SE345

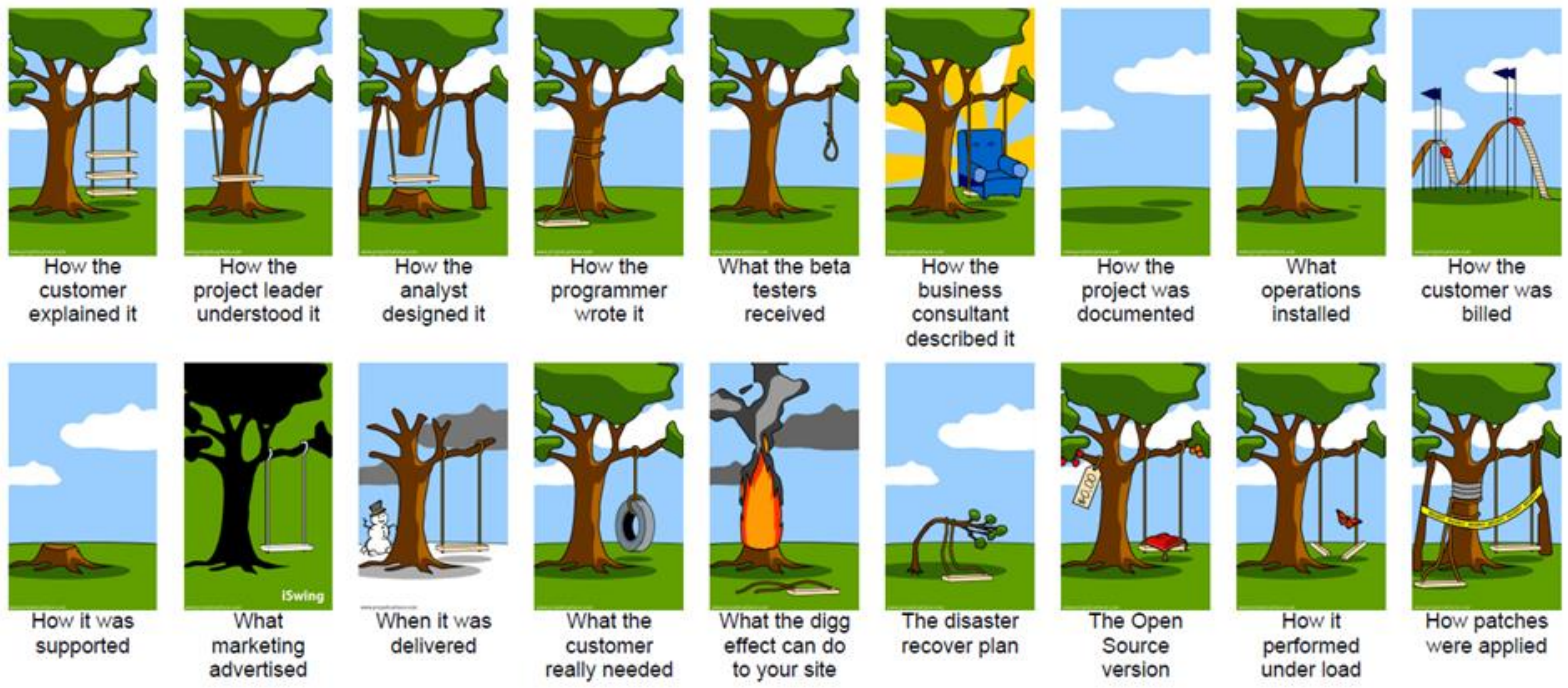
Atılım University
Dept of Software Engineering

Asst. Prof. Dr. Aylin AKCA-OKAN

Tentative Course Schedule

Wk	Subjects	Chapter
1	Introduction to Software Quality and Assurance	Chapter 1
2	Introduction to Software Quality and Assurance	Chapter 1
3	Software Quality Factors	Chapter 3
4	Overview of Components of the SQA System	Chapter 4
5	Overview of Components of the SQA System	Chapter 4
6	Presentation	Chapter 9
7	Midterm	
8	Integrating Quality Activities in Project Life Cycle	Chapter 7
9	Software Quality Metrics	Chapter 21
10	Reviews, Inspection and Audits	Chapter 8
11	Procedures and Work Instructions	Chapter 14
12	In-class Project	
13	Software Change Process	Chapter 18
14	SOA Process Standards	Chapter 23

Expectations vs Implementation
"if you can't say it, you can't do it"


How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How the business consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What marketing advertised
iSwing
When it was delivered
What the customer really needed
What the digg effect can do to your site
The disaster recover plan
The Open Source version
$0.00
How it performed under load
How patches were applied

Software Quality Assurance and Software Engineering

The characteristics of software engineering, especially the systematic, disciplined and quantitative approach at its core, make **the software engineering environment a good infrastructure for achieving SQA objectives**.

The **methodologies and tools** that are applied by software engineering determine, to a considerable extent, the **level of quality** to be expected from the software process and the maintenance services.

Software Quality Assurance and Software Testing

SQA

- Defined as a planned and systematic approach to the evaluation of the quality and adherence to software product standards, processes, and procedures.

- An umbrella activity that is applied throughout the software process.
- Consists of a means of monitoring the software engineering processes and methods used to ensure quality.
- An effective approach to produce high quality software.

- *Process-oriented activity*
- *Oriented to bug prevention*

Software Testing

- is the process of executing a system or component under specified conditions with the intent of finding defects/bugs and to verify that it satisfies specified requirements.

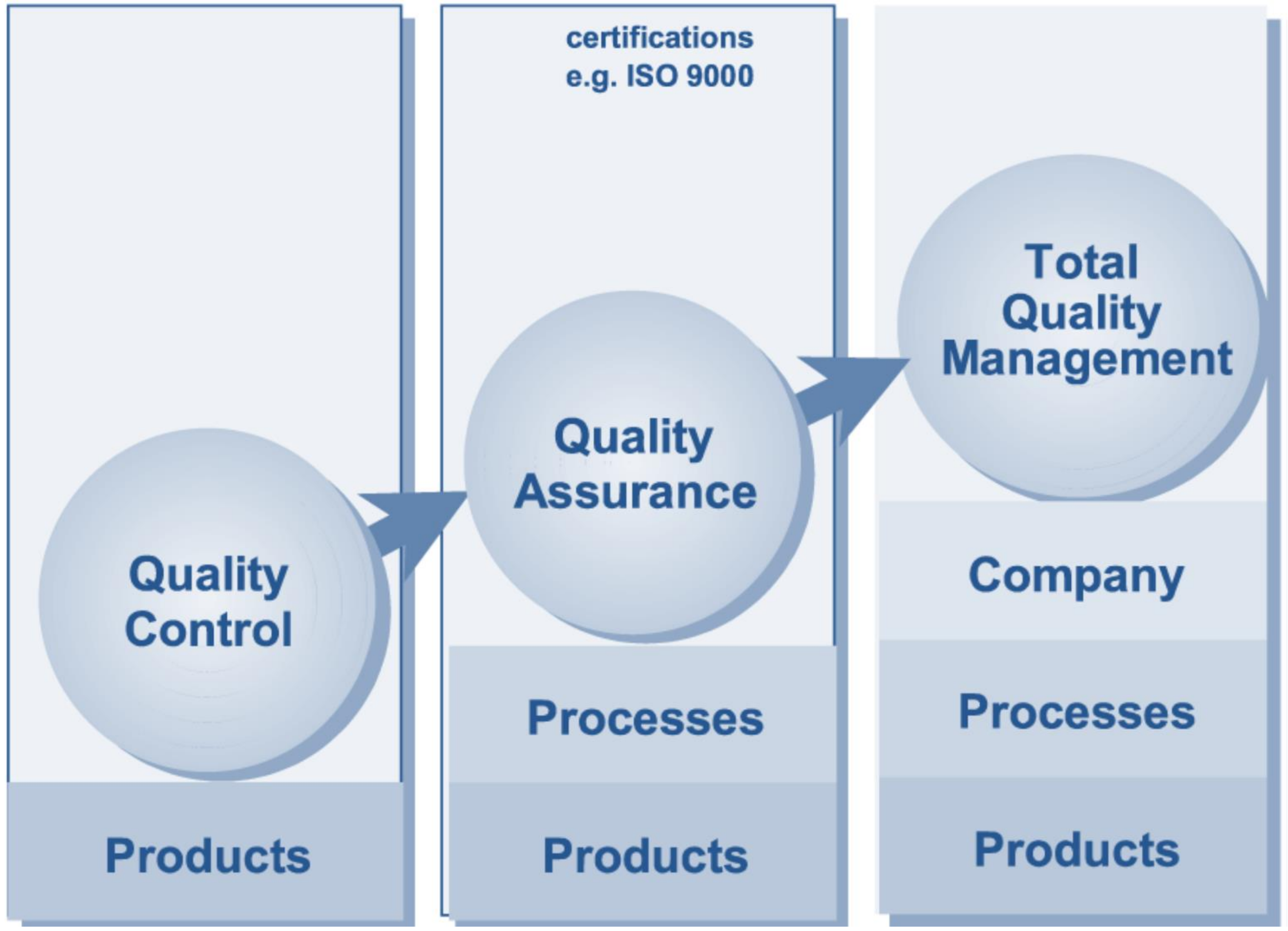
- Main goal → To detect bugs
- Have different levels
- Static testing vs. Dynamic testing
- Manual testing vs. Automated testing

- *Product-oriented activity*
- *Oriented to bug detection*

Quality Concepts

- Variation control is the heart of quality control
- Quality of design
  - refers to characteristics designers specify for the end product to be constructed
- Quality of conformance
  - the degree to which design specifications are followed in manufacturing the product
- **Quality control**
  - series of inspections, reviews, and tests used to ensure conformance of a work product (artefact) to its specifications
- **Quality assurance**
  - auditing and reporting procedures used to provide management with data needed to make proactive decisions

Software Quality Assurance and ...



