U-QASAR Methodology Guidelines (v. 2)

For Quality Objective setting and Metrics design



### 

**Steps of the Quality Objective Setting Workshop**

00:00 Preparation

00:15 Brainstorming quality objectives (5 min)

Using on post-it notes

00:20 Presenting and organizing the objectives (20 min)

Using on post-it notes & whiteboard

Each participant in turn presents their objectives and groups with others already on board

00:40 Voting for the objectives (5min)

Each participant has 3 votes

00:45 Split to the pairs/subgroups (5min)

00:50 Pair-work elaboration of the objectives and designing indicators & metrics

Quality Objectives elaborated in the order of the voting result

Drafting and designing on paper templates

Documenting the result using the U-QASAR Tool

As many objectives are elaborated as the time permits

01:45 Presenting the individual results of each pair/subgroup (15min)

02:00 Closing

Post questionnaire form

Discussion

# **Quality objectives, Indicators and Metrics**

Quality objectives help understanding and concretizing the desired quality characteristics and their current status. This helps communicating what is essential and important about the quality between different stakeholders. Furthermore, it makes it possible to follow the achievement of the desired quality characteristics.

Quality objectives can be defined for a product or process. We can also define quality objectives for a sub-process, e.g. testing or requirement engineering. Or we can have different quality objectives for different parts of the product or quality objectives may differ even at the feature level. For example if we think of web-email system or service, it seems natural that user interface should focus on learnability and usability quality objectives, while the backend systems quality objectives would be related to security and performance.

## Software quality objective

We defined a software quality objective as an objective related to an *external or internal* quality attribute of final or intermediate *software product or process*. In other words, the software quality objective characterizes the product or process from some quality viewpoint.

For product there are two types of objectives:

1. Success factors, i.e., objectives that are already on a reasonably good level
   * competitive advantage
2. Challenges, i.e., objectives that are currently on a unsatisfactory level
   * Improving the weaknesses

For process there are two types of objectives:

1. Continuous, e.g., unit test coverage must be 90% all the time
2. Step-wise / snapshot, e.g., 80% of requirements must be accepted by a certain date

-> typically there would be a sharp increase in the % accepted requirement on the eve of the deadline.

## The quality model

Software quality objectives form a quality model of a software product, project, or a company standard quality model.

Quality Objectives define the relevant quality aspects in a high enough level to show the value for the stakeholders.

-> “What is important?”

Quality indicators define the concrete ways to evaluate the status of the Quality Objective.

-> “What indicates the achievement or status of the objective?”

Metrics describe the detailed data that the quality indicators use to indicate the status.

-> “What data is needed for the indicator?”

Figure 1 shows how quality objective technical debt can be formulated with different quality indicators in two different projects.

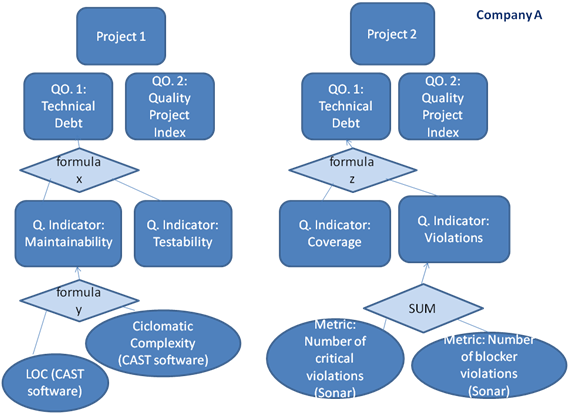


Figure 1. An example of having different quality indicators for the same quality objective

# Software product quality

Quality is a vague concept and it can mean different things to different people.

