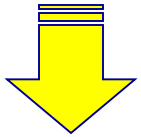
Software : Its Nature & Qualities

Fall, 2021

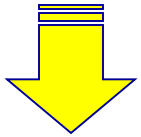
jehong@chungbuk.ac.kr

Revisit: Goal of Software Engineering

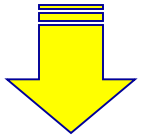
Produce Quality Software on time within budget that satisfies user's needs



Improve quality and productivity of software



Improve quality and productivity of system, product



Improve business performance

Software Qualities

Definitions

- DoD, 1985, "The degree to which the attributes of software enable it to perform its intended end use."
- ISO, 1986, "The totality of features and characteristics of a product or a service that bear on its ability to satisfy specified or implied needs."
- Kitchenham, 1986, "Fitness for needs".
  - Conformance to its specification
    : Is it a good solution?
  - Fitness for its intended purpose
    : Does it address the right problem?

⟶ Capability of a software product to conform to Users' Needs

Some Insights about Quality

Quality is not absolute

Quality is multidimensional

Quality is subject to constraints (people, money, time, tool)

Quality is about acceptable compromises

Quality criteria are not independent

Why software quality is different from other types of quality?

Software has no physical existence.

The lack of knowledge of client needs at the start.

The change of client needs over time.

The rapid rate of change in both hardware and software

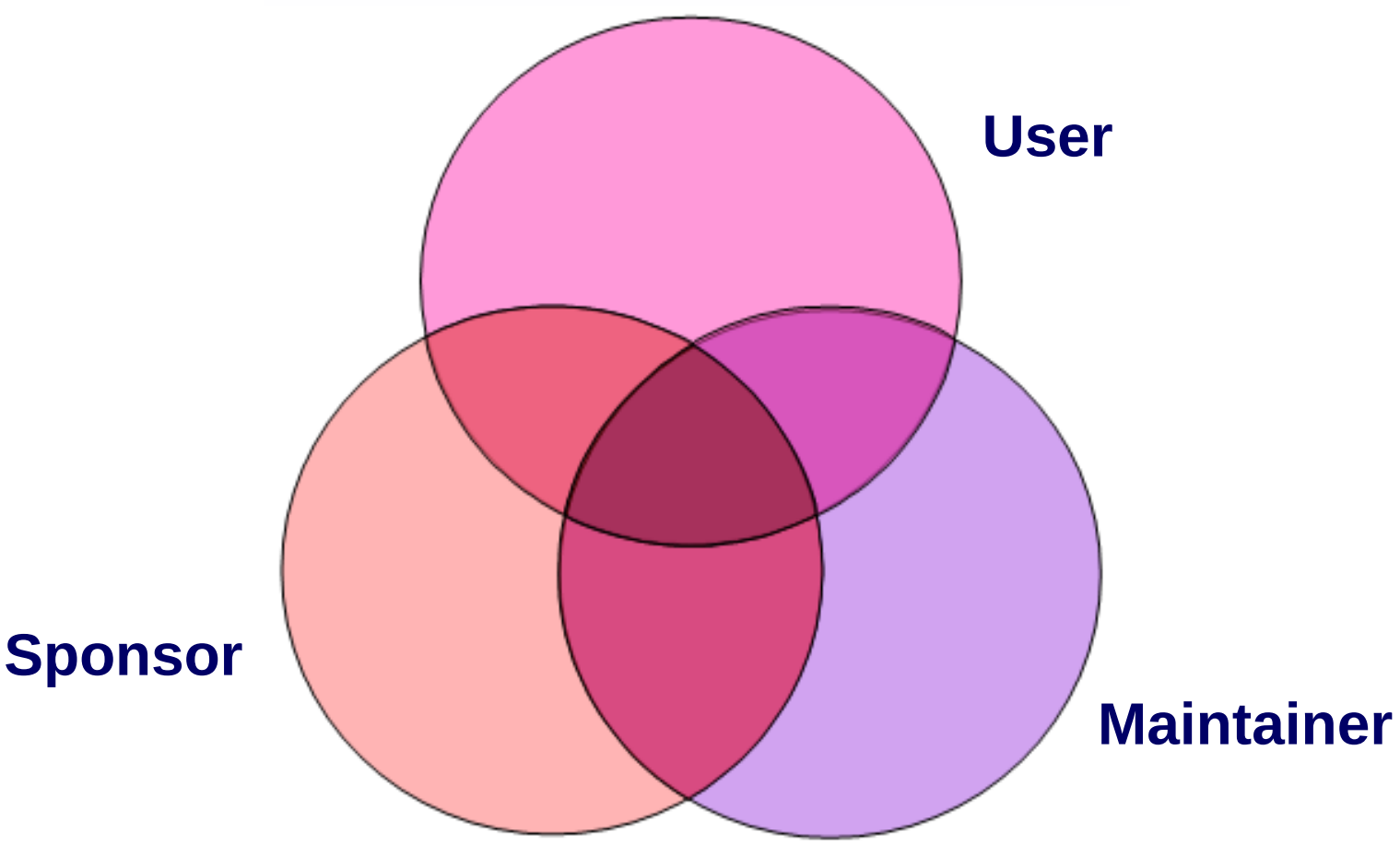
The high expectation of customers.