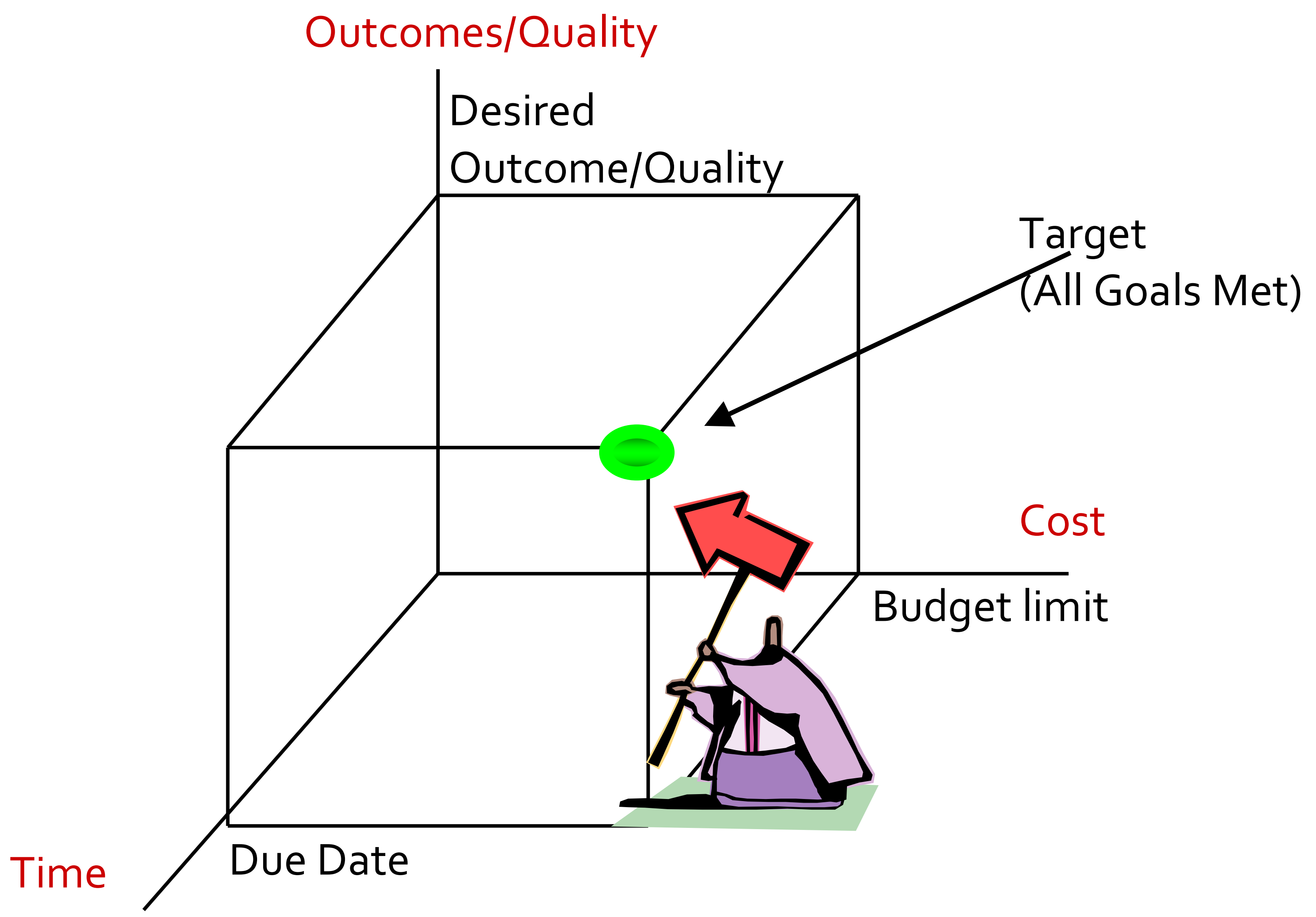
What are the Main Challenges of Software Development Nowadays?

High Cost

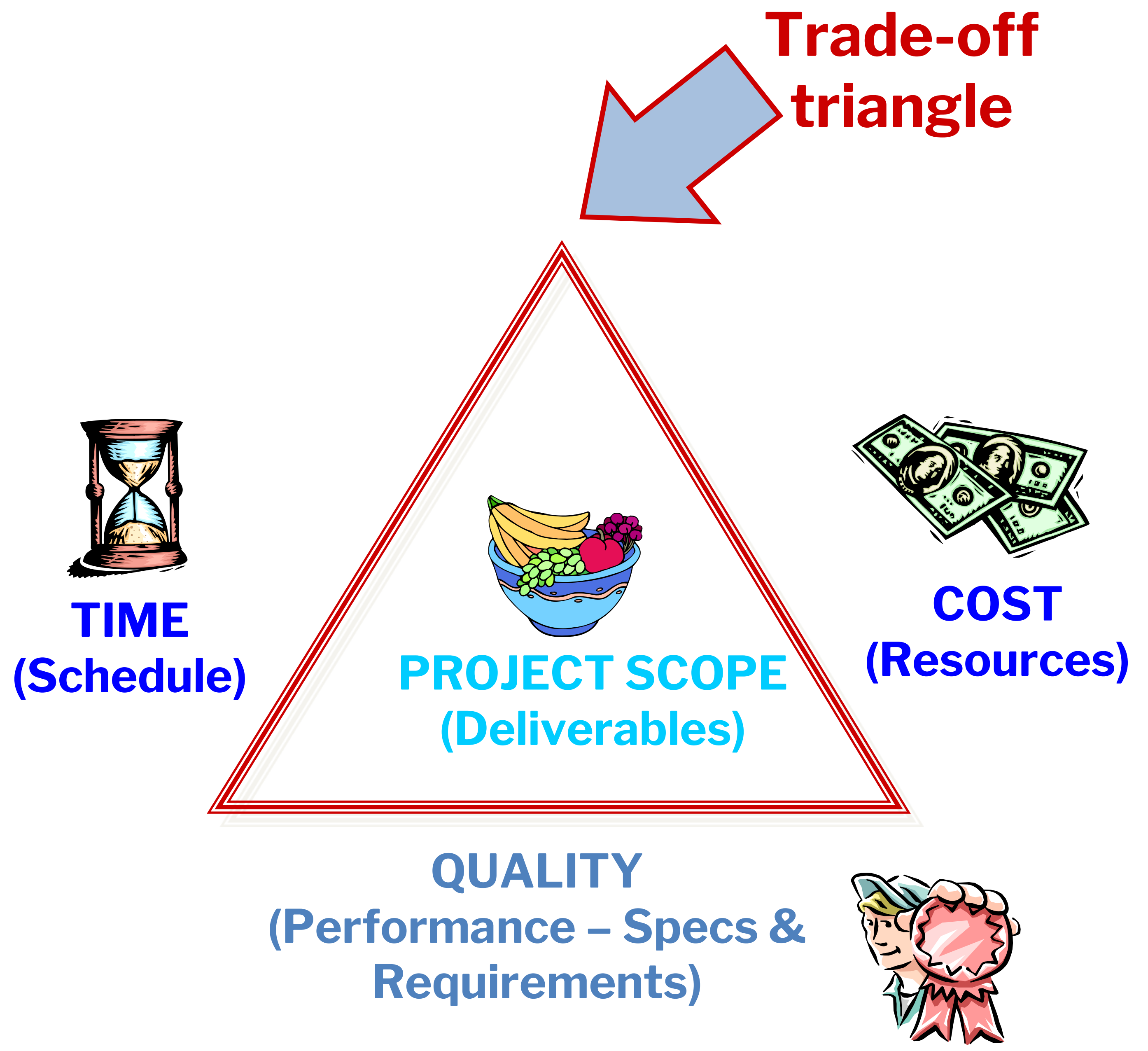
Difficult to deliver on Time

Low Quality

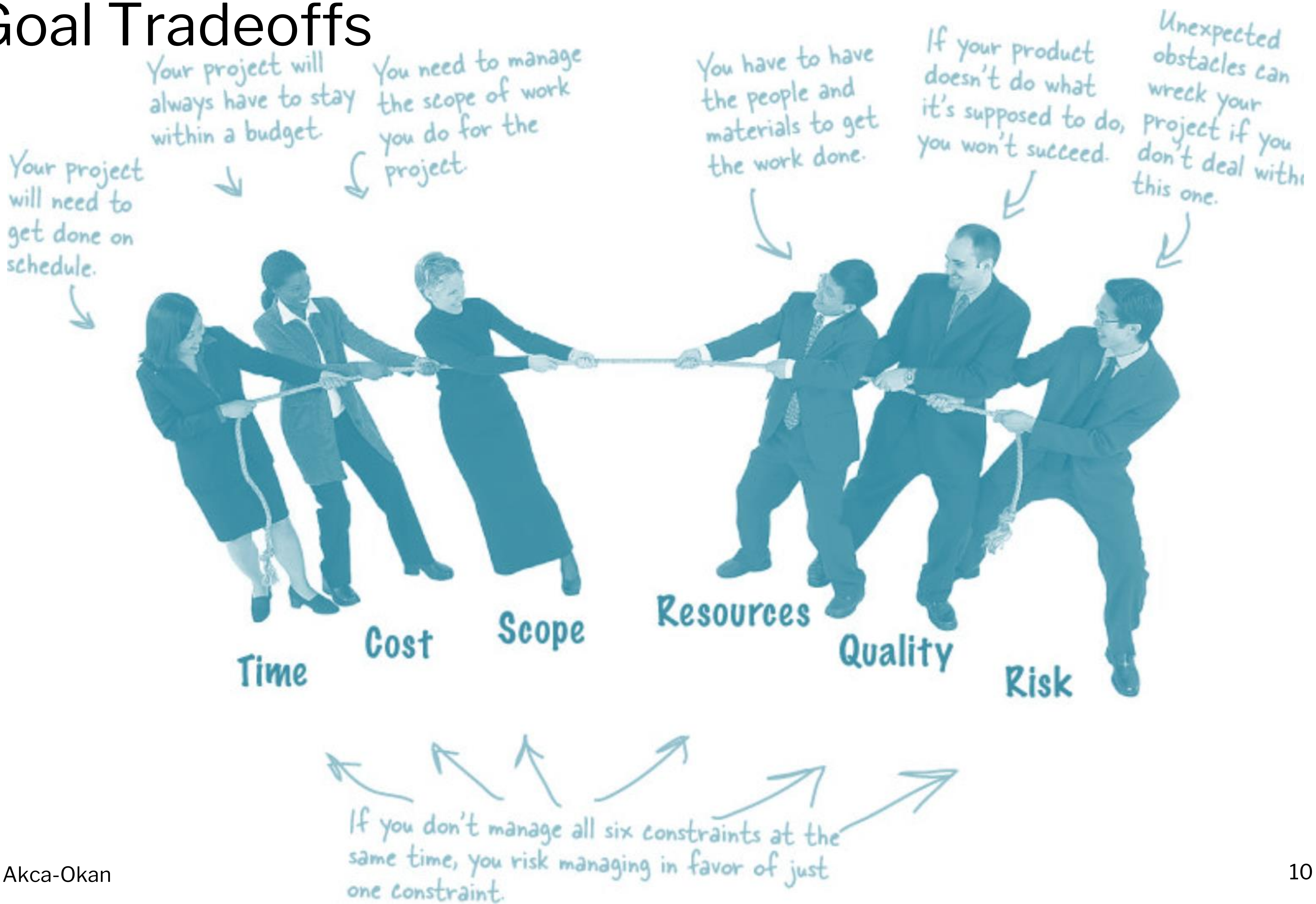
Project Goal Tradeoffs



have to know which of these are fixed & variable for every project


Trade-off triangle
TIME (Schedule)
PROJECT SCOPE (Deliverables)
COST (Resources)
QUALITY (Performance – Specs & Requirements)

Project Goal Tradeoffs


Your project will need to get done on schedule.
Your project will always have to stay within a budget.
You need to manage the scope of work you do for the project.
You have to have the people and materials to get the work done.
If your product doesn't do what it's supposed to do, you won't succeed.
Unexpected obstacles can wreck your project if you don't deal with this one.
Time
Cost
Scope
Resources
Quality
Risk
If you don't manage all six constraints at the same time, you risk managing in favor of just one constraint.

