AMDARIS

REQUIREMENTS TYPES

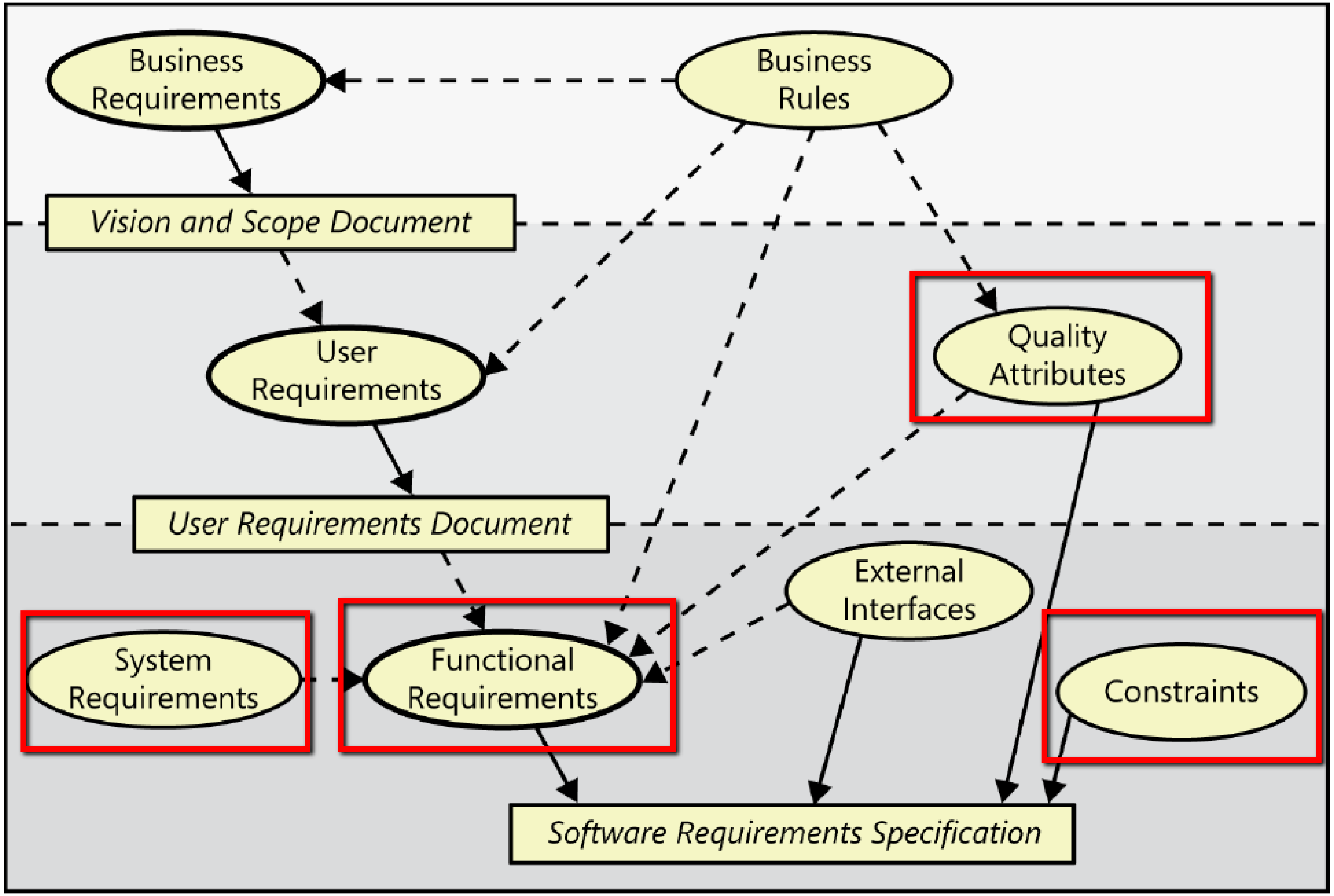
September 2019 | Olesea Oaserele

- Users naturally focus on specifying functional requirements, i.e., behavior that a software will exhibit under specific conditions.
- The success of a product represent more than just delivering the right functionality
- Users also have the expectations about *how well* the product will work:
  - How easy it is to use
  - How quickly it runs
  - How often it fails
  - How it handles exceptional conditions
  - How secure it is

REQUIREMENTS TYPES

- Business Requirements
- User Requirements
- Market Requirements
- Architectural/System Requirements
- Transition Requirements
- Functional Requirements
- Non-Functional Requirements
    - System Requirements
    - Quality attributes
    - Constraints

We will focus on functional and non-functional requirements


Business Requirements
Business Rules
Vision and Scope Document
User Requirements
Quality Attributes
User Requirements Document
External Interfaces
System Requirements
Functional Requirements
Constraints
Software Requirements Specification

REQUIREMENTS TYPES

- Business requirement - A high-level business objective of the organization that builds a product or of a customer who procures it.
- Business rule - A policy, guideline, standard, or regulation that defines or constrains some aspect of the business. Not a software requirement in itself, but the origin of several types of software requirements.
- Constraint - A restriction that is imposed on the choices available to the developer for the design and construction of a product.
- External interface requirement – A description of a connection between a software system and a user, another software system, or a hardware device

- Feature – One or more logically related system capabilities that provide value to a user and are described by a set of functional requirements.
- Functional requirement – A description of a behavior that a system will exhibit under specific conditions.
- Nonfunctional requirement – A description of a property or characteristic that a system must exhibit or a constraint that it must respect.
- Quality attribute – A kind of nonfunctional requirement that describes a service or performance characteristic of a product.
- System requirement – A top-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware.
- User requirement – A goal or task that specific classes of users must be able to perform with a system, or a desired product attribute.

FUNCTIONAL REQUIREMENTS

- A description of a behavior that a system will exhibit under specific conditions.
- Are derivate from business and user requirements
- Describe the ultimate aim of the product - what should it do.
- Describe what the developers must implement
- Represent the description of ways to accomplish users’ tasks (user requirements)
- Describe the business needs in order to accomplish business objectives (business requirements)