Quality of Software …

Which do you want?

What are the criteria for choosing the product?

Quality Factors



Qualities Classification

Quality

- External and Internal Qualities
- Product and Process Qualities

External Quality and Internal Quality

- Distinction is not sharp
- External quality: Visible to the users of the system.
- Internal quality: Concerns the developers of the system.

Product and Process Qualities

Closely related: use a process to produce the SW product.

Product quality:

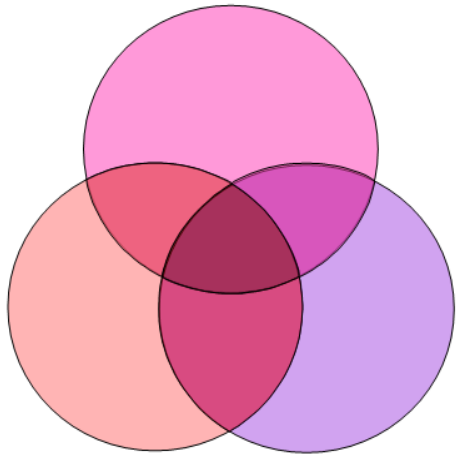
- Functionality
- Usability
- Efficiency
- Reliability, etc

Process quality

- Effectiveness of methods, tools
- Use of standards (standard compliance)
- Management, etc

Representative Qualities

Correctness, Reliability and Robustness
Performance
User Friendliness
Verifiability
Maintainability
Reusability
Portability
Understandability
Interoperability
Productivity
Timeliness
Visibility



Correctness, Reliability and Robustness

Open used interchangeably, meaning that the degree of the application's performing its functions as expected.

Correctness

Correct if the program behaves according to the specification of the functions.

Assumption :

- A specification of the system is available.
- It is possible to determine unambiguously if a program meets the specification.

