SS ZG622: Software Project Management (Lecture #9)

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

T V Rao, BITS-Pilani Off-campus Centre, Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

L9 : Sotware Quality –

Software Quality, Definitions, Costs, Standards

Source Courtesy: Some of the contents of this PPT are sourced from materials provided by publishers of prescribed books

Quality—A Philosophical View

Robert Persig [74] commented on the thing we call *quality:*

Quality . . . you know what it is, yet you don't know what it is. But that's self-contradictory. But some things are better than others, that is, they have more quality. But when you try to say what the quality is, apart from the things that have it, it all goes poof! There's nothing to talk about. But if you can't say what Quality is, how do you know what it is, or how do you know that it even exists? If no one knows what it is, then for all practical purposes it doesn't exist at all. But for all practical purposes it really does exist. What else are the grades based on? Why else would people pay fortunes for some things and throw others in the trash pile? Obviously some things are better than others . . . but what's the betterness? . . . So round and round you go, spinning mental wheels and nowhere finding anyplace to get traction. What the hell is Quality? What is it?

Quality—A Pragmatic View

The *transcendental view argues (like Persig) that quality* is something that you immediately recognize, but cannot explicitly define.

The *user view sees quality in terms of an end-user's* specific goals. If a product meets those goals, it exhibits quality.

The *manufacturer's view defines quality in terms of the* original specification of the product. If the product conforms to the spec, it exhibits quality.

The *product view suggests that quality can be tied to* inherent characteristics (e.g., functions and features) of a product.

Finally, the *value-based view measures quality based on* how much a customer is willing to pay for a product. In reality, quality encompasses all of these views and more.

The importance of software quality

- Increasing criticality of software
- The intangibility of software
- Project control concerns:
  - errors accumulate with each stage
  - errors become more expensive to remove the later they are found
  - it is difficult to control the error removal process (e.g. testing)

Software Quality

Software quality can be defined as:

*An **effective software process** applied in a manner that creates a **useful product** that provides measurable **value** for those who produce it and those who use it.*

- J Bessin, “The Business Value of Quality” (IBM DeveloperWorks)

Effective Software Process

- An *effective software process* establishes the infrastructure that supports any effort at building a high quality software product.
- The management aspects of process create the checks and balances that help avoid project chaos—a key contributor to poor quality.
- Software engineering practices allow the developer to analyze the problem and design a solid solution—both critical to building high quality software.
- Finally, umbrella activities such as change management and technical reviews have as much to do with quality as any other part of software engineering practice.

Useful Product

- A *useful product* delivers the content, functions, and features that the end-user desires
- But as important, it delivers these assets in a reliable, error free way.
- A useful product always satisfies those requirements that have been explicitly stated by stakeholders.
- In addition, it satisfies a set of implicit requirements (e.g., ease of use) that are expected of all high quality software.

Adding Value

- By *adding value for both the producer and user* of a software product, high quality software provides benefits for the software organization and the end-user community.
- The software organization gains added value because high quality software requires less maintenance effort, fewer bug fixes, and reduced customer support.
- The user community gains added value because the application provides a useful capability in a way that expedites some business process.
- The end result is:
  - (1) greater software product revenue,
  - (2) better profitability when an application supports a business process, and/or
  - (3) improved availability of information that is crucial for the business.

Quality Dimensions as per David Garvin[87]

Performance Quality. Does the software deliver all content, functions, and features that are specified as part of the requirements model in a way that provides value to the end-user?

Feature quality. Does the software provide features that surprise and delight first-time end-users?

Reliability. Does the software deliver all features and capability without failure? Is it available when it is needed? Does it deliver functionality that is error free?

Conformance. Does the software conform to local and external software standards that are relevant to the application? Does it conform to de facto design and coding conventions? For example, does the user interface conform to accepted design rules for menu selection or data input?

Quality Dimensions (contd...)

as per David Garvin[87]

