Software Development Quality Cheat Sheet

Quality Attributes

The attributes that define quality.

They cannot all be maximized because they conflict with each other. Trade-offs have to be made to find a global optimum.

Product Quality

(based on ISO 25010)

The attributes of the artefact we build.

Functional Suitability

Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.

Functional completeness

Degree to which the set of functions covers all the specified tasks and user objectives.

Functional correctness

Degree to which a product or system provides the correct results with the needed degree of precision.

Functional appropriateness

Degree to which the functions facilitate the accomplishment of specified tasks and objectives.

Performance efficiency

Performance relative to the amount of resources used under stated conditions.

Time behaviour

Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.

Resource utilization

Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.

Capacity

Degree to which the maximum limits of a product or system parameter meet requirements.

Compatibility

Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.

Co-existence

Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.

Interoperability

Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

Usability

Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

Appropriateness recognizability

Degree to which users can recognize whether a product or system is appropriate for their needs.

Learnability

Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.

Operability

Degree to which a product or system has attributes that make it easy to operate and control.

User error protection

Degree to which a system protects users against making errors.

User interface aesthetics

Degree to which a user interface enables pleasing and satisfying interaction for the user.

Accessibility

Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

Reliability

Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.

Maturity

Degree to which a system, product or component meets needs for reliability under normal operation.

Availability

Degree to which a system, product or component is operational and accessible when required for use.

Fault tolerance

Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

Recoverability

Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

Security

Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.

Confidentiality

Degree to which a product or system ensures that data are accessible only to those authorized to have access.

Integrity

Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

Non-repudiation

Degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.

Accountability

Degree to which the actions of an entity can be traced uniquely to the entity.

Authenticity

Degree to which the identity of a subject or resource can be proved to be the one claimed.

Maintainability

Degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements.

Modularity

Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

Reusability

Degree to which an asset can be used in more than one system, or in building other assets.

Analysability

Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.

Modifiability

Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

Testability

Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Portability

Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

Adaptability

Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

Installability

Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.

Replaceability

Degree to which a product can replace another specified software product for the same purpose in the same environment.

Service Quality

(partially based on SERVQUAL)

How our work is perceived by our customers.

Expectation

Degree to which actual outcomes match (or even exceed) the expectations of the stakeholders.

Effectiveness

Degree to which actual outcomes solve the needs of the stakeholders.

Efficiency

Degree to which the resources of the stakeholders (time, money) are not wasted.

Predictability

Degree to which actual outcomes can be predicted.

Reliability

Degree to which the stakeholders can rely on the provider's continuous quality of outcomes.

Responsiveness

Degree to which the service provider is willing and able to respond promptly and helpfully.

Assurance

Degree to which the service provider is able to convey trust and confidence.

Empathy

Degree to which the service provider is able to be approachable, caring, understanding and to relate to customer needs.

Tangibles

Degree to which physical facilities, equipment, personnel and communication materials are perceivable.

Mindset / Culture

The ideas, customs, and social behaviour of the environment the team works in.

Values

What we value.

Respect

We are honest and open in our thinking. Our credo is that valuing diversity leads us to the best solutions.

Diversity and Inclusion

Workplace diversity is understanding, accepting, and valuing differences between people.

Inclusion is a collaborative, supportive, and respectful environment that increases the participation and contribution of all employees.

Simplicity

The state or quality of being simple – easy to understand or explain. Consistent, small in size with few interactions consisting of few concepts and elements.

Culture of failure

A culture of failure – instead of a culture of blame – focuses on learning from failure instead of preventing failure. Thus, leading to innovation by encouraging people to achieve difficult goals.

Product Focus

A holistic view of the whole lifetime of the product leads to globally optimized solutions rather than local sub-optimization, often introduced by thinking in projects and phases.

Transparency

It is easy for others to see what actions are performed and why.

All Goals Visible

Product goals, team goals and individual goals are visible to everyone. Transparency helps alignment and empathy.

All Work Visible

Everything everyone is working on is visible to everyone. Visibility leads to better decision making and fewer surprises.

Work model

How we work.

Sustainable Pace

Working at a sustainable pace allows the team to keep a high level of quality forever. Building quality in leads to a sustainable pace (no fire-fighting, no staling).

Idle Time (Slack Time)

Having idle time is essential to be able to respond to changes quickly, have time to reflect, learn and be creative.