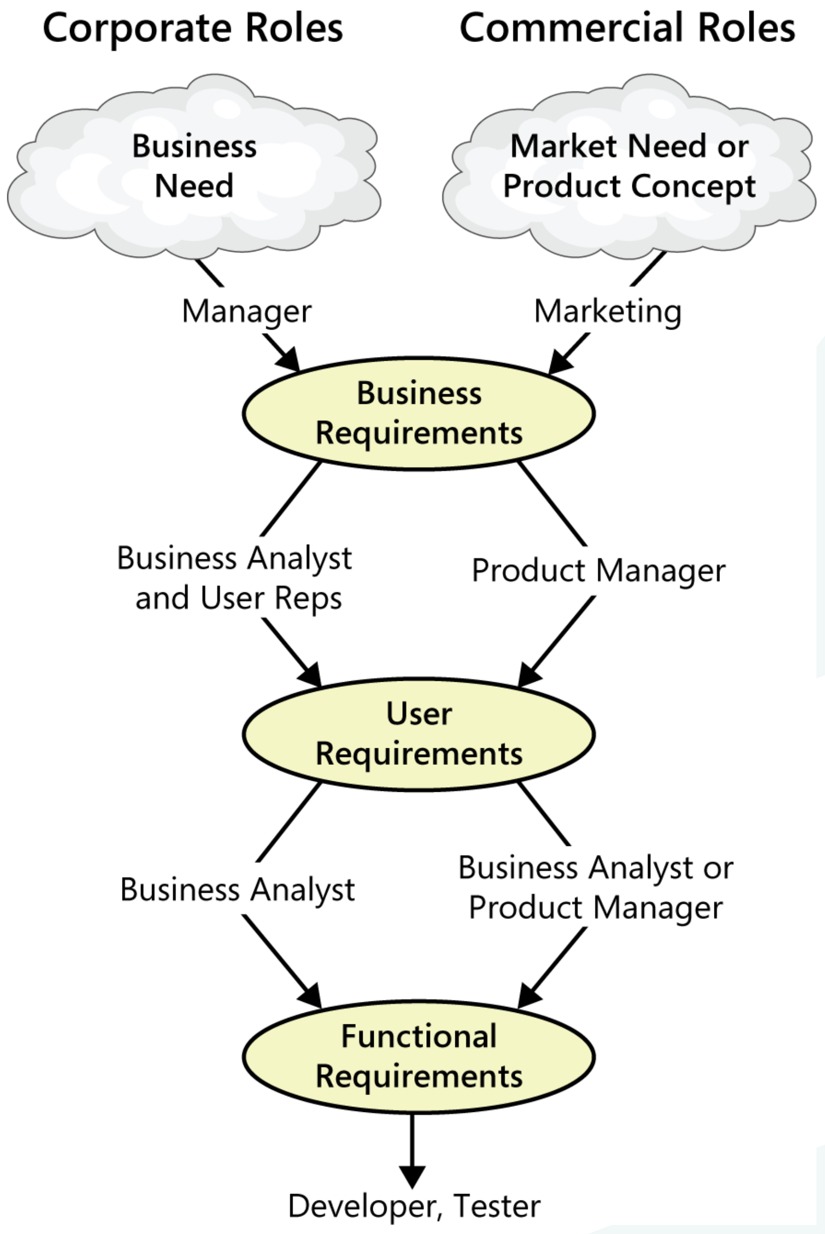
Functional requirements often are written in the form of the traditional "shall" statements: "The Passenger shall be able to print boarding passes for all flight segments for which he has checked in" or "If the Passenger's profile does not indicate a seating preference, the reservation system shall assign a seat."

The business analyst documents functional requirements in a *software requirements specification* (SRS) document, which describes as fully as necessary the expected behavior of the software system.
The functional requirements are evident: they describe the observable behavior of the system in different situations and conditions.

THREE LEVEL OF FUNCTIONAL REQUIREMENTS

How various stakeholders might participate in eliciting the three levels of requirements.

Alignment among the three levels is crucial for the project success.



SYSTEM REQUIREMENTS

- Describe the requirements for a product that is composed of multiple components or subsystems.
- A system can be all software or it can include both software and hardware subsystems.
- People and processes are part of a system, too.
- Often system requirements refers to the technical and hardware requirements in order to ensure the product works flawless.
- Or they can describe the behavior of the other elements of the system, so that the product will work and bring business value

Example of a “system” is the cashier’s workstation in a supermarket.
It could contain:

- a bar code scanner integrated with a scale,
- a hand-held bar code scanner
- a keyboard
- a display
- a cash drawer
- a card reader and PIN pad for your loyalty card and credit or debit card
- a change dispenser
- up to three printers for your purchase receipt, credit card receipt

These hardware devices are all interacting under software control.
The requirements for the system or product as a whole will describe how will all those interact and what should interface of each entity provide.

NON-FUNCTIONAL REQUIREMENTS

- Non-functional requirements can make or break the success of your product. There are lots of examples when the product failed even if all functionality were built by requirements and were serving the business objectives.

- They could serve as the origin of many functional requirements and can also drive significant architectural and design decisions.

- It’s far more costly to re-architect a completed system to achieve essential quality goals than to design for them at the outset.

- Some additional work on security at development time might avoid a lot of cost and user inconvenience.

- Non-functional requirements specify not **what** the system does, but rather **how well** it does those things.

ASPECTS OF NON-FUNCTIONAL REQUIREMENTS

- Non-functional requirements are also **known as**:
  - quality attributes
  - quality factors
  - quality of service requirements
  - constraints, and the
  - "–ilities"
- Non-functional requirements describe the **product's characteristics** in various dimensions:
  - performance
  - safety
  - availability
  - security
  - usability

- Some quality attributes are important to **users**, other are important to **developers** and maintainers.
- BAs must work with all groups of stakeholders to elicit all non-functional requirements
- **External interfaces** with other entities are also described in non-functional requirements:
  - user-software relationship
  - connections to other software systems
  - hardware components
  - communication interfaces
- Design and implementation **constraints** impose restrictions on the options available to the developer during construction of the product.

- Other class of non-functional requirements address the **environment** in which the system operates:
  - platform to hold the product
  - portability on different hard and software environment
  - compatibility with various systems
  - compliance requirements
  - regulatory requirements
  - certification requirements
- Also related to environment are **localisation** requirements that must take into account:
  - cultures,
  - languages,
  - laws,
  - currencies,
  - terminology,
  - spelling

DIFFICULTIES WITH QUALITY ATTRIBUTES

- Customers generally can not present their quality expectations explicitly
- Lack of awareness in the need of their specification
- Lack of understanding of the future product characteristics
- Problems with categorisation and relevance proving of the quality requirements
- Issues with signing-off the quality attributes because:
    - It’s hard to clearly specify the quality attributes
    - It’s hard to set some KPIs to evaluate them
    - There isn’t any precise specification of the quality
    - Sometimes there is no possibility of evaluating the quality but the human feelings