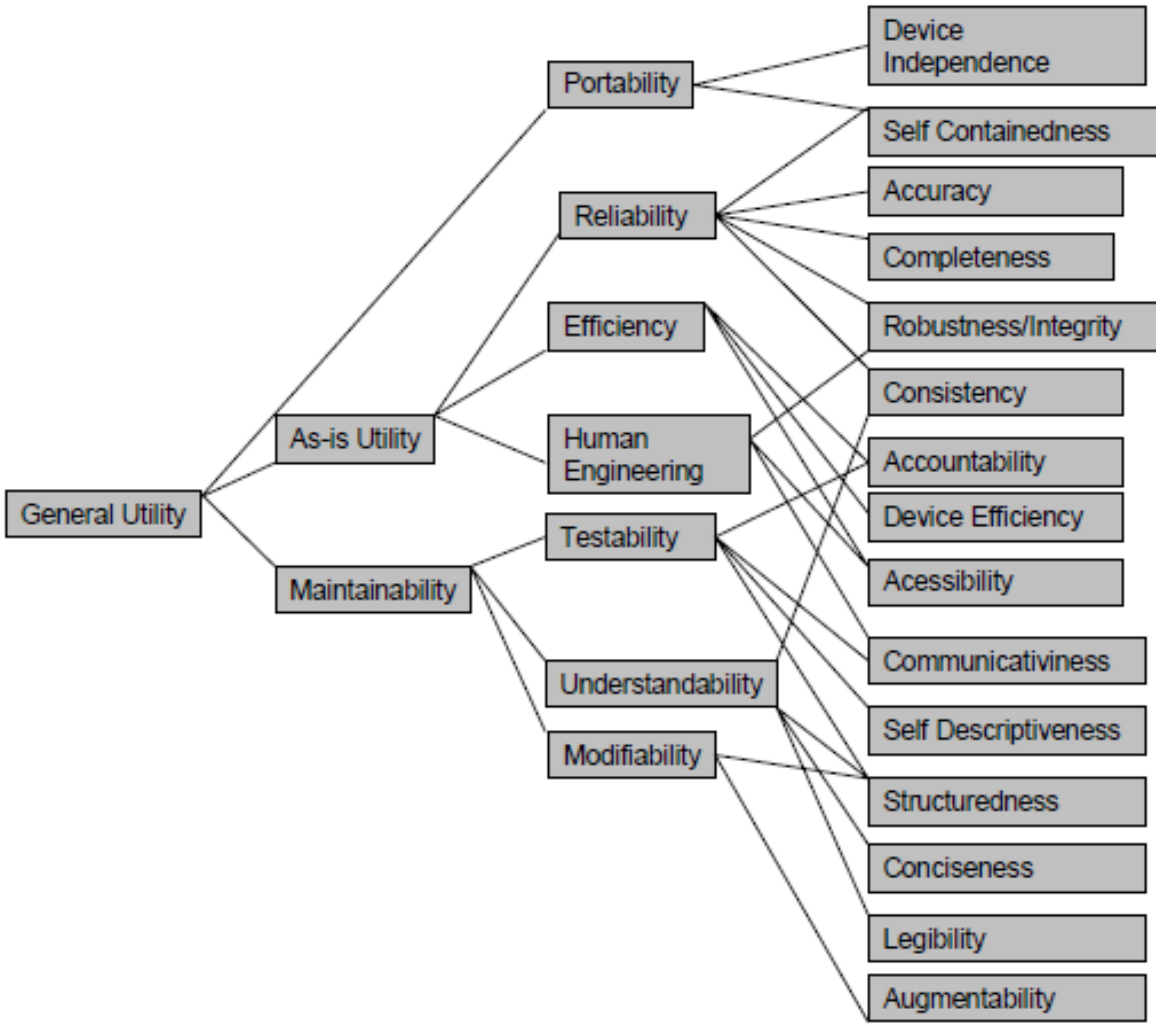
Durability. Can the software be maintained (changed) or corrected (debugged) without the inadvertent generation of unintended side effects? Will changes cause the error rate or reliability to degrade with time?

Serviceability. Can the software be maintained (changed) or corrected (debugged) in an acceptably short time period. Can support staff acquire all information they need to make changes or correct defects?

Aesthetics. Most of us would agree that an aesthetic entity has a certain elegance, a unique flow, and an obvious "presence" that are hard to quantify but evident nonetheless.

Perception. In some situations, you have a set of prejudices that will influence your perception of quality.

Barry Boehm's Quality Tree



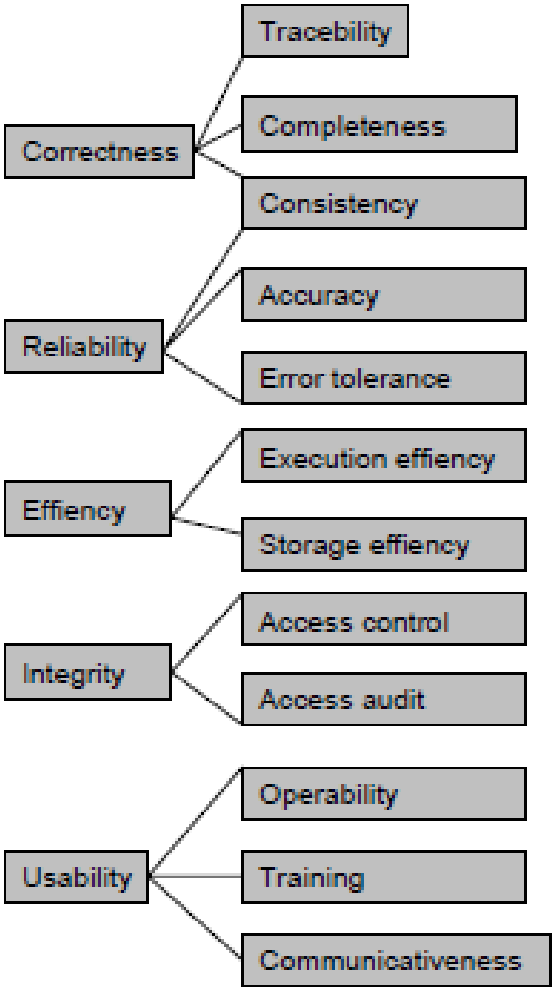
As-is utility : How well (reliably, efficiently, easily) can I use it as-is?

Maintainability : How easy is it to understand, modify and retest?

Portability : Can I still use it if I change my environment?

