Software Testing

Software quality

- Software quality is:
- 1. The degree to which a system, component, or process meets specified requirements.
- 2. The degree to which a system, component, or process meets customer or user needs or expectations.

An effective quality process must focus on

- Paying much attention to customer’s requirements
  - Even a bug-free program is a failure if it doesn't solve the customer's problem.
- Making efforts to continuously improve quality
  - It involves analyzing test results from the previous sprint to improve the testing strategy for the next one.
- Integrating measurement processes with product design and development
  - You cannot manage what you cannot measure. An effective process uses data to drive decisions rather than "gut feelings.“
  - Quality must be measured during design and development, not just after
- Pushing the quality concept down to the lowest level of the organization
  - Quality is not the sole responsibility of the "QA Department." It is a cultural mindset that must be shared by every developer, designer, and product manager.
  - This encourages **Total Quality Management (TQM)**.

An effective quality process must focus on

- Developing a system-level perspective with an emphasis on methodology and process
  - A "system-level perspective" means looking at the software as a whole rather than as isolated functions. This requires a standardized methodology (like Agile, DevOps, or V-Model).
- Eliminating waste through continuous improvement
  - Derived from **Lean Manufacturing**, eliminating waste (Muda) in software means removing any activity that does not add value to the final product.

Software Quality

- ISO 9126
  - The ISO 9126 quality model was developed by an expert group under the aegis of the International Organization for Standardization (ISO).
  - The document ISO 9126 defines six broad, independent categories of quality characteristics
  - functionality, reliability, usability, efficiency, maintainability, and portability
- CMM
  - In the CMM framework, a development process is evaluated on a scale of 1–5, commonly known as level 1 through level 5. For example, level1 is called the initial level, whereas level 5—optimized—is the highest level of process maturity.

