Software Quality

Software Quality

- Software quality is the “Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software”.

Software Quality

- The above definition emphasizes three main points:
- Software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.
- Specified standards define a set of development criteria that provide guidance for the development of software.
- Along with explicit requirements such as the desire for ease of use and good maintainability are also required.

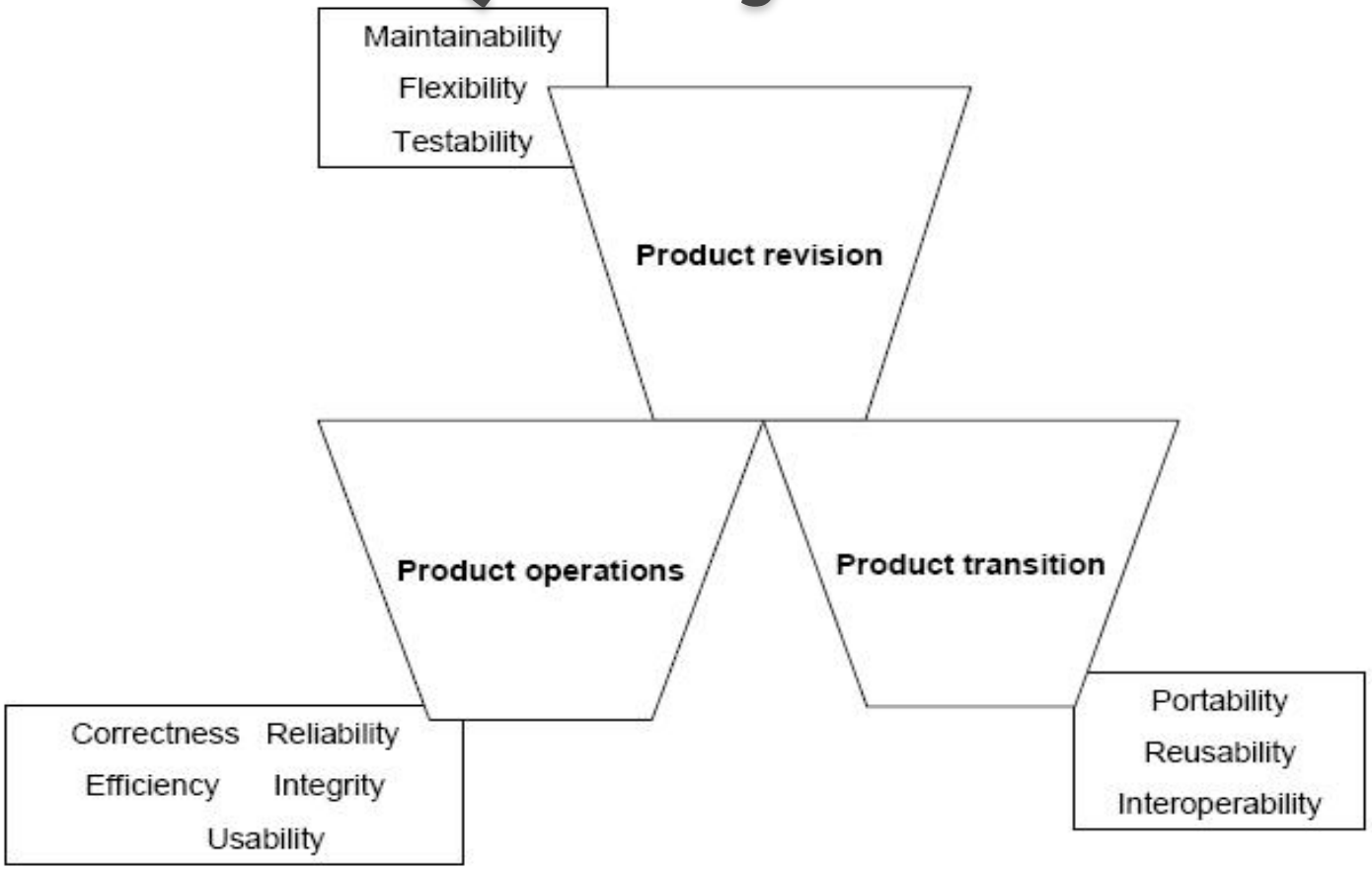
Software Quality Attributes

- **Functionality**: The capability to provide functions which meet specifications.
- **Reliability**: The capability to maintain a specified level of performance under stated conditions.
- **Usability**: The capability to be understood, learned, and used.
- **Efficiency**: The capability to provide appropriate performance relative to the amount of resources used.
- **Maintainability**: The capability to be modified for making corrections, enhancements or adaptation.
- **Portability**: The capability to be adapted for different specified environments without applying actions or means other than those provided for this purpose in the product.