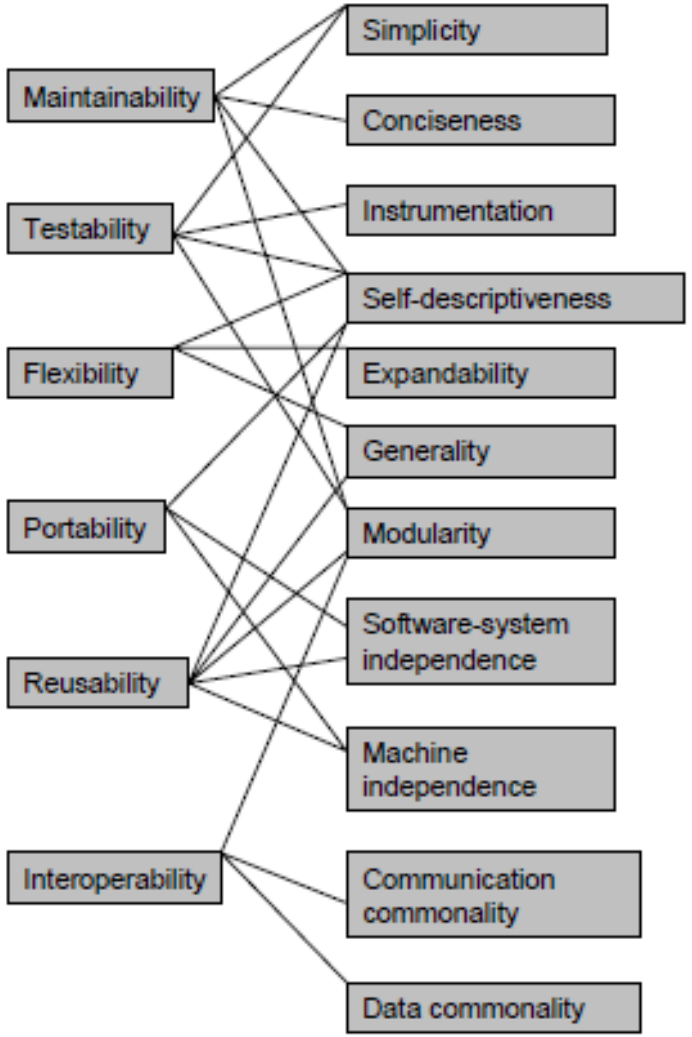
McCall's Triangle of Quality

McCall's Quality Model



11 Factors (To specify) : external view of the software

23 quality criteria (To build): internal view of the software



Quality specifications

Where there is a specific need for a quality, produce a quality specification

- Definition/description of the quality
- Scale: the unit of measurement
- Test: practical test of extent of quality
- Minimally acceptable: lowest acceptable value, if compensated for by higher quality level elsewhere
- Target range: desirable value
- Now: value that currently applies

ISO standards

ISO 9126 Software product quality

Attributes of software product quality

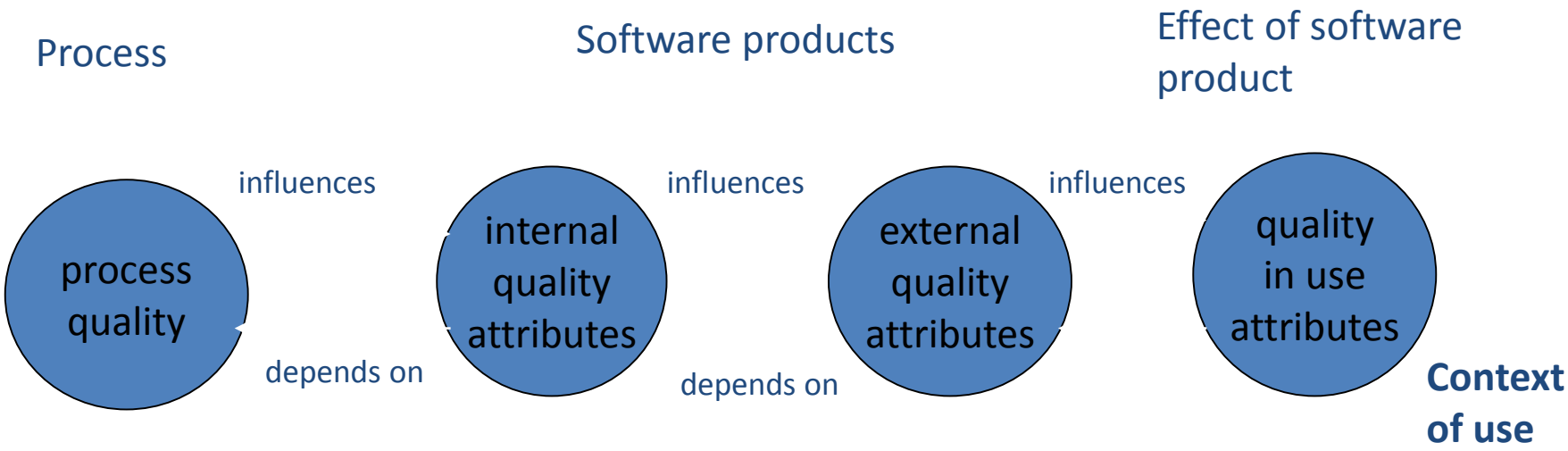
- External qualities i.e. apparent to the user of the deliverable
- Internal qualities i.e. apparent to the developers of the deliverables and the intermediate products