Reliability

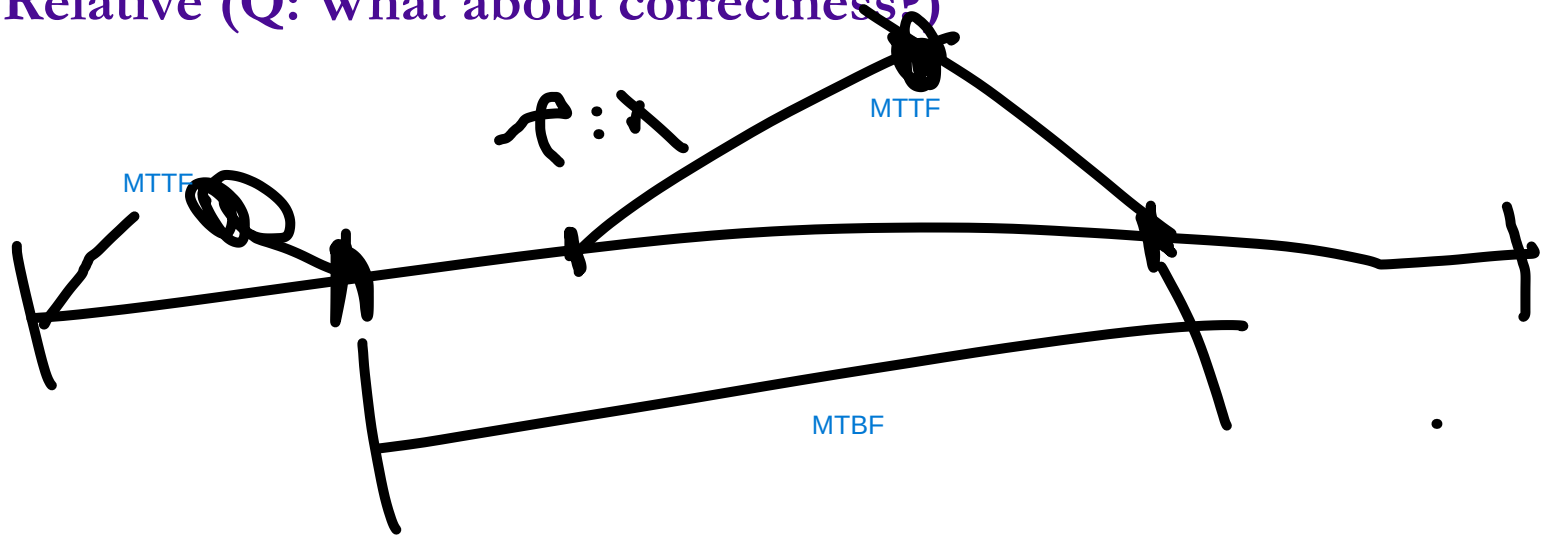
= dependability: reliable if the user can depend on it.

A measure of the frequency and criticality of product failure

Failure: An unacceptable effect or behavior under permissible operating conditions.

Can define in terms of statistical behavior: MTTF, MTBF

Relative (Q: What about correctness?)



Robustness

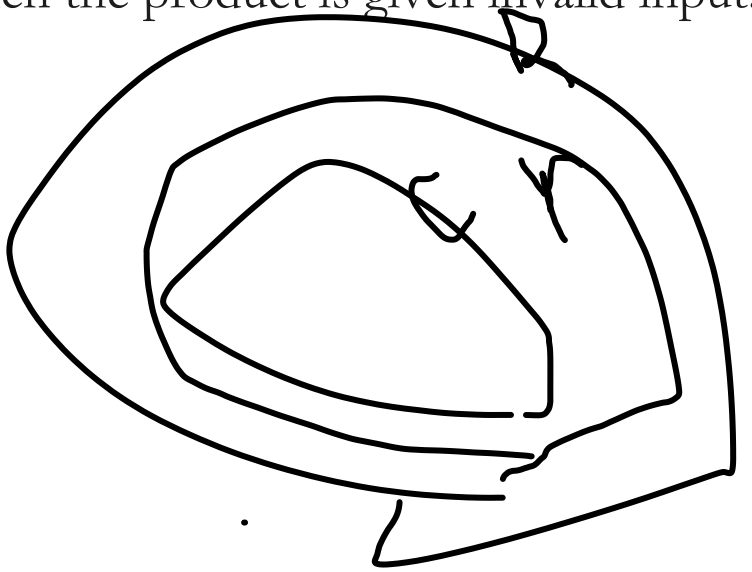
Robust if it behaves reasonably, even in circumstances that were not anticipated in the requirements specification.

A function of a number of factors such as

- The range of operating conditions.
- The possibility of unacceptable results with valid input.
- The acceptability of effects when the product is given invalid input.

Q: Relation to correctness

Q: Relation to reliability



Performance

Equates performance with efficiency(space, time).

Affects the usability of the system.

Evaluation

- Measurement (monitoring)
- Analysis
- Simulation

Q: When do we need to estimate performance?

Application of performance to process → Productivity

User Friendliness

Ease to use

Ease with which the system can be configured and adapted to the personalized environment.

Verifiability

Verifiable if its properties can be verified safely.

Performed either by formal analysis methods or by through testing.

Maintainability

Maintenance

- Corrective
- Adaptive
- Perfective
- Preventive

Software evolution (instead of maintenance)

Repairability and Evolvability

Maintainability(Cont.)