MODELS OF SOFTWARE QUALITY

MODELS OF SOFTWARE QUALITY

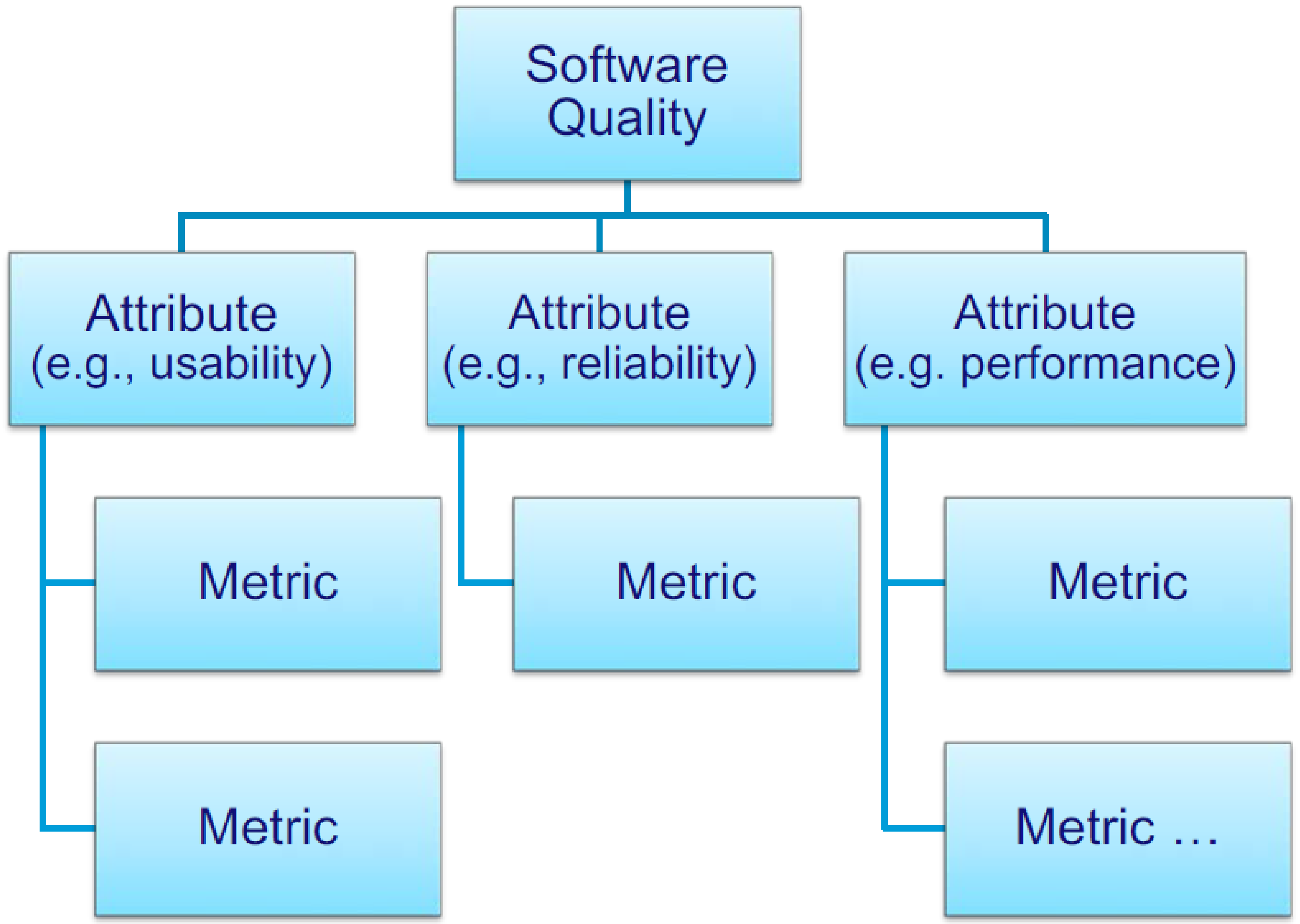
Some well-known models:

- Hierarchical general model of software quality
- Multi-level roadmap non-functional taxonomy
- McCall’s model
- Boehm’s model
- ISO/IEC 25010:2011 software quality model

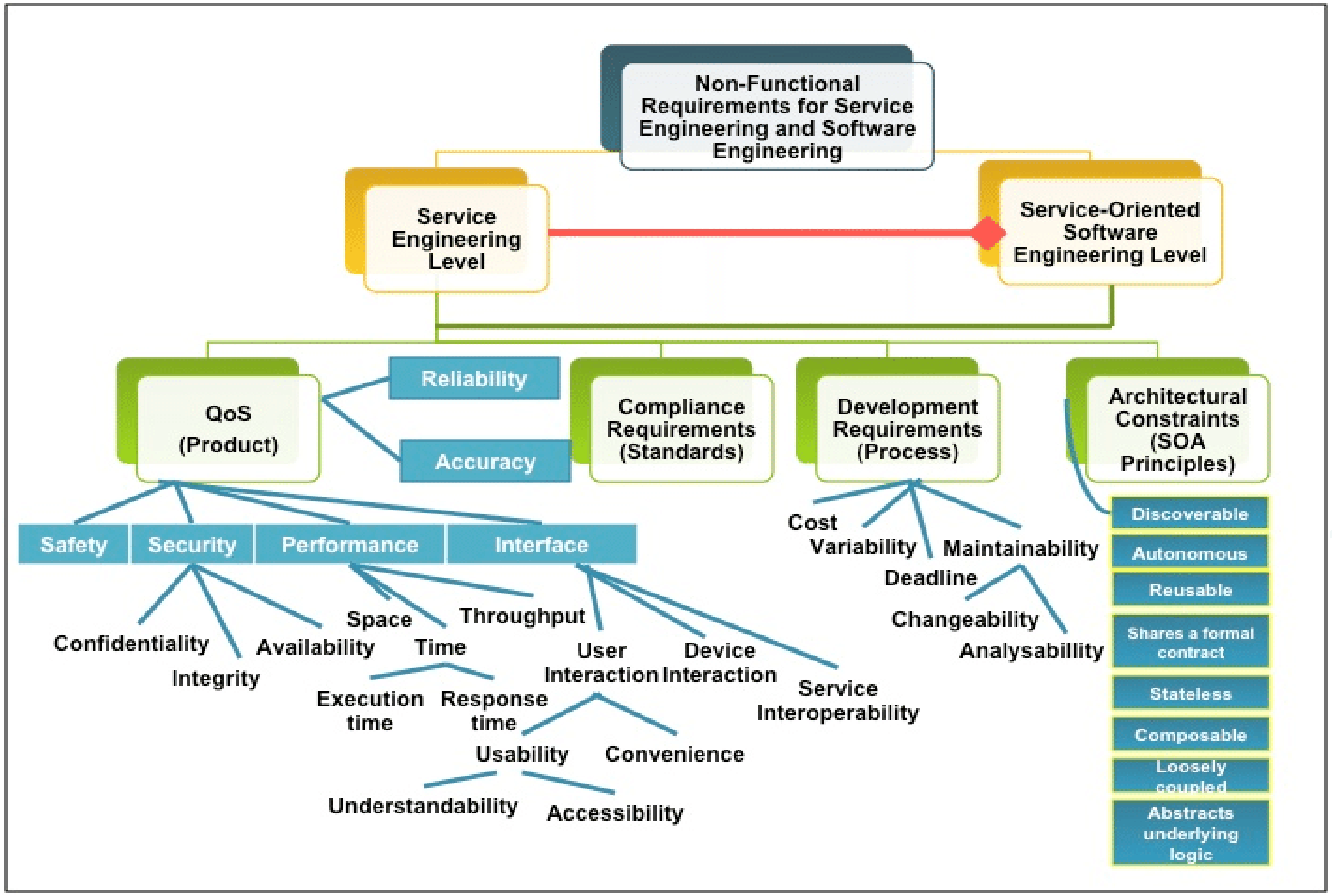
Lists of quality attributes:

- Wieger’s list
- Wikipedia’s list https://en.wikipedia.org/wiki/List_of_system_quality_attributes
- Volere Requirements Specification Template – will be discussed at Requirements specification course

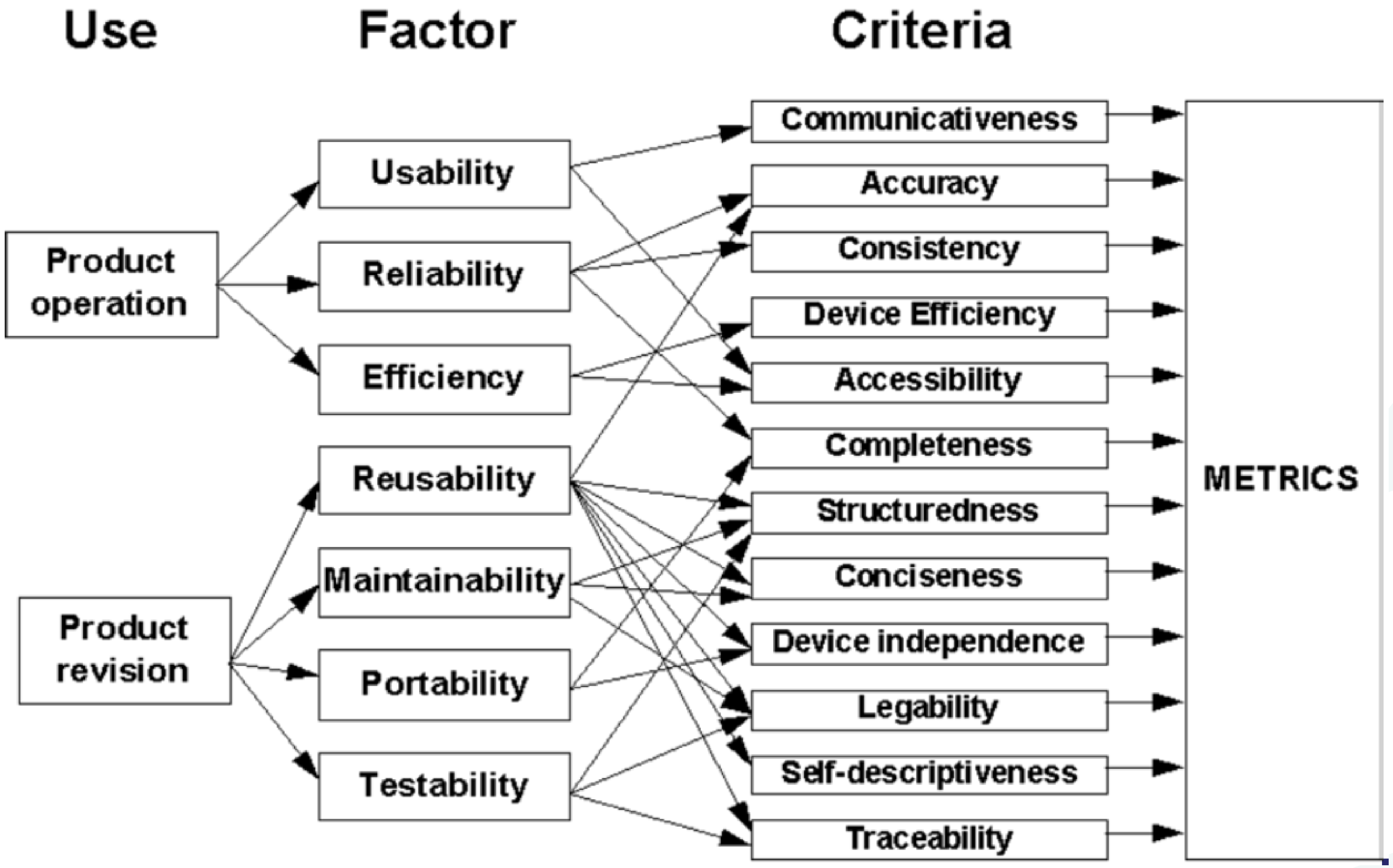
HIERARCHICAL MODEL OF SOFTWARE QUALITY


Software Quality
Attribute (e.g., usability)
Attribute (e.g., reliability)
Attribute (e.g. performance)
Metric
Metric
Metric
Metric
Metric
Metric …