ISO 14598 Procedures to carry out the assessment of the product qualities defined in ISO 9126

Types of quality assessment

- During software development, to assist developers to build software with the required qualities
- During software acquisition to allow a customer to compare and select the best quality product
- Independent evaluation by assessors rating a software product for a particular community of users

ISO 9126 Software Product Quality



Quality in use

- Effectiveness – ability to achieve user goals with accuracy and completeness
- Productivity – avoids excessive use of resources in achieving user goals
- Safety – within reasonable levels of risk of harm to people, business, software, property, environment etc,
- Satisfaction – happy users!

'users' include those maintain software as well as those who operate it.

ISO 9126 software qualities

Functionality	**does it satisfy user needs?**
Reliability	can the software maintain its level of performance?
Usability	how easy is it to use?
Efficiency	relates to the physical resources used during execution
Maintainability	relates to the effort needed to make changes to the software
Portability	how easy can it be moved to a new environment?

Sub-characteristics of Functionality

- Suitability
- Accuracy
- Interoperability
  - ability of software to interact with other software components
- Functionality compliance
  - degree to which software adheres to application-related standards or legal requirements e.g audit
- Security
  - control of access to the system

Sub-characteristics of Reliability

- Maturity
  - frequency of failure due to faults - the more the software has been used, the more faults will have been removed
- Fault-tolerance
- Recoverability
  - note that this is distinguished from ‘security’ - see above
- Reliability compliance
  - complies with standards relating to reliability

Sub-characteristics of Usability

- Understandability
  - easy to understand?
- Learnability
  - easy to learn?
- Operability
  - easy to use?
- Attractiveness – this is a recent addition
- Usability compliance
  - compliance with relevant standards

Sub-characteristics of Efficiency

- Time behaviour
  - e.g. response time
- Resource utilization
  - e.g. memory usage
- Efficiency compliance
  - compliance with relevant standards

Sub-characteristics of Maintainability

- "Analysability"
  - ease with which the cause of a failure can be found
- Changeability
  - how easy is software to change?
- Stability
  - low risk of modification having unexpected effects
- "Testability"
- Maintainability conformance

Sub-characteristics of portability

- Adaptability
- “Installability”
- Co-existence
  - Capability of co-existing with other independent software products
- “Replaceability”
  - factors giving ‘upwards’ compatibility - ‘downwards’ compatibility is excluded
- Portability conformance
  - Adherence to standards that support portability

Using ISO 9126 quality standards (development mode)

- Judge the importance of each quality for the application
  - for example, safety critical systems - *reliability* very important
  - real-time systems - *efficiency* important
- Select relevant external measurements within ISO 9126 framework for these qualities, for example
  - mean-time between failures for reliability
  - response-time for efficiency

Using ISO 9126 quality standards

- map measurement onto ratings scale to show degree of user satisfaction – for example response time

response (secs)	rating
<2	Exceeds requirement
2-5	Target range
6-10	Minimally acceptable
>10	Unacceptable

Using ISO 9126 quality standards

- Identify the relevant internal measurements and the intermediate products in which they would appear

e.g. at software design stage the estimated execution time for a transaction could be calculated

Using ISO 9126 for application software selection

- Rather than map engineering measurement to qualitative rating, map it to a score
- Rate the importance of each quality in the range 1-5
- Multiply quality and importance scores – see next slide

Response (secs)	Quality score
<2	5
2-3	4
4-5	3
6-7	2
8-9	1
>9	0

Weighted quality scores

		Product A		Product B	
Product quality	Importance rating (a)	Quality score (b)	Weighted score (a x b)	Quality score (c)	Weighted score (a x c)
usability	3	1	3	3	9
efficiency	4	2	8	2	8
maintain-ability	2	3	6	1	2
Overall totals			17		19

How do we achieve product quality?