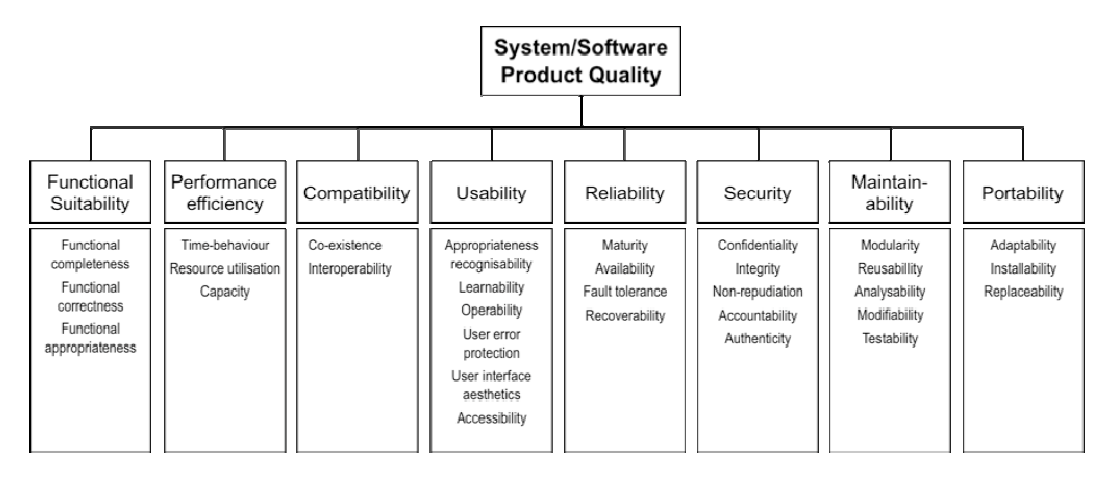


Figure 2. ISO/IEC 25010 quality model (ISO/IEC 2010)

***Functional suitability:*** degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions

***Performance efficiency:*** performance relative to the amount of resources used under stated conditions

***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

***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

***Reliability:*** degree to which a system, product or component performs specified functions under specified conditions for a specified period of time

***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

***Maintainability:*** degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers

***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

# Designing Quality Indicators and Metrics

Here is checklist of viewpoints to consider when designing indicators for Quality Objectives

**Purpose of the Measurements**

The most important aspect in measurement is to understand and define the purpose and goal of the measurements. Common examples of quality measurement goals are following:

* give feedback for developers and other stakeholders to track progress and improve
* set standard thresholds and identify deviations
* compare with other persons, teams, or organizations

The purpose of the measurement affects all aspects of data collection, validity of the measurements, interpretations of the results and so on.

**Measurability**

The measurability of indicators can differ significantly, which naturally affect how the measurements can be implemented and the overall feasibility of the measurement program.

**Measurable with quantitative numbers**

The easiest situation is when the desired quality indicators can be measured by numeric metrics. Examples of such metrics are lines of code, code coverage of tests, and numbers of defects. When numeric metrics exist the indicator is easy to make visible and track. However, one should still be careful not to make too straightforward conclusions based on seemingly "hard" data.

**Measurable by human assessment**

If direct and objective quantitative measurement cannot be found, it might still be possible to evaluate the indicator by human assessment. This assessment can be pure experience based and subjective or based on a specific assessment criteria. The results of the assessment can be expressed on a numeric scale or reported, e.g., as a written statement of the status of the indicator in question.

**Not measurable**

If the desired indicator cannot be measured with objective or subjective metrics, it is still possible to indirectly measure the means of achieving the desired quality. For example, if the desired quality characteristic is maintainability and it is too challenging to directly measure, it could be possible to use indirect process metrics of the practices that are used to achieve or improve maintainability. Such as code reviews, refactoring, etc.

**Manual or automated data collection**

The degree of automation in data collection affects the required effort in data collection as well as the availability and reliability of the measurement data. The more automated the data collection and presentation can be made the more reliably and frequently the data will be available.

Using tools for data collection makes it easier to get a consistent data collection, enabling comparison. Tools that project members use as part of their project-related activities already collect some of the measurement data you need. These tools that developers already use are good candidates for collecting data without putting an extra burden on them.

* Can you automate the metrics?
* If the metrics cannot be automated, consider carefully the feasibility of manual data collection.

**Target of the measurement**

The target of quality measurement can be either the development *process* or the developed software or system *product*. The specific targets for each sets of metrics need to be clearly defined. The process metrics are typically indirect indicators of software quality and used to ensure that the defined processes are followed. This is indirectly believed as leading to high quality end result.

Product metrics, instead, aim at measuring the achieved product quality by more direct means.

* Are you measuring the product or process?
* Try to find direct indicators and metrics for the product quality.

**Frequency of the Feedback**

Quality indicators and metrics are implemented to get information about the quality of software product and process. The frequency of getting this information, i.e., feedback-loop is a critical characteristic that affect the usefulness of the produced information. Depending on the purpose and needs of the collected measurement data the useful frequency can vary from seconds to weeks or even months.

* For what purpose the information is used?
* How early and often the information is needed for that purpose? What frequency would be unacceptably slow?