Small Steps

Everything we do, we do in small and complete steps by applying plan-do-check-act (PDCA) to reduce risks of doing the wrong thing.

DevOps

DevOps proposes close collaboration between Development and Operations in cross-functional teams. Both work with the same tools and apply the same practices and methods. Together with a strong focus on automation and monitoring, this leads to shorter development cycles, more frequent and dependable releases, closely aligned with business objectives.

Decision Making

When, where and how we make decisions.

Empirical Decision Making

Decisions are made based on historical data, not feelings and predictions.

Decision Making Roles

Who decides what: team, individual, position in the hierarchy.

Decision Making Methods

How we decide in what situation: consensus, consent, Roman vote, consultative decision by a single person, decide-and-inform.

Defer Decisions

Decide only what needs a decision now. Use the time to learn and make better informed decisions later.

Lean Process

Optimize flow of work items while respecting variation in the system.

(Almost) No Hand-offs

Handoffs – passing work from one person to another person – hinder flow of work. They introduce delays, additional pick-up time (time needed to start working), and high risk of miscommunication (single point of communication instead of continuous communication). Thus, resulting in lower quality and higher cycle time.

Low WiP

A lot of work-in-progress leads to longer feedback cycles, and therefore more risk of delivering the wrong thing.

Low Number of Queues

Queues (work is waiting to be picked up by the next process step) hinder flow of work because they introduce delays, and pick-up time (time needed to start working). Thus, resulting in longer cycle times, and higher WiP.

Pull System

Steps later in the process pull work from former steps. Pull systems reduce waste in the system and allow work to be delivered just-in-time.

Communication

Communication is the act of conveying intended meanings from one entity or group to another using mutually understood signs and semiotic rules.

Communication Map

A map showing the (desired) flow of information from producers to consumers.

Constructive Feedback

Focus on specific behaviours rather than making general statements. Keep feedback impersonal and goal-oriented. Offer feedback as soon after the action as possible.

Active Listening

Effective listening is active, requiring the hearer to "get inside the head" of the speaker so that he or she can understand the communication from the speaker's point of view.

Conflict Management

The process of limiting the negative aspects of conflict while increasing the positive aspects of conflict. The aim of conflict management is to enhance learning and group outcomes.

Facts over Assumptions

Transform assumption into facts and make the facts visible.

Team Organisation

The team and its environment are set up in a way that they can work.

Feedback Cycles

Establish feedback cycles in processes and organisational structures so that they are as short as possible to be able to inspect and adapt to the progress of the development of the product. Thus, reducing time, cost and risks.

Continuous Learning & Improvement

Individuals and the team invest continuously in learning to be able to adapt to changes in the environment and the team. The team is supported by mentoring and coaching.

Communities (of Practices)

A group of people who share a craft or profession. They exchange knowledge and improve their skills together.

Document Management

Track, manage and store documents so that all involved people have proper access and traceability.

Team
People working together in a committed way to achieve a common goal or mission. The work is interdependent, and the team members share responsibility and hold themselves accountable for attaining the results. [MIT]

Composition
How we staff the product team.

Team Alignment
Everyone works towards the same goal(s) and vision.

Team Autonomy
Teams can decide on team matters on their own. There exists no mandatory synchronisation of decisions over team boundaries.

Team End-to-End Responsibility
The product team as a whole is responsible for all activities regarding the product: from ideas to operations. The team cannot delegate responsibility to others.

Cross-functional Teams
A team of members with different functional expertise and from all levels of an organisation. The team should ideally have all skills needed to build and operate the product on its own.

(T-/π-) M-Shaped Team Members
Team members have a few areas of deep expertise and general knowledge of all other topics.

Collaboration
How the product team works together to build the product.

Collaboration on whole value stream
Collaboration happens between everybody on the whole value stream.

Collective (Code) Ownership
Collective Ownership encourages everyone to contribute new ideas to all segments of the product. Anyone can change anything to add functionality, fix bugs, improve designs or refactor. No one person becomes a bottleneck for changes.

Retrospectives
Inspect and adapt the way the team works.

Shared Task Responsibility
The whole team – not an individual – is responsible for the outcome of a task. No blame game but helping each other out.

Team Charter
The team defines the guidelines and boundaries of how they want to work together: team purpose, scope, goals, …

Product Discovery
What product can we build that has so much value that the customer/user is willing to pay for?

