JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2003**

Vol. 2, No. 5, September - October 2003

Using Quality Models to Engineer Quality Requirements

Donald Firesmith, Software Engineering Institute, U.S.A.

Abstract

There are a great number of different kinds of quality requirements. Consisting of a hierarchy of quality factors including associated quality characteristics and quality measures, a quality model provides a structured foundation on which to identify, analyze, and specify these quality requirements.

1 THE PROBLEM

As has been noted in previous columns, quality requirements are critical to the success of any application. Even if an application fulfilled all of its functional requirements by providing all of its required features and fulfilling each and every one of its use cases, it can still be totally unacceptable if its availability is too little, its capacity is too low, its performance is to slow, it is not interoperable with other systems, it has numerous security vulnerabilities, and it is not considered to be user friendly by its end users. In fact, if functional requirements were the only requirements that mattered, then there would be no need for architecture and delivering an application consisting of a single huge monolithic module would be acceptable.

Thankfully, most requirements engineers, architects, developers, and testers clearly understand that at least some quality requirements are critical to the success of any application. But which ones?

The situation is relatively straight forward with functional requirements. They are relatively uniform in nature[1] and have several well known methods (e.g., functional decomposition and use case modeling) that can be used to successfully identify, analyze, and specify them. But there are a large number of quite different types of quality requirements, and these different types of quality requirements require quite different types of analysis methods. For example, you can use asset-based threat analysis and misuse cases to analyze security requirements, but these techniques will be quite

[1] Most functional requirements can be written in the form "The application shall enable the [a specific kind of user] to…"

Cite this column as follows: Donald Firesmith: "Using Quality Models to Engineer Quality Requirements", in *Journal of Object Technology*, vol. 2, no. 5, September-October 2003, pp. 67-75. http://www.jot.fm/issues/issue_2003_09/column6

inappropriate for analyzing other types of quality requirements such as performance (e.g., throughput, response-time, jitter, and scheduling) requirements or dependability (e.g., availability, reliability, and robustness) requirements.

How are the members of a requirements team to deal with the inherent complexity of engineering many different types of quality requirements, many of which have their own different analysis techniques? How can requirements engineers ensure the following?

- No important types of quality requirements fall through the cracks.
- The quality requirements must be organized into a hierarchy that is both logical and understandable.
- The terms for (and meanings of) the different types of quality requirements must be standardized so that communication among stakeholders is clear.
- The quality requirements properly capture the different types of quality.
- The quality requirements are unambiguous, measurable, and testable.

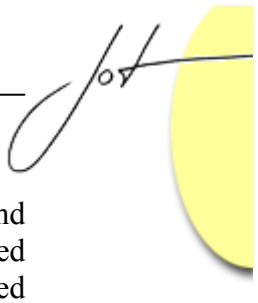
2 QUALITY MODELS

As noted above, when specifying quality requirements, a way is needed to organize, clarify, and standardize the relevant meanings of the term "quality" when applied to software-intensive systems. If requirements engineers do this first, they will form a proper foundation for identifying, analyzing, and specifying the large number of quality requirements that are needed on any significant endeavor.

This then is one role of a quality model, a concept that comes from the quality and measurement communities: to make the general term "quality" specific and useful when engineering requirements. A quality model first decomposes the general concept of quality to create a taxonomy of its component quality factors and subfactors (i.e., aspects, attributes, or characteristics). The quality model then provides specific quality criteria (i.e., descriptions) and measures (i.e., means of measurement) that can be used to turn these general high-level quality factors into detailed and specific measurable descriptions that can be used to specify the associated aspect of quality or to determine during testing if that aspect of quality actually exists. By mandating minimum levels of quality measures for quality criteria for relevant quality factors and subfactors, requirements engineers can obtain unambiguous and testable quality requirements.

Scope

The typical scope of a quality model is one or more related applications. Thus, a quality model is used to document or analyze the required or actual quality of an application. Another common scope for a quality model is one or more related components, whether these components are being bought or developed, either as part of a larger application or as generally reusable components. Another less common scope for a quality model would be to specify or measure the quality of a [data, contact, or reuse] center's services. Finally, one can also use a quality model to document or analyze the planned or actual



quality of a business unit (e.g., specifying the required interoperability, performance, and security of a business organization). Thus, just as quality requirements can be engineered for an application, component, center, or business, quality models can also be developed for each of these four scopes.

A quality model can also have another kind of scope. A quality model can be:

- A general industry standard quality model, such as the ISO 9126 quality model [1] or the OPF quality model [2].
- An organizational quality model, such as one that is used on all projects within a given development organization (e.g., all applications within a program or related projects or a product line of applications).
- An endeavor-specific quality model, such as one that is used on a single project developing a single application.