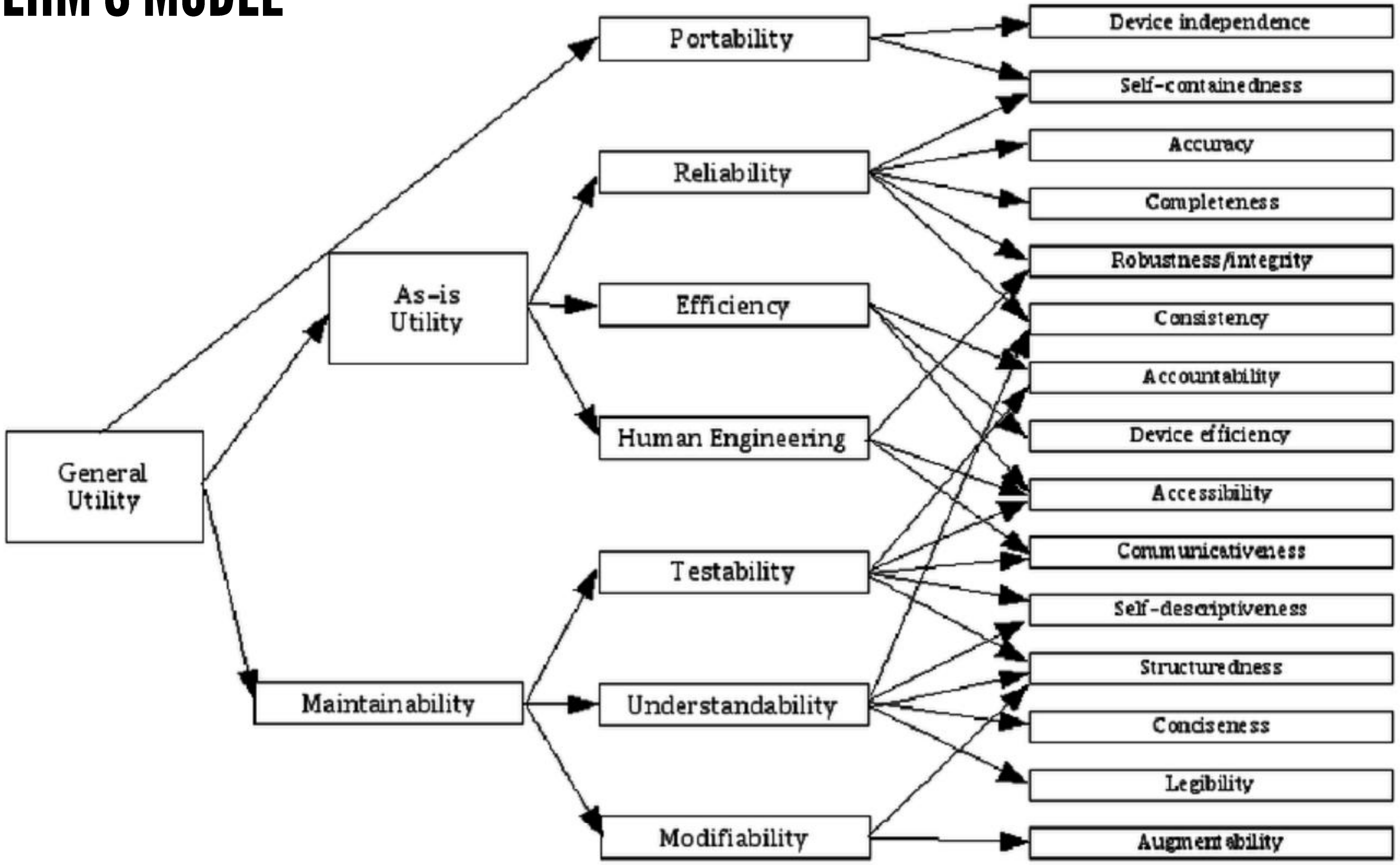
MULTI LEVEL ROADMAP TAXONOMY



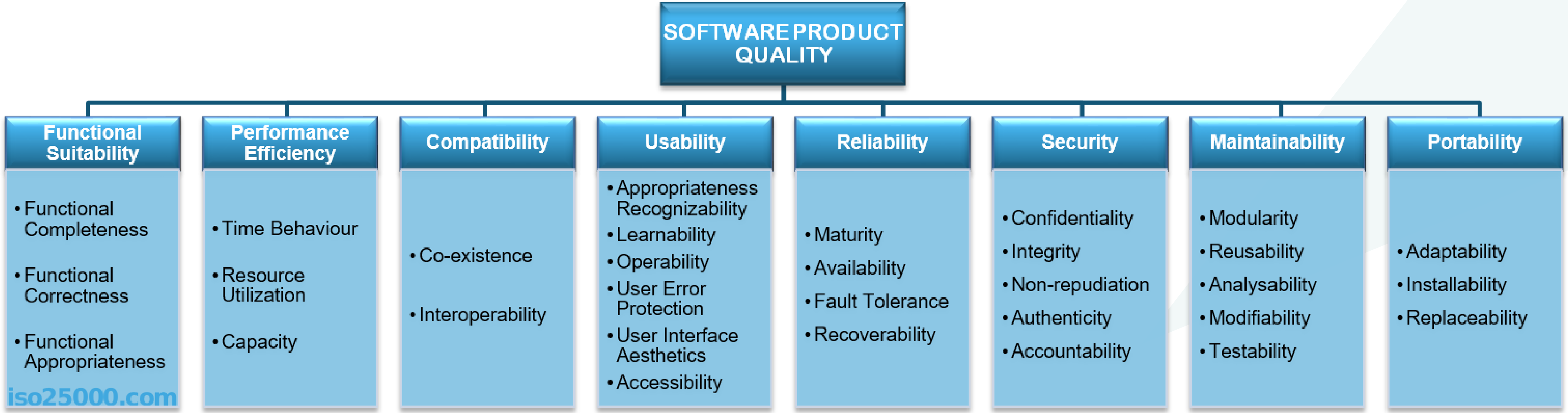
MCCALL'S MODEL



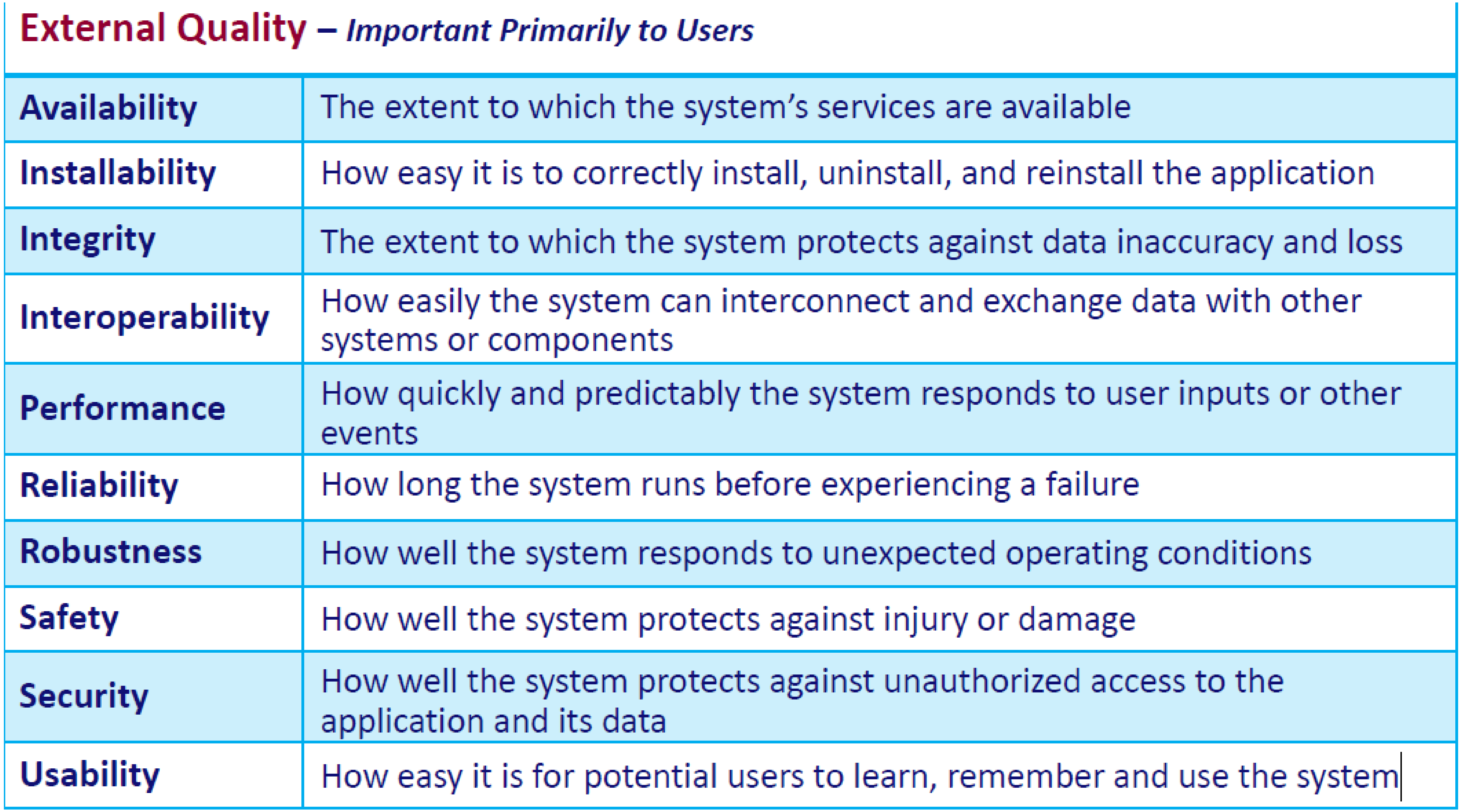
BOEHM'S MODEL



ISO/IEC 25010 SOFTWARE QUALITY CHARACTERISTICS

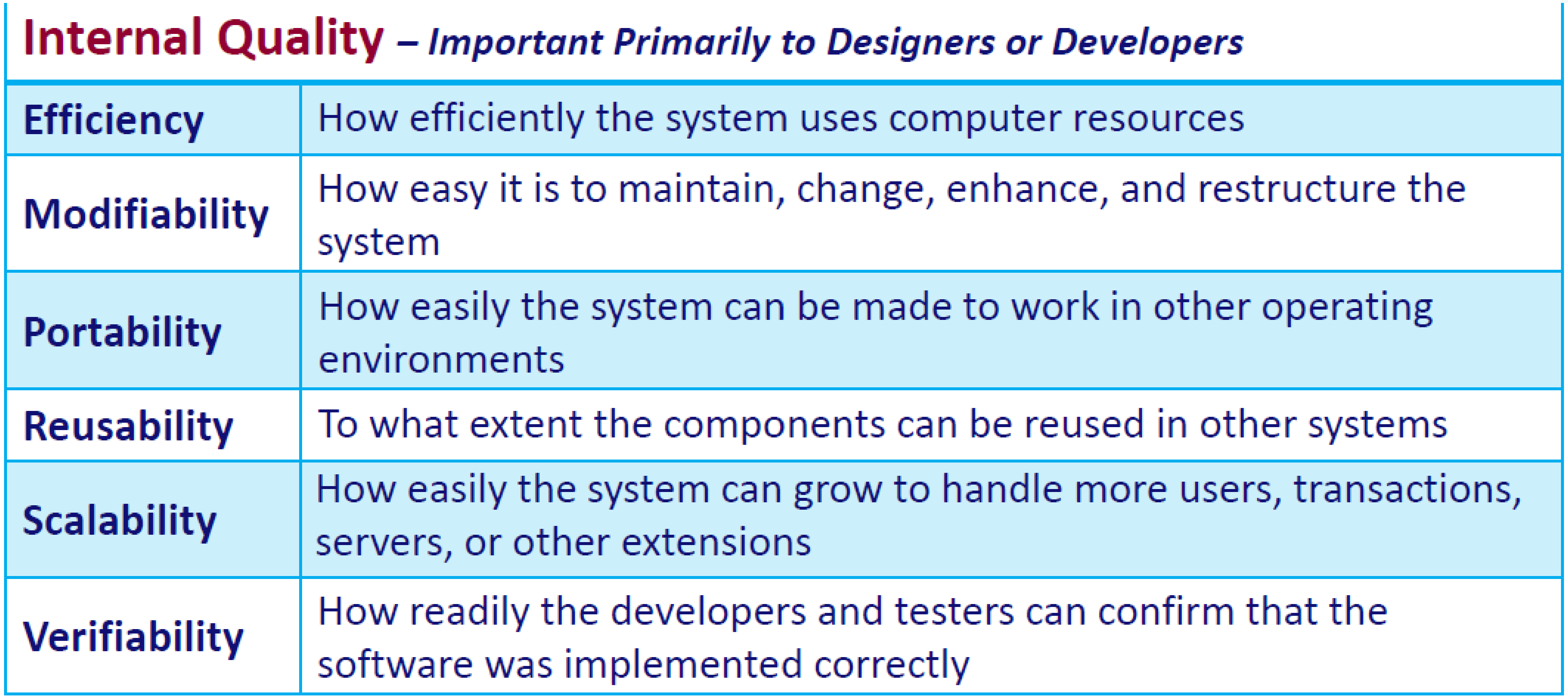


WIEGER'S LIST



External Quality – *Important Primarily to Users*	
Availability	The extent to which the system's services are available
Installability	How easy it is to correctly install, uninstall, and reinstall the application
Integrity	The extent to which the system protects against data inaccuracy and loss
Interoperability	How easily the system can interconnect and exchange data with other systems or components
Performance	How quickly and predictably the system responds to user inputs or other events
Reliability	How long the system runs before experiencing a failure
Robustness	How well the system responds to unexpected operating conditions
Safety	How well the system protects against injury or damage
Security	How well the system protects against unauthorized access to the application and its data
Usability	How easy it is for potential users to learn, remember and use the system

WIEGER’S LIST



Internal Quality – ***Important Primarily to Designers or Developers***

Efficiency	How efficiently the system uses computer resources
Modifiability	How easy it is to maintain, change, enhance, and restructure the system
Portability	How easily the system can be made to work in other operating environments
Reusability	To what extent the components can be reused in other systems
Scalability	How easily the system can grow to handle more users, transactions, servers, or other extensions
Verifiability	How readily the developers and testers can confirm that the software was implemented correctly

QUALITY ATTRIBUTES PER TYPE OF SOFTWARE

Software/System Type	Quality Attributes that are of particular importance
Corporate Applications	Availability, performance, scalability, security, usability, integrity, interoperability
Desktop and Mobile Applications	Usability, performance, security
Web/Internet Apps	Usability, reliability, security, availability, scalability, maintainability
Embedded Systems (or Real-time systems) *(e.g., In-Car system, Washing machine system, etc.)*	Performance, efficiency, reliability, robustness, safety, security, usability