Five Views of Software Quality

Transcendental view:

- Quality is viewed to be something ideal. In this case, no effort is made to express it using concrete measures.

User view:

- In this view, the user is concerned with whether or not the product is fit for use. The product should meet user needs and expectations.

Manufacturing view:

- In this view, quality is seen as conforming to requirements.

Product view:

- *If a product is manufactured with good internal characteristics then it will have good external qualities.*

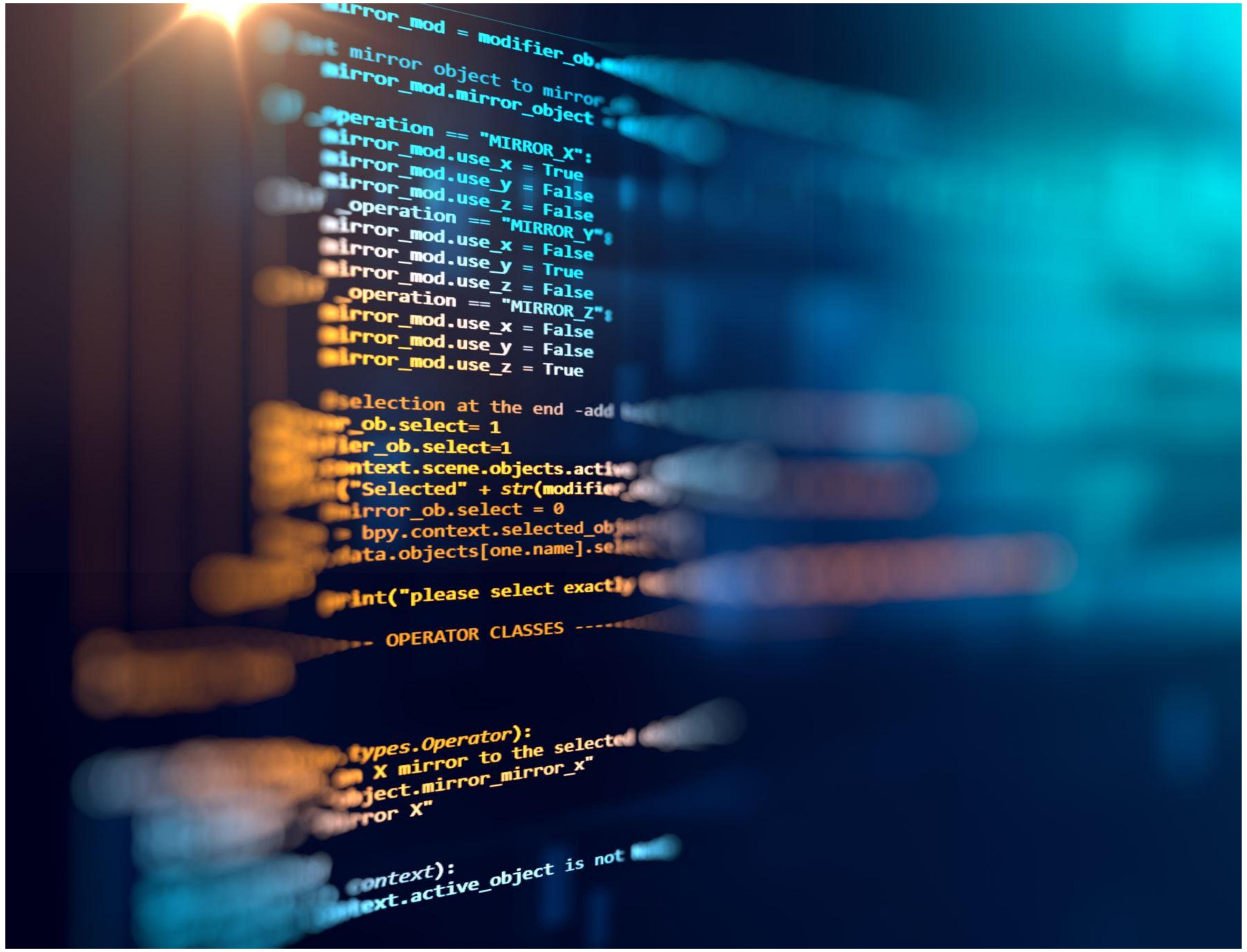
Value-Based view:

- The central view in this case is how much a customer is willing to pay for a certain level of quality.

Software Defects

The IEEE Standard 610.12 defines the following terms related to defects:

- **Failure:** the inability of a system or component to perform its required functions within specified performance requirements.
- **Fault:** An incorrect step, process, or data definition in a computer program.
- **Error:** A human action that produces an incorrect result.



Software Errors, Faults and Failures

An **error** can be a **grammatical error** in one or more of the code lines, or a **logical error** in carrying out one or more of the client's requirements.

Not all software errors become **software faults.** In some cases, the software error can cause improper functioning of the software. In many other cases, erroneous code lines will not affect the functionality of the software as a whole.

A failure is said to occur whenever the external behavior of a system does not conform to that prescribed in the system specification. A software fault becomes a software failure only when it is "activated"

Discussion Question - 1

1. List the four components of a software system.
2. How does the quality of each component contribute to the quality of the developed software?
3. How does the quality of each component contribute to the quality of the software maintenance?

Discussion Question - 2

George Wise is an exceptional programmer. Testing his software modules reveals very few errors, far fewer than the team's average. He keeps his schedule promptly, and only rarely he is late in completing his task. He always finds original ways to solve programming difficulties, and uses an original, individual version of the coding style.

He dislikes preparing the required documentation, and rarely does he do it according to the team's templates.

A day after completing a challenging task, on time, he was called to the office of the department's chief software engineer. Instead of being praised for his accomplishments (as he expected), he was warned by the company's chief software engineer that he would be fired unless he began to fully comply with team's coding and documentation instructions.

1. Do you agree with the position taken by the department's chief software engineer?
2. If yes, could you suggest why his/her position was so determined/decisive?

Discussion Question - 3

- To know that quality has improved, it would be helpful to be able to measure quality.
- How can we measure quality?

Mini case-1

Just half a year ago we launched our new product - the radar detector, The firmware RD-8.1, embedded in this product, seems to be the cause for its success.

However, when we began planning the development of a European version of the product, we discovered that although the products will be almost identical, our software development department needs to create new firmware; almost all the design and programming will be new.

Mini case-2

Believe it or not, our software package “Blackboard” for school teachers, launched just three months ago, is already installed in 187 schools. The development team just returned from a week in Hawaii, their vacation bonus.

However, we have been receiving sudden daily complaints from the “Blackboard” maintenance team. They claim that the lack of failure-detection features in the software, in addition to the poor programmer's manual, has caused them to invest more time than estimated to deal with bugs or add minor software changes that were agreed as part of purchasing contracts with clients.

Common points in the cases

- The software projects satisfactorily fulfilled the basic requirements to perform the correct calculations.
- Apparently, the software packages successfully fulfilled the correctness requirements.
- The software projects suffered from poor performance in important areas such as software maintenance, software reliability, software reuse, and user training.
- Apparently, the software packages fail to fulfill the maintainability, training usability, reliability, and reusability requirements.

Difficulties faced by users, developers, and maintenance staff

Common points in the cases

The common cause, for poor performance of the software projects developed in these areas, was a lack in essential parts of the project requirements.

We need comprehensive requirements set that covers all attributes of software and aspects of its use needed throughout the software life cycle.
Usability, reusability, maintainability, etc. are expected to achieve increased user satisfaction, and higher efficiency of development and maintenance teams.

Problems could have been avoided if the relevant requirements had been included in the requirement document.

Software Quality Requirements

- Requirement Documentation (Specification) is one of the most important elements for achieving software quality
- Need to explore what constitutes a good software requirements document.
- Some SQA Models suggest 11-15 factors categorized (some fewer, some more)

