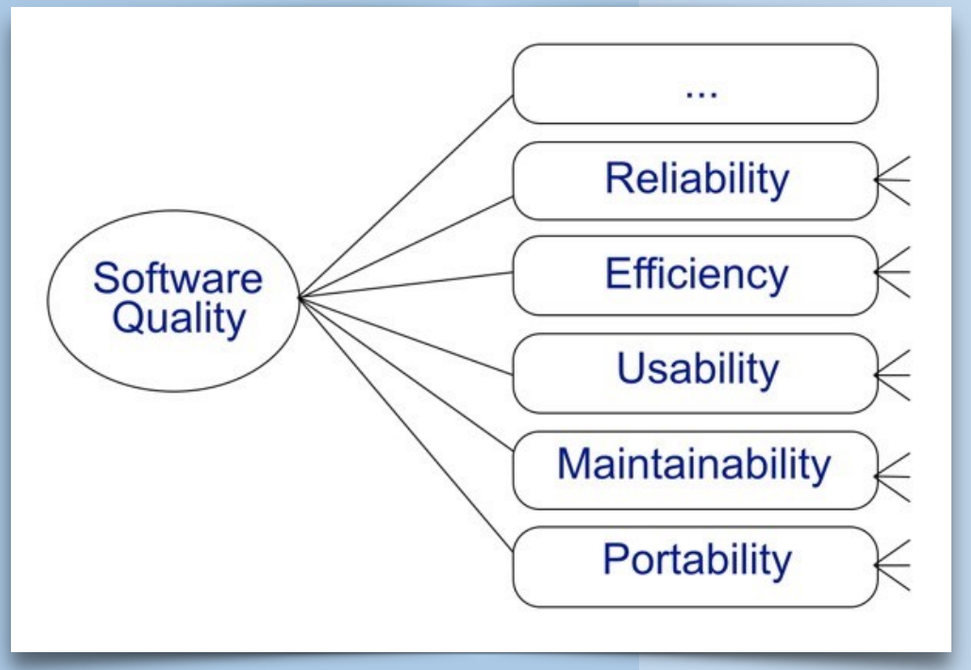
Introduction to Software Engineering

Software Quality



Roadmap

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

Sources

- *Software Engineering*. Ian Sommerville. Addison-Wesley, 10th edition, 2015
- *Software Engineering: A Practitioner's Approach*. Roger S. Pressman. McGraw Hill; 8th edition, 2003.
- *Fundamentals of Software Engineering*, C. Ghezzi, M. Jazayeri, D. Mandroli, Prentice-Hall; 2nd edition, 2002.

Roadmap

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

Which one would you choose? (and why?)

What’s the difference between the two?

Try to enumerate as many differences as you can?

How do these differences translate to “quality”?

What is Quality?

Software Quality is *conformance to*:

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

Problems with Software Quality

- There is *tension* between:
  - —**customer** quality requirements (efficiency, reliability, etc.)
  - —**developer** quality requirements (maintainability, reusability, etc.)
  - —**organisation** quality requirements (standard conformance, portfolio management)