McCall's Quality Factors

- McCall, Richard, and Walters proposed a useful categorization of software quality factors.
- This model attempts to bride the gap between users and developers by focusing on a number of software quality factor that reflect both the users view and developers priorities.
- This model has three major perspectives for defining and identifying the quality of a software product:
- Its operational characteristics
- Its ability to undergo change
- Its adaptability to new environments.

McCall's Quality Factors



- **Product Operations** : This identifies quality factors that influence the extent to which the software fulfils its specification.

- **Correctness** : The extent to which a functionality matches its specification.
- **Reliability** : The systems ability not to fail/ the extent to which the system fails.
- **Efficiency** : Further categorized into execution efficiency and storage efficiency and generally means the usage of system resources, example: processor time, memory.
- **Integrity** : The protection of program from unauthorized access.
- **Usability** : The ease of using software.

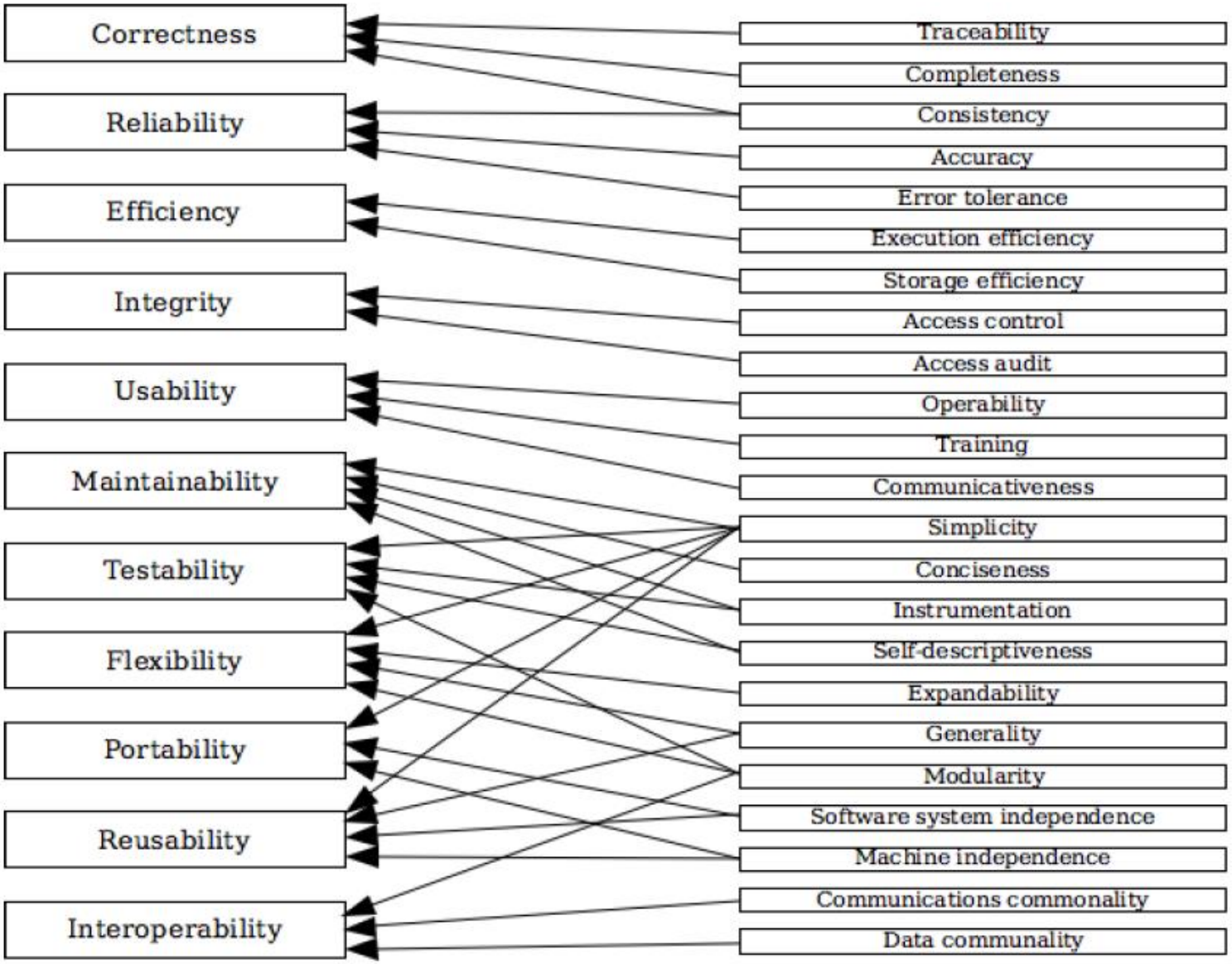
- **Product Revision** : This identifies quality factors that influence the ability to change the software product.