The problem: quality attributes tend to *retrospectively* measurable

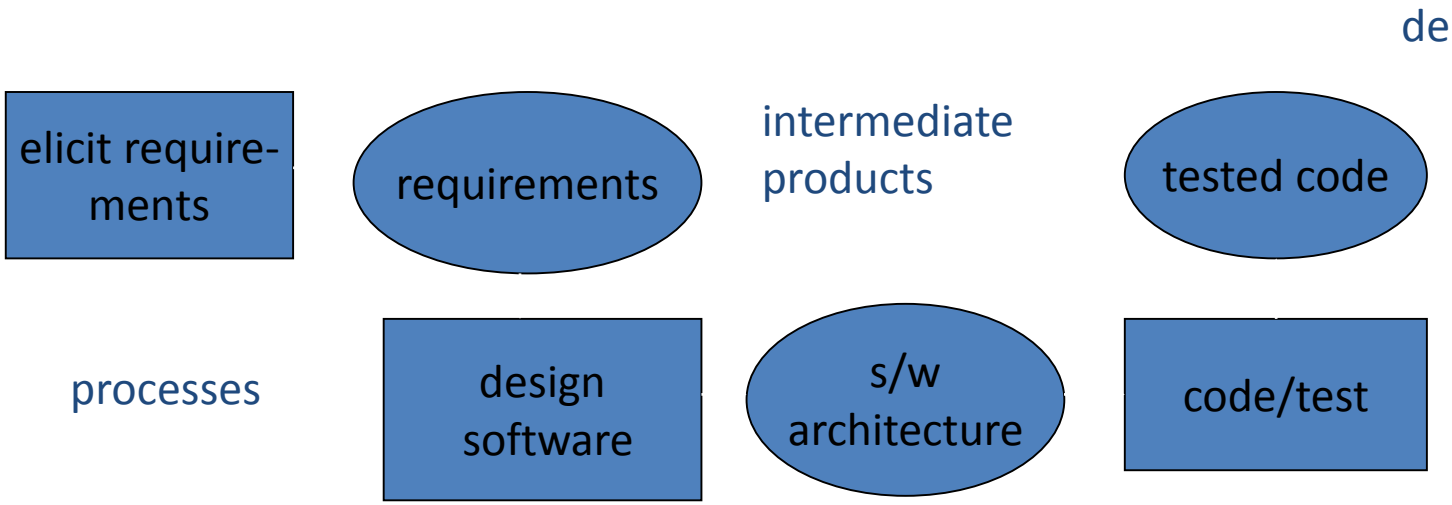
- need to be able to examine processes by which product is created beforehand
- the production process is a network of sub-processes
- output from one process forms the input to the next
- errors can enter the process at any stage

Correction of errors

- Errors are more expensive to correct at later stages
  - need to rework more stages
  - later stages are more detailed and less able to absorb change
- Barry Boehm
  - Error typically 10 times more expensive to correct at coding stage than at requirements stage
  - 100 times more expensive at maintenance stage

ISO Standards: Development Life Cycles

A development life cycle (like ISO 12207) indicates the sequence of *processes* that will produce the software *deliverable* and the *intermediate products* that will pass between the processes.



For each activity, *define*:

- Entry requirements
  - these have to be in place before an activity can be started
  - example: ‘a comprehensive set of test data and expected results be prepared and independently reviewed against the system requirement before program testing can commence’

For each activity, *define*:

- Implementation requirements
  - these define how the process is to be conducted
  - example ‘whenever an error is found and corrected, *all* test runs must be completed, including those previously successfully passed’

For each activity, *define*:

- Exit requirements
  - an activity will not be completed until these requirements have been met
  - example: ‘the testing phase is finished only when all tests have been run in succession with no outstanding errors’

ISO 25010 Product Quality Characteristics

ISO/IEC 25010, which supersedes ISO/IEC 9126-1, was issued in March 2011. There are changes to product quality definitions

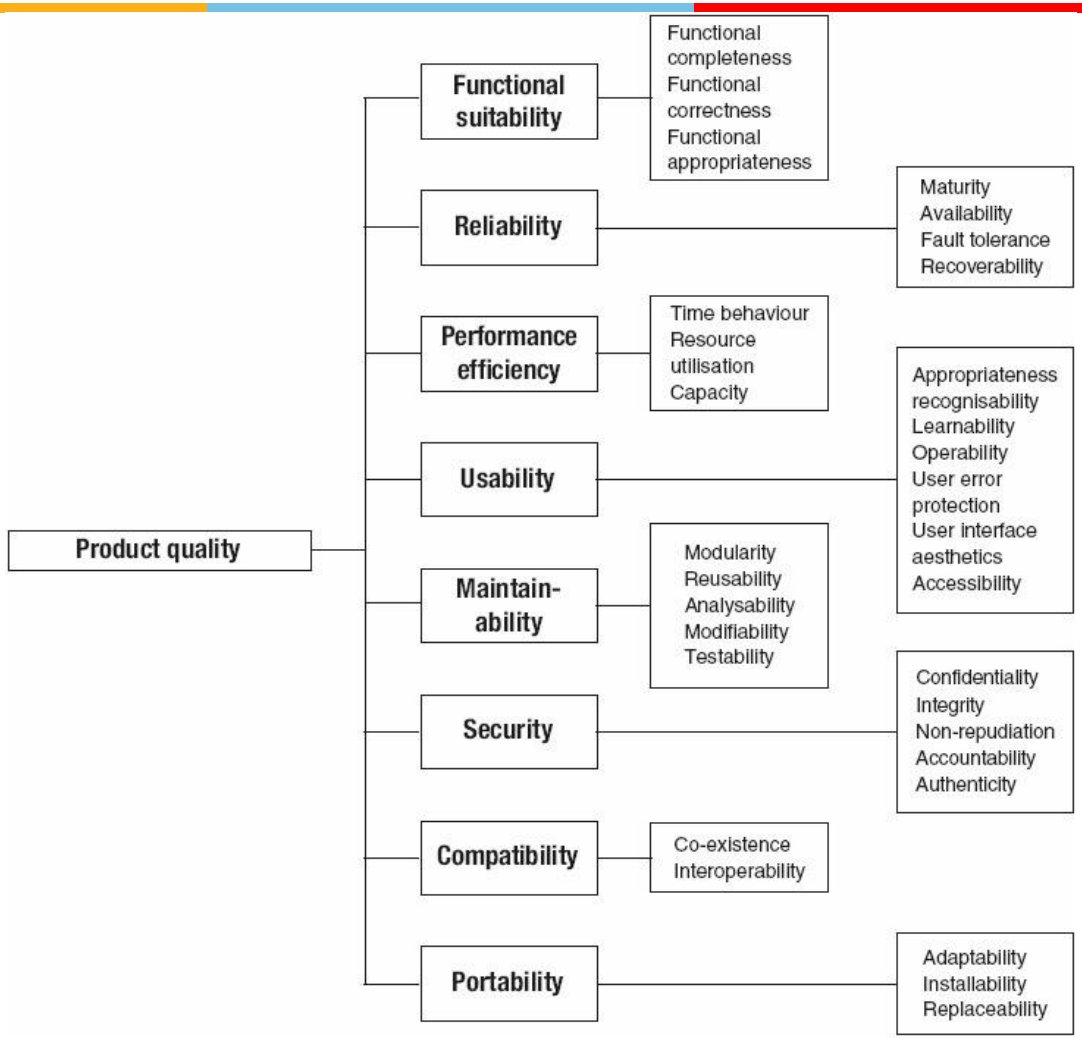
- Functionality is renamed Functional suitability.
- Efficiency is renamed Performance efficiency.
- Compatibility is a new characteristic.
- Usability.
- Reliability.
- Security is a new characteristic with subcharacteristics of Confidentiality (data accessible only by those authorized), Integrity (protection from unauthorized modification), Non-repudiation (actions can be proven to have taken place), Accountability (actions can be traced to who did them), and Authenticity (identity can be proved to be the one claimed).
- Maintainability .
- Portability.