STEPS IN EXPLORING NON-FUNCTIONAL REQUIREMENTS

QUALITY ATTRIBUTES PER TYPE OF SOFTWARE

1. Start with a broad taxonomy

- Wieger's
- Multi-level
- McCall's
- ISO/IEC 25010
- Wikipedia,…

2. Reduce the list

- Engage stakeholders to assess which of the attributes are likely to be important
- Elicit other needs

3. Prioritize the attributes

- Select the most important from the important
- Possibly further reduce the list

4. Elicit specific expectations for each attribute

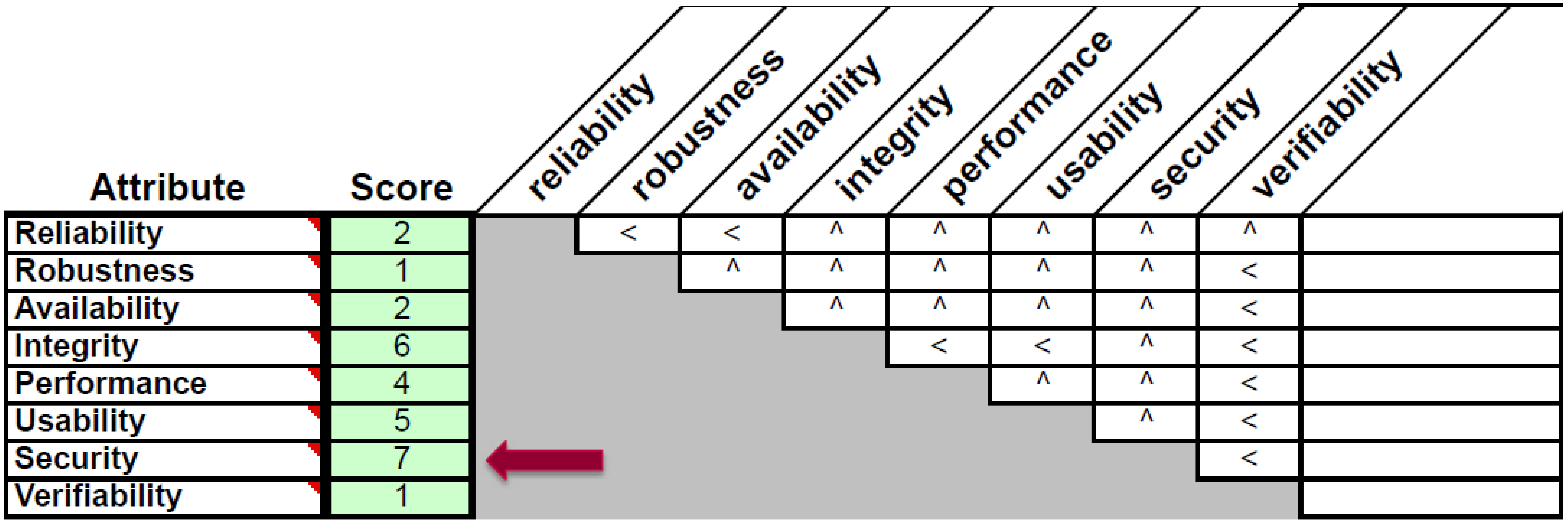
- Set measures for each attribute
- Define expectations per attribute

5. Specify well-structured quality requirements

A spreadsheet that could help you with the analysis: http://www.clarrus.com/resources/articles/software-quality-attributes/

PRIORITISE THE ATTRIBUTES

- Perform a pairwise comparison and identify which is most important in each pair
- If the left side attribute is more important, identify with a '<' character,
- otherwise use a '^' character
- Score = Sum of all 'arrows' pointing to the attribute



Attribute	Score	reliability	robustness	availability	integrity	performance	usability	security	verifiability	
Reliability	2		<	<	^	^	^	^	^	
Robustness	1			^	^	^	^	^	<	
Availability	2				^	^	^	^	<	
Integrity	6					<	<	^	<	
Performance	4						^	^	<	
Usability	5							^	<	
Security	7								<	
Verifiability	1									

ELICIT SPECIFIC EXPECTATIONS FOR EACH ATTRIBUTE

Users won’t know how to answer questions such as:
“What are your interoperability requirements?”
“How reliable your system should be?”
Sample questions during **elicitation**:

- What would be a reasonable or acceptable response time for a typical query performed by the user?
- What would users consider an unacceptable response time for a typical query?
- How many simultaneous users are you expecting on average?
- What is the maximum number of simultaneous users that you would anticipate?
- What times of the day, week, month or year have much heavier usage than usual?

SPECIFY WELL-STRUCTURED NON-FUNCTIONAL REQUIREMENTS

- Quality requirement need to be measurable to establish a precise agreement on expectations.
- If it is not measurable, there is no point in specifying it
- If you are not able to determine if you have achieved a desired goal, it can be a **not verifiable** requirement
- The aim is to make the most or all requirements verifiable

About the specification of requirements we’ll discuss at the Requirements Specification courses

QUALITY ATTRIBUTES DESCRIBED

USABILITY – A BIG TOPIC ABOUT QUALITY

- How easy it is for potential users to learn, remember and use the system. Also referred to as "*ease of use", "user-friendliness",* or even *"human engineering"*
- It is an important topic and contains:
    - Ease of use, ease of learning
    - Memorability
    - Error avoidance, handling, and recovery
    - Efficiency of interactions
    - Accessibility
    - Ergonomics
- Many of these are not strictly verifiable

USABILITY – THINGS TO CONSIDER FOR METRICS

- Success rate – percentage of tasks that users complete correctly
- The time a task requires – the average time needed for a specific type of user to complete a particular task correctly.
- How many transactions the user can complete correctly in a given time period.
- What percentage of a set of tasks the user can complete correctly without needing help.
- Error rate – how many errors the user makes when completing a task.
- How many tries it takes the user to accomplish a particular task, like finding a specific function buried somewhere in the menus.
- The number of interactions (mouse clicks, keystrokes, touch-screen gestures) required to get to a piece of information or to accomplish a task.
- User’s subjective level of satisfaction – comparable between versions/releases/similar tools

USABILITY VS USER INTERFACE REQUIREMENTS

User interface (UI) requirements involve mainly:

- Conventions and standards that have been developed for the “human-computer interface”
- Look and feel for various Operational Systems
- Compliance to application standards
- Compliance to company branding standards
- Requirements related to various groups of users

Following standards and conventions usually help to increase **usability**:
E.g., using autocompletion/autocorrection, tooltips, menu options, similarity to other familiar systems, wizards, etc.

USABILITY – EXAMPLES

Ex1: After one hour training, a user shall be able to complete 20 data entry with an error rate less than 1 in 20, in no more than 30 minutes.

