# Software Quality Factors (Attributes)

**McCall's Classic Model for Software Quality Factors**

McCall factor model continues to provide a practical, up-to-date method for classifying software quality requirements. McCall’s factor model classifies all software requirements into 11 software quality factors. These 11 factors are grouped into three categories:

* **Product Operation Factors:** According to McCall’s model, five software quality factors are included in this category, all of which deal with requirements that directly affect the daily operation of the software. These factors are:
  + **Correctness** – is related to the outputs of software systems, such as a query display of a customer’s balance in the sales accounting information system or the air supply as a function of temperature specified by the firmware of an industrial control unit.
  + **Reliability** – deals with failures to provide service. The requirements can refer to the entire system or one or more of its separate functions.
  + **Efficiency** – deals with the hardware resources needed to perform all the functions of the software system in conformance with all other requirements.
  + **Integrity** – defined to cope with risks of “nonfriendly” unauthorized attempts to damage the software system and its performance.
  + **Usability** – deals with the scope of staff resources needed to train a new employee and to operate the software system.
* **Product Revision Factors:** These factors deal with those requirements that affect the complete range of software maintenance activities. The three quality factors included in this category are:
  + **Maintainability** – determines the efforts needed by users and maintenance personnel to identify the reasons for software failure, to correct the failure, and to verify the success of the correction.
  + **Flexibility** – deals with the capabilities and efforts required to support adaptive maintenance activities.
  + **Testability** – deals with the testing process of a software

system, as well as with its operation.

* **Product Transition Factors:** A category that pertains to the adaptation of software to other environments, and its interaction with other software systems. Three factors that are included are:
  + **Portability** – relates to the adaptation of a software system to other environments consisting of different hardware, different operating systems, and so forth.
  + **Reusability** – deals with “two-directional” requirements. One direction is the use of a software module, or an entire application, taken from an existing software product in a new software project currently being developed. The other direction relates to a requirement to develop modules or a group of modules, or even an entire project, in a way to enable their reuse in future projects.
  + **Interoperability** – focuses on creating interfaces with other software systems or equipment firmware. It sometimes specifies the name(s) of the software or firmware to which an interface is required.

# ISO/IEC 25010 Model and Other Alternative Models of Software Quality

It was developed by a joint ISO/IEC international professional team and is of significant importance. The model includes the eight following factors, while four of them were already included in McCall’s model.

* **Functional suitability** – the capability to fulfill the functions needed by the customer, stated or implied (not necessarily the specified requirement). A significant similarity exists in the interoperability factors of McCall’s model.
* **Performance efficiency** – relates to the amount of hardware resources required to fulfill the software system tasks. A significant similarity exists in the efficiency factor of McCall’s model.
* **Compatibility** – refers to the capability of a software system or component to exchange information with other software systems or components and perform other system required functions, sharing its hardware system and software environment.
* **Usability** – deals with the scope of staff resources needed to train a new employee and to operate the software system.
* **Reliability** – deals with failures to provide service.
* **Security** – relates to the capability of a system product to protect the software system, data stores, and information produced from

the reading, modification, or destruction by unauthorized persons or systems. A significant similarity exists in the integrity factor described in McCall’s model.

* **Maintainability** – determines the efforts needed by users and maintenance personnel to identify the reasons for software failure, to correct the failure, and to verify the success of the correction.
* **Portability** – relates to the adaptation of a software system to other environments consisting of different hardware, different operating systems, and so forth.

# Alternative Software Quality Models

These alternative models propose 14 additional software quality factors, several of which show similarities to McCall’s factors and also overlap each other.

* **Effectiveness** – relates to the successful completion of tasks, including schedule and error frequency considerations.
* **Evolvability** – the efforts required to fulfill future requirements for software system changes, and to adapt the system to technological developments and changes in the operational environment
* **Expandability** – the future efforts required to serve larger populations, improve service, or add new applications, in order to improve system performance
* **Extensibility** – the efforts required to fulfill future requirements to enhance the software product to meet new requirements, resulting from economic and technological developments
* **Human Engineering** – deals with the “man-machine” user interface with the application or software, the ease to understand and work with the application, the ease in performing any communication involved with working with the application.
* **Manageability –** the administrative tools that support software modification during the software development and maintenance periods
* **Modifiability** – the efforts that will be needed to modify the software product according to the specific requirements of customers
* **Productivity** – relates to the rate at which the software product performs tasks.
* **Safety** – designed to eliminate conditions that may be hazardous to equipment and equipment operators as a result of errors in process control software
* **Satisfaction** – the user perception of the extent the software product meets user expectations in relation to the requirements.
* **Supportability** – the ease of performing install tasks and various maintenance tasks of error corrections, the adaption of the software product to specific customer needs
* **Survivability** – the continuity of service
* **Understandability -** the user’s capability to find out how to use the software for particular tasks, and to grasp the conditions of use
* **Verifiability** – defines design and programming features that enable efficient verification of the design and programming.