Components of a Quality Model

As illustrated in the following figure, a quality model consists of a hierarchy of the following components:

- **Quality Factor Groups**, which are groups of related quality factors.
- **Quality Factors**, which are high-level aspects, attributes, or characteristics of an application, component, center, or business organization.
- **Quality Subfactors**, which are lower-level quality factors that are components of other quality factors or subfactors.
- **Quality Criteria**, which are specific descriptions that provide evidence either for or against the existence of a specific quality factor or subfactor.
- **Quality Measures**, which are metrics that quantify quality criteria and thus make them measurable, objective, and unambiguous.

By requiring a combination of a specific quality criterion and one of its associated quality measures, a quality requirement mandates a required amount of the associated quality factor or quality subfactor. Thus, quality requirements are based on the components of a quality model, and an endeavor's quality model forms the theoretical foundation for its associated quality requirements. Therefore, producing and documenting a quality model is an endeavor's first step towards producing a complete and consistent set of quality requirements.

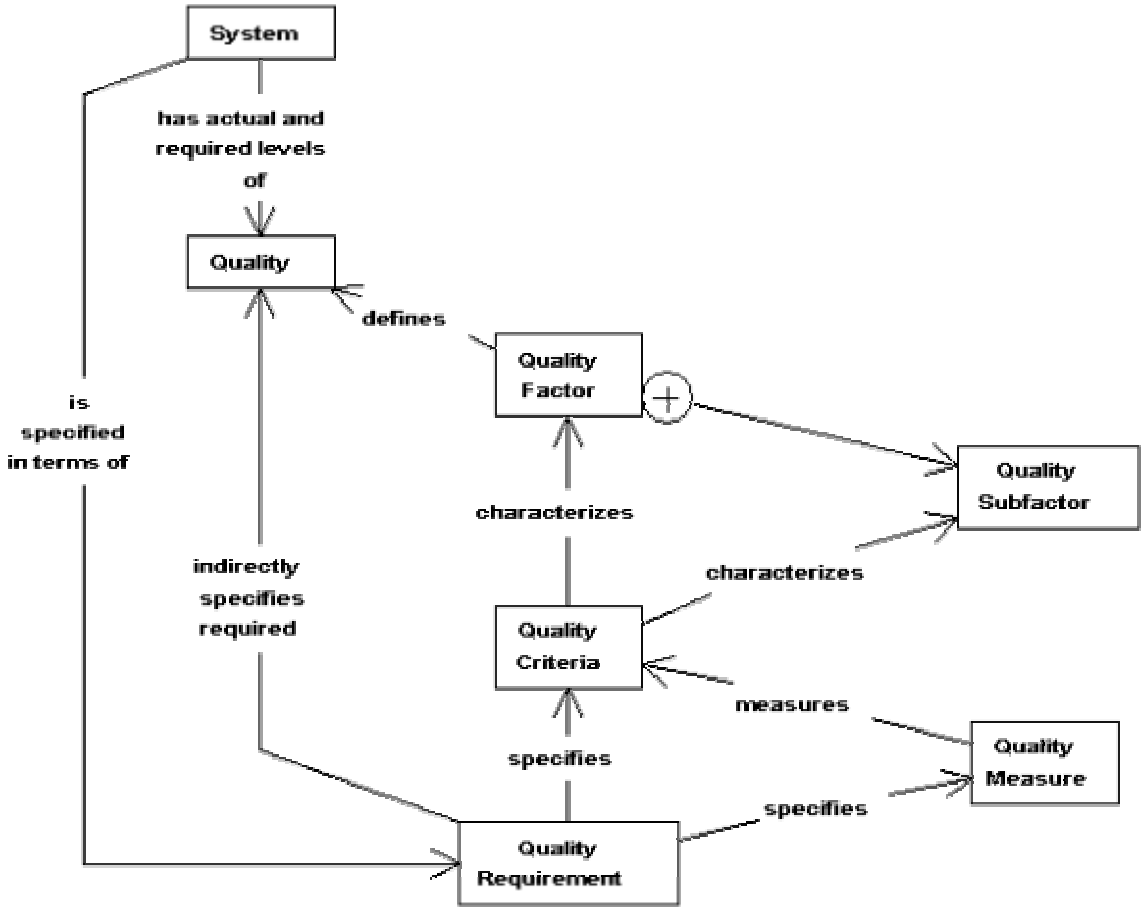


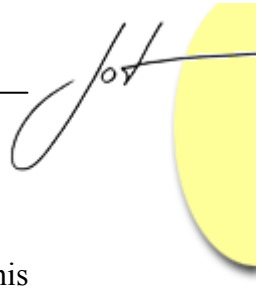
Fig. 1: Relationships Involving Components of a Quality Model

Quality Factor Groups

A quality factor group is a major grouping of related quality factors. Quality factor groups provide the highest-level decomposition of quality models into their component parts.

