of the requirements)

Efficiency, Usability

Efficiency (Performance)

- *Use of resources* such as computing time, memory
  - Affects user-friendliness and scalability
  - Hardware technology changes fast!
  - *First do it, then do it right, then do it fast*

- For process, resources are manpower, time and money
  - relates to the “productivity” of a process

Efficiency, Usability ...

Usability (User Friendliness, Human Factors)

- The *degree* to which the human users find the system (process) *both "easy to use" and useful*
  - — Depends a lot on the target audience (novices vs. experts)
  - — Often a system has various kinds of users (end-users, operators, installers)
  - — Typically expressed in "amount of time to learn the system"

Maintainability

- *External product attributes* (evolvability also applies to process)

Maintainability

- How easy it is to *change* a system after its initial release
  - software entropy ⇒ maintainability gradually decreases over time

Maintainability ...

Is often refined to ...

Repairability

> How much work is needed to *correct* a defect

Evolvability (Adaptability)

> How much work is needed to *adapt* to changing requirements (both system and process)

Portability

> How much work is needed to *port* to new environment or platforms

Verifiability, Understandability

- *Internal (and external) product attribute*

Verifiability

- How easy it is to *verify* whether desired attributes are there?
  - — internally: e.g., verify requirements, code inspections
  - — externally: e.g., testing, efficiency

Understandability

- How easy it is to *understand* the system
  - — internally: contributes to maintainability
  - — externally: contributes to usability

Productivity, Timeliness, Visibility

- *External process attribute* (visibility also internal)

Productivity

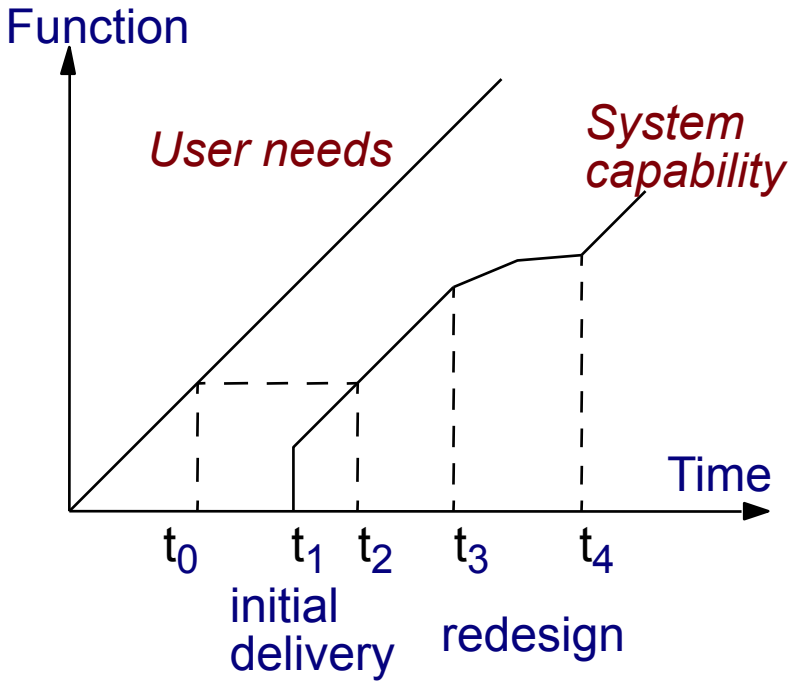
- Amount of product produced by a process for a given number of resources
  - productivity among individuals varies a lot
  - often:

$$\text{productivity}\left(\sum \text{individuals}\right) < \sum \text{productivity (individuals)}$$

Productivity, Timeliness, Visibility ...

Timeliness

- Ability to *deliver the product on time*
  - — important for marketing ("short time to market")
  - — often a reason to sacrifice other quality attributes
  - — incremental development may provide an answer



Productivity, Timeliness, Visibility ...