- **Maintainability** : Effort required to locate and fix a fault in the program within its operating environment.
- **Flexibility** : The ease of making changes required as dictated by business by changes in the operating environment.
- **Testability** : The ease of testing program to ensure that it is error-free and meets its specification, i.e., validating the software requirements.

- **Product Transition** : This identifies quality factors that influence the ability to adapt the software to new environments.

- **Portability** : The effort required to transfer a program from one environment to another.
- **Re-usability** : The ease of reusing software in a different context.
- **Interoperability**: The effort required to couple the system to another system.

- The model further details the three types of quality characteristics in a hierarchy of factors, criteria and metrics:
- **11 Factors(To specify)**: They describe the external view of the software, as viewed by the users.
- **23 quality criteria(To build):** They describe the internal view of the software, as seen by the developer.
- **Metrics(To control):** They are defined and used to provide a scale and method of measurement.


Correctness
Reliability
Efficiency
Integrity
Usability
Maintainability
Testability
Flexibility
Portability
Reusability
Interoperability
Traceability
Completeness
Consistency
Accuracy
Error tolerance
Execution efficiency
Storage efficiency
Access control
Access audit
Operability
Training
Communicativeness
Simplicity
Conciseness
Instrumentation
Self-descriptiveness
Expandability
Generality
Modularity
Software system independence
Machine independence
Communications commonality
Data communality

McCall's Software Metrics

- Auditability
- Accuracy
- Communication commonality
- Completeness
- Consistency
- Data commonality
- Error tolerance
- Execution efficiency
- Expandability
- Generality
- Hardware independence
- Instrumentation
- Modularity
- Operability
- Security
- Self-documentation
- Simplicity
- Software system independence
- Traceability
- Training

- **Auditability:** The ease with which conformance to standards can be checked
- **Accuracy:** The precision of computations and control
- **Communication commonality**: The degree to which standard interfaces, protocols and bandwidth are used
- **Completeness:** The degree to which full implementation of the required function has been achieved
- **Conciseness:** The compactness of the program in terms of lines of code
- **Consistency:** The use of uniform design and documentation techniques throughout the software development protocol

- **Data commonality**: The use of standard data structures and types throughout the program
- **Error tolerance**: The damage that occurs when a program encounters an error
- **Execution efficiency**: The run-time performance of the program
- **Expandability:** The degree to which architectural, data or procedural design can be extended
- **Generality:** The breadth of potential application of program components
- **Hardware independence**: The degree to which the software is decoupled from the hardware on which it operates
- **Instrumentation:** The degree to which the program monitors its own operations and identifies errors that do occur

- **Modularity**: The functional independence of program components
- **Operability**: The ease of operation of the program
- **Security**: The availability of mechanisms that control or protect programs and data
- **Self-documentation**: The degree to which the source code provides meaningful documentation
- **Simplicity**: The degree to which a program can be understood without difficulty
- **Traceability**: The ability to trace a design representation or actual program component back to requirements

- **Software system independence:** The degree to which the program is independent of non-standard programming language features, operating system characteristics and other environmental constraints
- **Training:** The degree to which the software assists in enabling new users to apply the system

Table 2 **Relationships between McCall's quality factors and metrics**

Quality factor / Software quality metric	Correctness	Reliability	Efficiency	Integrity	Maintainability	Flexibility	Testability	Portability	Reusability	Interoperability	Usability
Auditability				X			X				
Accuracy		X									
Communication commonality										X	
Completeness	X										
Conciseness			X		X	X					
Consistency	X	X			X	X					
Data commonality										X	
Error tolerance		X									
Execution efficiency			X								
Expandability						X					
Generality						X		X	X	X	
Hardware independence								X	X		
Instrumentation				X	X		X				
Modularity		X			X	X	X	X	X	X	
Operability			X								X
Security				X							
Self-documentation					X	X	X	X	X		
Simplicity		X			X	X	X				
Software system independence								X	X		
Traceability	X										
Training											X