Product quality model of ISO/IEC 25010

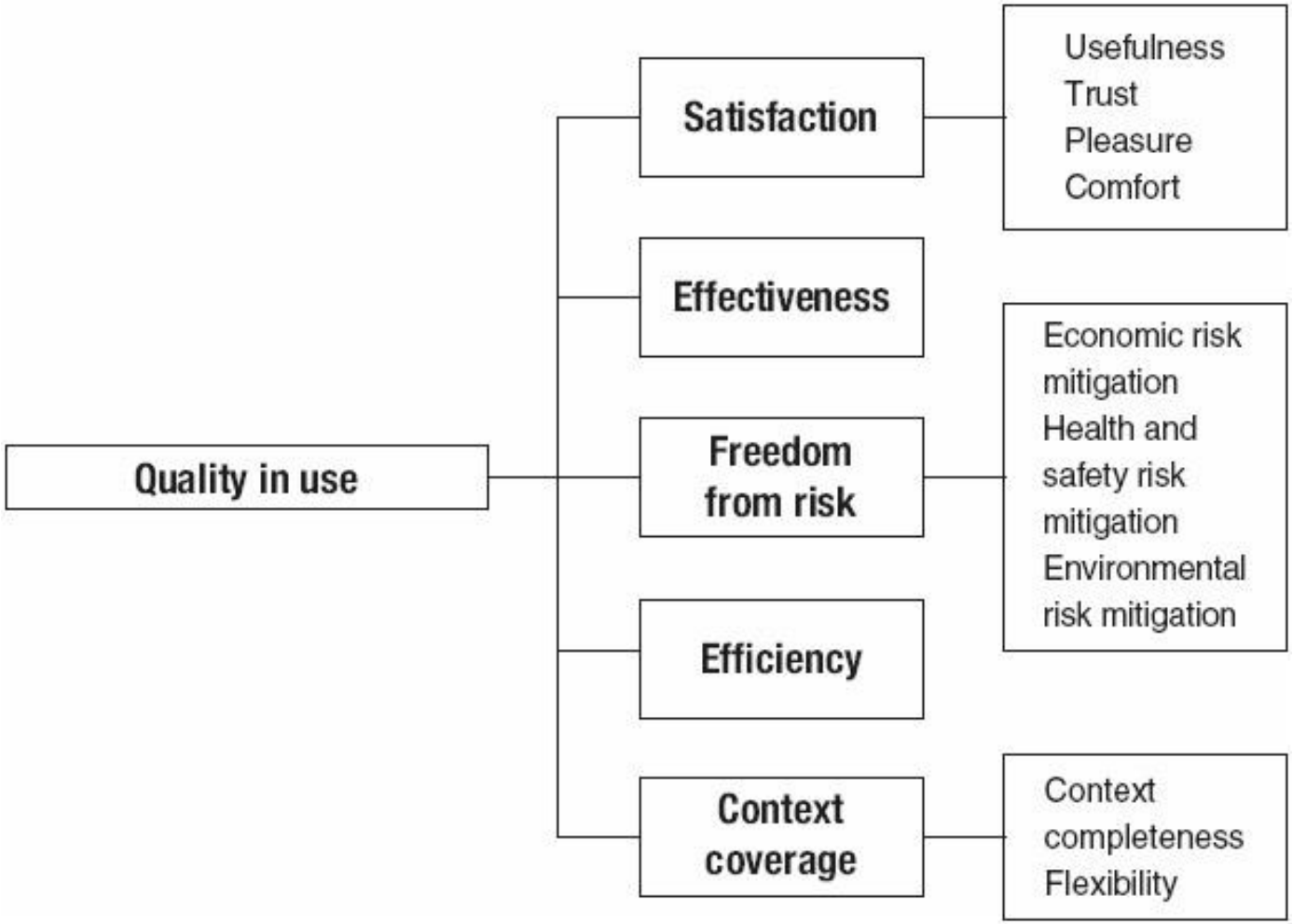
ISO/IEC 25010 grew from quality models of Boehm and McCall which were already the basis of the old ISO/IEC 9126