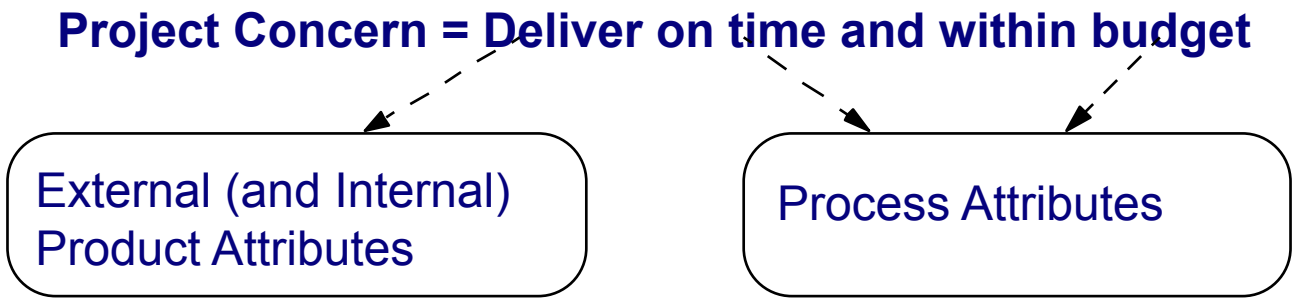
Visibility (Transparency)

- Current process steps and project status are accessible
  - — important for management
  - — also deal with staff turn-over

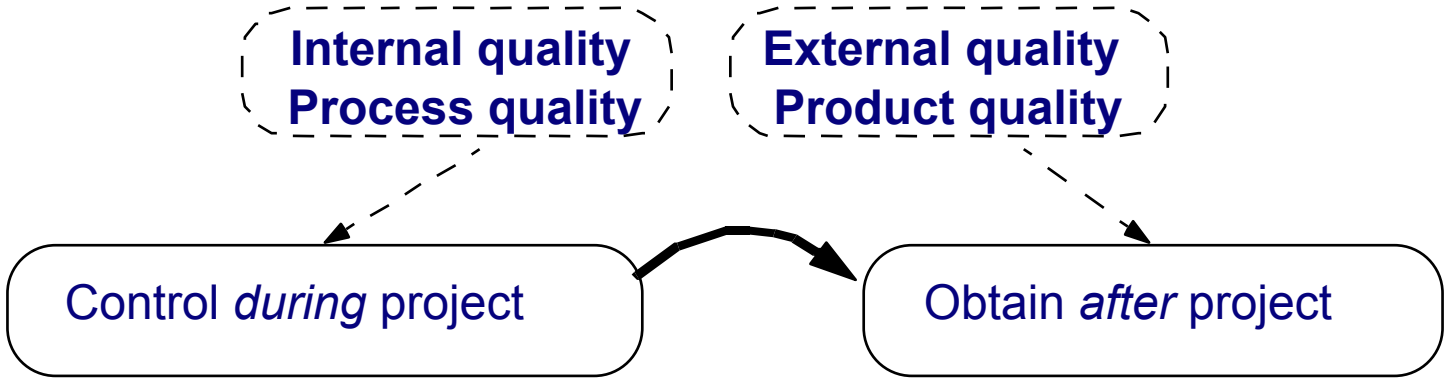
Roadmap

- What is quality?
- Quality Attributes
- **Quality Assurance: Planning and Reviewing**
- Quality System and Standards

Quality Control Assumption



Assumptions:



Otherwise, quality is mere coincidence!

The Quality Plan

A quality plan should:

- set out desired product qualities and how these are assessed
  - define the most significant quality attributes
- define the quality assessment process
  - i.e., the controls used to ensure quality
- set out which organisational standards should be applied
  - may define new standards, i.e., if new tools or methods are used

NB: Quality Management should be separate from project management to ensure independence

Software Quality Controls

1. Reviews
   - *Inspections* for defect removal (product)
   - *Progress Assessment Reviews* (product and process)
   - *Quality reviews* (product and standards)

2. Automated Software Assessment
   - *Measure* software atributes and compare to standards (e.g., defect rate, cohesion, etc.)

Types of Quality Reviews

A quality review is carried out by a group of people who carefully examine part or all of a software system and its associated documentation.

