# Software Quality Characteristics

Go through the list and think about your product/features. Add specifics for your context, and transform the list to your own.

## **Capability.** *Can the product perform valuable functions?*

- *Completeness:* all important functions wanted by end users are available.
- *Accuracy:* any output or calculation in the product is correct and presented with significant digits.
- *Efficiency:* performs its actions in an efficient manner (without doing what it's not supposed to do.)
- *Interoperability:* different features interact with each other in the best way.
- *Concurrency:* ability to perform multiple parallel tasks, and run at the same time as other processes.
- *Extensibility:* ability for customers to add features or change behavior.

## **Reliability**. *Can you trust the product in many and difficult situations?*

- *Stability:* the product shouldn't cause crashes, unhandled exceptions or script errors.
- *Robustness:* the product handles foreseen and unforeseen errors gracefully.
- *Recoverability:* it is possible to recover and continue using the product after a fatal error.
- *Resource Usage:* appropriate usage of memory, storage and other resources.
- *Data Integrity:* all types of data remain intact throughout the product.
- *Safety:* the product will not be part of damaging people or possessions.
- *Disaster Recovery:* what if something really, really bad happens?
- *Trustworthiness:* is the product's behavior consistent, predictable, and trustworthy?

## **Usability**. *Is the product easy to use?*

- *Affordance:* product invites to discover possibilities of the product.
- *Intuitiveness:* it is easy to understand and explain what the product can do.
- *Minimalism:* there is nothing redundant about the product's content or appearance.
- *Learnability:* it is fast and easy to learn and remember how to use the product.
- *Memorability:* once you have learnt how to do something you don't forget it.
- *Discoverability:* the product's capabilities can be discovered by systematic exploration of the user interface.
- *Operability:* an experienced user can perform common actions very fast.
- *Interactivity:* the product has easy-to-understand states and possibilities of interacting with the application (via GUI or API).
- *Errors:* there are informative error messages, difficult to make mistakes and easy to repair after making them.
- *Consistency:* behavior is the same throughout the product, and there is one look & feel.
- *Tailorability:* default settings and behavior can be specified for flexibility.
- *Accessibility:* the product is possible to use for as many people as possible, and meets applicable accessibility standards.
- *Documentation:* there is a Help that helps, and matches the functionality.

## **Charisma.** *Does the product have "it"?*

- *Satisfaction:* how does it feel after using the product?
- *Professionalism:* does the product have the appropriate flair of professionalism and feel fit for purpose?
- *Attractiveness:* are all types of aspects of the product "good-looking"?
- *Curiosity:* will users get interested and try out what they can do with the product?
- *Entrancement:* do users get hooked, have fun, in a flow, and fully engaged when using the product?
- *Hype*: does the product use too much or too little of the latest and greatest technologies/ideas?
- *Expectancy:* the product exceeds expectations and meets the needs you didn't know you had.
- *Attitude:* do the product and its information have the right attitude and speak to you with the right language and style?
- *Directness:* are (first) impressions impressive?
- *Story*: are there compelling stories about the product's inception, construction or usage?

## **Security.** *Does the product protect against unwanted usage?*

- *Authentication*: the product's identifications of the users.
- *Authorization*: the product's handling of what an authenticated user can see and do.
- *Privacy*: ability to not disclose data that is protected to unauthorized users.
- *Security holes*: product should not invite to social engineering vulnerabilities.
- *Secrecy:* the ability to under no circumstances disclose information about the underlying systems.
- *Invulnerability:* ability to withstand penetration attempts.
- *Compliance:* security standards the product adheres to.

## **Performance.** *Is the product fast enough?*

- *Capacity:* the many limits of the product, also under load, stress or slow network.
- *Responsiveness:* the speed of which an action is (perceived as) performed.
- *Availability:* the system is available for use when it should be.
- *Throughput:* the products ability to process many, many things.
- *Feedback:* is the feedback from the system on user actions appropriate?
- *Scalability*: how well does the product scale up, out or down?

### **IT-bility.** *Is the product easy to install, maintain and support?*

- *System requirements:* ability to run on supported configurations, and handle different environments or missing components.
- *Installability:* product can be installed on intended platforms with appropriate footprint.
- *Upgrades:* ease of upgrading to a newer version without loss of configuration and settings.
- *Uninstallation:* are all files (except user's or system files) and other resources should be removed when uninstalling?
- *Configuration:* can the installation be configured in various ways or places to support customer's usage?
- *Deployability:* product can be rolled-out by IT department to different types of (restricted) users and environments.
- *Maintainability:* are the product and its artifacts easy to maintain and support for customers?
- *Testability:* how effectively can the deployed product be tested by the customer?

### **Compatibility.** *How well does the product interact with software and environments?*

- *Hardware Compatibility*: the product can be used with applicable configurations of hardware components.
- *Application Compatibility:* the product, and its data, works with other applications customers are likely to use.
- *Operating System Compatibility*: the product can run on intended operating systems, and follows typical behavior.
- *Configuration Compatibility*: product's ability to blend in with any configurations of the environment.
- *Backward Compatibility:* can the product do everything the last version could?
- *Forward Compatibility:* will the product be able to use artifacts or interfaces of future versions?
- *Sustainability:* effects on the environment, e.g. energy efficiency, switch-offs, power-saving modes, support work from home.
- *Standards Conformance:* the product conforms to applicable standards, regulations or laws.

# Internal Software Quality Characteristics

These characteristics are not directly experienced by end users, but can be equally important for successful products.

### **Supportability.** *Can customers' usage and problems be supported?*

- *Identifiers:* is it easy to identify parts of the product and their versions, or specific errors?
- *Diagnostics:* is it possible to find out details regarding customer situations?
- *Troubleshootable:* is it easy to pinpoint errors (e.g. log files) and get help?
- *Debugging:* can you observe the internal states of the software when needed?
- *Versatility:* ability to use the product in more ways than it was originally designed for.

### **Testability.** *Is it easy to check and test the product?*

- *Traceability:* the product logs actions at appropriate levels and in usable format.
- *Controllability:* ability to independently set states, objects or variables.
- *Observability:* ability to observe things that should be tested.
- *Monitorability:* can the product give hints on what/how it is doing?
- *Stability:* changes to the software are controlled, and not too frequent.
- *Automation:* are there public or hidden programmatic interface that can be used?
- *Information:* ability for testers to learn what needs to be learned...
- *Auditability:* can the product and its creation be validated?

### **Maintainability.** *Can the product be maintained and extended at low cost?*

- *Flexibility:* the ability to change the product as required by customers.
- *Extensibility:* will it be easy to add features in the future?
- *Simplicity:* the code is not more complex than needed, and does not obscure test design, execution and evaluation.
- *Readability:* the code is adequately documented and easy to read and understand.
- *Transparency:* Is it easy to understand the underlying structures?
- *Modularity:* the code is split into manageable pieces.
- *Refactorability:* are you satisfied with the unit tests?
- *Analyzability:* ability to find causes for defects or other code of interest.

### **Portability.** *Is transferring of the product to different environments enabled?*

- *Reusability:* can parts of the product be re-used elsewhere?
- *Adaptability:* is it easy to change the product to support a different environment?
- *Compatibility:* does the product comply with common interfaces or official standards?
- *Internationalization*: it is easy to translate the product.
- *Localization:* are all parts of the product adjusted to meet the needs of the targeted culture/country?
- *User Interface-robustness*: will the product look equally good when translated?