- *Functional suitability* means that the product fits to the functional needs and requirements of the user and customer.
- *Performance efficiency* describes how well the product responds to user requests and how efficient it is in its execution.
- *Compatibility* defines the quality that a product does not disturb or can even work together with other products.
- *Usability* indicates how easy the system to use. This includes how fast it can be learned, if interface is attractive, and if challenged persons are able to use it
- *Reliability* is about product failures and availability.
- *Security* includes ability to keep data intact and confidential and that the product can make sure that its users are who they claim to be.
- *Maintainability* describes that the system should be designed and programmed in a way that it is easy to comprehend, change and test.
- *Portability* helps to bring the product to another platform (programming language, operating system, hardware). It means that the necessary changes can be easily done and it can be easily installed

Product quality model of ISO/IEC 25010



Quality in use model of ISO/IEC 25010



Cost of Quality

Prevention costs:

Quality planning
Formal technical reviews
Test equipment
Training

Internal failure costs:

Rework
Repair
failure mode analysis

Appraisal Costs:

Technical reviews
Data Collection
Testing & Debugging

External failure costs:

complaint resolution
product return and replacement
help line support
warranty work

Quality and Risk

"People bet their jobs, their comforts, their safety, their entertainment, their decisions, and their very lives on computer software. It better be right."

Example:

Throughout the month of November, 2000 at a hospital in Panama, 28 patients received massive overdoses of gamma rays during treatment for a variety of cancers. In the months that followed, five of these patients died from radiation poisoning and 15 others developed serious complications. What caused this tragedy? A software package, developed by a U.S. company, was modified by hospital technicians to compute modified doses of radiation for each patient.

Negligence and Liability

The story is all too common. A governmental or corporate entity hires a major software developer or consulting company to analyze requirements and then design and construct a software-based "system" to support some major activity.

The system might support a major corporate function (e.g., pension management) or some governmental function (e.g., healthcare administration or homeland security).

Work begins with the best of intentions on both sides, but by the time the system is delivered, things have gone bad.

The system is late, fails to deliver desired features and functions, is error-prone, and does not meet with customer approval.

Litigation ensues.

Quality and Security

"Software security relates entirely and completely to quality. You must think about security, reliability, availability, dependability—at the beginning, in the design, architecture, test, and coding phases, all through the software life cycle [process]. Even people aware of the software security problem have focused on late lifecycle stuff. The earlier you find the software problem, the better. And there are two kinds of software problems. One is bugs, which are implementation problems. The other is software flaws—architectural problems in the design. People pay too much attention to bugs and not enough on flaws."

Gary McGraw (ComputerWorld)

Confusion About Quality

Many words (ending with *ility*) are cited for Quality:

Augmentability	Compatibility
Expandability	Flexibility
Interoperability	Maintainability
Manageability	Modifiability
Operability	Portability
Reliability	Scalability
Survivability	Understandability
Usability	Testability
Traceability	Verifiability

Capers Jones., Software Engineering Best Practices

Method in Chaos – Analysis of Vista in 2009

When Windows Vista is judged against the list, it can be seen how abstract and difficult to apply the list to commercial software

Quality Trait	Observation for Vista	Quality Trait	Observation for Vista
Augmentability	Difficult to apply	Compatibility	Poor
Expandability	Fairly good	Flexibility	Difficult to apply
Interoperability	Difficult to apply	Maintainability	Probably Poor
Manageability	Difficult to apply	Modifiability	Probably Poor
Operability	Difficult to apply	Portability	Poor
Reliability	Improving	Scalability	Marginal
Survivability	Difficult to apply	Understandability	Poor
Usability	Questionable	Testability	Poor, Complex
Traceability	Poor, Complex	Verifiability	Difficult to apply

Capers Jones., Software Engineering Best Practices

Achieving Software Quality

Critical success factors:

Software Engineering Methods

Project Management Techniques

Quality Control

Quality Assurance

Achieving Software Quality

Software Engineering Methods

- Understand the problem to be solved
- Create design that conforms to the problem
- Design and Software exhibit quality dimensions

Project Management Techniques

- Use estimation for achievable delivery dates
- Understand schedule dependencies and avoid short cuts
- Conduct risk planning