Software Quality Attributes are also called Software Quality Factors

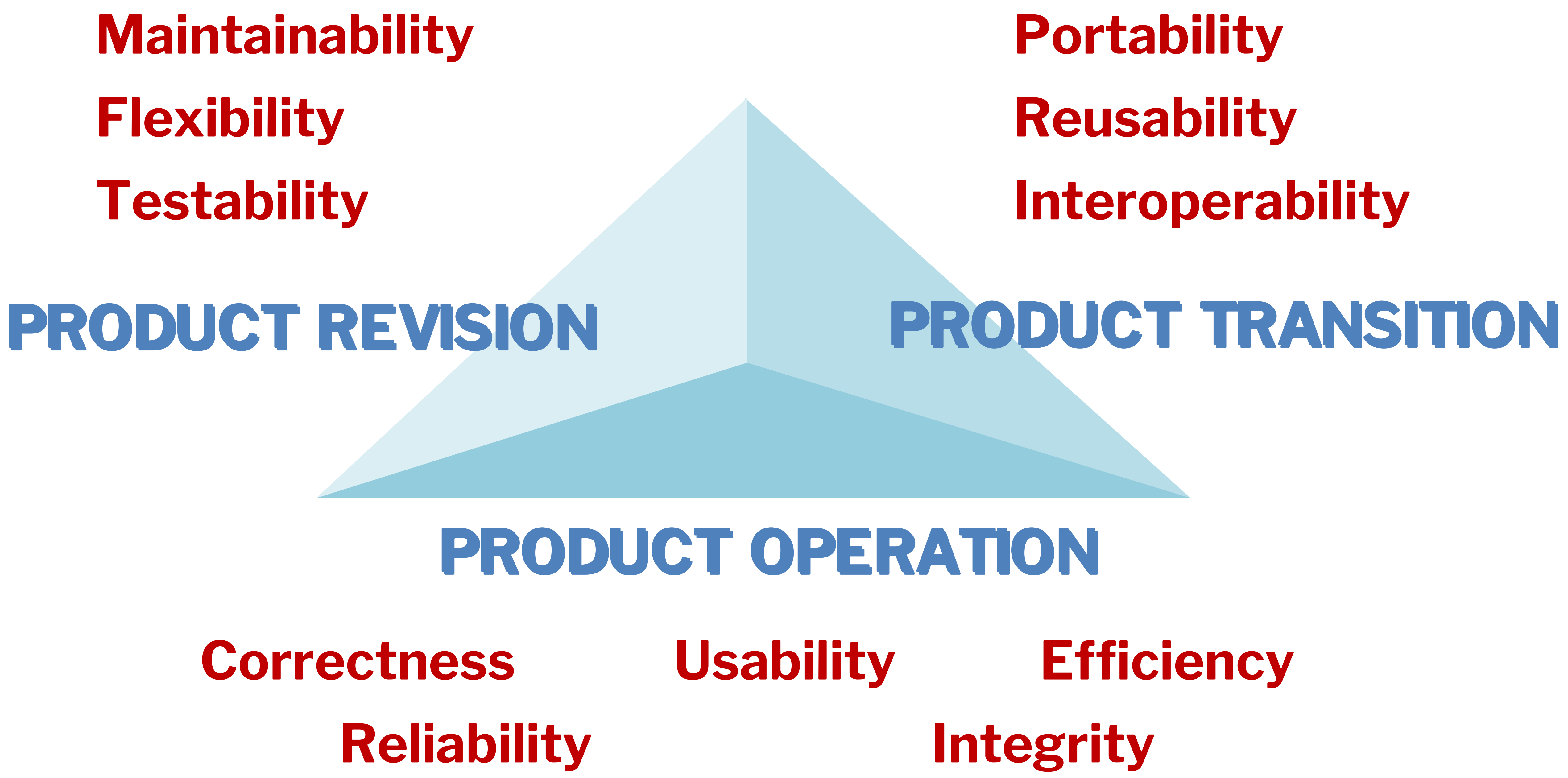
The need!

- Need for improving poor requirements documents is widespread .
- Frequently lack quality factors such as: usability, reusability, maintainability, etc.
- Software industry groups the long list of related attributes into what we call quality factors, sometimes non-functional requirements.
- Natural to assume an unequal emphasis on all quality factors. Emphasis varies from project to project: scalability, maintainability, reliability, portability, etc.

In Software Engineering, we concentrate on capturing, designing, implementing, and deploying with emphasis on functional requirements.

- Little (not none!) emphasis on the non-functional requirements (quality factors).
- More and more emphasis now placed on quality factors
- Can be a critical factor in satisfying overall requirements.

McCall’s Triangle of Quality



Evaluation of Software Quality Framework

- Quality requirements that the software product must meet
- Quality factors – Management-oriented attributes of software that contribute to its quality
- Quality subfactors – Decompositions of a quality factor to its technical components
- Metrics – quantitative measures of the degree to which given attributes (factors) are present

Example

- Quality requirement – "The product will be easy to use"

- Understanding
  - Learning time: Time for new user to gain basic understanding of features of the software
- Ease of learning
  - Learning time: Time for new user to learn how to perform basic functions of the software
- Operability
  - Operation time: Time required for a user to perform operation(s) of the software
- Communicativeness
  - Human factors: Number of negative comments from new users regarding ergonomics, human factors, etc.

- Quality factor(s) – Usability (An attribute that bears on the effort needed for use and on the assessment of such use by users)

- Quality subfactors – Understandability, ease of learning, operability, communicativeness
  - Understandability – The amount of effort required to understand software
  - Ease of learning – The degree to which user effort required to learn how to use the software is minimised
  - Operability – The degree to which the effort required to perform an operation is minimised
  - Communicativeness – The degree to which software is designed in accordance with the psychological characteristics of users

Product operation factors

Correctness: extent to which a program fulfills its specification.

Reliability: ability not to fail.

Efficiency: use of resources execution and storage.

Integrity: protection of the program from unauthorized access.

Usability: ease of use of the software.

Product revision factors

Maintainability:
effort required to locate and fix a fault in a program.

Flexibility:
ease of making changes required by changes in operating environment.

Testability:
ease of testing the program to ensure that it is error-free and meets its specification.

Product transition factors

Portability:
effort required to transfer a program from one environment to another system.

Reusability:
ease of re-using software in a different context.

Interoperability:
effort required to couple a system to another system.

McCall’s Factor Model
Product Operation Factors - Correctness

- The presence and appropriateness of a set of functions for specified tasks, lack of bugs and defects: <u>measured in terms of defect rate (# bugs per line of code)</u>

- Software system’s required outputs should be correct (such as a query display of a customer’s balance, sales invoice printout and red alarms when temperature rises above 250°F).

A specification is required for each output, such as:

- ◎ The accuracy of the output information.
- ◎ The completeness of the output information.
- ◎ The up-to-datedness/timeliness of the output information (defined as the time between the event and its consideration by the software system).
- ◎ The standards for coding and documenting the software system should be correct.

McCall’s Factor Model
Product Operation Factors - Reliability

Capability to maintain its level of performance under stated conditions for a stated period of time

- Reliability requirements deal with failures to provide service.
- Measured in terms of failure rate (#failures per hour), Mean time between failures (MTBF); Mean time to failure (MTTF)
- They determine the maximum allowed software system failure rate, the maximum allowed percentage of a software system's downtime, and the maximum allowed recovery times.

Example:

The failure frequency of a heart-monitoring unit that will operate in a hospital’s intensive care ward is required to be less than one in 20 years. Its heart attack detection function is required to have a failure rate of less than one per million cases. Downtime for a system will not be more than ten minutes per month.

McCall's Factor Model
Product Operation Factors - Efficiency

- Relationship between the level of performance of the software and the amount of resources used, under stated conditions: <u>measured in terms of speed</u>
- deal with the hardware resources needed to perform all the functions of the software system.

The main hardware resources to be considered are:

- ◎ the computer's processing capacity (MHz).
- ◎ storage capability in terms of memory and disk capacity (MB, GB, TB).
- ◎ the data communication capability of the communication lines (KBPS , MBPS, GBPS).

McCall’s Factor Model
Product Operation Factors - Integrity

These requirements deal with the software system security; i.e. requirements to prevent unauthorized access, and distinguish between

- limited group who will be allowed to add and change data (“write permit”)
- the rest (unauthorized group)

These requirements are defined to cope with risks of “non-friendly” unauthorized attempts to damage the software system and its performance.

McCall's Factor Model
Product Operation Factors - Usability

Deal with the scope of staff resources needed to train a new employee or user and to operate the software system.

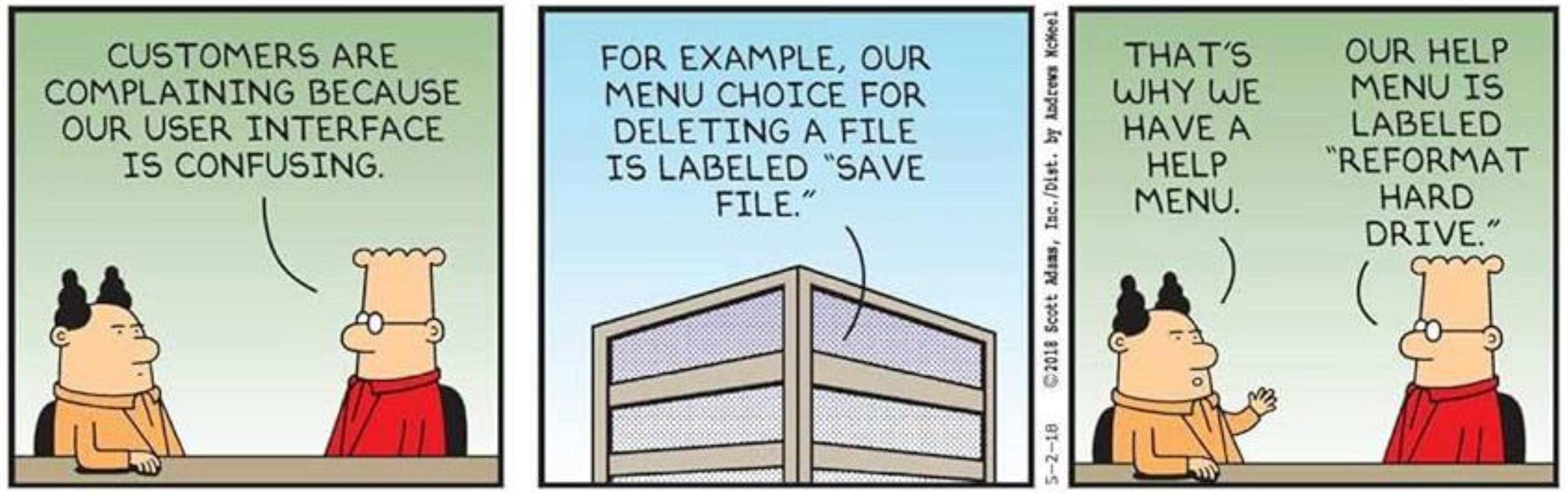
The usability requirements relate to the operation usability:

- the productivity of the user, that is, the average number of transactions performed per hour,
- training usability, the average time spent training a new employee.

Example: The software usability requirements for the new help desk system initiated by a home appliance manufacturing company lists the following specification requirements:

(1) A staff member should be able to handle at least 60 service calls a day.