- Reviews should be *recorded and records maintained*
  - Software or documents may be *"signed off"* at a review
  - Progress to the next development stage is thereby *approved*

Types of Quality Reviews …

Review type	***Principal purpose***
Formal Technical Reviews (a.k.a. design or program inspections)	Driven by *checklist* > detect detailed errors in any product > mismatches between requirements and product > check whether standards have been followed.
Progress reviews	Driven by *budgets, plans and schedules* > check whether project runs according to plan > requires precise milestones > both a process and a product review

Review Meetings

Review meetings should:

- typically involve 3-5 people
- require a maximum of 2 hours advance preparation
- last less than 2 hours

Review Minutes

The review report should *summarize*:

1. *What* was reviewed
2. *Who* reviewed it?
3. *What* were the findings and conclusions?

The review should *conclude* whether the product is:

1. *Accepted* without modification
2. *Provisionally accepted*, subject to corrections (no follow-up review)
3. *Rejected*, subject to corrections and follow-up review

Review Guidelines

1. Review the *product*, not the producer
2. Set an *agenda* and maintain it
3. *Limit debate* and rebuttal
4. *Identify problem areas*, but don't attempt to solve every problem noted
5. Take *written notes*
6. *Limit the number of participants* and insist upon advance preparation
7. Develop a *checklist* for each product that is likely to be reviewed
8. *Allocate resources* and time schedule for reviews
9. Conduct meaningful *training* for all reviewers
10. *Review* your early reviews

Sample Review Checklists (I)

Software Project Planning

1. Is software scope unambiguously defined and bounded?
2. Are resources adequate for scope?
3. Have risks in all important categories been defined?
4. Are tasks properly defined and sequenced?
5. Is the basis for cost estimation reasonable?
6. Have historical productivity and quality data been used?
7. Is the schedule consistent?

...

Sample Review Checklists (II)

Requirements Analysis

1. Is information domain analysis complete, consistent and accurate?
2. Does the data model properly reflect data objects, attributes and relationships?
3. Are all requirements traceable to system level?
4. Has prototyping been conducted for the user/customer?
5. Are requirements consistent with schedule, resources and budget?

...

Sample Review Checklists (III)

Design

1. Has modularity been achieved?
2. Are interfaces defined for modules and external system elements?
3. Are the data structures consistent with the information domain?
4. Are the data structures consistent with the requirements?
5. Has maintainability been considered?

...

Sample Review Checklists (IV)

Code

1. Does the code reflect the design documentation?
2. Has proper use of language conventions been made?
3. Have coding standards been observed?
4. Are there incorrect or ambiguous comments?

...

Sample Review Checklists (V)

Testing

1. Have test resources and tools been identified and acquired?
2. Have both white and black box tests been specified?
3. Have all the independent logic paths been tested?
4. Have test cases been identified and listed with expected results?
5. Are timing and performance to be tested?

Review Results

Comments made during the review should be *classified*.

- ***No action.***
  - — No change to the software or documentation is required.
- ***Refer for repair.***
  - — Designer or programmer should correct an identified fault.
- ***Reconsider overall design.***
  - — The problem identified in the review impacts other parts of the design.

Requirements and specification errors may have to be referred to the client.

Roadmap

- What is quality?
- Quality Attributes
- Quality Assurance: Planning and Reviewing
- **Quality System and Standards**

Product and Process Standards

Product standards define *characteristics that all components should exhibit.*

Process standards define *how the software process should be enacted*.

Product standards
Design review form
Document naming standards
Procedure header format
Java conventions
Project plan format
Change request form

Process standards
Design review conduct
Submission of documents
Version release process
Project plan approval process
Change control process
Test recording process

Potential Problems with Standards

- Not always seen as *relevant and up-to-date* by software engineers
- May involve too much *bureaucratic form filling*
- May require *tedious manual work* if unsupported by software tools
  - *Limit overhead to effectively apply standards*

Sample Java Code Conventions

4.2 Wrapping Lines

When an expression will not fit on a single line, break it according to these general principles:

- Break after a comma.
- Break before an operator.
- Prefer higher-level breaks to lower-level breaks.
- Align the new line with the beginning of the expression at the same level on the previous line.
- If the above rules lead to confusing code or to code that's squished up against the right margin, just indent 8 spaces instead.

Sample Java Code Conventions ...