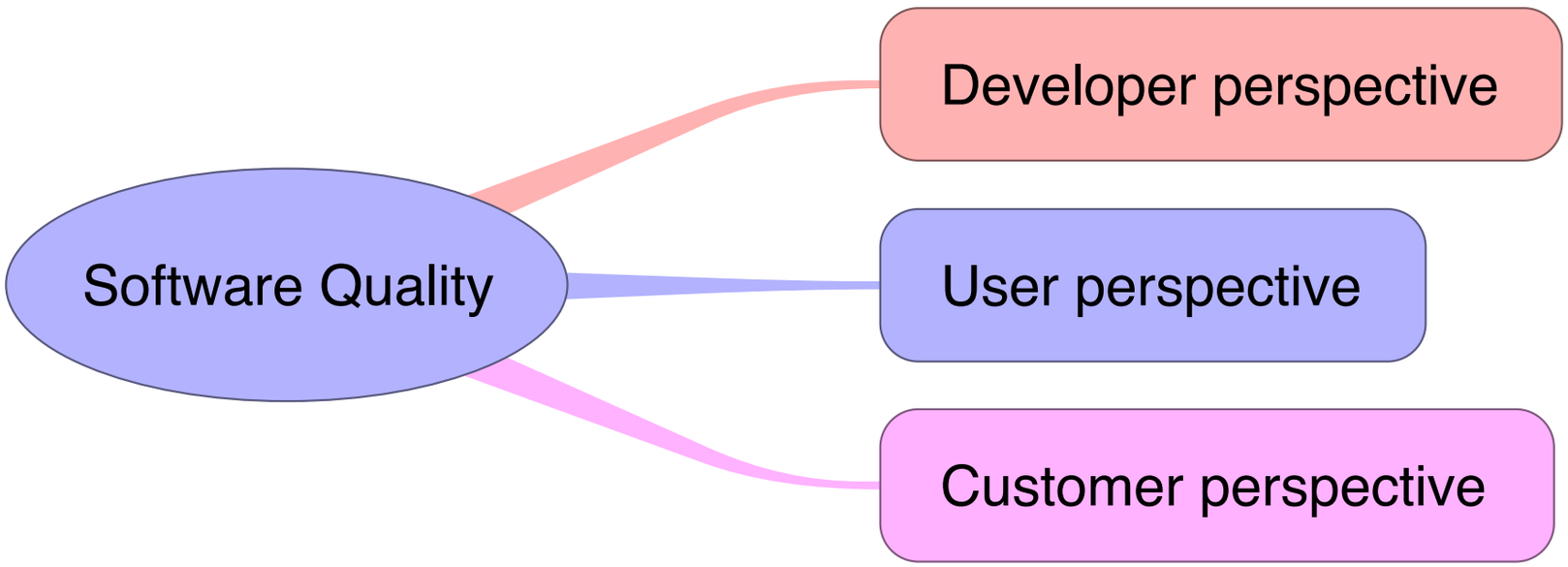
Quality management is not just about reducing defects!

How can we specify quality?

- Software specifications are usually *incomplete and often inconsistent*

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

What is Software Quality?



What attributes do you expect of software of good “quality”?

Come up with as many attributes as you can. Then try to categorize these attributes from the perspectives of various stakeholders, such as the developer, the end user, and the customer paying for the software.

A “mind map” can be a good way to structure and organize these attributes:

https://en.wikipedia.org/wiki/Mind_map

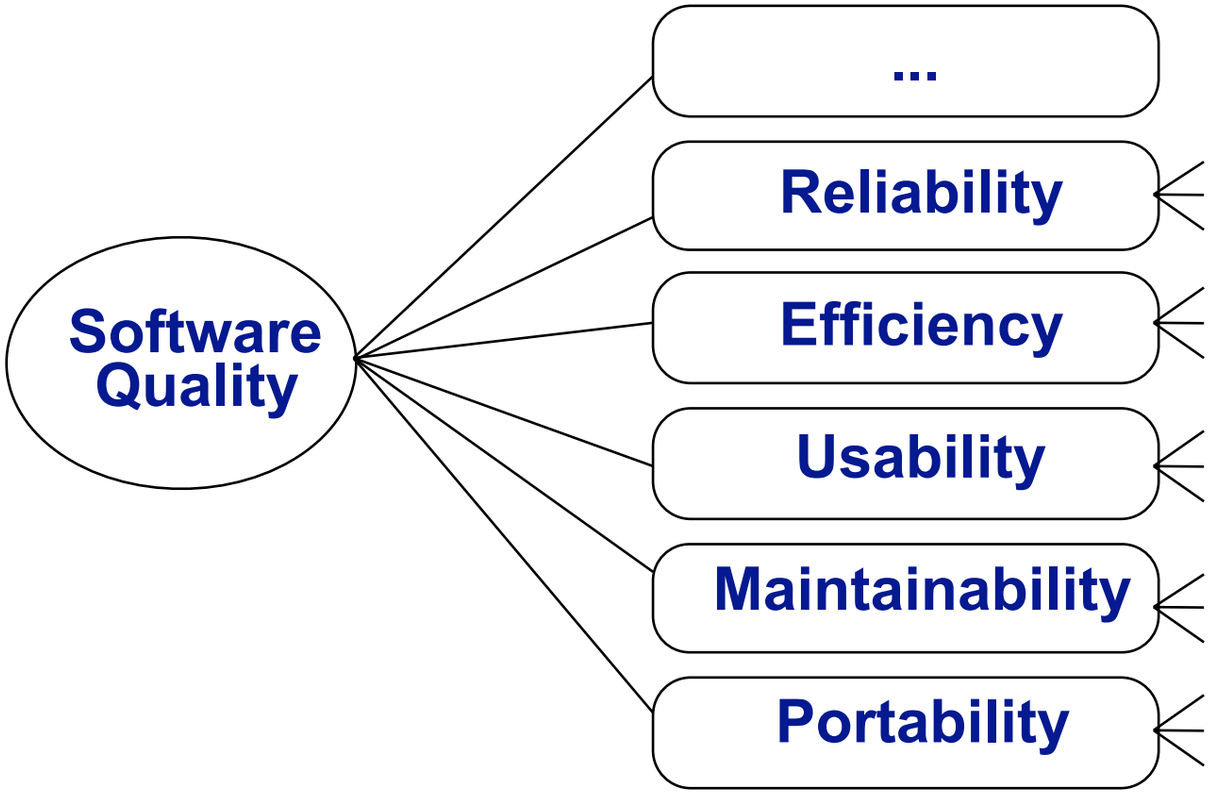
Roadmap

- 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 attribute

may be refined into further sub-attributes

Software quality attributes are sometimes referred to as “ilities” (since many end with the suffix “ility”). Notice how many of those listed here are related to non-functional requirements.

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)

In the end we are interested in ensuring the *quality of the end product*.

A reasonable assumption, however, is that we cannot produce a product of acceptable quality, unless the *process* we use to develop it as well as the *resources* used to produce it also are of adequate quality. Hence a sound engineering approach to quality considers these aspects as well.

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
  - —e.g. Efficiency

The key point is that an *internal* quality attribute can be checked in isolation: *Are classes documented with Javadoc? Is the code adequately covered by tests?*

An *external* quality attribute can only be checked in a deployment context. *Will the system survive invalid inputs? Can users learn to use the system effectively without reading a manual?*

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

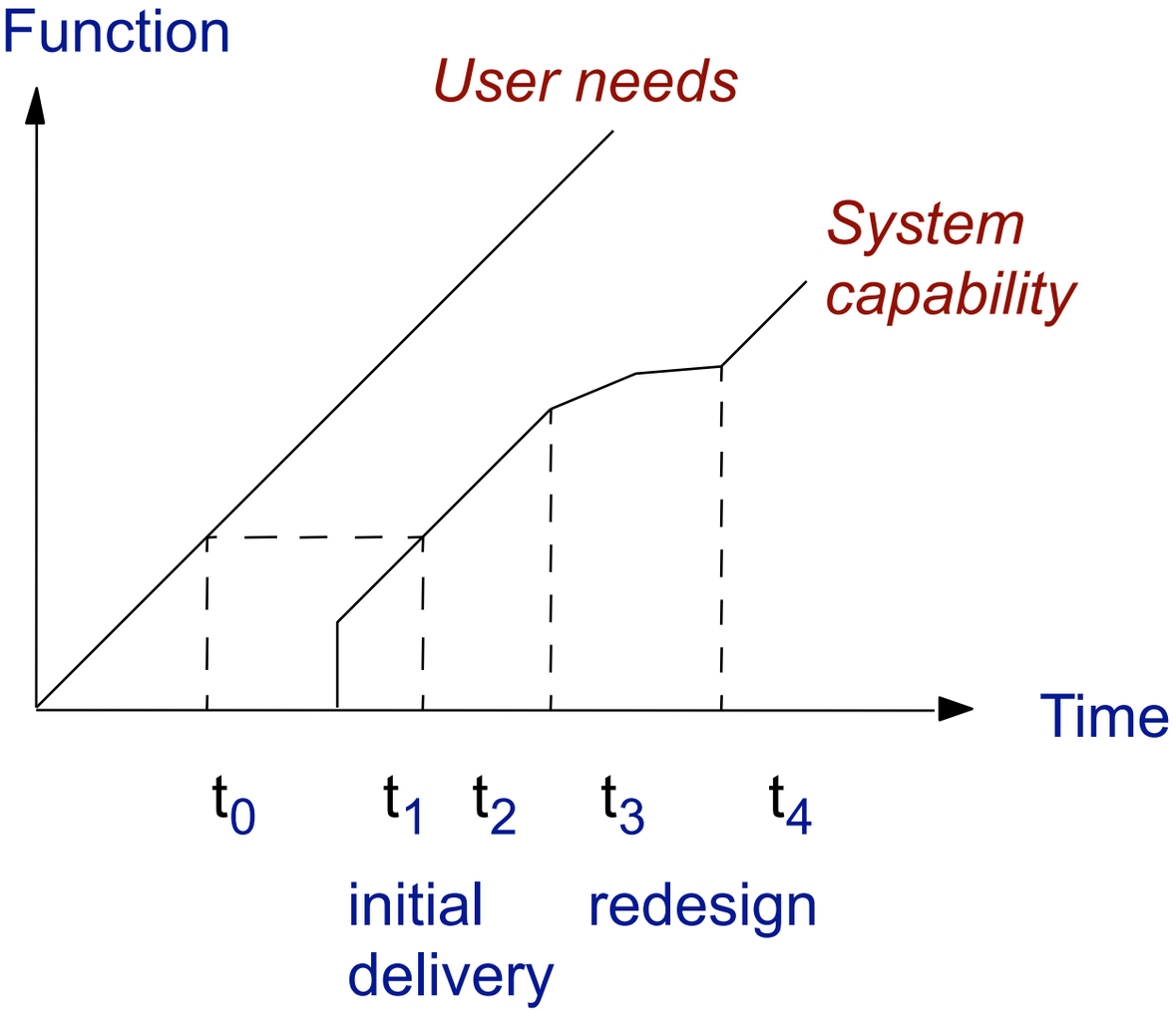
- Amount of product produced by a process for a given number of resources
  - —productivity among individuals varies a lot
  - —often:

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

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 ...

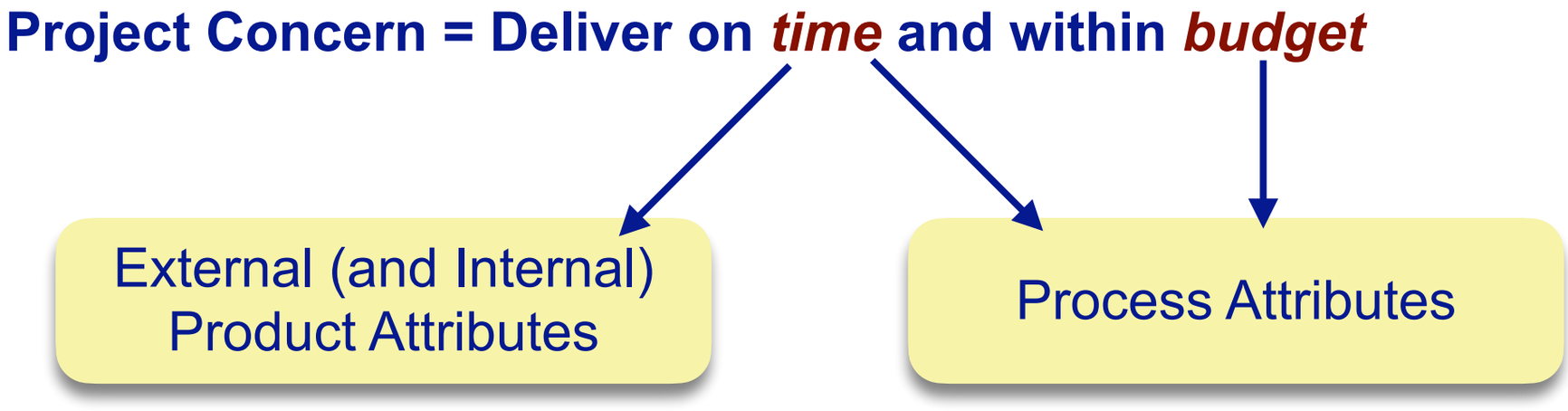
Visibility (Transparency)

- 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:

Control *during* project

Internal quality
Process quality

Obtain *after* project

External quality
Product quality

Otherwise, quality is mere coincidence!