Boehm's Quality Model(1978)

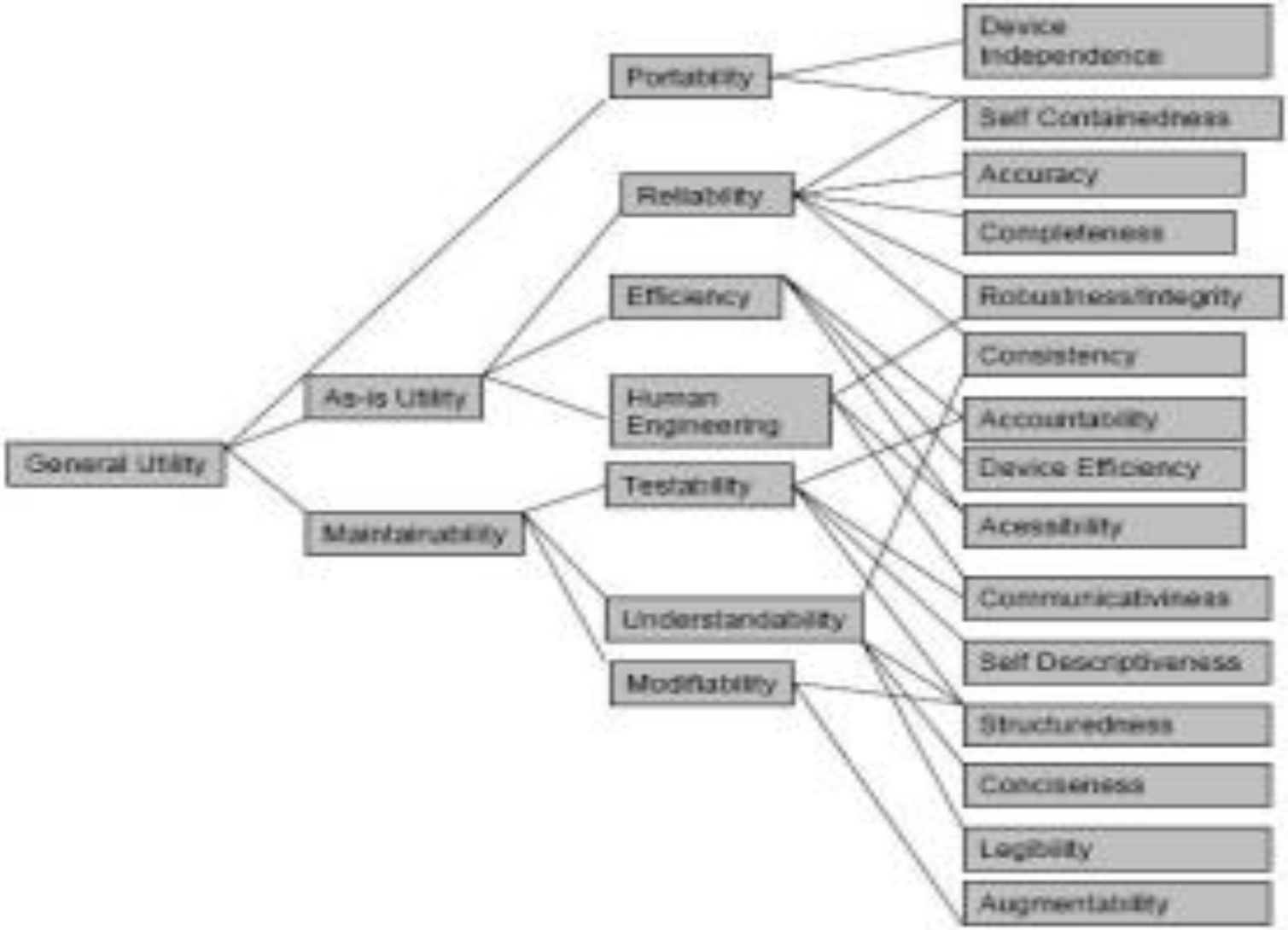
- Boehm's model is similar to the McCall Quality Model in that it also presents a hierarchical quality model structured around high-level characteristics, intermediate level characteristics, primitive characteristics – each of which contributes to the overall quality level.

