From Requirements to Release Criteria:

Specifying, Demonstrating, and Monitoring Product Quality

Erik Simmons
Intel Corporation
JF1-46
2111 NE 25th Ave.
Hillsboro, OR 97214-5961
erik.simmons@intel.com

Version 1.1, 08/17/01

Prepared for the Pacific Northwest Software Quality Conference, 2001

Key Words: Product Quality; Quality Requirements; Quantified Quality; Release Criteria

Author Biography

Erik Simmons has 15 years experience in multiple aspects of software and quality engineering. Erik currently works as Platform Quality Engineer in the Platform Quality Methods group, part of the Corporate Quality Network at Intel Corporation. He is responsible for Requirements Engineering practices at Intel, and lends support to several other corporate software and product quality initiatives. Erik is a member of the Pacific Northwest Software Quality Conference Board of Directors and the Steering Committee of the Rose City SPIN. He holds a Masters degree in mathematical modeling and a Bachelors degree in applied mathematics from Humboldt State University in California, and was appointed to the Clinical Faculty of Oregon Health Sciences University in 1991.

Abstract

Without good planning and monitoring, product quality is often a matter of chance. Specifying quality requirements is an excellent first step to predictable, managed product quality, but it is just the beginning. Quality specification, quality planning, data generation, quality monitoring, and quality reporting all work together to ensure that quality levels are known throughout the project. Quality release criteria are a good way to ensure that product quality drives interim milestone releases and the final product release. The breadth of quality dimensions can force teams to learn new concepts and skills in order to demonstrate the qualities associated with the release criteria. Despite this challenge, quality specification and quality monitoring are excellent practices that help monitor end product quality throughout the lifecycle.

Introduction

If you ask a typical product or project manager what his biggest challenge is, you will often get answers involving cost and schedule. Only occasionally in the commercial product development arena will you hear product managers mention product quality as a significant concern. There are several reasons for this:

- Recent market trends have placed tremendous pressure on companies to be the first to market with a product
- Venture capital funds expect to see a product quickly as a sign of progress
- Products are more complex than ever before, lengthening development schedules

There is some rationale behind a manager's focus on attributes other than quality at first. When bringing a new product to market, the innovators that make up the first wave of customers are much more concerned with features and the newness of the product than with price or quality. Later, when the early majority begins to buy, quality and price become greater concerns [Rogers95].

Despite this reasoning, without good planning and monitoring, product quality is a matter of chance. Numerous teams have launched products only to find out that required qualities were missing or inadequate, leading to expensive and embarrassing retrofits, product recall, or outright product failure in the marketplace. In order to prevent such mishaps, a product development team must perform several activities:

- Quality specification
- Quality planning
- Data generation
- Quality monitoring
- Quality reporting

These activities are not strictly sequential, but can overlap or occur in parallel. Each activity is introduced below and is then illustrated in the example that follows.

Quality Specification

Quality specification is the process of establishing and documenting the product's quality requirements[1].

Quality requirements are often overlooked when specifying the requirements for a product. This could be because the requirements team does not know what qualities should be included, does not know how to write quality requirements well, or is pressured by existing culture and practices to focus solely on functional requirements.

Quality has many dimensions. A holistic view of quality must consider all aspects of quality for all stakeholders, rather than focusing narrowly on defect detection and removal. By taking this approach, known in some circles as "big Q" rather than "little q", the resulting quality specification will thoroughly cover product quality.

[1] Quality requirements are called "non-functional requirements" in many sources, but this seems to be a poor term for what are essentially qualities representing how well (fast, reliable, etc.) the product's functions must deliver what they provide. In fact, many students laugh out loud when the term is introduced during requirements classes, joking that it must refer to the features in the product that don't work. The same term is also applied to some design constraints, further clouding its meaning.

Although there is no standard categorization of quality requirements, the set of categories shown in Figure 1 is typical. Organizations typically customize such lists and add details based on the type of product they build and areas in which they have faced problems in the past. For example, software performance can be segmented into ‘time’ (speed) and ‘space’ (resource usage). Security contains many facets, including integrity, availability, confidentiality, and accuracy [Chung00]. Availability can be separated into reliability, maintainability, and integrity [Gilb88]. One exhaustive list of quality requirement categories has161 elements [Chung00]. Overlaps within definitions are extremely common within the different sources. The best advice is to establish a well-defined list of categories and maintain it over time, based on project experience and need.

- Performance
- Availability
- Usability
- Supportability
- Security
- Maintainability
- Manufacturability
- Testability
- Extensibility
- Scalability

Figure 1. Example Quality Requirement Categories.