Repairability

- Repairable if a software system allows the correction of defects with a limited amount of work.
- Q: How many hours needed to fix an error in the maintenance phase?
- Right modularization promotes repairability. Q: Why?

A. Nobody Know

- Availability, serviceability in computer engineering
- Q: What the difference the repairability between software and hardware ?
- Improved through the use of proper tools: HL PL, CASE, etc.
- Q: Relation to reliability

Maintainability(Cont.)

Evolvability

- Software are modified over time
  - To provide new functions.
  - To change existing functions.
- Important in more large and complex software
- Achieved by modularization
- Decreased with each release of a new version
- Q: Thus, what we may need?
- Affect to economic, business impact
- Fostered evolvability with product family and software architecture

Reusability

Use existing components to build a new product.

Examples: Scientific libraries, MFC, eclipse Plug-in, etc

Reuse levels

- People
- Requirements
- Design
- Code

Other levels of Reuse

- Module → Function → Software → System.

Application of reuse to process

- Software methodology
- Life cycle model

Portability

Portable if it can run in different environments

- Hardware platforms
- Software platforms

Achieved by modularization

- Q: How to do this modularization ?

Understandability

An internal product quality.

Object-oriented paradigm claims ease to understand.

Enhanced by abstraction and modularity

Q: Relation to maintainability ?

Interoperability

Ability of a system to coexist and cooperate with other systems

Achieved by interface standardization

Open system concept

Productivity

A quality of the software production process in efficiency and performance

Difficult to measure:

- Simple metric: SLOC
- Functionality based metric : FP

Timeliness

Ability to deliver a product on time.

Time-to-Market challenges

Requires

- careful scheduling,
- accurate work estimation, and
- clearly specified milestones.

Achieved by incremental delivery

Visibility

Visible if all of its steps and its current status are documented.

Transparency

- the step and the status of the project are available and accessible for external examination

Allows to weigh the impact of their actions and thus guides in making decisions.

Requirement specification and design specification

Product is visible

- clearly structured as a collection of modules
- clearly understandable functions, and
- available and accurate documentation

Q: Why important visibility ?

Security

Secure if an idea implemented to protect software against malicious attack and other hacker risks

The software continues to its function correctly under such potential risks.

Necessary to provide integrity, authentication and availability.

Security vulnerability: a weakness/flaw that can actually be exploited by an attacker, which requires the flaw to be

- accessible: attacker has to be able to get at it
- exploitable: attacker has to be able to do some damage with it

Security risk, Security vulnerabilities and incident statistics, annualized loss expectancy are some of the metrics.

Safety

Software Safety

- Being Free from software hazards (IEEE Std-1228, 1994)
- Concerned with avoiding hazardous situations and alerting the correct systems if the situation becomes unsafe

Software Hazard

- SW (operational) condition that is prerequisite of accidents

Accident

- Unintended event causing death, injury, disease, environmental damage or property losses

How to measure Software Safety?

Quality Requirements in Specific Application Areas

- Information Systems
- Real-time Systems
- Distributed Systems
- Embedded Systems

Information Systems

Storage and retrieval of data

Examples: banking systems, library-cataloging systems, etc.

Qualities

- Data integrity
- Security
- Data availability
- Transaction performance
- User friendliness

Real-time Systems

Respond within a predefined and strict time periods

Examples: factory-monitoring systems, missile guidance systems, mouse-handling software

Control-oriented systems

Scheduling in OS level

- Deadline
- Priority

Qualities

- Respond time requirements (correctness criterion)
- Safety

Distributed Systems

Parallelism and Communication

- Task allocations, Partitioning

The degree of distribution

- Data
- Control
- Hardware

Examples: middleware in client/server systems, groupware, etc.

Qualities

- System availability
- Code mobility

Embedded Systems

Software is one of many components.

Has no (or limited) interface to end-user.

Examples: Airplanes, robots, microwave ovens, dishwashers, automobiles,etc.