# Software Compliance with Quality Factors

A set of explanatory criteria (subfactors) for each factor is believed to bridge the gap, and help customers and software developers specify quality requirements, define review questions, prepare test plans, and develop software quality metrics. As a result, the criteria help to examine the degree software projects that comply with the software quality factors.

|  |  |
| --- | --- |
| **Software Quality Factors** | **Criteria (Subfactors)** |
| **Correctness** | Accuracy Completeness Up-to-dateness Availability |
| **Reliability** | System and Application Reliability Failure Recovery  Hardware Failure Recovery |
| **Efficiency** | Efficiency of Processing Efficiency of Storage Efficiency of Communication  Efficiency of Power Usage |
| **Integrity** | Access Control Access Audit |
| **Usability** | Operability  Learning and Training Ability |
| **Maintainability** | Simplicity Modularity  Self-descriptiveness Consistency |

|  |  |
| --- | --- |
| **Flexibility** | Modularity Generality Simplicity  Self-descriptiveness |
| **Testability** | Simplicity  Failure maintenance Traceability |
| **Portability** | Software system independence |
| **Reusability** | Modularity  Self-descriptiveness Consistency  Software system independence Generality  Simplicity |
| **Interoperability** | Commonality System compatibility  Software system independence  Modularity |

As you have probably noticed, several criteria (subfactors) relate to more than one factor. This reflects the fact that several criteria contribute to successful compliance in more than one factor. It also reflects the extensive overlapping and the many similarities of software quality factors.

# Business Models and Choice of Software Engineering Practices

**Business Model:** Describes the rationale of how an organization creates, delivers, and captures value (economic, social, or other forms of value). The essence of a business model is that it defines the manner by which the business enterprise delivers value to customers, entices customers to pay for value, and converts those payments to profit.

# Five Main Business Models in the Software Industry

1. **Custom Systems Written on Contract:** The organization makes profits by selling tailored software development services for clients. The profits made by the supplier depend on his ability to remain within budget and to deliver on schedule, as defined in the contract, a product that performs as intended.
2. **Custom Software Written In-house:** The organization develops software to improve organizational efficiency. (e.g., your current internal IT organization)
3. **Commercial Software:** The company makes profits by developing and selling software to other organizations. (e.g., Oracle)
4. **Mass-market Software:** The company makes profits by developing and selling software to consumers. (e.g., Microsoft and Adobe)
5. **Commercial and Mass-market Firmware:** The company makes profits by selling software in embedded hardware and systems. (e.g., digital cameras)

# Situational Factors (Risks) that Influence the Choice of Software Engineering Practices

* + **Criticality** – the potential to cause harm to the user or prejudice the interests of the purchaser varies depending on the type of product. Take medical databases for example, a doctor might make a mistake because of wrong data from such a database.
  + **Uncertainty of Users’ Wants and Needs:** The requirements for software that implements a familiar process in an organization are better known than the requirements for a consumer product that is so new that the end-users do not even know what they want. For instance, users may not agree on the steps in the process, their demands may require technology that does not exist, business needs may change during the project, and sometimes people completely change their minds.
  + **Range of Environments:** Software written for use in a specific organization only has to be compatible with its own computer environment, whereas software sold to a mass market must work in a wide range of environments. In general, the purchasing organization has identified a small set of target environments in order to avoid cost increases. The result is a range of environments that are clearly defined and relatively small compared with other cultures.
  + **Cost of Fixing Errors:** It will cost more to fix an error after the product is built than it would if the error was discovered during the requirements phase of a project. For example, distributing corrections for certain software applications (e.g., embedded software of an automobile) is usually far costlier than fixing a website.
  + **Regulations:** Regulatory bodies and contractual clauses may require the use of software practices other than those that would normally be adopted. Certain situations require process audits to check whether a process was followed at the time of producing the software. Defense software (e.g., for a fighter or commercial plane) must comply with a huge list of regulations, most of which concern the software development process.
  + **Project Size:** Projects that take several years and require hundreds of developers are common in certain organizations, whereas in other organizations, shorter projects developed by a single team are more typical. Several dozen people work for more than two (2) years on the average-size project, but hundreds of people over several years are required for large projects.
  + **Communication:** There are a certain number of factors, in addition to project scope, that can increase the quantity of person-to-person communication or make communications more difficult. Certain factors seem to occur more often within certain cultures, whereas others happen at random:
    - **Concurrent Developer–Developer Communication:** Communication with other people on the same project is affected by how the work is distributed.
    - **Developer–Maintainer Communication:** Communication with developers is greatly facilitated when they work in the same area.

# Communication between Managers and Developers:

Progress reports must be sent to upper management.

* + **Organization’s Culture:** The organization has a culture that defines how people work. There are four types of organizational cultures:
    - **Control Culture:** Motivated by the need for power and security (e.g., IBM).
    - **Skill Culture:** defined by the need to make full use of one’s skills (e.g., Microsoft).
    - **Collaborative Culture:** As illustrated by Hewlett-Packard (HP), is motivated by a need to belong (e.g., Facebook).
    - **Thriving Culture:** Motivated by self-actualization, and can be seen in start-up organizations (e.g., Adobe).

**References:**

Criticality. (n.d). Citing sources. Retrieved from [http://www.cs.swan.ac.uk](http://www.cs.swan.ac.uk/)

Cost of Fixing Errors. (n.d). Citing sources. Retrieved from https://ntrs.nasa.gov

Galin, D. (2018). Software Quality Assurance – Concepts and Practice: IEEE Computer Society, Inc.

Laporte, C. and April, A. (2018). Software Quality Assurance: IEEE Computer Society, Inc. Software Business Model. (n.d). Retrieved from https://[www.cs.utexas.edu](http://www.cs.utexas.edu/)