- The high-level characteristics represent basic high-level requirements of actual use to which evaluation of software quality could be put – the general utility of software.
- The high-level characteristics address three main questions that a buyer of software has:
- As-is utility : How well (easily, reliably, efficiently) can I use it as-is?
- Maintainability: How easy is it to understand, modify and retest?
- Portability : Can I still use it if I change my environment?

- The intermediate level characteristic represents Boehm's 7 quality factors that together represent the qualities expected from a software system:
- **Portability (General utility characteristics)**: Code possesses the characteristic portability to the extent that it can be operated easily and well on computer configurations other than its current one.
- **Reliability (As-is utility characteristics)**: Code possesses the characteristic reliability to the extent that it can be expected to perform its intended functions satisfactorily.
- **Efficiency (As-is utility characteristics)**: Code possesses the characteristic efficiency to the extent that it fulfils its purpose without waste of resources.
- **Usability (As-is utility characteristics, Human Engineering)**: Code possesses the characteristic usability to the extent that it is reliable, efficient and human-engineered.

- **Testability (Maintainability characteristics):** Code possesses the characteristic testability to the extent that it facilitates the establishment of verification criteria and supports evaluation of its performance.
- **Understandability (Maintainability characteristics):** Code possesses the characteristic understandability to the extent that its purpose is clear to the inspector.
- **Flexibility (Maintainability characteristics, Modifiability):** Code possesses the characteristic modifiability to the extent that it facilitates the incorporation of changes, once the nature of the desired change has been determined.

- The lowest level structure of the characteristics hierarchy in Boehm's model is the primitive characteristics metrics hierarchy.
- The primitive characteristics provide the foundation for defining qualities metrics – which was one of the goals when Boehm constructed his quality model.
- Consequently, the model presents one more metrics supposedly measuring a given primitive characteristic.


General Utility
As-is Utility
Maintainability
Portability
Reliability
Efficiency
Human Engineering
Testability
Understandability
Modifiability
Device Independence
Self Containedness
Accuracy
Completeness
Robustness/Integrity
Consistency
Accountability
Device Efficiency
Accessibility
Communicativeness
Self Descriptiveness
Structuredness
Conciseness
Legibility
Augmentability

- Though Boehm's and McCall's models might appear very similar, the difference is that McCall's model primarily focuses on the precise measurement of the high-level characteristics "As-is utility", whereas Boehm's quality model is based on a wider range of characteristics with an extended and detailed focus on primarily maintainability.

FURPS

Usability

Functionality

Performance

FURPS+

Reliability

Supportability

- The FURPS- categories are of two types: Functional(F) and Non-functional(URPS). These categories can be used as both product requirements as well as in the assessment of product quality.

FURPS/FURPS+

- The characteristics that are taken into consideration in FURPS model are:
- Functionality- includes feature sets, capabilities and security;
- Usability -includes human factors, consistency in the user interface, online and context-sensitive help, wizards, user documentation, and training materials;

- **Reliability:** includes frequency and severity of failure, recoverability, predictability, accuracy, and mean time between failure (MTBF).
- **Performance:** prescribes conditions on functional requirements such as speed, efficiency, availability, accuracy, throughput, response time, recovery time, and resource usage.
- **Supportability:** includes testability, extensibility, adaptability, maintainability, compatibility, Configurability, serviceability, install ability, and localizability / internationalization.

Dromey's Quality Model

- Dromey has built a quality evaluation framework that analyzes the quality of software components through the measurement of tangible quality properties.
- Dromey gives the following examples of what he means by software components for each of the different models:
- Variables, functions, statements, etc. can be considered components of the Implementation model.
- A requirement can be considered a component of the requirements model.
- A module can be considered a component of the design model.

- According to Dromey, all these components possess intrinsic properties that can be classified into four categories:
- **Correctness** : Evaluates if some basic principles are violated.
- **Internal** : Measure how well a component has been deployed according to its intended use.
- **Contextual** : Deals with the external influences by and on the use of a component.
- **Descriptive** : Measure the descriptiveness of a component (for example, does it have a meaningful name.)