For example, the quality factors might be grouped as follows:

- **Developer-Oriented Quality Factors**, which are quality factors that primarily affect members of the development and maintenance organizations during the current or future development of an application or component.
- **User-Oriented Quality Factors**, which are quality factors that primarily affect members of the user organizations during actual usage of an application or component.



Quality Factors

Earlier, I observed that merely meeting functional requirements is insufficient. This means that quality goes beyond what something does to how well it does what it does. Looked at another way, quality has to do with the degree to which something possesses a combination of desirable characteristics, attributes, aspects, or traits. It is these factors, such as availability and reliability that determine whether or not something is of high quality.

Thus, a quality factor (a.k.a., quality attribute) is an important high-level aspect, attribute, or characteristic of the quality of one or more work products (e.g., application, component, or documents). A quality factor thus characterizes one part of the overall quality.

Just as the term "quality" is at too high of a level of abstraction to be really useful for identifying and organizing a large number of non-functional requirements, quality factors are themselves at too high of a level to be unambiguous and testable quality requirements. This is a major reason why it is inadequate to specify that an application be "portable" or "secure." However, portability or security are reasonable classifications for grouping quality requirements or ensuring that certain classes of quality requirements do not "fall through the cracks."

There are many different quality factors such as availability, extensibility, performance, reusability, security, or usability. Because many of the quality factors end in the letters "ility", they are often referred to as the "ilities". Notice that quality factors are typically written either as individual nouns or as noun phrases.

Like the term quality, quality factors are themselves are often decomposed into smaller, more manageable components called quality subfactors. Quality factors are thus used to organize these quality subfactors and their associated quality criteria, quality measures, and quality requirements.

Quality Subfactors

A quality subfactor is a quality factor that is a component part of another quality factor or higher-level quality subfactor. As more detailed kinds of quality factors, quality subfactors are also important high-level aspects, attributes, or characteristics of one or more work products (e.g., application, component, or documents) that characterizes part of their overall quality. As quality factors, they are also typically designated with nouns or noun phrases.

Just as decomposing quality into quality factors is too large of a grouping, quality subfactors are themselves still too large to be directly turned into quality requirements. Thus, stating that an application shall have high throughput (a subfactor of performance) is actually still a vague goal rather than an unambiguous and testable quality requirement.

If the quality subfactors of different quality factors overlap, there could be confusion as to which quality factor is intended when talking about a quality subfactor. Because of

this, a good quality model forms a tree-like hierarchy and different quality factors are decomposed into different quality subfactors.

For example, the performance quality factor (which deals with proper handling of time) is decomposed into quality subfactors as follows:

- **Performance** is the degree to which timing characteristics are adequate. Performance includes the following subfactors:
  - **Jitter** is the precision (i.e., variability) of the time when one or more events occur.
  - **Latency** is the time it takes to actually provide a requested service or allow access to a resource.
  - **Response time** is the degree to which the time it takes to *initially respond* to a request for a service or to access a resource.
  - **Scheduleability** is the degree to which events and behaviors can be scheduled.
  - **Throughput** is the number of times that a service is provided within a specified unit of time.

Quality Criteria

A quality criterion is a specific description that provides evidence either for or against the existence of a specific quality factor or subfactor. Thus, quality criteria go a long way towards making the high-level quality factors detailed enough to be unambiguous and testable. They lack only the addition of quality measures to make them sufficiently complete and specific to form the basis for detailed quality requirements.

If quality is the trunk of the tree and the quality factors and subfactors are the branches, then quality criteria are the twigs. There are many more quality criteria than quality factors because there are typically numerous criteria per factor. Quality criteria also tend to be more domain-specific and less reusable than quality factors because they tend to be specific descriptions of specific applications, components, centers, or business units.

