ISO 9126 Software Quality Characteristics

An overview of the ISO 9126-1 software quality model definition, with an explanation of the major characteristics.

SQA Definition | SQA in Theory | SQA in Practice | SQA and Software Articles
The CMMi easy button | The Software Testing Directory

ISO 9126 Software Quality Model

Article Purpose

The purpose of this article is to present an overview of the ISO 9126 standard and to give a detailed description of the software quality model used by this standard.

ISO 9126 is an international standard for the evaluation of software. The standard is divided into four parts which addresses, respectively, the following subjects: quality model; external metrics; internal metrics; and quality in use metrics. ISO 9126 Part one, referred to as ISO 9126-1 is an extension of previous work done by McCall (1977), Boehm (1978), FURPS and others in defining a set of software quality characteristics.

ISO9126-1 represents the latest (and ongoing) research into characterizing software for the purposes of software quality control, software quality assurance and software process improvement (SPI). This article defines the characteristics identified by ISO 9126-1. The other parts of ISO 9126, concerning metrics or measurements for these characteristics, are essential for SQC, SQA and SPI but the main concern of this article is the definition of the basic ISO 9126 Quality Model.

The ISO 9126 documentation itself, from the official ISO 9126 documentation, can only be purchased and is subject to copyright. SQA.net only reproduces the basic structure of the ISO 9126 standard and any descriptions, commentary or guidance are original material based on public domain information as well as our own experience.

The ISO 9126-1 software quality model identifies **6 main quality characteristics**, namely:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability

- Portability

These characteristics are broken down into subcharacteristics, a high level table is shown below. It is at the subcharacteristic level that measurement for SPI will occur. The main characteristics of the ISO9126-1 quality model, can be defined as follows:-

Functionality
Functionality is the essential purpose of any product or service. For certain items this is relatively easy to define, for example *a ship's anchor* has the function of holding a ship at a given location. The more functions a product has, e.g. an ATM machine, then the more complicated it becomes to define it's functionality. For software a list of functions can be specified, i.e. a sales order processing systems should be able to *record customer information* so that it can be used to reference a sales order. A sales order system should also provide the following functions:

- Record sales order product, price and quantity.
- Calculate total price.
- Calculate appropriate sales tax.
- Calculate date available to ship, based on inventory.
- Generate purchase orders when stock falls below a given threshold.

The list goes on and on but the main point to note is that functionality is expressed as a totality of essential functions that the software product provides. It is also important to note that the presence or absence of these functions in a software product can be verified as either existing or not, in that it is a Boolean (either a yes or no answer). The other software characteristics listed (i.e. usability) are only present to some degree, i.e. not a simple on or off. Many people get confused between overall process functionality (in which software plays a part) and software functionality. This is partly due to the fact that Data Flow Diagrams (DFDs) and other modeling tools can depict process functionality (as a set of data in\data out conversions) and software functionality. Consider a sales order process, that has both manual and software components. A function of the sales order process could be to record the sales order but we could implement a *hard copy* filing cabinet for the actual orders and only use software for calculating the price, tax and ship date. In this way the functionality of the software is limited to those calculation functions. SPI, or Software Process Improvement is different from overall Process Improvement or Process Re-engineering, ISO 9126-1 and other software quality models do not help measure overall Process costs\benefits but only the software component. The relationship between software functionality within an overall business process is outside the scope of ISO 9126 and it is only the software functionality, or essential purpose of the software component, that is of interest for ISO 9126.

Following functionality, there are 5 other software attributes that characterize the usefulness of the software in a given environment.
Each of the following characteristics can only be measured (and are assumed to exist) when the functionality of a given system is present. In this way, for example, a system can not possess *usability* characteristics if the system does not function correctly (the two just don't go together).

Reliability
Once a software system is functioning, as specified, and delivered the reliability characteristic defines the capability of the system to maintain its service provision under defined conditions for defined periods of time. One aspect of this characteristic is *fault tolerance* that is the ability of a system to withstand component failure. For example if the network goes down for 20 seconds then comes back the system should be able to recover and continue functioning.

Usability
Usability only exists with regard to functionality and refers to the ease of use for a given function. For example a function of an ATM machine is to dispense cash as requested. Placing common amounts on the screen for selection, i.e. $20.00, $40.00, $100.00 etc, does not impact the function of the ATM but addresses the Usability of the function. The ability to learn how to use a system (learnability) is also a major subcharacteristic of usability.