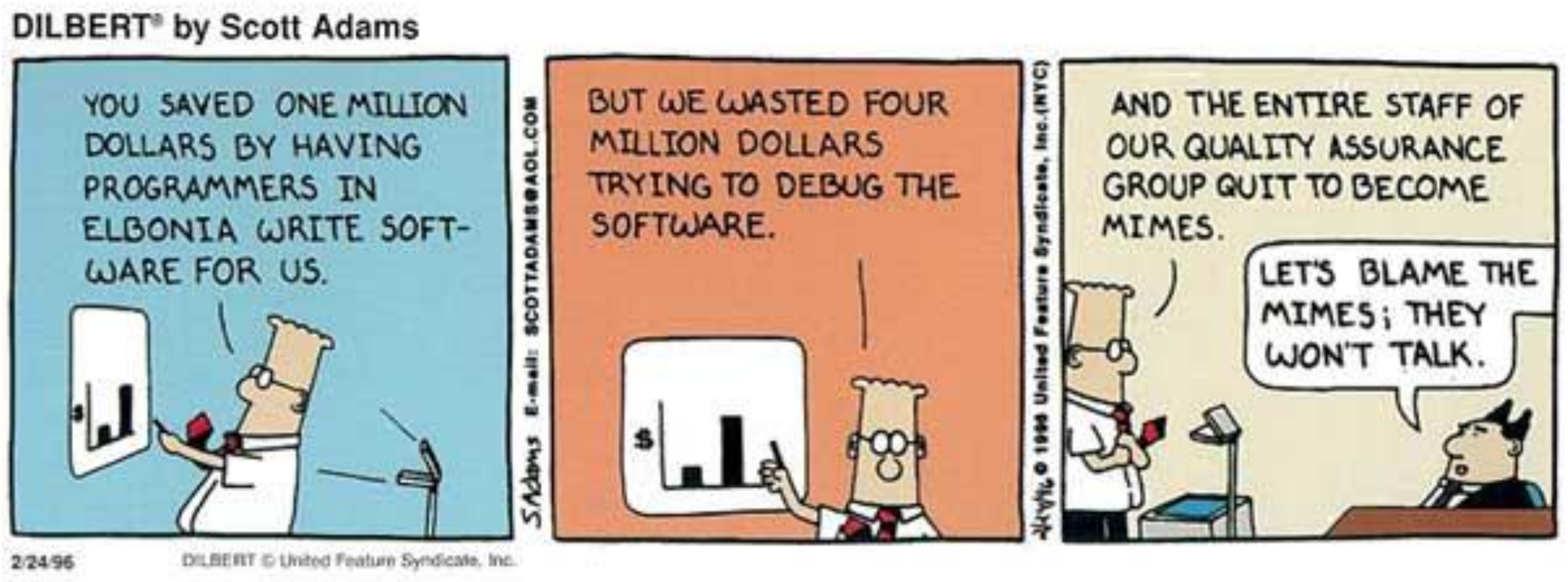
(2) Training a new employee will take no more than 2 days (16 training hours), at the end of which the trainee will be able to handle 45 service calls a day.


CUSTOMERS ARE COMPLAINING BECAUSE OUR USER INTERFACE IS CONFUSING.
FOR EXAMPLE, OUR MENU CHOICE FOR DELETING A FILE IS LABELED "SAVE FILE."
THAT'S WHY WE HAVE A HELP MENU.
OUR HELP MENU IS LABELED "REFORMAT HARD DRIVE."
©2018 Scott Adams, Inc./Dist. by Andrews McMeel
5-2-18

McCall’s Factor Model
Product Revision Factors - Maintainability

Maintainability requirements determine:

- ◎ The effort needed to make modifications, including corrections, improvements or adaptation of software to changes in environment, requirements, and functional specifications; is easy to change and adapt to new requirements.
- ◎ The efforts that will be needed by users and maintenance personnel to identify the reasons for software failures, to correct the failures, to verify the success of the corrections.


DILBERT® by Scott Adams
YOU SAVED ONE MILLION DOLLARS BY HAVING PROGRAMMERS IN ELBONIA WRITE SOFT-WARE FOR US.
BUT WE WASTED FOUR MILLION DOLLARS TRYING TO DEBUG THE SOFTWARE.
AND THE ENTIRE STAFF OF OUR QUALITY ASSURANCE GROUP QUIT TO BECOME MIMES.
LET'S BLAME THE MIMES; THEY WON'T TALK.
S.Adams E-mail: SCOTTADAMS@AOL.COM
© 1996 United Feature Syndicate, Inc.(NYC)
2/24/96
DILBERT © United Feature Syndicate, Inc.

McCall’s Factor Model
Product Revision Factors - Maintainability - cont.

Deals with the modular structure of the software, internal program documentation, programmer manual, architectural and detail design and corresponding documentation.

- **Typical maintainability requirements:**

(a) The programming will adhere to the company standards and guidelines.

(b) size of module <= 30 statements.

McCall’s Factor Model
Product Revision Factors - Flexibility

The capabilities and efforts required to adapt a software package to changes in the environment/different types of customers that use the software perhaps a little differently.

No software changes are needed if the package is planned for flexibility.

Example:

- TSS (teacher support software) deals with the documentation of student achievements, the calculation of final grades, the printing of term grade documents, and the automatic printing of warning letters to parents of failing students.

- The software specifications included the following flexibility requirements: The software should be suitable for teachers of all subjects and all school levels (elementary, junior and high schools).

McCall’s Factor Model
Product Revision Factors - Testability

Deal with the testing process of a software system, as well as with its operation.

Testability requirements for the ease of testing are related to special features in the programs that help the tester, for instance by:

(a) providing predefined intermediate results and log files.

(b) automatic diagnostics performed by the software system prior to starting the system, to find out whether all components of the software system are in working order and to obtain a report about the detected faults.

McCall’s Factor Model
Product Revision Factors - Testability

Example:

- An industrial computerised control unit is programmed to calculate various measures of production status, report the performance level of the machinery, and operate a warning signal in predefined situations.

- One testability requirement demanded was to develop a set of standard test data with known expected correct reactions in each stage.
- This standard test data is to be run every morning, before production begins, to check whether the computerized unit reacts properly.

McCall’s Factor Model
Product Transition Factors - Portability

Relates to the adaptation of a software system to other environments consisting of different hardware, different operating systems.

These requirements make it possible to continue using the same basic software in diverse situations or to use it simultaneously in diverse hardware and operating systems situations.

Example:

- A software package designed and programmed to operate in a Windows 2000 environment is required to allow low-cost transfer to Linux environments.

McCall’s Factor Model
Product Transition Factors - Reusability

Two directions:

1. Use of a software module/ application, taken from an existing software product in a new project currently being developed.
2. Develop modules/project, in a way to enable their reuse in future projects.

Reused software is expected to be tested by its developers, and already corrected according to failures experienced by former users.

Thus, the reuse of software is expected to save development resources, shorten the development schedule, and provide higher quality software modules.

McCall’s Factor Model
Product Transition Factors - Reusability

Example:

A list of the software modules for the pool project designed and programmed in a way that will allow its reuse in the spa's future software system planned to be developed next year. The entrance checks of membership cards module

- The club members visit recording module
- The pool's restaurant billing module
- The processing of membership renewal payments

McCall’s Factor Model
Product Transition Factors - Interoperability

Software’s ability to interact with other specified systems Interoperability requirements focus on creating interfaces with other software systems or firmware.

Interoperability requirements can specify the name(s) of the software for which interface is required. Frequently these will be known ahead of time and plans can be made to provide for this requirement during design time.

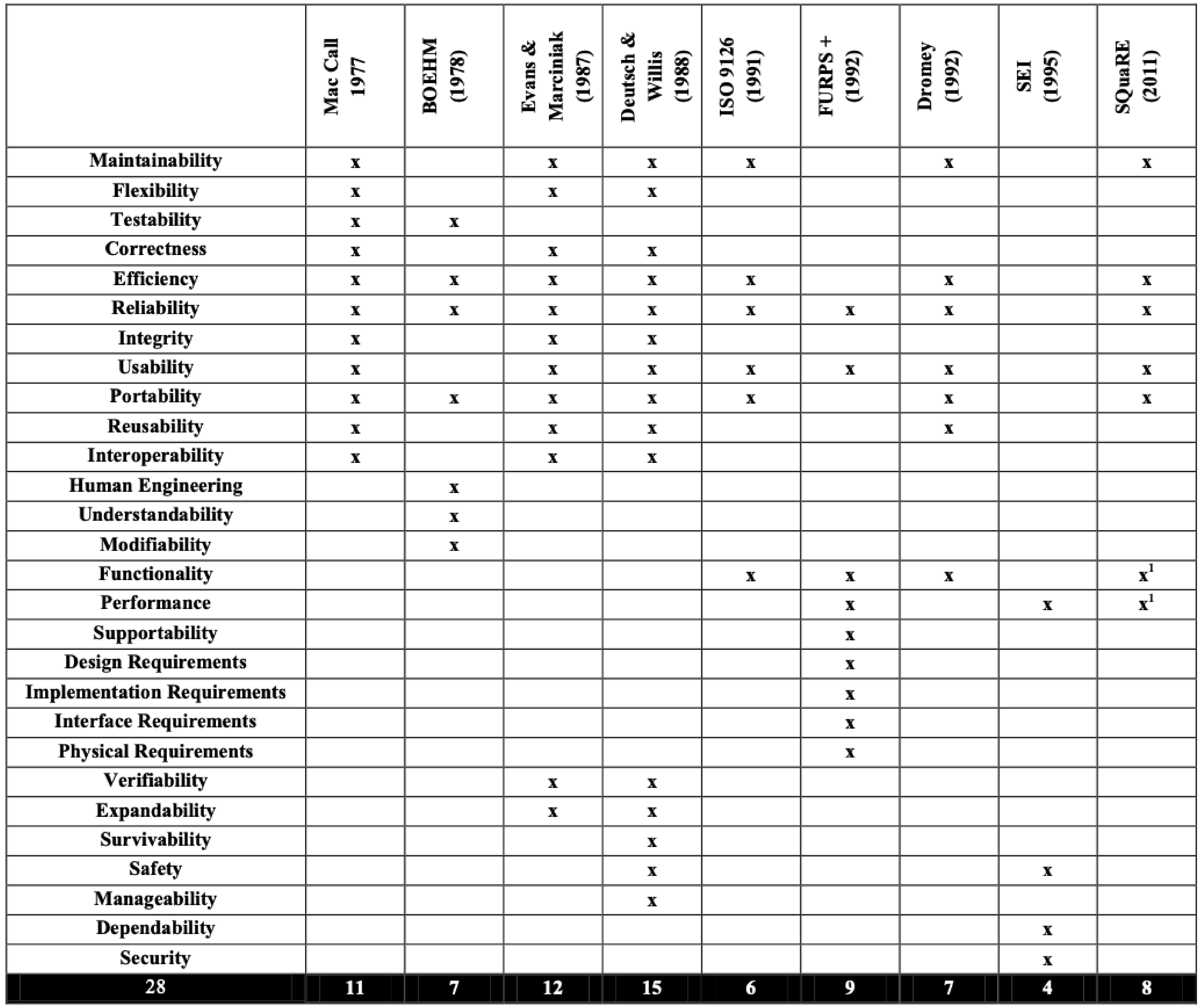
They may also specify the accepted standard output structure in a specific industry or application area.