Quantifying Quality Requirements

To be meaningful, quality requirements must be verifiable and measurable. Stating quality requirements precisely in natural language is difficult. Authors often use weak words such as *fast, easy, simple, minimal, as needed, adequate, normal,* and a host of others [Wilson96]. These qualitative terms leave the meaning of a requirement open to different interpretation by each reader. Such ambiguity leads to missed expectations in the final product, and prevents accurate testing.

Planguage is a technique to quantify qualitative statements that works very well on quality requirements. Planguage is a keyword-driven specification language that improves communication about complex ideas. Easy to learn and use, Planguage also provides many benefits [Simmons01, Gilb01, Gilb97a, Gilb97b]. A basic description of Planguage is given in Appendix A. Planguage is used to document the quality requirements in the example later in this paper.

Setting Target Values

Before we can quantify quality requirements, the levels of achievement for each quality requirement must be defined through target setting. Target setting is most often based on analysis, benchmarking, or standards. Each type of quality requirement (performance, reliability, scalability, etc.) can require a specific approach to target setting in order to obtain accurate, realistic, and meaningful targets. Setting quantitative targets for these attributes is challenging at first, forcing a team to think in new ways about the product it is building. However, most of the methods are not difficult and the results more than justify the effort. The toughest part of the process is often finding the necessary data or applicable standards.

Quality Planning

If quality only made it as far as a requirements specification, it would be of little value. The additional activities of quality planning, data generation, quality monitoring, and quality reporting are the keys to obtaining real value from quality requirements.

Many existing sources discuss the need to use quality requirements to guide the subsequent product design, and for traceabililty between requirements and design as a form of verification. But this is not enough. Quality must permeate beyond the specification and design to become one of the guiding principles for interim milestone reviews and the final release of the product.

Capturing the important product qualities from the requirements as quality release criteria helps ensure this happens.

Quality Release Criteria

Quality release criteria (QRC) are used to assess and monitor the risk to interim milestone releases and the final product release. Quality release criteria cover a diverse set of quality attributes for a product. Some of the criteria are related to product quality requirements. Others are related to SQA, test, software configuration management, or project management data.

The QRC are grouped within assessments, which makes quality reporting and quality monitoring simpler. Assessments similar to those used at Intel are shown in Table 1 [Gatlin00]. Some assessments only apply to certain situations or categories of products. The assessments are always used as a point of departure on a project. Each project must customize and adapt the assessments and underlying quality release criteria to its needs.

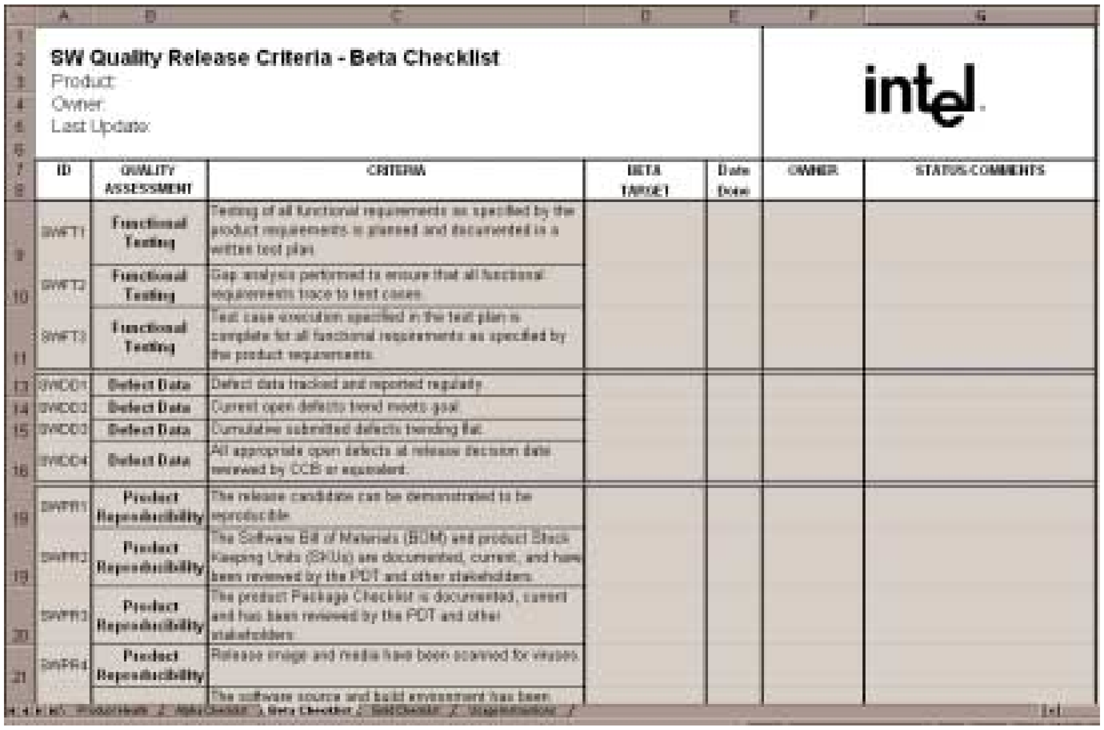
Table 1. Assessments Containing Quality Release Criteria.