Efficiency

This characteristic is concerned with the system resources used when providing the required functionality. The amount of disk space, memory, network etc. provides a good indication of this characteristic. As with a number of these characteristics, there are overlaps. For example the usability of a system is influenced by the system's Performance, in that if a system takes 3 hours to respond the system would not be easy to use although the essential issue is a performance or efficiency characteristic.

Maintainability
The ability to identify and fix a fault within a software component is what the maintainability characteristic addresses. In other software quality models this characteristic is referenced as supportability. Maintainability is impacted by code readability or complexity as well as modularization. Anything that helps with identifying the cause of a fault and then fixing the fault is the

complexity as well as modularization. Anything that helps with identifying the cause of a fault and then fixing the fault is the concern of maintainability. Also the ability to verify (or test) a system, i.e. testability, is one of the subcharacteristics of maintainability.

Portability
This characteristic refers to how well the software can adopt to changes in its environment or with its requirements. The subcharacteristics of this characteristic include adaptability. Object oriented design and implementation practices can contribute to the extent to which this characteristic is present in a given system.

The full table of Characteristics and Subcharacteristics for the ISO 9126-1 Quality Model is:-

Characteristics	Subcharacteristics	Definitions
	Suitability	This is the essential Functionality characteristic and refers to the appropriateness (to specification) of the functions of the software.
	Accurateness	This refers to the correctness of the functions, an ATM may provide a cash dispensing function but is the amount correct?
Functionality	Interoperability	A given software component or system does not typically function in isolation. This subcharacteristic concerns the ability of a software component to interact with other components or systems.
	Compliance	Where appropriate certain industry (or government) laws and guidelines need to be complied with, i.e. SOX. This subcharacteristic addresses the compliant capability of software.
	Security	This subcharacteristic relates to unauthorized access to the software functions.
	Maturity	This subcharacteristic concerns frequency of failure of the software.
Reliability	Fault tolerance	The ability of software to withstand (and recover) from component, or environmental, failure.
	Recoverability	Ability to bring back a failed system to full operation, including data and network connections.
	Understandability	Determines the ease of which the systems functions can be understood, relates to user mental models in Human Computer Interaction methods.
Usability	Learnability	Learning effort for different users, i.e. novice, expert, casual etc.
	Operability	Ability of the software to be easily operated by a given user in a given environment.
Efficiency	Time behavior	Characterizes response times for a given thru put, i.e. transaction rate.
	Resource behavior	Characterizes resources used, i.e. memory, cpu, disk and network usage.
	Analyzability	Characterizes the ability to identify the root cause of a failure within the software.
Maintainability	Changeability	Characterizes the amount of effort to change a system.
	Stability	Characterizes the sensitivity to change of a given system that is the negative impact that may be caused by system changes.
	Testability	Characterizes the effort needed to verify (test) a system change.

	Adaptability	Characterizes the ability of the system to change to new specifications or operating environments.
Portability	Installability	Characterizes the effort required to install the software.
	Conformance	Similar to compliance for functionality, but this characteristic relates to portability. One example would be Open SQL conformance which relates to portability of database used.
	Replaceability	Characterizes the *plug and play* aspect of software components, that is how easy is it to exchange a given software component within a specified environment.

ISO 9126 Observations

For the most part, the overall structure of ISO9126-1 is similar to past models, McCall (1977) and Boehm (1978), although there are a couple of notable differences. Compliance comes under the functionality characteristic, this can be attributed to government initiatives like SOX. In many requirements specifications all characteristics, that are specified, that are not pure functional requirements are specified as *Non-Functional* requirements. It is interesting to note, with ISO9126, that compliance is seen as a functional characteristic.

Using the ISO 9126 (or any other quality model) for derivation of system requirements brings clarity of definition of purpose and operating capability .
For example a *rules engine* approach to compliance would enable greater adaptability, should the compliance rules change. The functionality for compliance could be implemented in other ways but these other implementation methods may not produce as strong an adaptability characteristic as a rules, or some other component based, architecture.

Also, a designer typically will need to make trade offs between two or more characteristics when designing the system. Consider highly modularized code, this code is usually easy to maintain, i.e. has a good *changeability* characteristic, but may not perform as well (for cpu resource, as unstructed program code). On a similar vein a normalized database may not perform as well as a not normalized database. These trade offs need to be identified, so that informed design decisions can be made.

Although ISO 9126-1 is the latest proposal for a useful Quality Model, of software characteristics, it is unlikely to be the last. One thing is certain, the requirements (including compliance) and operating environment of software will be continually changing and with this change will come the continuing search to find useful characteristics that facilitate measurement and control of the software production process.

No guarantee (or claim) is made regarding the accuracy of this information. Any questions or comments should be sent to:-
agoodyear@yahoo.com