- Dromey focuses on the relationship between the quality attributes and the sub-attributes, as well as attempts to connect software product properties with software quality attributes.
- Product properties that influence quality.
- High level quality attributes.
- Means of linking the product properties with the quality attributes.

- Dromey's Quality Model is further structured around a 5 step process:
- Choose a set of high-level quality attributes necessary for the evaluation.
- List components/modules in your system.
- Identify quality-carrying properties for the components/modules (qualities of the component that have the most impact on the product properties from the list above).
- Determine how each property effects the quality attributes.
- Evaluate the model and identify weaknesses.

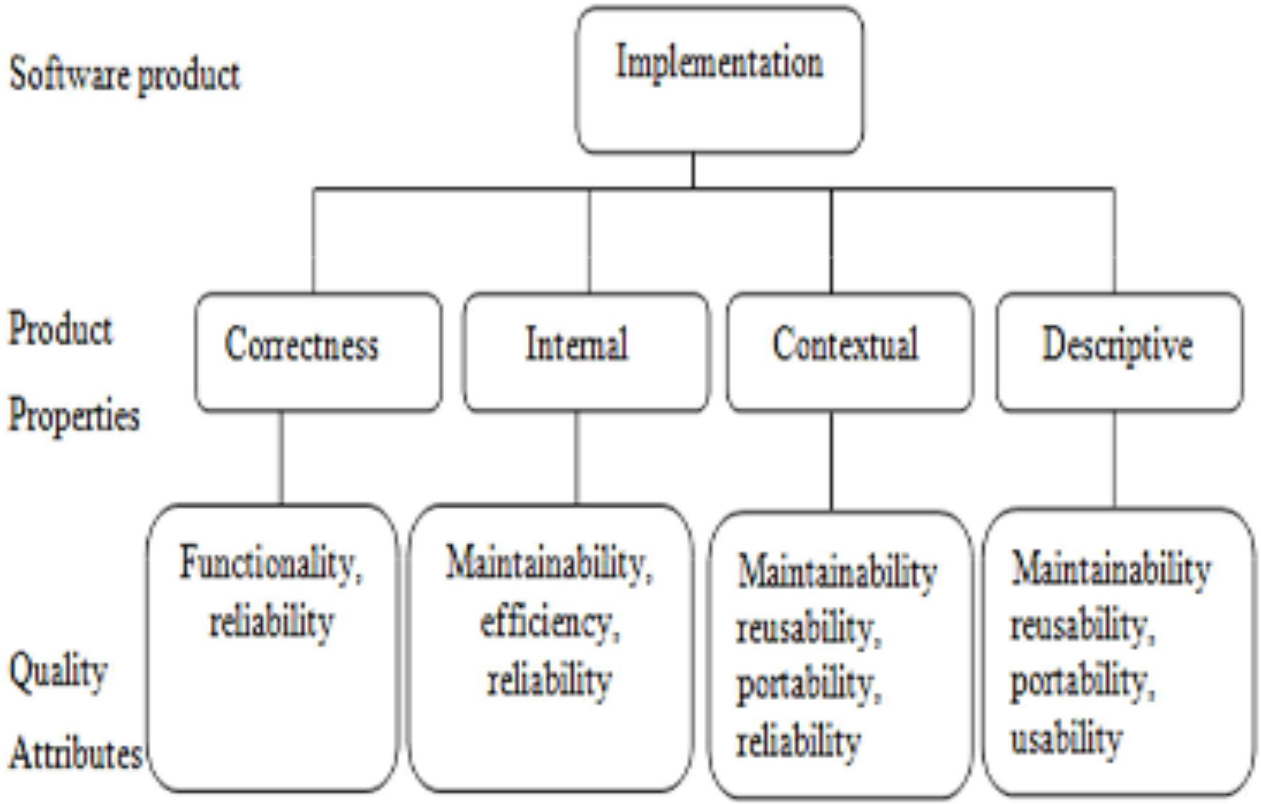


Fig.4 Principles of Dromey's Quality Model

ISO 9126

The ISO 9126 standard was based on the McCall and Boehm models[7].Besides being structured in basically the same manner as these models ISO 9126 also includes functionality as a parameter, as well as identifying both internal and external quality characteristics of software products.

