

Table of Contents

Code Quality 3

Code Quality

- [7 axis of code quality](#)
- [Software functional quality](#)
- [Software structural quality](#)
- <https://www.perforce.com/blog/sca/what-code-quality-and-how-improve-it>
- <https://codingsans.com/blog/code-quality>
- [code analysis](#)
- <https://www.tiobe.com/>

Other

- <https://hackernoon.com/how-680-developers-say-we-can-increase-code-quality-339908809d19>
 - improving code quality;
 - facilitating team collaboration;
 - homogenizing code standards and code styles;
 - identifying bugs early in the development process;
 - onboarding new developers and spreading best practices and knowledge within the organization.
- <https://www.ibm.com/garage/method/practices/learn/developer-analytics/>

Snippet from [Wikipedia](#): **Software quality**

In the context of software engineering, **software quality** refers to two related but distinct notions:

- Software's functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications. That attribute can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile product. It is the degree to which the correct software was produced.
- Software structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability. It has a lot more to do with the degree to which the software works as needed.

Many aspects of structural quality can be evaluated only statically through the analysis of the software inner structure, its source code (see Software metrics), at the unit level, and at the system level (sometimes referred to as end-to-end testing), which is in effect how its architecture adheres to sound principles of software architecture outlined in a paper on the topic by Object Management Group (OMG).

However some structural qualities, such as usability, can be assessed only dynamically (users or others acting on their behalf interact with the software or, at least, some prototype or partial implementation; even the interaction with a mock version made in cardboard represents a dynamic test because such version can be considered a prototype). Other aspects, such as reliability, might involve not only the software but also the underlying hardware, therefore, it can be assessed both statically and dynamically (stress test).

Functional quality is typically assessed dynamically but it is also possible to use static tests (such as software reviews).

Historically, the structure, classification and terminology of attributes and metrics applicable to software quality management have been derived or extracted from the ISO 9126 and the subsequent ISO/IEC 25000 standard. Based on these models (see Models), the Consortium for IT Software Quality (CISQ) has defined five major desirable structural characteristics needed for a piece of software to provide business value: Reliability, Efficiency, Security, Maintainability and (adequate) Size.

Software quality measurement quantifies to what extent a software program or system rates along each of these five dimensions. An aggregated measure of software quality can be computed through a qualitative or a quantitative scoring scheme or a mix of both and then a weighting system reflecting the priorities. This view of software quality being positioned on a linear continuum is supplemented by the analysis of "critical programming errors" that under specific circumstances can lead to catastrophic outages or performance degradations that make a given system unsuitable for use regardless of rating based on aggregated measurements. Such programming errors found at the system level represent up to 90 percent of production issues, whilst at the unit-level, even if far more numerous, programming errors account for less than 10 percent of production issues (see also Ninety-ninety rule). As a consequence, code quality without the context of the whole system, as W. Edwards Deming described it, has limited value.

To view, explore, analyze, and communicate software quality measurements, concepts and techniques of information visualization provide visual, interactive means useful, in particular, if several software quality measures have to be related to each other or to components of a software or system. For example, software maps represent a specialized approach that "can express and combine information about software development, software quality, and system dynamics".

Software quality also plays a role in the release phase of a software project. Specifically, the quality and establishment of the release processes (also patch processes), configuration management are important parts of an overall software engineering process.

[Creative Commons Attribution-Share Alike 4.0](#)

From:

<https://almbok.com/> - **ALMBoK.com**

Permanent link:

https://almbok.com/dev/code_quality

Last update: **2022/08/10 05:26**