Qualities

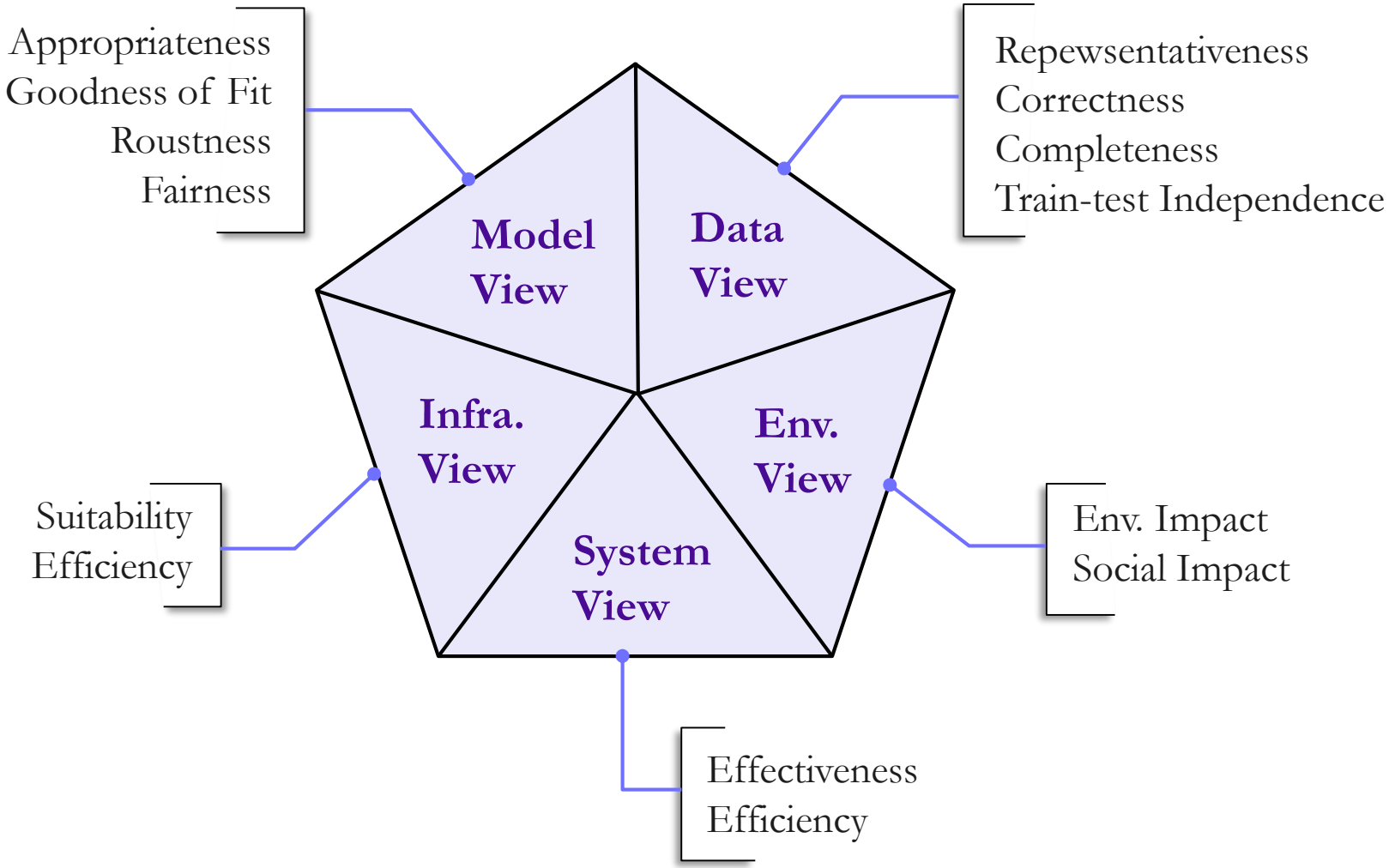
- Reliability
- Adaptability
- Memory and performance efficiency

Quality Factors for AI/ML Systems

Fundamentally different from traditional software

- The relationship between the input and the outcome of the model is only defined for a subset of the data, which leads to uncertainty in model outcomes for previously unseen data.
- Common development principles from software engineering, such as encapsulation and modularity, have to be rethought, e.g., neural networks cannot simply be cut into smaller sub-nets and reused as modules.
- Development and integration of ML components is a multi-disciplinary approach: It requires knowledge about the application domain, knowledge about how to construct ML models, and finally, knowledge about software engineering
- The algorithms executing the model play a far less significant role than the data used for training and testing