Product Vision
A clear, concise description of a company's long-term aspirational goal for its product. Everybody involved should be able to tell why we're creating the product and the shared objective.

Design Studio
A lean UX method that helps a team to come up with a lot of ideas to solve a well-framed problem.

Design Thinking
Design thinking is a method for practical, creative resolution of problems.

Impact Mapping
Strategic planning technique based upon analysis of impacts on customers dependent on decision variants.

Product Prototype
An early sample, model or release of a product to test a concept or process or to act as a thing to be learned from.

User Research
Gather user inputs via support cases, surveys, user diaries, job shadowing or contextual inquiries.

Business Model Canvas
Tool to visualise components of a business model in a visual structure to design, discuss and analyse new or existing business models. It generates a shared understanding and helps to make decisions while helping to ensure that nothing vital is forgotten.

Empathy Map
Tool to help teams develop deep, shared understanding for and empathy with their customer needs and desires as a means to improve the product and service.

Product Production
Build the product.

Preconditions
Conditions a team must discuss and agree upon in advance.

1) Define Quality
Define what quality means in your context/team/product by specifying quality scenarios. Make trade-off decisions.

2) Define checks
Define how we check whether we met our defined quality goals.

3) Commit to quality
Everybody involved commits to the "Defined Quality". There will be no shortcuts or compromises for local optimisation.

Define how to handle exceptions.

Environment
How to deal with the surroundings of the product team.

Systems Thinking
Concepts and tools to develop an understanding of the interdependent structures of dynamic systems. Helps individuals and teams to identify leverage points that lead to desired outcomes.

Stakeholder Management
Create positive relationships with stakeholders through the appropriate management of their expectations and agreed objectives.

Risk Management
Manage personnel, technical, cultural, timing, success risks by mitigating likelihood and cost in case of occurrence.

Process
The steps, decisions and actions we make during product production.

Continuous Delivery
Delivering software in short cycles by optimizing the value stream from *idea* to *production*. Reduces cost, time and risks of delivering.

Rolling Wave Planning
The process of project planning in waves as the project proceeds, and later details become clearer. This is an approach that iteratively plans for a project as it unfolds.

Decompose Work Vertically (by value)
We split the whole product into mostly independent pieces that provide value by themselves.

WiP Limits
Work in progress limits define the maximum of work items in a specific state (e.g. analysis), thus limiting work in progress overall.

Zero Bug Policy
When a bug is found it is either fixed immediately or never. Don't waste time managing bug lists and bug-cascades.

Definition of Done
Each Team has its Definition of Done or consistent acceptance criteria across all work items. A Definition of Done drives the quality of work and is used to assess when a work item has been completed.

Definition of Ready
Having a Definition of Ready means that work items must be immediately actionable. The team must be able to determine what needs to be done. Ready work items should be clear, concise, and most importantly, actionable.

Root Cause Analysis
Solve the underlying problems, not symptoms.

Continuous Documentation
Documentation is created automatically as part of Continuous Delivery by combining manually built pieces and automatically extracted information from source code, configuration etc.

RACI Matrix
Describes the participation by various roles in completing tasks or deliverables. **R**esponsible, **A**ccountable, **C**onsulted, **I**nformed.

Architecture & Design
Design the parts of the product and their interactions.

Clean Architecture
In a clean architecture, dependencies between parts of the software (methods, classes, components) are directed from more concrete (e.g. system boundary) to more abstract pieces (e.g. business logic). Clean architecture leads to testability, flexibility, and independence from UI, DB, and other external agencies.

Modularization, Decomposition/Composition
The system is broken down into almost independent parts that are orchestrated to provide the system's behaviour.

Evolutionary Architecture/Design
Evolutionary architecture and design support only today's requirements but are open for adaptation for future needs. This leads to reduced development costs and time – now and in the future.

Design/Architecture Options
Design software in a way that keeps as many options (for future decisions) open for as long as possible.

Predictive & Reflective Design
Predictive design: Designing for code that is yet to be written. What people usually think of as "design."
Reflective design: Designing based on code that has already been written in order to improve that code.

Feature Toggles
Feature toggles allow switching features on and off in software. Helps to achieve continuous delivery and reduces risks through releasing features gradually.

Team Architecture Workshops
Frequent workshop for the whole team to respond to how the architecture supported the implementation of features and to anticipate needed changes to the architecture for future features.