**Who needs and uses the collected information?**

When measurements target individual people and their activities the data is always sensitive. It is important to decide on what level of the organization the data is used and who gets access to it. People must know who is using the measurements and for what purpose.

When the data is used outside the team, the situation is always more sensitive and also the visibility of how the data is used and by whom requires more careful attention. It is always important to remember, that measuring something affects the measured attribute. If people feel that they are evaluated and judged based on metrics, they change their behaviour, intentionally or not, so that the measured attribute gets better - or what they think is better.

* Who will need and get access to the collected information?
* Would it be possible to restrict the access?

# Quality Objective Template

|  |  |
| --- | --- |
| Name: | Short name for the objective. |
| Description: | Description that documents the objective on higher level than a single measure. |
| Rationale: | Motivates why the objective is important. Gains and risks; E.g., costs, business benefits, customer viewpoint. |
| Related factors: | Tentative ideas of what contributes, or prevents achieving this objective |
| One or more quality indicators | Indicators and associated metrics to indicate achieving this objective:  For each quality indicator:  Description to understand exactly what is the indicator and a measure for it, including the formula if aggregating several metrics.  Calculation formula if combines multiple metrics or is otherwise needed.  Current level of the indicator  Target levels of the indicator   * Lower breakpoint (useless, if below this level) * Higher breakpoint (competitive advantage, if above) * Target level (current realistic target)   Measuring interval, continuous or snapshot, how often  Easiness of collecting (We have it / Requires some work / Impossible)  Details of all the metrics that the indicator consists of. |

**Example of an elaborated objective**

|  |  |
| --- | --- |
| Name: | Easy updateability |
| Description: | Updating the software should be quick and easy, ideally possible without deep technical or product knowledge |
| Rationale: | Direct cost savings related to updates. Reduced risk of errors during updates. |
| Related factors: | Robustness of the software, configurability, quality of installer software |
| Quality indicator 1 | Updating effort  Amount of average working hours used in making an update for a single customer installation. Measured as hours from the hour reporting system.  Current level: 3 h  Target levels:   * Lower breakpoint: 8h * Higher breakpoint: 1h * Target level: 15 min |
| Easiness of collecting: | Manual data exists, adapter to hour reporting system requires around 1 person week of work. |

# Evaluating the designed metrics

Here is a ten-question checklist to help evaluating the validity of your metrics (Kaner & Bond):

1. *What is the purpose of this measure?* Examples could be evaluating project status or staff performance, a self-assessment, or to inform external actors.
2. *What is the scope of this measure?* Are you collecting data one single time, a single project, or is it supposed to be measured continuously in all projects from now on? It may be that metrics that are good on a smaller scale will fail to give the right result, because other factors also affect the outcome.
3. *What attribute are we trying to measure?* If you have clear idea of what you actually want to measure, it is easier to establish a good relationship between what you measure and the idea.
4. *What is the natural scale of the attribute we are trying to measure?* Some attributes are easy to decide how to measure, such as length. It is more difficult to decide on the correct scales for attributes such as skills, code quality or testing thoroughness.
5. *What is the natural variability of the attribute?* It is natural that there is some variation in an attribute. What is the natural variation of the amount of tests a person can go through each day? And even more important - what causes these natural variations to occur?
6. *What is the metric (the function that assigns a value to the attribute)? What measuring instruments do we use to perform the measurement?* Examples of metrics are counting (bugs, number of items in the backlog, lines of code), matching (“This requirement is equally complex to that one”), comparing (code quality) and timing (time to finish a requirement). This can be done manually or automatically.
7. *What is the natural scale for this metric?* The scale of the attribute you want to measure and the metric you decide to use can differ. For example, thoroughness of testing cannot be measured on a natural scale, but having an expert evaluate and rank different test artifacts according to quality follows an ordinal scale.
8. *What is the natural variability of readings from this instrument?* This means the normal measurement error, and includes errors that we may do something about (systematic error) and those beyond our possible or reasonable control (random error).
9. *What is the relationship of the attribute to the metric value?* This is the important construct validity problem: How do we know that the metric measures that attribute in a good way?
10. *What are the natural and foreseeable side effects of using this instrument?* Introducing metrics to measure an attribute can be a positive thing. On the other hand it can also give way to a set of unforeseen and negative results, and the less tightly linked a measure is to the underlying attribute, the more side effects can occur. For example, if quality of a coder is measured by number of finished requirements per time, this could lead to developers cherry picking the simple tasks.