Quality Factors for AI/ML Systems



Standards for Software Quality

There are so many quality factors

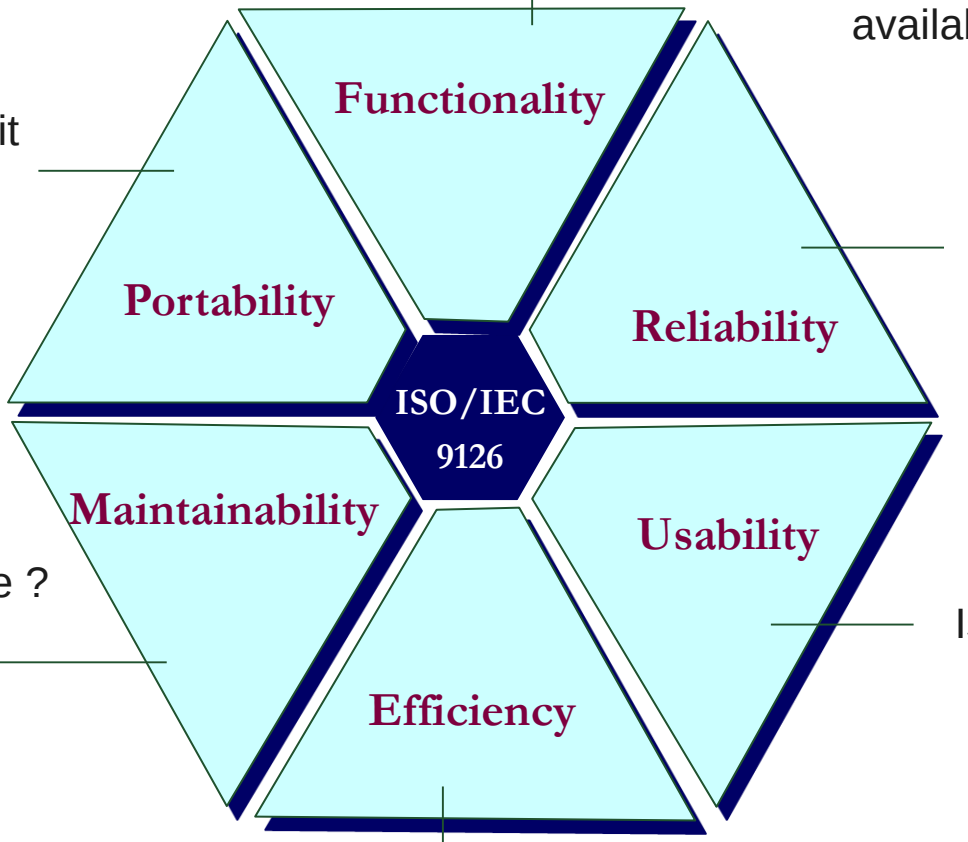
- Those factors are not independent each others !
- Some quality factors are too closely related
  - difficult to investigate / evaluate
  - difficult to apply to a system

Standards for software quality

- ISO 9126 : Software engineering — Product quality
- ISO 25010 : Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models
- and more …

ISO 9126: Six Quality Characteristics (2/3)

Six quality characteristics of a software



Are the required function available in the software ?

How reliable in the software ?

Is the software easy to use ?

How efficient the software ?

How easy to modify the software ?

How easy to transit the software to another environ-ment ?