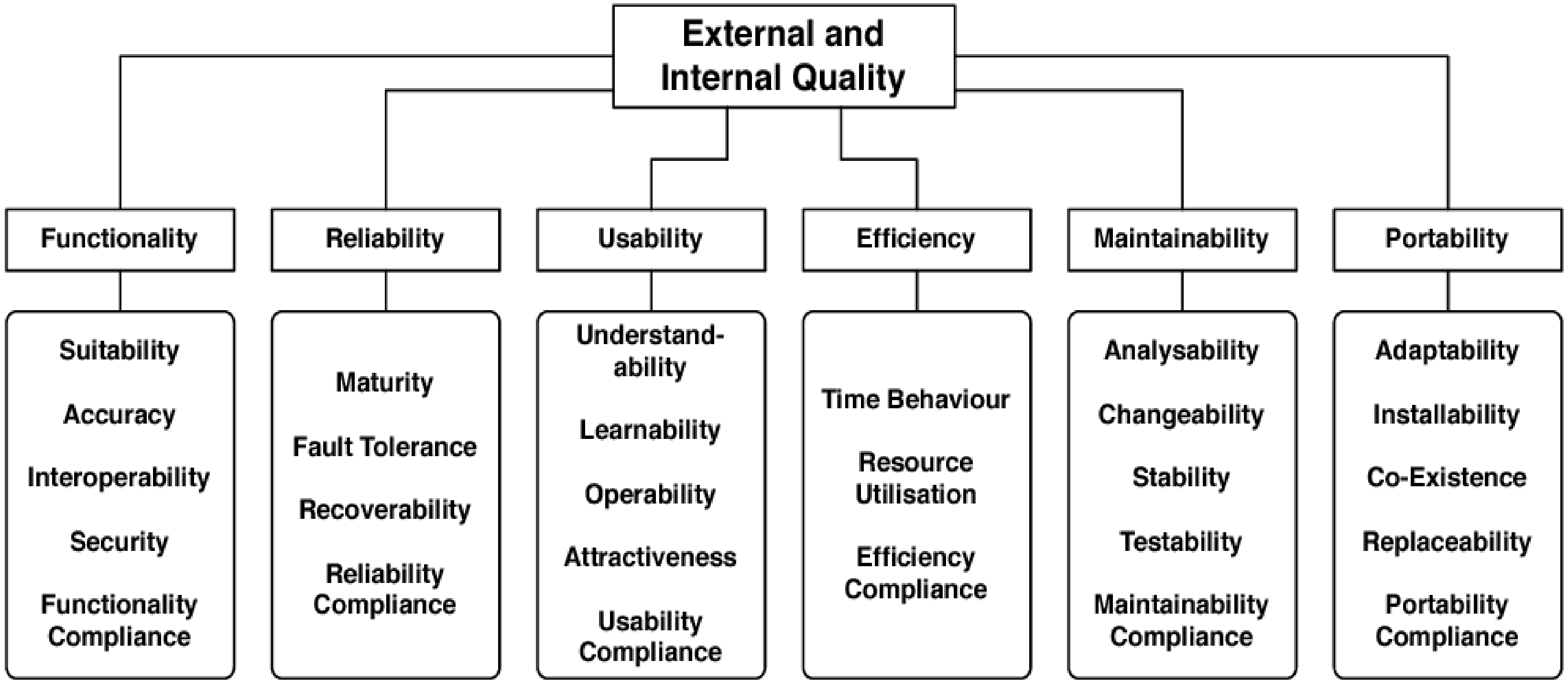


Figure 1: The ISO/IEC 9126-1 Model for Internal and External Quality

Functionality

- This attribute refers to the software's ability to provide the functions that meet stated and implied needs. In simple terms, it's about whether the software does what it's supposed to do. A software product's functionality is determined by factors like its **suitability, accuracy, interoperability, security, and compliance** with regulations.
- Example: An e-commerce website should have the functionality to allow users to add items to a shopping cart, process payments securely, and track orders. Without these core functions, the website fails its primary purpose.

Reliability

- This characteristic measures the software's ability to maintain its performance level under specific conditions for a defined period. A reliable system is one that you can count on to be available and to perform correctly without frequent failures. Sub-characteristics include **maturity** (the frequency of failures), **fault tolerance** (the ability to operate even when a component fails), and **recoverability** (the ability to recover data and return to a specified state after a failure).