Example:
The medical laboratory information system is required to interface with a medical clinic information system, in order to transmit patients' test results automatically to the physician's clinic.

McCall Quality Factors - Summary

Category	Main Concern	Key Focus
Product Operation	How well the software performs its intended functions	User experience, correctness, reliability
Product Revision	How easily the software can evolve	Maintainability, testability, flexibility
Product Transition	How well software adapts or integrates elsewhere	Portability, reusability, interoperability

McCall’s Factor Model & Alternatives



	Mac Call 1977	BOEHM (1978)	Evans & Marciniak (1987)	Deutsch & Willis (1988)	ISO 9126 (1991)	FURPS + (1992)	Dromey (1992)	SEI (1995)	SQuaRE (2011)
Maintainability	x		x	x	x		x		x
Flexibility	x		x	x					
Testability	x	x							
Correctness	x		x	x					
Efficiency	x	x	x	x	x		x		x
Reliability	x	x	x	x	x	x	x		x
Integrity	x		x	x					
Usability	x		x	x	x	x	x		x
Portability	x	x	x	x	x		x		x
Reusability	x		x	x			x		
Interoperability	x		x	x					
Human Engineering		x							
Understandability		x							
Modifiability		x							
Functionality					x	x	x		x[1]
Performance						x		x	x[1]
Supportability						x			
Design Requirements						x			
Implementation Requirements						x			
Interface Requirements						x			
Physical Requirements						x			
Verifiability			x	x					
Expandability			x	x					
Survivability				x					
Safety				x				x	
Manageability				x					
Dependability								x	
Security								x	
28	11	7	12	15	6	9	7	4	8

(1) For SQuaRE: Functionality is Functional suitability and Performance is Performance efficiency

https://www.researchgate.net/publication/303792652_Comparative_Study_of_Software_Quality_Models

McCall's Factor Model & Alternatives

No.	Software quality factor	McCall's classic model	Alternative factor models	
			Evans and Marciniak model	Deutsch and Willis model
1	Correctness	+	+	+
2	Reliability	+	+	+
3	Efficiency	+	+	+
4	Integrity	+	+	+
5	Usability	+	+	+
6	Maintainability	+	+	+
7	Flexibility	+	+	+
8	Testability	+		
9	Portability	+	+	+
10	Reusability	+	+	+
11	Interoperability	+	+	+
12	Verifiability		+	+
13	Expandability		+	+
14	Safety			+
15	Manageability			+
16	Survivability			+

These models do not differ substantially from McCall's model. The McCall factor model provides a practical, up-to-date method for classifying software requirements (Pressman, 2000).

Other Quality Factors
Verifiability

addresses design and programming features that allow for efficient verification of design and programming

- This does not refer to outputs; rather, structure of code, design elements and their dependencies, coupling, cohesion, patterns, etc

Most verifiability requirements refer to modularity, simplicity, and adherence to documentation and programming guidelines

A significant similarity exists between the verifiability factor and McCall's testability factor.

Other Quality Factors
Expandability

- refers to future efforts that will be needed to serve larger populations, improve service, or add new applications in order to improve usability.
- refers to scalability and extensibility to provide more usability.

Essentially, this is McCall’s flexibility.

Other Quality Factors
Safety

- aimed at eliminating conditions that may be hazardous to equipment and equipment operators, as a result of errors in process control software.
- address conditions that could bring the equipment or application down especially for controlling software, as in setting alarms or sounding warnings.

Especially important to process control/real time software such as that running conveyor belts or instrumentation for ordinance.

Example:

In a chemical plant, a computerized system controls the flow of acid according to pressure and temperature changes occurring during production. Safety requirements refer to the system’s computerized reactions in cases of dangerous situations and also specify what kinds of alarms are needed in each case.

Other Quality Factors
Manageability

- refer to the administrative tools that support software modification during the software development and maintenance periods.
- refer to tools primarily administrative to control versions, configurations and change management / tracking.

We must have tools to manage versions and various configurations that may vary from customer to customer.

Example:

“Chemilog” is a software system that automatically logs the flows of chemicals into various containers to allow for later analysis of the efficiency of production units. The development and issue of new versions and releases of “Chemilog” are controlled by the Software Development Board whose members act according to the company’s software modifications procedures.

Other Quality Factors
Survivability

refer to MTBF or continuity of service , as well as MTTR (mean time to recover). These define

- the minimum time allowed between failures of the system,
- the maximum time permitted for recovery of service, two factors that pertain to service continuity.

Significant similarity exists between the survivability factor and the McCall's reliability factor.

Example:

The probability that unrecoverable damage to the betting files will occur in case of any system failure is to be limited to less than one in a million in a betting software.

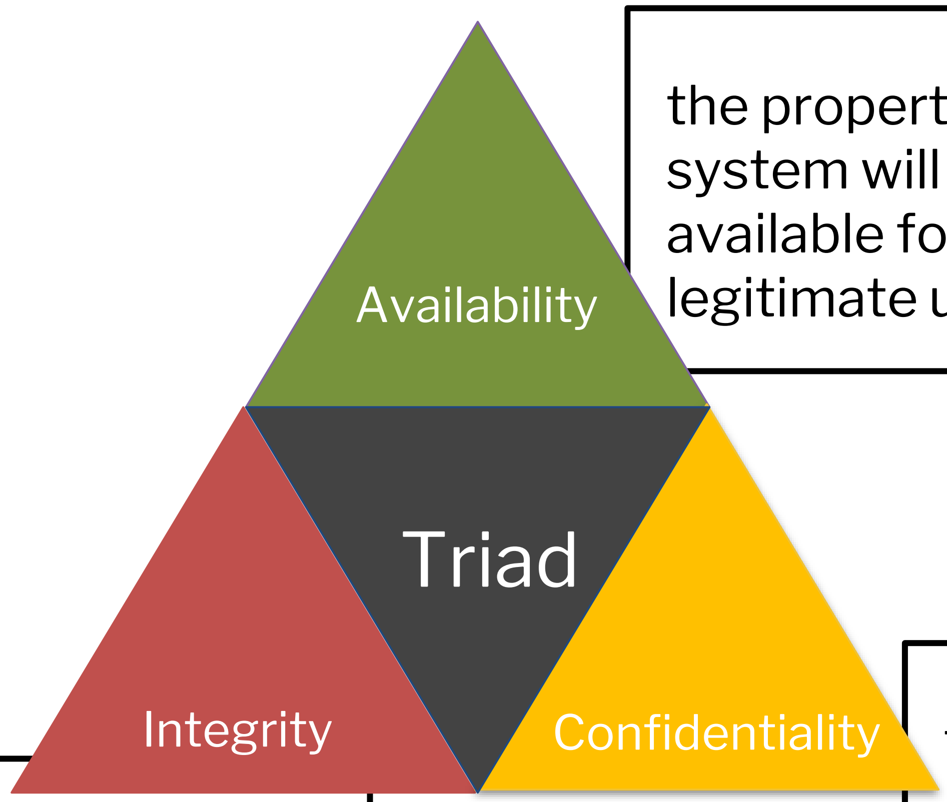
Other Quality Factors
Security

It is a measure of the system’s ability to protect data and information from unauthorized access while still providing access to people and systems that are authorised.

Example:

In an online healthcare system, patients can access their medical records, communicate with their doctors, and schedule appointments. Only the patient and authorized healthcare providers can access medical records. Patients can specify which doctors have permission to view their data.

focuses on three characteristics (CIA Triad)



Availability: the property that the system will be available for legitimate use

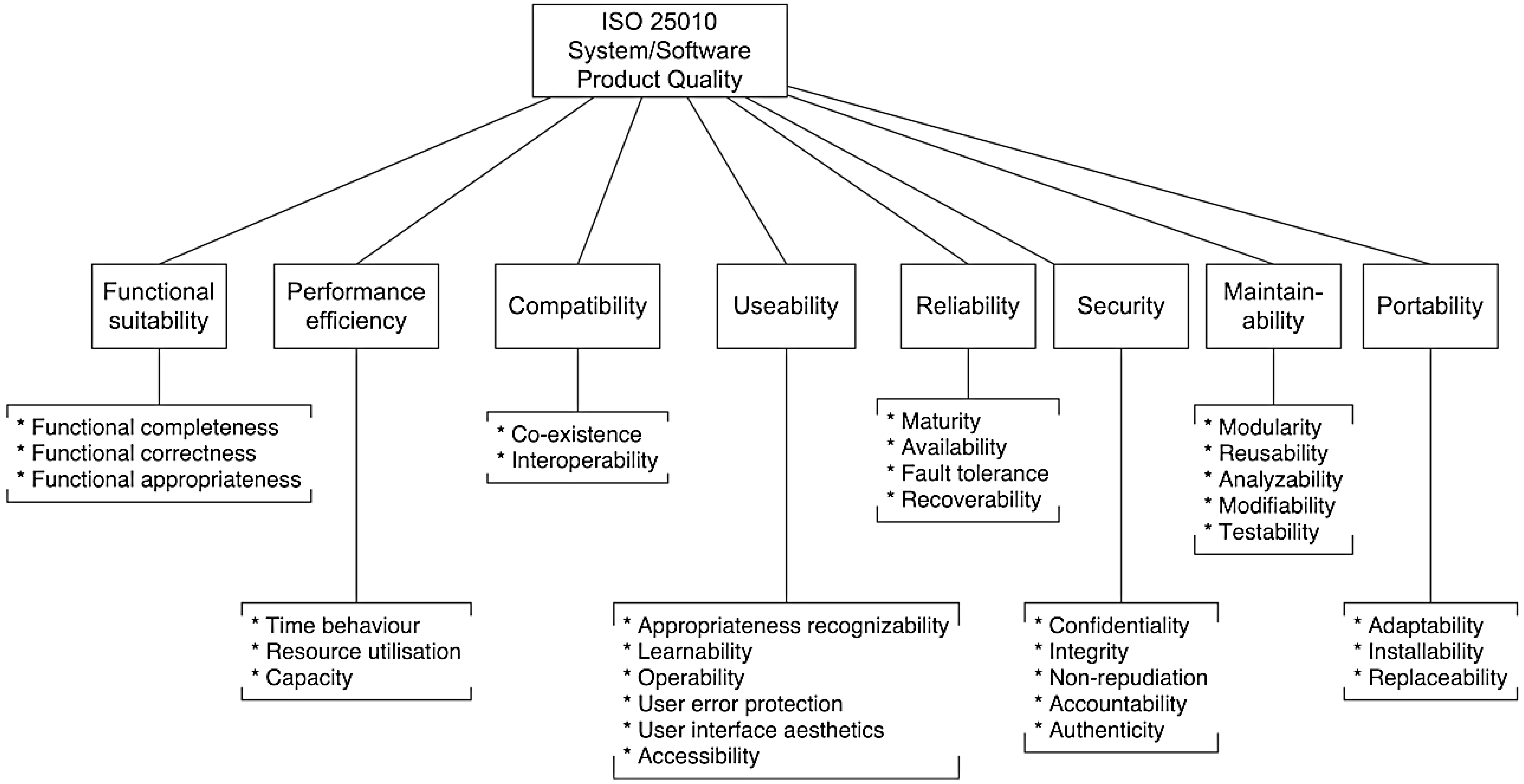
Integrity: the property that data or services are not subject to unauthorized manipulation