Ex 2: A librarian user who has never used the CCC Library System before shall be able to catalogue a book correctly with no more than 30 minutes of orientation.

Ex3: All functions on the File menu shall have shortcut keys defined that use the Control key pressed simultaneously with one other key. The X, Y, Z, … file menu commands shall use the same shortcut keys that MS Office Word 2016 uses.

PERFORMANCE

- Performance is an external quality attribute.
- It is closely related to the internal quality attribute of efficiency.
- How quickly and predictably the system responds to user inputs or other events.

Ex. 1: The interpreter component shall parse at least 5000 error-free statements per minute.
Ex. 2: 95% of the transactions shall be processed in less than 1 second.
Ex. 3: The application should work without response time degradation with a load of 100k concurrent users.
Ex. 4: Webpages shall fully download in an average of 3 seconds or less over a 100 megabits/second Internet connection in a standard office PC (standard office PC shall be described in the requirement)

INTEROPERABILITY

- How easily the system can interconnect and exchange data with other systems or components.
- Can be related to connection to various external devices
- Or to the way and easiness the system exchange data with other systems or devices
- Or to the level of acceptance of various data standards or forms

Ex: The system shall be able to import a valid BPMN v2 process model file from Bizagi (version 2.8 or 2.7) and Signavio (version 9.0 or 9.2)

INTEGRITY

- The extent to which the system protects itself against data inaccuracy and loss.
- How well the system is able to work with and manage corrupted data

Ex.1: After performing a file backup, the system shall verify the backup copy against the original and report any discrepancies.

Ex.2: The system shall confirm that an encoded structure of a BPMNv2 process model file imported from a third-party process modeling tool represents a valid structure.

RELIABILITY

- The ability to continue to operate under predefined positive or negative conditions.
- Unlike physical objects, software does not wear out, so their reliability shall not decrease with time
- Robustness and availability are closely related to reliability.
- Failures occur due to:
  - improper inputs
  - errors in the software code
  - components that are not available
  - hardware failures, …

RELIABILITY – THINGS TO CONSIDER FOR MEASURES

It is very difficult to quantify the Reliability

- Mean Time Between Failures (MTBF) – The average length of time the system runs before failing
- Rate of failure occurrence – Percentage of failed requests
- Percentage of operations that are completed correctly
- How long the system runs before experiencing a failure.

Some questions to ask during Elicitation:

- How would you judge whether this system was reliable enough?
- What would you consider to be a critical failure, as opposed to nuisance?
- Under what conditions could a failure have severe repercussions on your business operations?
- No one likes to see a system crash, but are there certain parts of the system that absolutely have to be super-reliable?

RELIABILITY – STATING THE REQUIREMENTS

For some systems reliability is crucially important

- Flight management systems
- Air traffic control systems
- Magnetic resonance systems

Such systems should also be designed for high verifiability, so that it becomes easier to detect defects.

Ex.1: The train acceleration control software shall have a mean time between failures of the order of 109 hours.

Ex. 2: No more than 5 experimental runs out of 1000 can be lost because of software failures.

Ex. 3: The mean-time between failures of the card reader component shall be at least 90 days.

AVAILABILITY

- The extent to which the system's services are available.
- Ask stakeholders what percentage of up time is really needed and whether there are any times for which availability is imperative to meet business or safety objectives.

Ex.1: The system must be 100% operational 99.9% of the calendar time during its first year of operation.
Ex.2: The system shall have less than 1hr downtime per three months.
Ex.3: The system shall be at least 99.5 percent available on weekdays between 6:00 a.m. and midnight local time, and at least 99.95 percent available on weekdays between 4:00 p.m. and 6:00 p.m. local time.

ROBUSTNESS

- How well the system responds to unexpected operating conditions.
- Or Software's capability of maintaining a specified performance level in cases of operation faults
- Also known as Fault tolerance

Metrics to quantify:

- Time to restart after failure
- Percentage of events causing failure
- Probability of data corruption on failure

Ex.1: If the editor fails before the user saves the file, the editor shall be able to recover all changes made in the file being edited up to one minute prior to the failure the next time the same user starts the program.

SECURITY

How well the system protects against unauthorized access to the application and its data.

Ex. 1: The system shall lock a user’s account after four consecutive unsuccessful attempts within a period of five minutes.
Ex. 2: The system shall log all attempts to access secure data by users having insufficient privilege levels.
Ex. 3: A user shall have to change the temporary password assigned by the security administrator to a previously unused password immediately following the first successful logon with the
temporary password.

SCALABILITY

- Ability of the system to handle load increases without decreasing performance, or the possibility to rapidly increase the load.
- How easily the system can grow to handle more users, transactions, servers, or other extensions.

Ex. 1: The product shall be capable of processing the existing 100,000 customers. This number is expected to grow to 500,000 within three years.
Ex. 2: The product shall be able to process 50,000 transactions an hour within two years of its launch.

QUALITY TRADE-OFFS

Minus - increasing the attribute in that row adversely affects the attribute in the column. Vice versa is not true.

Blank cell - attribute in the row has little impact on the attribute in the column.

Plus - increasing the attribute in that row positively affects the attribute in the column.

	reliability	robustness	availability	integrity	flexibility	usability	interoperability	efficiency	testability	maintainability	reusability	portability
reliability		+	+		+	+		-	+	+		
robustness	+		+			+		-				
availability	+	+										
integrity						-	-	-	-		-	
flexibility	+			-				-	+	+		+
usability		+						-	-			
interoperability				-	+			-				+
efficiency	-	-			-	-	-		-	-		-
testability	+		+		+	+		-		+		
maintainability	+		+		+			-	+			
reusability	-			-	+		+	-	+	+		+
portability					+	-	+	-	+	-	+	

CONSTRAINTS

Restrictions applied to the actions that the system or its users are allowed to perform.
Certain actions **must** or **must not** or **may not** be performed, or that **only** certain people or roles can perform particular actions.
Constraints are often derived into functional requirements

Constraints could have different origins
Organizational policies
Ex.1: A loan applicant who is less than 18 years old must have a parent or a legal guardian as cosigner on the loan.
Ex.2: A library patron may have a maximum of 10 items on hold at any time.
Ex.3: Insurance correspondence may not display more than four digits of the policyholder’s Social Security number.

Government regulations

Ex.1: All software applications must comply with government regulations for usage by visually impaired persons.

Ex.2: Airline pilots must receive at least 8 continuous hours of rest in every 24-hour period.

Ex.3: Individual federal income tax returns must be postmarked by midnight on the first business day after April 14 unless an extension has been granted.

Industry standards

Ex.1: Mortgage loan applicants must satisfy the Federal Housing Authority qualification standards.

Ex.2: Web applications may not contain any HTML tags or attributes that are deprecated according to the HTML 5 standard.

THANK YOU

AMDARIS