To deal with the large number of criteria and to make them reusable, criteria can often be grouped into common reusable criteria types that have endeavor-specific instances. For example, consider the types of quality criteria that are associated with the integrity subfactor of the security quality factor. These quality criteria types describing integrity could include:

- Protect Data Transmissions from Corruption
- Detect Corruption of Transmitted Data
- Respond to Corruption of Transmitted Data
- Protect Stored Data from Corruption
- Detect Corruption of Stored Data
- Respond to Corruption of Stored Data
- Protect Software Components from Corruption

- Protect Hardware Components from Corruption

These types (classes) of quality criteria can often be parameterized in the quality models and specific instances of the parameterized classes of criteria can then used to produce quality requirements. For example, the first quality criteria type above could be parameterized as follows:

- "The A protects B transmissions over C networks from corruption by D attack." whereby:
  - A ∈ {business, center, application, component}
  - B ∈ {all, personal, business confidential, classified}
  - C ∈ {all, public, internal}
  - D ∈ {all, sophisticated, unsophisticated}

Then, this parameterized quality criteria type could be instantiated to produce a specific integrity quality criterion for protecting transmissions from corruption as "The application protects personal transmissions over all public networks from corruption by unsophisticated attack."

Quality Measures

A quality measure is a metric that quantifies a quality criterion. Quality measures thus provide numerical values specifying or estimating the quality of a work product or process by measuring the degree to which it possess a specific quality factor or subfactor. The measurement community has published many documents [3] concerning how to measure the quality criteria of the various kinds of quality factors.

Quality Requirements

To specify a quality requirement is to specify a minimum acceptable amount of a quality factor, and this is done by specifying a minimum amount of a quality measure for a quality criterion for that quality factor. Continuing the above example, a quality requirement for integrity could be "The Internet auction website shall protect a minimum of 99.9% of all transmissions with buyers over the web from corruption by an unsophisticated attack."

3 ENGINEERING QUALITY REQUIREMENTS USING QUALITY MODELS

To engineer the quality requirements for an application, one could perform the following process in an iterative, incremental, parallel, and time-boxed manner:

1. **Select Quality Model**. Select a relevant quality model on which to base the quality requirements. If possible, reuse an existing quality model. Extend or tailor the quality model as required.

2. **Select Relevant Quality Factors and Subfactors**. For each quality factor and subfactor in the quality model, determine its relevance to the current endeavor. Consider the functional, data, and interface requirements. Brainstorm with stakeholder representatives and subject matter experts.
3. **Produce Quality Criteria**. For each quality factor and subfactor, evaluate the relevant functional, data, and interface requirements to identify/determine associated quality criteria. Where practical, reuse parameterized quality criteria types that can be instantiated to produce endeavor-specific quality criteria. Brainstorm with stakeholder representatives and subject matter experts.
4. **Select Related Quality Measures**. For each relevant quality factor and associated quality criteria, determine the appropriate quality measure.
5. **Specify Quality Requirements**. For each quality criteria and associated quality measure for each quality factor and subfactor, determine a minimum acceptable amount of that quality measure for that quality criteria[2] and specify the associated requirement using a standard format.
6. **Evaluate the Quality Requirements**. Validate the resulting quality requirement with its stakeholders. Verify the quality of the quality requirement against associated standards and guidelines (e.g., quality characteristics such as completeness, feasibility, implementability, lack of ambiguity, testability, and understandability, etc.). Ensure that the resulting requirements specifications / repositories are complete in the sense that adequate quality requirements exist for all relevant quality factors and subfactors in the quality model.

[2] This is the most difficult part of the process and may require different analysis methods for different types of quality requirements. For example, the minimum level of security requirements may require using an asset-based threat and vulnerability analysis, whereas a quite different approach may be needed to specify the minimum levels of performance.

4 CONCLUSION

Quality requirements can and should be based on a standard quality model with standard quality factors, subfactors, criteria types, and measures. Using such a quality model produces the following benefits:

- All important and relevant types of quality requirements are engineered, resulting in a more complete requirements specification.
- The resulting quality requirements are organized into a logical and understandable hierarchy that is easy to use and learn.
- Communication among stakeholders regarding the quality requirements uses standardized terms with clearly documented meanings.
- The quality requirements properly capture the different types of quality.

REFERENCES

[1] **ISO/IEC 9126-1:** Information technology - Software quality characteristics and metrics - Part 1: Quality characteristics and subcharacteristics, International Organization for Standardization, International Electrotechnical Commission, Geneva, 1991.

[2] OPEN Process Framework Quality Model available at http://www.donald-firesmith.com/Components/WorkProducts/ModelSet/QualityModel/QualityModel.html

[3] **ISO/IEC 9126-2:** Information technology - Software quality characteristics and metrics - Part 2: External metrics, International Organization for Standardization, International Electrotechnical Commission, Geneva, 1991.

About the author

Donald Firesmith is a senior member of the technical staff at the Software Engineering Institute. He has worked exclusively with object technology since 1984 and has written 5 books on the subject. He is currently writing a book on requirements engineering. Most recently, he has developed a 1000+ page informational website on the OPEN Process Framework at http://www.donald-firesmith.com. He can be reached at dgf@sei.cmu.edu.