Confidentiality: the property that data or services are protected from unauthorized access

Most up-to-date Quality Model ISO25010

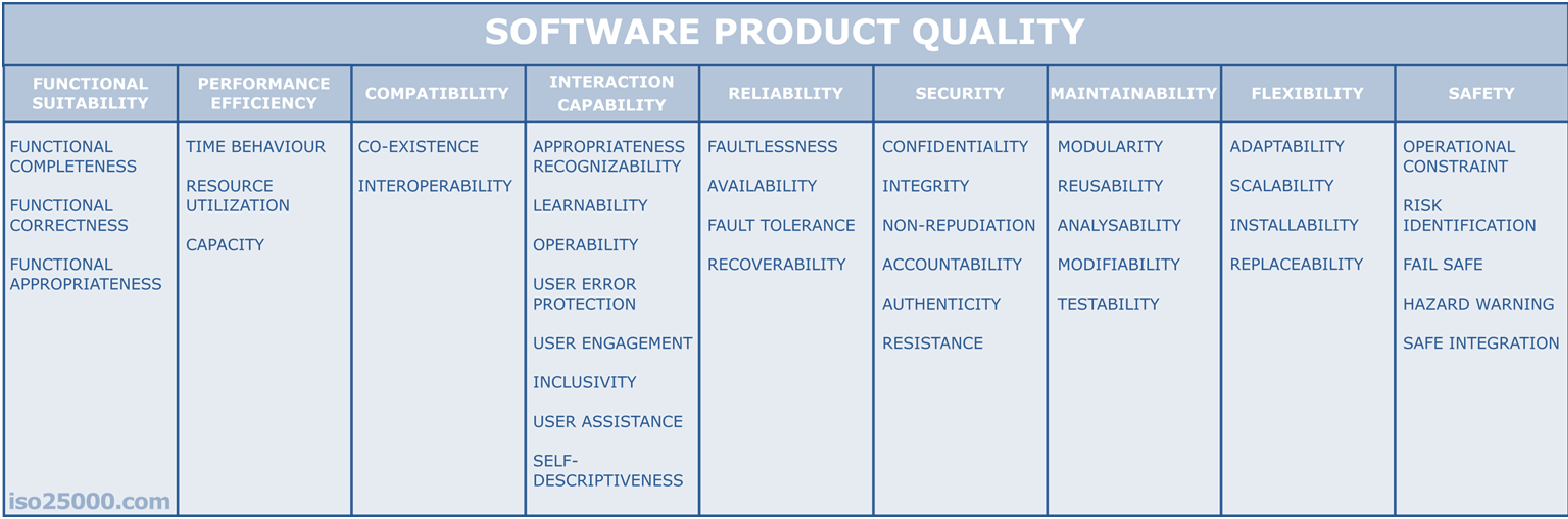
Each factor provides a dimension for assessing and continuously enhancing software quality through its lifecycle.

These factors form the basis for modern software evaluation, quality assurance, and continuous improvement in software development and deployment.



2011 Model

Most up-to-date Quality Model ISO25010



SOFTWARE PRODUCT QUALITY								
FUNCTIONAL SUITABILITY	PERFORMANCE EFFICIENCY	COMPATIBILITY	INTERACTION CAPABILITY	RELIABILITY	SECURITY	MAINTAINABILITY	FLEXIBILITY	SAFETY
FUNCTIONAL COMPLETENESS	TIME BEHAVIOUR	CO-EXISTENCE	APPROPRIATENESS RECOGNIZABILITY	FAULTLESSNESS	CONFIDENTIALITY	MODULARITY	ADAPTABILITY	OPERATIONAL CONSTRAINT
FUNCTIONAL CORRECTNESS	RESOURCE UTILIZATION	INTEROPERABILITY	LEARNABILITY	AVAILABILITY	INTEGRITY	REUSABILITY	SCALABILITY	RISK IDENTIFICATION
FUNCTIONAL APPROPRIATENESS	CAPACITY		OPERABILITY	FAULT TOLERANCE	NON-REPUDIATION	ANALYSABILITY	INSTALLABILITY	FAIL SAFE
			USER ERROR PROTECTION	RECOVERABILITY	ACCOUNTABILITY	MODIFIABILITY	REPLACEABILITY	HAZARD WARNING
			USER ENGAGEMENT		AUTHENTICITY	TESTABILITY		SAFE INTEGRATION
			INCLUSIVITY		RESISTANCE			
			USER ASSISTANCE					
			SELF-DESCRIPTIVENESS					

2022 Model

Most up-to-date Quality Model ISO25010

- **Functional Suitability**
  - Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.
  - Sub-characteristics:
    - Functional completeness - Degree to which the set of functions covers all the specified tasks and intended users' objectives.
    - Functional correctness - Degree to which a product or system provides accurate results when used by intended users.
    - Functional appropriateness - Degree to which the functions facilitate the accomplishment of specified tasks and objectives.

Most up-to-date Quality Model ISO25010

- **Performance Efficiency**
  - Degree to which a product performs its functions within specified time and throughput parameters and is efficient in the use of resources (such as CPU, memory, storage, network devices, energy, materials...) under specified conditions.
  - Sub-characteristics:
    - Time behaviour - Degree to which the response time and throughput rates of a product or system, when performing its functions, meet requirements.
    - Resource utilisation - Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
    - Capacity - Degree to which the maximum limits of a product or system parameter meet requirements.

Most up-to-date Quality Model ISO25010

Compatibility

- Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same common environment and resources.
- Sub-characteristics:
  - Co-existence - Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
  - Interoperability - Degree to which a system, product or component can exchange information with other products and mutually use the information that has been exchanged.

Most up-to-date Quality Model ISO25010

- **Interaction Capability (ex-Usability)**
  - Degree to which a product or system can be interacted with by specified users to exchange information via the user interface to complete specific tasks. Sub-characteristics:
    - Appropriateness recognizability - Degree to which users can recognise whether a product or system is appropriate for their needs.
    - Learnability - Degree to which the functions of a product or system can be learnt to be used by specified users within a specified amount of time.
    - Operability - Degree to which a product or system has attributes that make it easy to operate and control.
    - User error protection - Degree to which a system prevents users against operation errors.
    - User engagement - Degree to which a user interface presents functions and information in an inviting and motivating manner encouraging continued interaction.
    - Inclusivity - Degree to which a product or system can be used by people of various backgrounds (such as people of various ages, abilities, cultures, ethnicities, languages, genders, economic situations, etc.).
    - User assistance - Degree to which a product can be used by people with the broadest range of characteristics and capabilities to achieve specified goals in a specified context of use.
    - Self-descriptiveness - Degree to which a product presents appropriate information, where needed by the user, to make its capabilities and use immediately obvious to the user without excessive interactions with a product or other resources (such as user documentation, help desks or other users).

Most up-to-date Quality Model ISO25010

- **Reliability**
  - Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.
  - Sub-characteristics:
    - Faultlessness - Degree to which a system, product or component performs specified functions without fault under normal operation.
    - Availability - Degree to which a system, product or component is operational and accessible when required for use.
    - Fault tolerance - Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
    - Recoverability - Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

Most up-to-date Quality Model ISO25010

- **Security**
  - Degree to which a product or system defends against attack patterns by malicious actors and protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorisation.
  - Sub-characteristics:
    - Confidentiality - Degree to which a product or system ensures that data are accessible only to those authorised to have access.
    - Integrity - Degree to which a system, product or component ensures that the state of its system and data are protected from unauthorized modification or deletion either by malicious action or computer error.
    - Non-repudiation - Degree to which actions or events can be proven to have taken place so that the events or actions cannot be repudiated later.
    - Accountability - Degree to which the actions of an entity can be traced uniquely to the entity.
    - Authenticity - Degree to which the identity of a subject or resource can be proved to be the one claimed.
    - Resistance - Degree to which the product or system sustains operations while under attack from a malicious actor.

Most up-to-date Quality Model ISO25010

- **Maintainability**
  - Degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements.
  - Sub-characteristics:
    - Modularity - Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
    - Reusability - Degree to which a product can be used as an asset in more than one system, or in building other assets.
    - Analysability - Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
    - Modifiability - Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
    - Testability - Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Most up-to-date Quality Model ISO25010

- **Flexibility (~ex-Portability)**
  - Degree to which a product can be adapted to changes in its requirements, contexts of use or system environment.
  - Sub-characteristics:
    - Adaptability - Degree to which a product or system can effectively and efficiently be adapted for or transferred to different hardware, software or other operational or usage environments.
    - Scalability - Degree to which a product can handle growing or shrinking workloads or to adapt its capacity to handle variability.
    - Installability - Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.
    - Replaceability - Degree to which a product can replace another specified software product for the same purpose in the same environment.