Achieving Software Quality

Quality Control

- Reviews
- Inspection
- Testing
- Measurements and Feedback

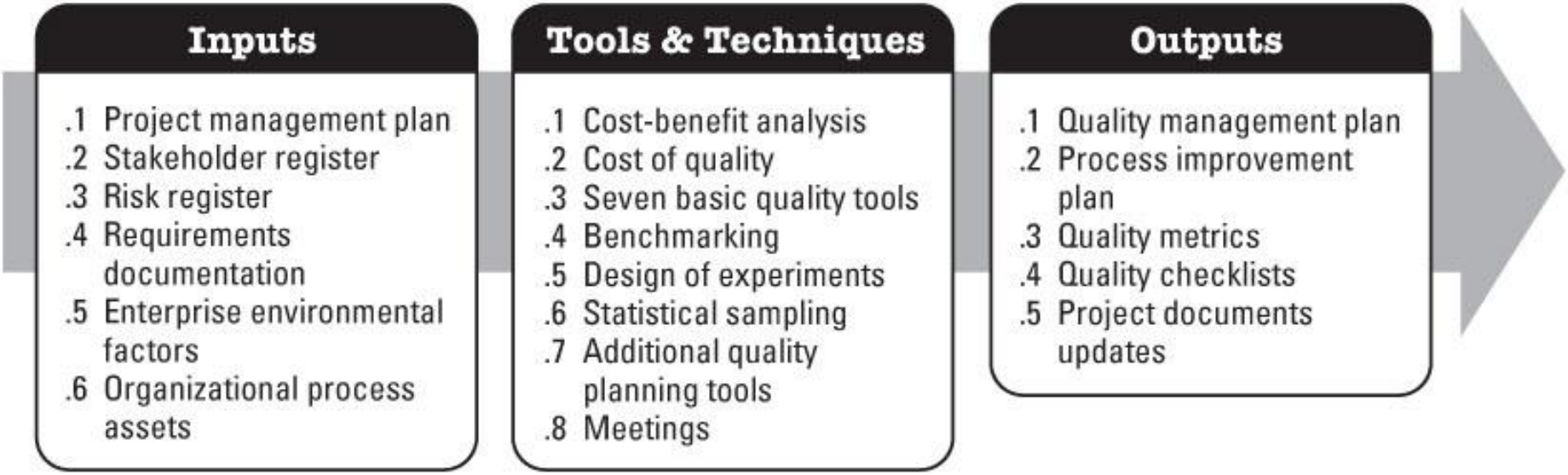
Quality Assurance

- Establish infrastructure for software engineering
- Audit effectiveness and completeness of quality control

Project Quality Management (PMBOK)

- **Plan Quality Management**—The process of identifying quality requirements and/or standards for the project and its deliverables and documenting how the project will demonstrate compliance with quality requirements.
- **Perform Quality Assurance**—The process of auditing the quality requirements and the results from quality control measurements to ensure that appropriate quality standards and operational definitions are used.
- **Control Quality**—The process of monitoring and recording results of executing the quality activities to assess performance and recommend necessary changes.

Project Quality Management (PMBOK)



Plan Quality Management

Project Quality Management (PMBOK)

Tools & Techniques

- **Cost-Benefit Analysis**
  - The primary benefits of meeting quality requirements include less rework, higher productivity, lower costs, increased stakeholder satisfaction, and increased profitability. A cost-benefit analysis for each quality activity compares the cost of the quality step to the expected benefit
- **Cost of Quality**
  - Cost of quality includes all costs incurred over the life of the product by investment in preventing nonconformance to requirements, appraising the product or service for conformance to requirements, and failing to meet requirements (rework). Failure costs are often categorized into internal (found by the project) and external (found by the customer).
- **Benchmarking**
  - Benchmarking involves comparing actual or planned project practices to those of comparable projects to identify best practices, generate ideas for improvement, and provide a basis for measuring performance
- **Design of Experiments**
  - Design of experiments (DOE) is a statistical method for identifying which factors may influence specific variables of a product or process under development

Project Quality Management (PMBOK)

Tools & Techniques

- **Seven Basic Quality Tools**
  - The seven basic quality tools, also known in the industry as 7QC Tools, are used to solve quality-related problems
  - *Cause-and-effect diagrams*, (aka fishbone diagrams or Ishikawa diagrams) The problem statement placed at the head of the fishbone is used as a starting point to trace the problem's source back to its actionable root cause
  - *Flowcharts*, display the sequence of steps and the branching possibilities that exist for a process that transforms one or more inputs into one or more outputs. Flowcharts prove useful in understanding and estimating the cost of quality in a process
  - *Checksheets*, (aka tally sheets) used as a checklist when gathering data
  - *Control charts*, are used to determine whether or not a process is stable or has predictable performance; control charts may also be used to monitor cost and schedule variances, volume, and frequency of scope changes etc.
  - *Pareto diagrams*, identify the vital few sources causing most effects
  - *Histograms*, *Scatter diagrams*

Project Quality Management (PMBOK)

Perform Quality Assurance

Project Quality Management (PMBOK)

Control Quality

Summary – Software Quality

- Software quality is the concern of every software process stakeholder.
- If a software team stresses quality in all software engineering activities, it reduces the amount of rework that must be done. This results in lower costs and improved time-to-market.
- To achieve high quality software, four elements must be present:
  - proven software engineering process and practice,
  - solid project management,
  - comprehensive quality control, and
  - the presence of a quality assurance infrastructure.
- Quality software meets its customer's needs, performs accurately and reliably, and provides value to all who use it.

Thank You