Quality assurance in engineering is concerned with *delivering a product* of a *given quality* (to be defined) within *given time and budget constraints*.

The delivery time clearly depends on the availability of the product as well as on the production process. The budget is clearly a process attribute.

How do you ensure quality? The assumption is that *quality depends on the process*. We can monitor both the product and the process quality internally during development. This (we hope) will give us the desired product quality upon delivery and deployment.

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

The desired quality attributes always depend on the given project. These must be defined as part of the quality plan. (NB: like any aspect of a project, they may change, but they still need to be defined.)

Quality attributes are useless if they cannot be checked. The quality plan must define the process by which quality is checked and controlled.

Another important aspect concerns standards to be adopted. Standard may cover any aspect of the process, from coding and testing standards to privacy.

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 attributes and compare to standards (e.g., defect rate, cohesion, etc.)

Reviews are widely understood to be the most effective way to detect problems early in software projects. Reviews are typically carried out to assess the quality of any deliverable (i.e., a design document, a piece of code), but may also be carried out to assess the process itself.

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?

...

Take care that such checklists must be tailored to the quality goals of a specific project. The sample questions listed here are fairly general and generic, but this need not always be the case.

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?
5. Are all public interfaces tested?
6. Is test coverage at least 90%?

...

Sample coding standards

6.1 `@Override` : always used

A method is marked with the `@Override` annotation whenever it is legal. This includes a class method overriding a superclass method, a class method implementing an interface method, and an interface method respecifying a superinterface method.

Exception: `@Override` may be omitted when the parent method is `@Deprecated`.

From the *Google Java Style Guide*
https://google.github.io/styleguide/javaguide.html

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.*

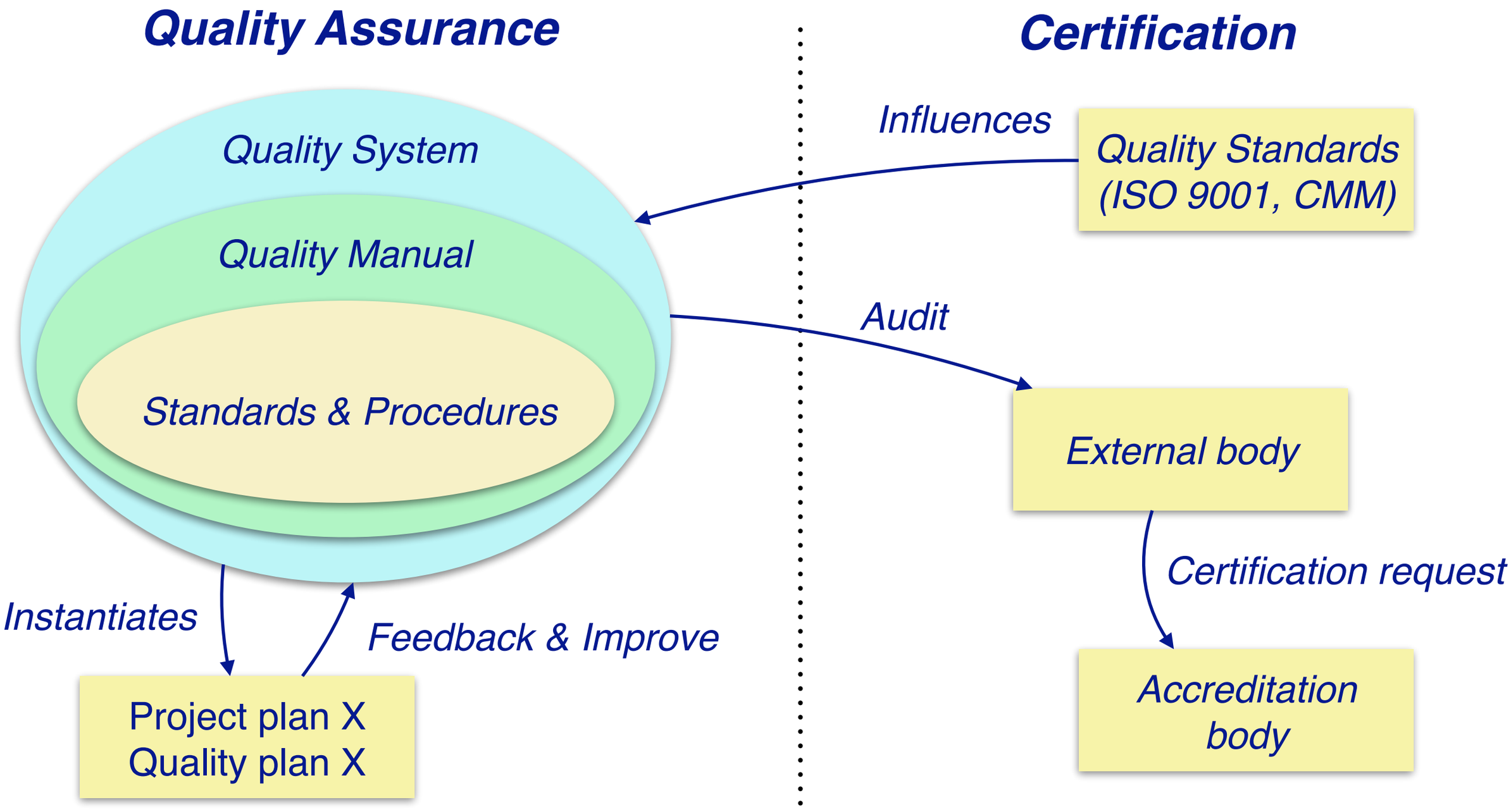
Process standards define *how the software process should be enacted*.