- Example: An online banking system must be highly reliable. If the network goes down for a few seconds, it should be able to recover and continue the transaction without losing data or failing.

Usability

- Usability is about how easy it is for users to understand, learn, and operate the software. A usable product is intuitive and requires minimal effort from the user. Its sub-characteristics include **understandability**, **learnability**, **operability**, and **attractiveness**.
- **Example**: A mobile app with a clean and intuitive interface, clear icons, and straightforward navigation is highly usable. A user should be able to figure out how to complete a task, such as ordering food, without needing a manual.

Efficiency

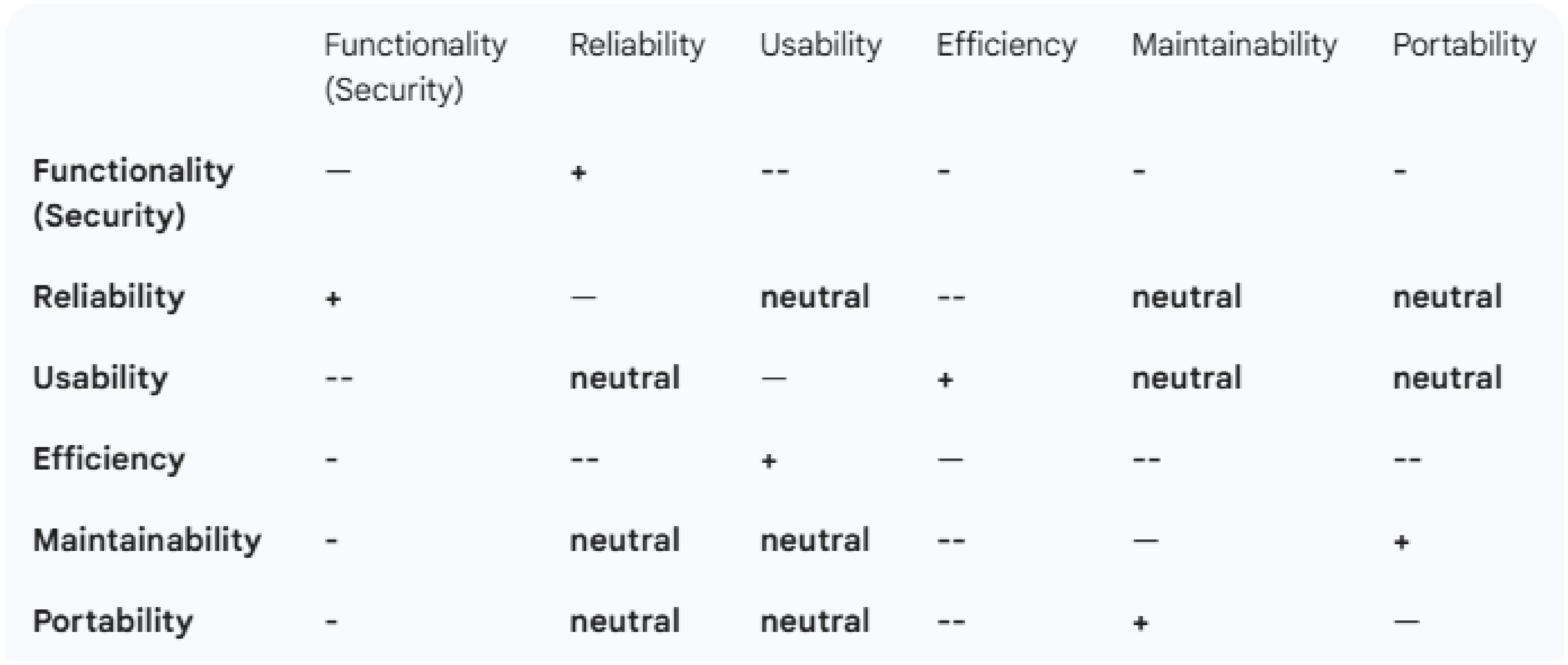
- This refers to the relationship between the software's performance level and the resources it uses. An efficient system performs its functions using an appropriate amount of resources, like processing time, memory, or disk space. This is measured by **time behavior** (response time and throughput) and **resource utilization**.
- **Example**: A video editing software that renders a video quickly without consuming excessive CPU and memory is considered efficient.