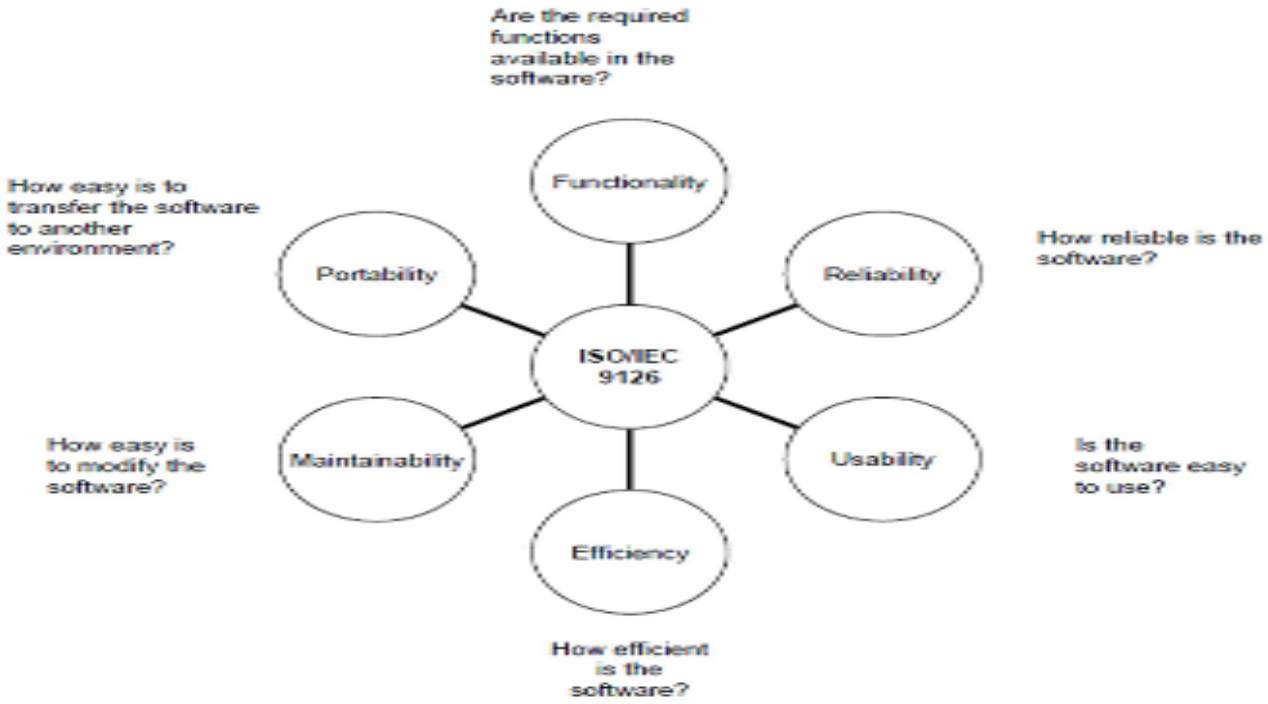


Fig.3 The ISO 9126 quality model

- ISO classifies software quality in a structured set of characteristics and sub-characteristics as follows:
- **Functionality** : A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
  - Suitability
  - Accuracy
  - Interoperability
  - Security
  - Functionality Compliance
- **Reliability** : A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
  - Maturity
  - Fault Tolerance
  - Recoverability
  - Reliability Compliance

- **Usability** : A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.
  - Understandability
  - Learn ability
  - Operability
  - Attractiveness
  - Usability Compliance
- **Efficiency** : A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.
  - Time Behaviour
  - Resource Utilization
  - Efficiency Compliance

- **Maintainability:** A set of attributes that bear on the effort needed to make specified modifications.
  - Analyzability
  - Changeability
  - Stability
  - Testability
  - Maintainability Compliance
- **Portability:** A set of attributes that bear on the ability of software to be transferred from one environment to another.
  - Adaptability
  - Installability
  - Co-Existence
  - Replaceability
  - Portability Compliance

F. Comparison between five quality models

Factors/Attributes/Characteristics	McCall	Boehm	Dromey	FURPS	ISO 9126
Maintainability	*		*		*
Flexibility	*				
Testability	*	*			
Correctness	*				
Efficiency	*	*	*		*
Reliability	*	*	*	*	*
Integrity	*				
Usability	*		*	*	*
Portability	*	*	*		*
Reusability	*		*		
Interoperability	*				
Human Engineering		*			
Understandability		*			
Modifiability		*			
Functionality			*	*	*
Performance				*	
Supportabilty				*	
	17	11	7	7	5	6