Most up-to-date Quality Model ISO25010

- **Safety**
  - Degree to which a product under defined conditions to avoid a state in which human life, health, property, or the environment is endangered.
  - Sub-characteristics:
    - Operational constraint - Degree to which a product or system constrains its operation to within safe parameters or states when encountering operational hazard.
    - Risk identification - Degree to which a product can identify a course of events or operations that can expose life, property or environment to unacceptable risk.
    - Fail safe - Degree to which a product can automatically place itself in a safe operating mode, or to revert to a safe condition in the event of a failure.
    - Hazard warning - Degree to which a product or system provides warnings of unacceptable risks to operations or internal controls so that they can react in sufficient time to sustain safe operations.
    - Safe integration - Degree to which a product can maintain safety during and after integration with one or more components.

Most up-to-date Quality Model ISO25010

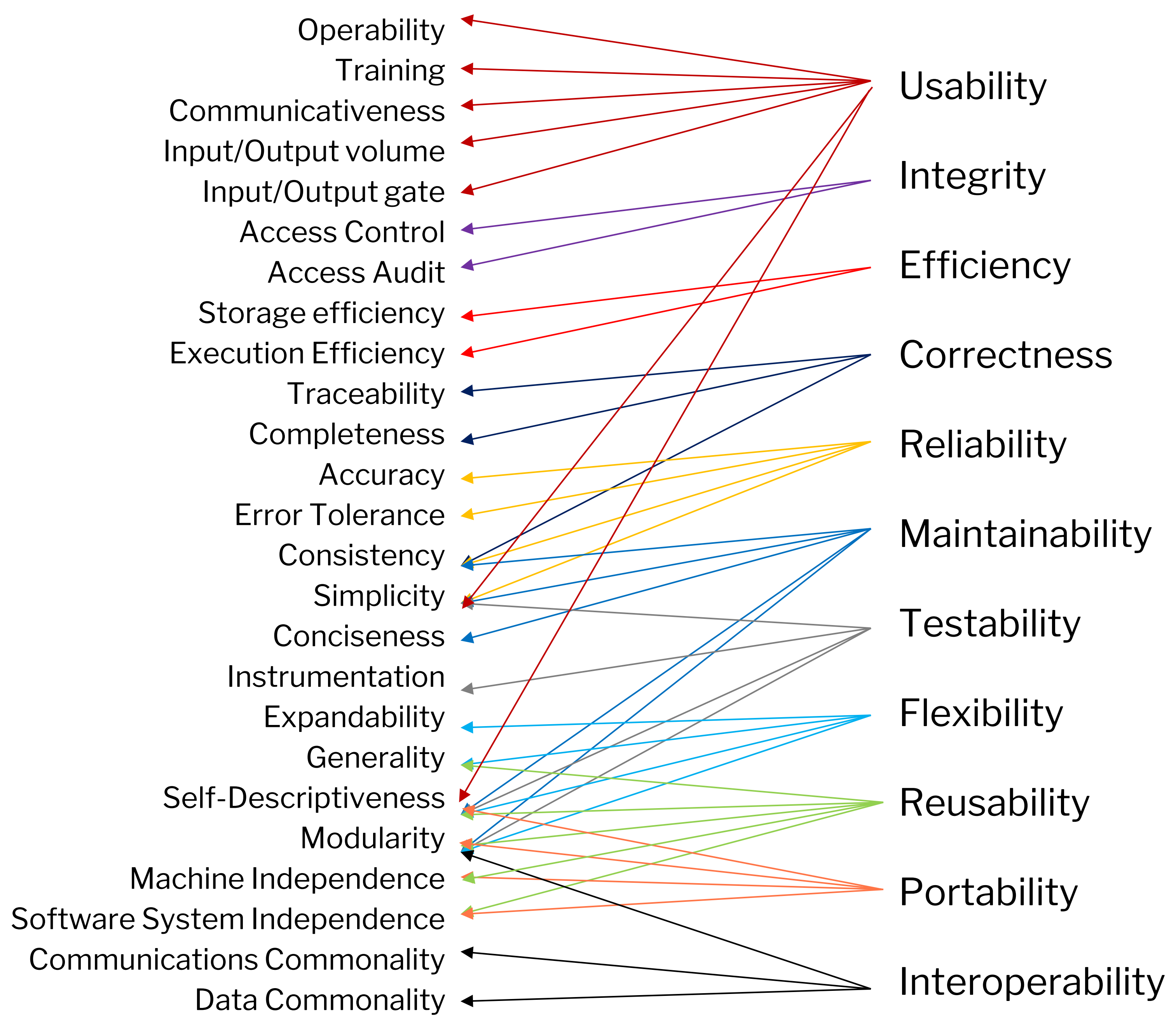
OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTERNOON!
IS THE SOFTWARE FINISHED?
SOFTWARE IS NEVER FINISHED.
DID YOU FIX ALL OF THE BUGS?
THERE'S NO WAY TO KNOW.
I CAN'T MANAGE YOU IF YOU DON'T LEARN TO LIE.
OKAY, THE SOFTWARE WILL BE PERFECT IN 2.3 DAYS.

1. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010
2. https://www.perforce.com/blog/qac/what-is-iso-25010
3. https://www.iso.org/obp/ui/
4. https://blog.pacificcert.com/iso-25010-software-product-quality-model/
5. https://www.workingsoftware.dev/the-ultimate-guide-to-write-non-functional-requirements/
6. https://tiobe.com/files/TIOBEQualityIndicator_v4_10.pdf
7. https://www.it-cisq.org/cisq-supplements-isoiec-25000-series-with-automated-quality-characteristic-measures/
8. https://www.diva-portal.org/smash/get/diva2:1530284/FULLTEXT01.pdf
9. https://quality.arc42.org/articles/iso-25010-shortcomings

Bridging the Gap: Factors, Criteria, and Subfactors

- **The Compliance Challenge:** Quality factors (e.g., Maintainability) are often **general attributes**. The challenge is bridging the gap between this general definition and explicit review questions requiring **quantitative measurement**.
- **The Solution:** **Criteria (Subfactors):** Each quality factor is represented by several explanatory criteria (subfactors).
    - These criteria are quantitative and qualitative.
    - They enable defining measurable software requirements and developing **software quality metrics**.
    - *Example:*
        - **Modularity** is a key sub-factor for both Maintainability and Flexibility.
        - **Accuracy** and **Completeness** are sub-factors for Correctness.
- **Preference for Quantification:** Quantitative measures are usually preferred over qualitative measures because they provide **more objective assessments** of software performance.

Match the factors



Match the factors

#	Scenario (Short Case)	Quality Factor
1	A **university portal** displays students' grades but fails to show GPA or ranking, even though both are in the requirements.	**Functional Suitability** (completeness, correctness, appropriateness)
2	A **desktop design tool** works on Windows but fails to install on macOS or Linux.	**Flexibility** (adaptability, scalability, installability, replaceability)
3	A **surgical robot software** freezes when multiple sensors send conflicting data, risking patient safety.	**Safety** (operational constraint, risk identification, fail safe, hazard warning, safe integration)
4	A **public transport mobile app** slows down significantly during rush hours as more users query real-time data.	**Performance Efficiency** (time behaviour, resource utilization, capacity)
5	A **smart home system** connects to lights made by another vendor.	**Compatibility** (co-existence, interoperability)
6	A **banking app** uses small icons and complex menus, confusing elderly users.	**Interaction Capability** (learnability, operability, user error protection, inclusivity)
7	A **remote patient monitoring system** crashes when the internet connection drops.	**Reliability** (fault tolerance, availability, recoverability)
8	A **school grading web system** allows URL-based access to other students' records.	**Security** (confidentiality, integrity, accountability, resistance)
9	A **retail POS system** has duplicated logic in every module, making updates slow and error-prone.	**Maintainability** (modularity, reusability, analysability, modifiability, testability)

1. Interaction Capability
2. Safety
3. Performance Efficiency
4. Functional Suitability
5. Reliability
6. Maintainability
7. Compatibility
8. Flexibility
9. Security

Discussion Question

The City of Mountain View has decided to develop a software package that will serve the youth clubs operated by the city. The software's main tasks will be:

- Follow-up of monthly payments of the members.
- Preparing lists of participants in the various courses offered by the clubs.
- Production of reminder notices to course participants who fail to appear regularly.
- Statistical reports about membership and participation in club activities.

The city has already implemented the following software packages:

- Tax collection
- Public Library
- School follow-up and achievements control
- Water consumption billing.

The City Council has asked the Information Technology Unit to report the council about the possibilities for reuse of the city software packages already available to the city in the youth club software package.

(1) Could you suggest which modules of the existing city software packages could be reused in the new software? List your assumptions about the contents of the existing software packages and the required new software.

(2) Could you grade the reused modules suggested in (1) according to the scope of adaptation efforts required to apply the reused module in the youth club software package?

Quality Factors and Specification

- Defining quality requirements should involve bodies such as the client's requirements document and the developer's additional requirements document.
- The overall goal is to define requirements that cover all aspects of software use throughout the life cycle.

- **Four Dimensions of Quality (Chemuturi):** Quality must be assured across four interconnected dimensions:
  1. **Specification Quality:** How well specifications are defined. Tools used include standards, guidelines, and checklists. This is where quality factors are initially defined.
  2. **Design Quality:** How well the product is designed to fulfill specifications.
  3. **Development/Construction Quality:** Adhering to standards for code quality.
  4. **Conformance Quality:** Ensuring quality is built in at every stage, measured via quality metrics (e.g., defect removal efficiency, defect density).

Quality Factors in Testing and Metrics

Quality factors guide how the software is validated and measured throughout the project life cycle.

- **Requirement-Driven Testing:** Testing classes should be classified based on McCall's factors to ensure full coverage of the requirements.
    - **Operation Factor Testing:** Includes correctness tests (software correctness, user manual, availability), reliability tests, stress tests, security tests, and usability tests.
    - **Revision Factor Testing:** Includes maintainability correctness tests, flexibility tests, and testability tests.
    - **Transition Factor Testing:** Includes portability tests, reused software correctness tests, and interoperability tests.

- **Quality Metrics:** Factors form the basis for **Software Product Metrics**. These are quantitative measures related to attributes like functionality, reliability, usability, efficiency, maintainability, and portability.
    - The degree of compliance of a software product to its required factors is measured by these metrics.

Trade-offs Between Quality Factors

It is important to note that quality factors are generally **not completely independent**.

- **Inversely Proportional:** Trying to improve one factor may degrade another.
  - *Example:* Making code more portable may **reduce its efficiency**.
  - *Example:* If **high performance** is required, **reliability suffers**.

- **Positive Impact:** Enhancing one factor may positively impact another.
  - *Example:* Enhancing a system's **correctness** will increase its **reliability**.
  - *Example:* Enhancing **testability** will improve **maintainability**.