Six Quality Characteristics (3/3)

Characteristics & sub-characteristics of the six qualities

Characteristics	Sub-characteristics
? Functionality	Suitability, Accurateness, Interoperability, Compliance, Security
? Reliability	Maturity, Fault-Tolerance, Recoverability
? Usability	Understandability, Learnability, Operability
? Efficiency	Time behaviour, Resource behaviour
? Maintainability	Analyzability, Changeability, Stability, Testability
? Portability	Adaptability, Installability, Conformance, Replaceability

ISO 9126

ISO 25010: Eight Quality Characteristics (1/3)

ISO/IEC 25010(2011), Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models

Replacing the 9126 Standards, currently

Differentiates between

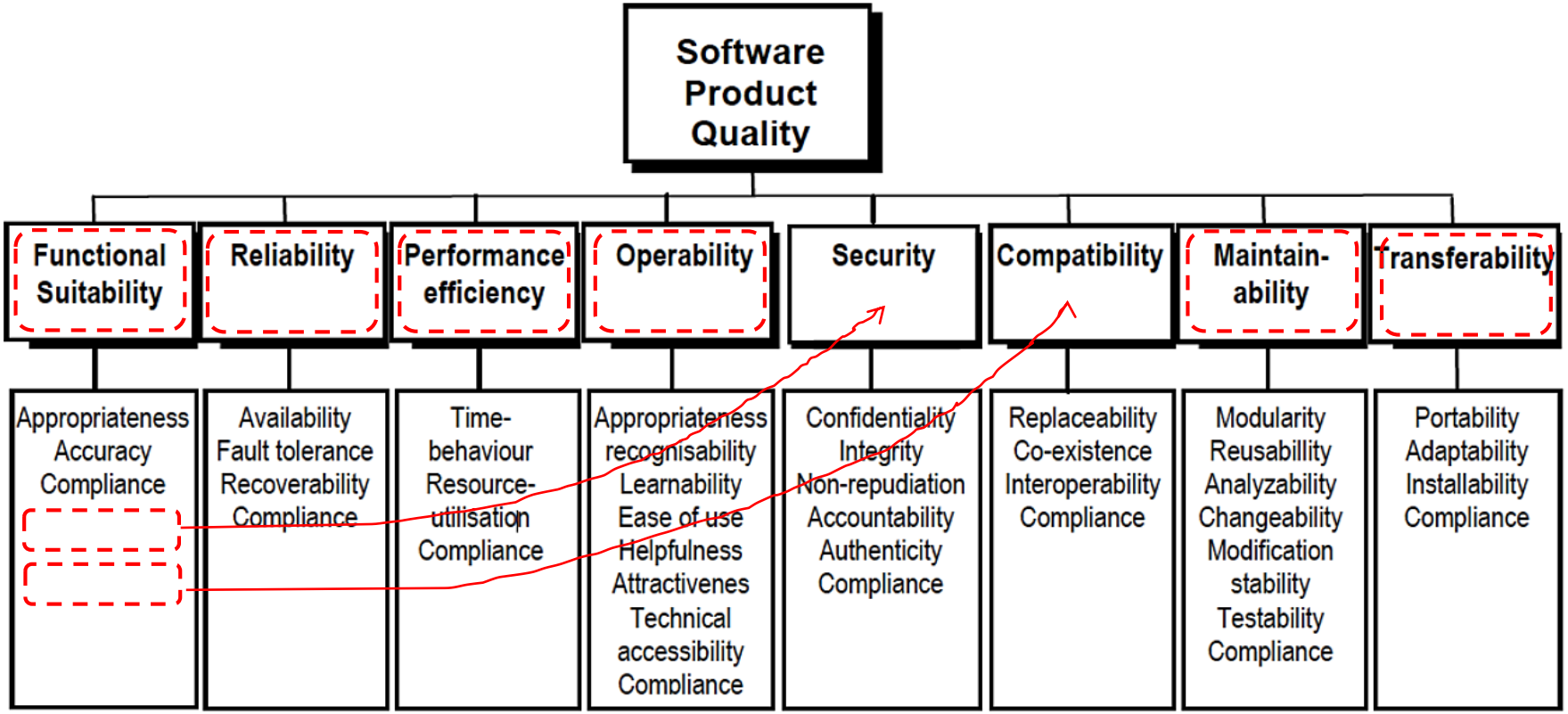
- Product quality model
- Quality in use model

External and internal quality have a specific meaning in the ISO/IEC 25010 framework:

- External quality assesses the characteristics of the product quality model by black-box measurement.
- Internal quality assesses the characteristics of the product quality model by glass-box (i.e., white-box) measurement, i.e. measuring system properties based on knowledge about the internal structure of the software

Eight Quality Characteristics (1/3)