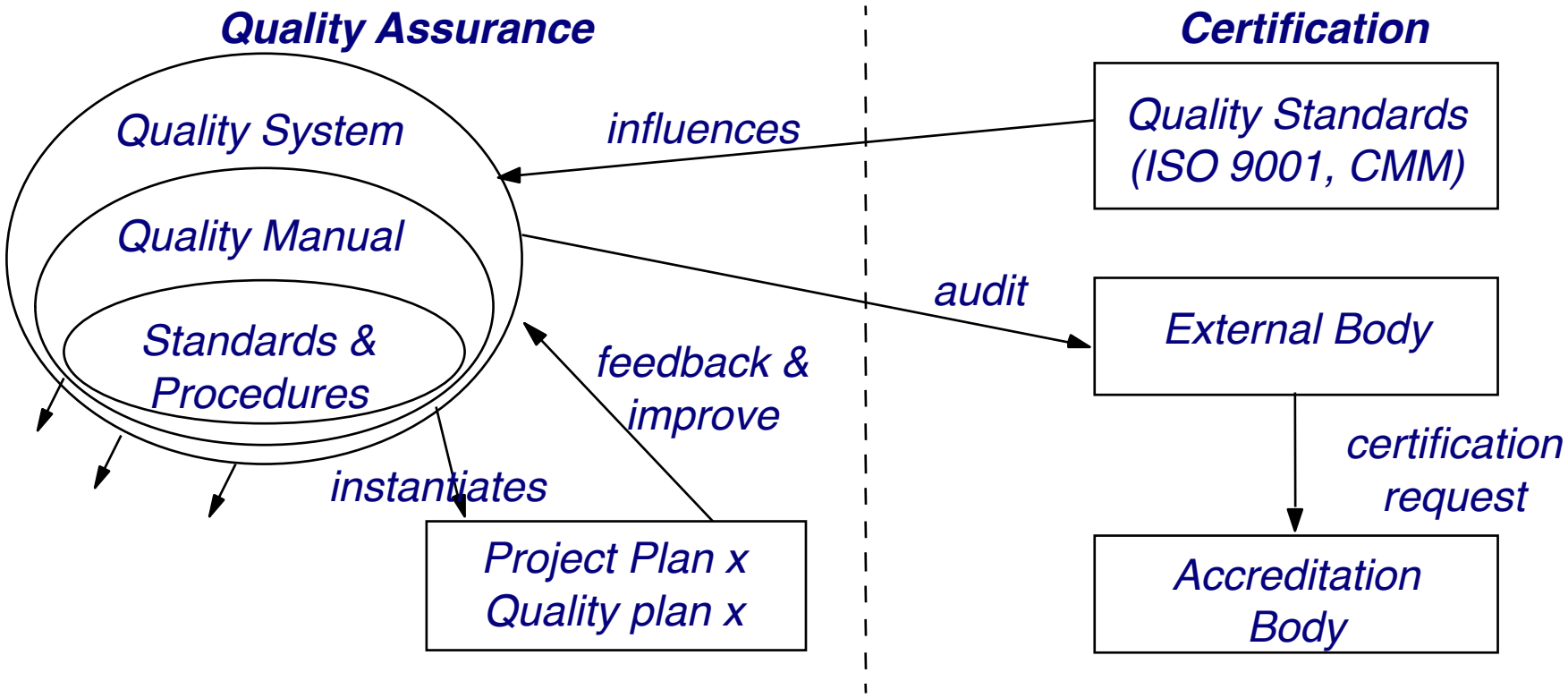
10.3 Constants

Numerical constants (literals) should not be coded directly, except for -1, 0, and 1, which can appear in a for loop as counter values.

Source: http://java.sun.com/docs/codeconv/CodeConventions.pdf

Quality System

A Quality Plan should be an instance of an organization's *Quality System*



Customers may require an externally reviewed quality system

ISO 9000

ISO 9000 is an international set of standards for *quality management* applicable to a range of organisations from manufacturing to service industries.