Product standards	*Process standards*
Design review form	Design review conduct
Document naming standards	Submission of documents
Procedure header format	Version release process
Java conventions	Project plan approval process
Project plan format	Change control process
Change request form	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*

Quality System



Customers may require an externally reviewed quality system

Quality Assurance in an organisation is formalized as a “Quality System”, which includes a “Quality Manual” as well as numerous Standards and Procedures. The quality system must be instantiated individually for a given project X. The quality system will typically be influenced by existing external standards. Customers increasingly require that the quality system of an organisation providing some service be formally audited and accredited by external bodies.

This scheme holds not only for software engineering or even just for engineering, but for many different kinds of organisations (such as universities).

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/

ISO 90003 (a few of the points)

- **The quality policy is a formal statement** from management
- The business makes decisions about the quality system based on **recorded data.**
- The quality system is **regularly audited and evaluated** for conformance and effectiveness.
- The business has **created systems for communicating with customers** about product information, inquiries, contracts, orders, feedback, and complaints.
- The business **regularly reviews performance** through internal audits and meetings. The business determines whether the quality system is working and what improvements can be made. It has a documented procedure for internal audits.
- The business **deals with past problems and potential problems**. It keeps records of these activities and the resulting decisions, and monitors their effectiveness.
- The business has **documented procedures for dealing with actual and potential non-conformances** (problems involving suppliers, customers, or internal problems).

Capability Maturity Model (CMM)

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

Level 5: *Optimizing*
Improvement is fed back into QA process

Level 4: *Managed*
QA Process + quantitative data collection

Level 3: *Defined*
QA process is defined and institutionalized

Level 2: *Repeatable*
Formal QA procedures in place

Quality depends on individual project managers!

Level 1: *Initial (ad hoc)*
No effective QA procedures, quality is “luck”

Quality depends on individuals!

The Capability Maturity Model (CMM) categorizes the "maturity" of software development contractors. The model was developed at the Software Engineering Institute (SEI) in the late 80s and early 90s by Watts Humphrey.

Any software development team gets level 1 "for free". Here quality is achieved purely through the skill and "luck" of individual developers. At level 2, individual managers help to ensure quality.

At the highest level, not only are QA processes defined, but data are gathered and used to optimize and improve the process.

Beware that there is no guarantee that a contractor at CMM level 5 will necessarily deliver better quality than another at level 1.

https://en.wikipedia.org/wiki/Capability_Maturity_Model

How to evaluate your process?

The Joel Test

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?
7. Do you have a spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

This refreshing blog post from 2000 (but still very relevant) summarizes some very down-to-earth ways to assess the maturity of your own software development processes.

http://www.joelonsoftware.com/articles/fog0000000043.html

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?