ISO/IEC 25010(2011), Product Quality Model

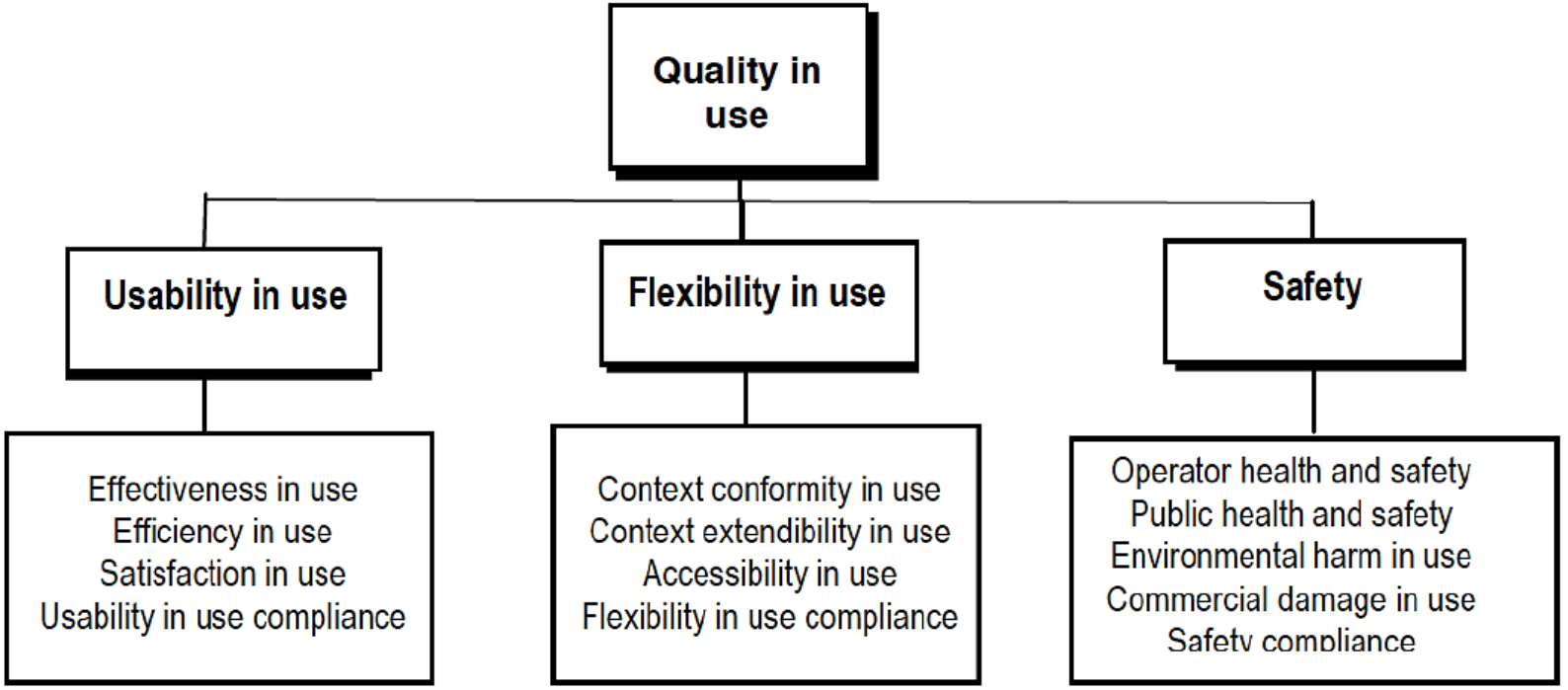


What is the relatiobship with the 9126?

: ISO/IEC 9126 Structure

Eight Quality Characteristics (1/3)

ISO/IEC 25010(2011), Quality in Use Model



Only achieved in a realistic system environment (in operation).

Summary and Discussion

Definition of Software Quality

- Fitness for needs

Representative Quality Factors

- Correctness, Reliability, Performance, User friendliness, ...
- Maintainability, Reusability, Understandability, Visibility, ...
- Quality factors for AI/ML systems

Why software quality is so important ?

How to measure software quality, for example maintainability ?