Code Quality
How easily the developers can interact with the code base.

Clean Code
The code is clean if it can be understood easily – by everyone on the team. With understandability comes readability, changeability, extensibility and maintainability. All the things needed to keep software development going over a long time at a constant pace.

Coding Conventions
How the source code should look. A common style helps readability, understandability and navigability.

Continuous Refactoring
Improve quality aspects of the code in small steps without changing its behaviour before (preparatory), during (TDD-refactor) and after (simplify, make more comprehensible) adding new functionality.

Pair Programming/Pairing/Teaming (Mob Programming)
Two (or more) people work together to solve a problem in different roles: The Driver (the one writing) focuses on the details, the Navigator focuses on the big picture.

Push/PR Review
A peer team member reviews the changes before code and/or artefacts are merged into the shared repository.

Code Review
Systematic examination of source code to find mistakes overlooked during development. Kinds of reviews: walkthroughs, inspections.

Compile Time Safety over Tests
Checking correctness with the compiler should be given priority over testing. The feedback cycle is much quicker.

Test Driven Development
Short iterations of transforming part of a requirement into a failing test, writing just enough production code to make the test succeed and refactoring the code base for simplicity.

Operation Tests
Test an operation – everything that happens as the result of a stimulus to a process running in the system.

Functionality Tests
Test an atomic business functionality (a business functionality that cannot be split any more).

Algorithm Tests
Test a single algorithm (code with an `if`, `switch`, dictionary lookup, …).

Test After
Tests are written after the production code was written to increase confidence that the code behaves correctly.

Process Boundary Tests
Test the boundary of a process by verifying that the outside can correctly communicate with the inside of the process. The triggered operation is simulated to decouple these tests from business logic changes.

Exploratory Testing
Simultaneous learning, test design, test execution and test result interpretation.

System Parts Interaction Tests
Tests that verify that the individual parts of the system (services, processes) can interact with each other and can be run locally.

System Smoke Testing
Test the most important functionalities after deployment to ensure basic functionality of the software system.

Scenario Tests
Test whole user scenarios at the system level.

Approval Tests
Compare current behaviour or test output against a validated previous run.

Defect Driven Testing
When a defect is found, it is reproduced by a test and then fixed. The defect cannot reoccur.

Usability Validation
Validate the intended use with sample users.

System Usability Scale (SUS)
Creates a score based on a set of standardised questions so that you can check if you improved the usability of the product with the product increment.

Usability Testing
Usability Testing is a type of testing done from an end-user's perspective to determine if the system is easily usable.

Non-Functional Tests
Test the way software operates, not specific functionality.

Performance Testing
Determines how the system performs regarding responsiveness and stability under a particular workload.

Load Testing
Simulates multiple users accessing the system concurrently and measures the quality of service.

Security Testing
Reveals flaws in the security mechanisms of the system to protect data and maintain functionality as intended.

Maintainability Testing
Determines how easily the system can be maintained (corrective, perfective, adaptive, preventive maintenance).

Interoperability Testing
Determines how well the software inter-operates with other software components, software or systems.

Compatibility Testing
Determines how compatible the software is with different platforms, environments, operating systems, browsers, other systems, …

Reliability Testing
Determines the reliability of the software (probability of failure-free software operation for a specific period of time in a specific environment).

Installability Testing
Tests the installation procedures of the software (install, uninstall, partial and full upgrade).

Monitoring
Define, measure and react to metrics on the production environment.

Incidents
Failures and warnings from the production environment and user reports.

System Behaviour Monitoring
Monitors the production environment with telemetry about performance, load and reliability.

Test Tenants
A tenant on the productive environment that can be used to test new and existing functionality without effect on real tenants.

Testing in Production
Executing tests on the production environment.

Product Release
Ship the product to the customers and users.

Version Control
Manage changes to source code and other artefacts so that older versions can be reproduced.

Versioning (Product)
The version number of the product (and its parts) identify the version of the source code and other artefacts it was built from.

Continuous Integration
Members of a team integrate their work frequently (multiple integrations per day). Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

Continuous Deployment
Extension to Continuous Integration to deploy the integrated software to a test or production environment automatically. And in case of failure, trigger human intervention.

Configuration Management
Establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life.

Urs Enzler, Marco Ravicini www.bbv.ch June 2019 V1.1