Maintainability

- Maintainability is the ease with which a software product can be modified, either to correct defects, add new features, or adapt to a changing environment. This is a critical factor for long-term software health. Key sub-characteristics include **analyzability** (the effort to diagnose a problem), **changeability** (the effort to implement a change), **stability** (the risk of unexpected effects from modifications), and **testability** (the effort to test a modified product)
- **Example**: A well-structured software with modular code and clear documentation is easy to maintain. A developer can quickly identify where a bug is located and fix it without causing new issues.

Portability

- This characteristic measures the software's ability to be transferred from one environment to another. This could be moving it to a different operating system, a new hardware platform, or even just a new browser. It includes sub-characteristics like **adaptability**, **installability**, **co-existence**, and **replaceability**.
- A web application that works seamlessly on different browsers (Chrome, Firefox, Safari) and operating systems (Windows, macOS, Linux) is highly portable. It can be easily installed and run in a new environment with minimal changes.



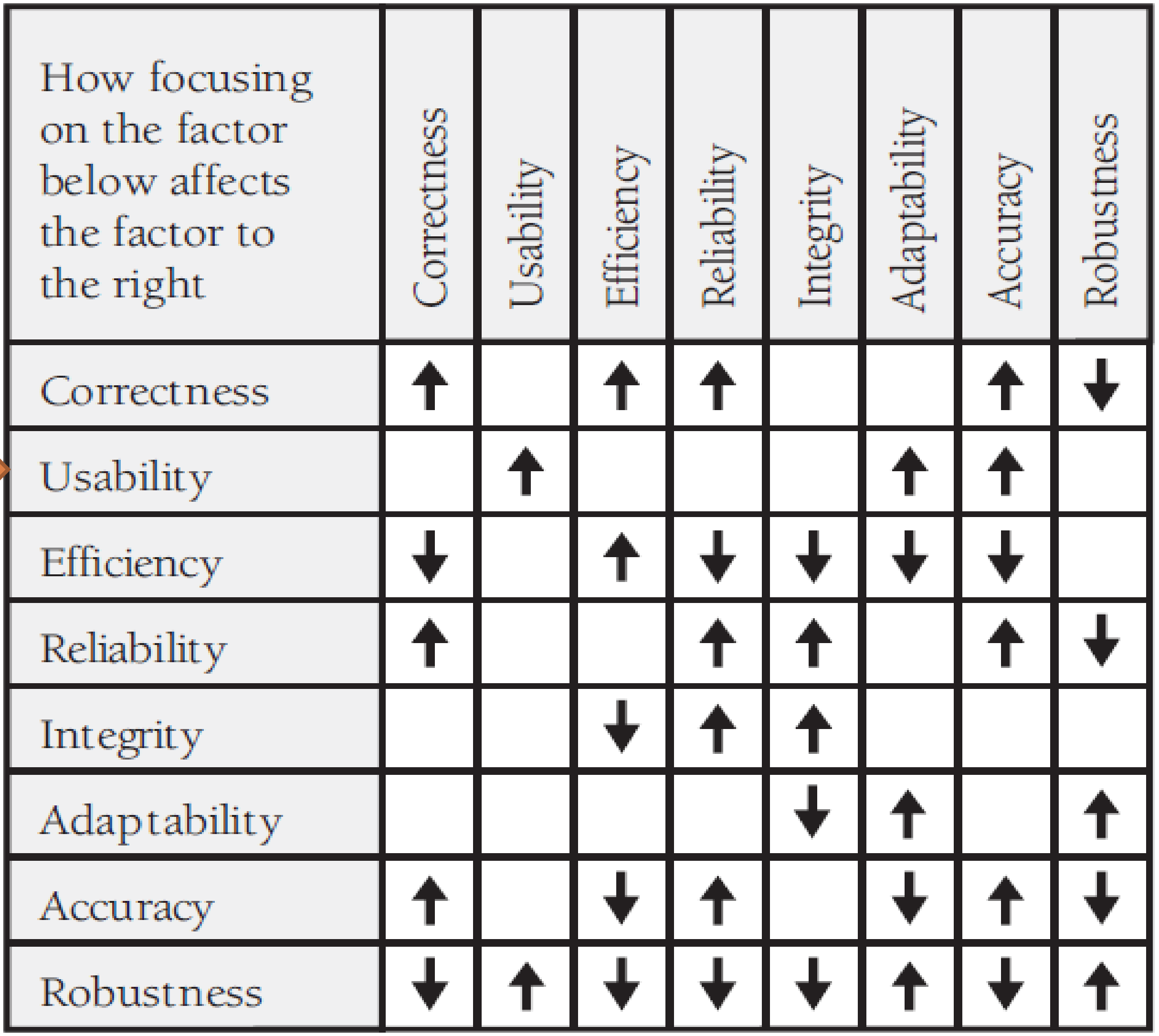
	Functionality (Security)	Reliability	Usability	Efficiency	Maintainability	Portability
Functionality (Security)	—	+	--	-	-	-
Reliability	+	—	neutral	--	neutral	neutral
Usability	--	neutral	—	+	neutral	neutral
Efficiency	-	--	+	—	--	--
Maintainability	-	neutral	neutral	--	—	+
Portability	-	neutral	neutral	--	+	—

Characteristics of Quality

- External
  - Correctness (?), Usability, Efficiency, Reliability, Integrity, Adaptability, Accuracy, Robustness
- Internal
  - Maintainability, Readability, Testability, Reusability, Portability, Flexibility, Understandability

How focusing on the factor below affects the factor to the right	Correctness	Usability	Efficiency	Reliability	Integrity	Adaptability	Accuracy	Robustness
Correctness	↑		↑	↑			↑	↓
Usability		↑				↑	↑	
Efficiency	↓		↑	↓	↓	↓	↓	
Reliability	↑			↑	↑		↑	↓
Integrity			↓	↑	↑			
Adaptability					↓	↑		↑
Accuracy	↑		↓	↑		↓	↑	↓
Robustness	↓	↑	↓	↓	↓	↑	↓	↑

Helps it	↑
Hurts it	↓



How focusing on the factor below affects the factor to the right	Correctness	Usability	Efficiency	Reliability	Integrity	Adaptability	Accuracy	Robustness
Correctness	↑		↑	↑			↑	↓
Usability		↑				↑	↑	
Efficiency	↓		↑	↓	↓	↓	↓	
Reliability	↑			↑	↑		↑	↓
Integrity			↓	↑	↑			
Adaptability					↓	↑		↑
Accuracy	↑		↓	↑		↓	↑	↓
Robustness	↓	↑	↓	↓	↓	↑	↓	↑

Helps it	↑
Hurts it	↓

How focusing on the factor below affects the factor to the right	Correctness	Usability	Efficiency	Reliability	Integrity	Adaptability	Accuracy	Robustness
Correctness	↑		↑	↑			↑	↓
Usability		↑				↑	↑	
Efficiency	↓		↑	↓	↓	↓	↓	
Reliability	↑			↑	↑		↑	↓
Integrity			↓	↑	↑			
Adaptability					↓	↑		↑
Accuracy	↑		↓	↑		↓	↑	↓
Robustness	↓	↑	↓	↓	↓	↑	↓	↑

Helps it ↑

Hurts it ↓