ISO 9001 is a *generic model of the quality process*, applicable to organisations whose business processes range all the way from design and development, to production, installation and servicing;

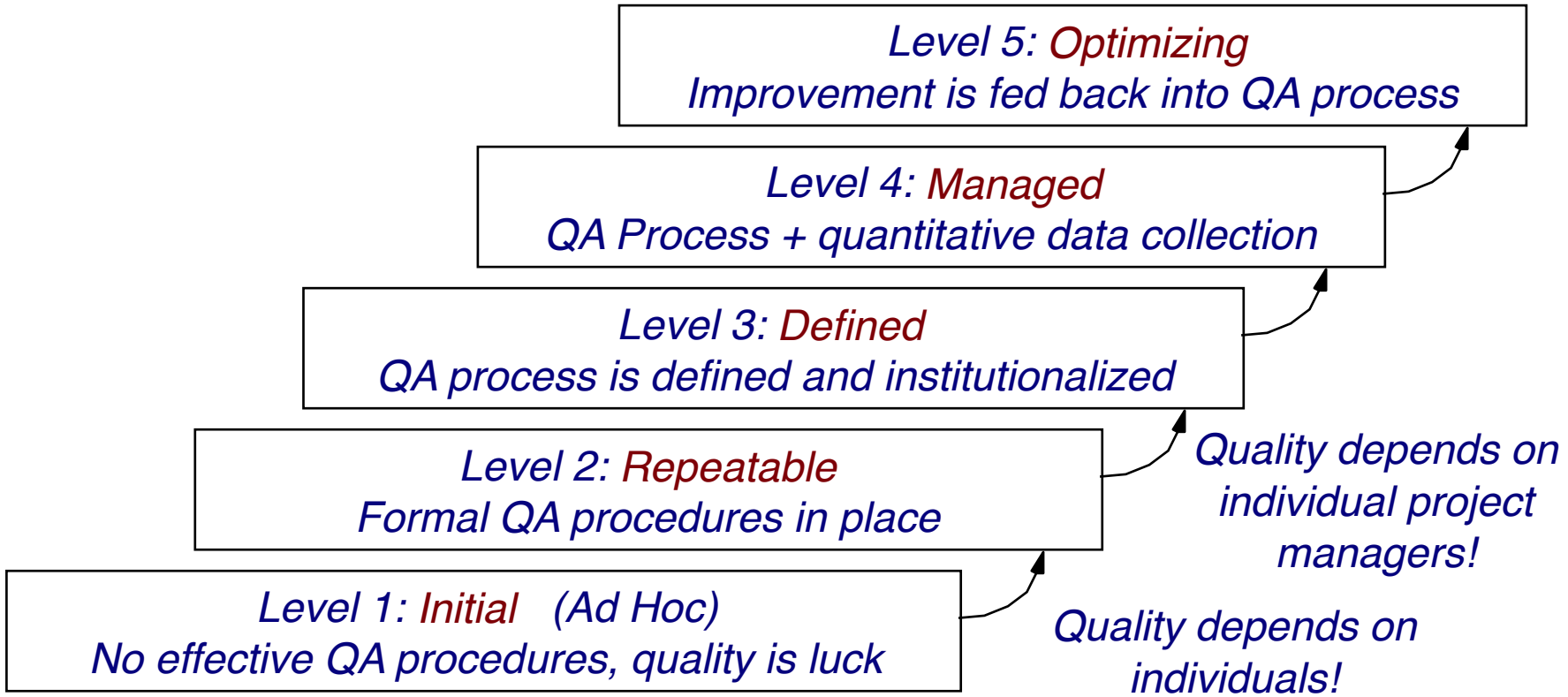
- ISO 9001 must be *instantiated for each organisation*
- ISO 9000-3 *interprets ISO 9001 for the software developer*

ISO = International Organisation for Standardization

- ISO main site: http://www.iso.ch/
- ISO 9000 main site: http://www.tc176.org/

Capability Maturity Model (CMM)

The SEI process maturity model classifies how well contractors manage software processes



What you should know!

- Can a correctly functioning piece of software still have poor quality?
- What's the difference between an external and an internal quality attribute?
- And between a product and a process attribute?
- Why should quality management be separate from project management?
- How should you organize and run a review meeting?
- What information should be recorded in the review minutes?

Can you answer the following questions?

- Why does a project need a quality plan?
- Why are coding standards important?
- What would you include in a documentation review checklist?
- How often should reviews be scheduled?
- Would you trust software developed by an ISO 9000 certified company?
- And if it were CMM level 5?

License