Assessment	**Description**
Functional Testing	Evaluation of the testing of features and functions to determine whether the testing is adequate to ensure quality.
Product Reproducibility	Evaluation of the ability to archive, reproduce, and rebuild the software product, and that the consistency of the product package with the contents of the released product has been verified.
Legal Compliance	Evaluation of product compliance with legal requirements.
Continuous Operation	A method for detecting time-dependant product defects such as memory leaks and race conditions. It is also one indicator of product reliability over time.
Defect Data	Evaluation of product defect information and verification of correct product operation.
Ease of Use	Evaluation of the ease of use testing to ensure it covers at least the product's user interface and all features related to ease of product use.
Install Testing	Evaluation of install and uninstall testing to verify correct, reliable install and uninstall of product is possible.
Customer Documentation	Evaluation of product user documentation for accuracy and consistency with product operation.
Customer Acceptance	Evaluation of the testing performed to determine whether the product meets its customer acceptance criteria.
Standards Compliance	Evaluation of product compliance with product standards including Intel engineering or other standards. Process standards are specifically excluded.
Compatibility Testing	Evaluation of the product under test to perform its required functions while sharing the same environment with one or more systems or components.
Performance Measurement	Evaluation of the performance attributes of the software under test to determine whether the validation effort is adequate for ensuring that performance goals are met.
Regression Testing	Evaluation of the testing of the product that determines whether changes have caused unintended effects and whether the changed product still meets its requirements.

Assessments can contain different quality release criteria and associated targets for the alpha, beta, and gold milestones. For example, at Intel the Defect Data assessment (named SWDD) is defined to contain the following four criteria at the beta and gold milestones, but only SWDD1 at the alpha milestone:

SWDD1. Defect data tracked and reported regularly.
SWDD2. Current open defects trend meets goal.
SWDD3. All open defects at release decision date reviewed by the Change Authority.
SWDD4. Defects waived by the Change Authority for release are documented in Release Notes.

This system of definition is quite natural to implement in the form of a spreadsheet, with separate worksheets for the alpha, beta, and gold milestones. An example subset of the release criteria for the beta milestone is given in Figure 2.



SW Quality Release Criteria - Beta Checklist

Product:

Owner:

Last Update:

ID	QUALITY ASSESSMENT	CRITERIA	BETA TARGET	Date Done	OWNER	STATUS/COMMENTS
SWFT1	Functional Testing	Testing of all functional requirements as specified by the product requirements is planned and documented in a written test plan.				
SWFT2	Functional Testing	Gap analysis performed to ensure that all functional requirements trace to test cases.				
SWFT3	Functional Testing	Test case execution specified in the test plan is complete for all functional requirements as specified by the product requirements.				
SWDD1	Defect Data	Defect data tracked and reported regularly.				
SWDD2	Defect Data	Current open defects trend meets goal.				
SWDD3	Defect Data	Cumulative submitted defects trending flat.				
SWDD4	Defect Data	All appropriate open defects at release decision date reviewed by CCB or equivalent.				
SWPR1	Product Reproducibility	The release candidate can be demonstrated to be reproducible.				
SWPR2	Product Reproducibility	The Software Bill of Materials (BOM) and product Stock Keeping Units (SKUs) are documented, current, and have been reviewed by the PDT and other stakeholders.				
SWPR3	Product Reproducibility	The product Package Checklist is documented, current and has been reviewed by the PDT and other stakeholders.				
SWPR4	Product Reproducibility	Release image and media have been scanned for viruses.				
		The software source and build environment has been				

Figure 2. Example QRC Spreadsheet, Beta Milestone.

Data Generation

Because the quality release criteria cover an array of topics, the raw data to evaluate the QRC come from many sources. Functional testing, inspections, audits, and project metrics are all contributors to at least some of the criteria. Initially, the breadth of the quality release criteria places pressure on the data generation capabilities of most organizations, since evaluating all the necessary criteria requires many new skills, techniques, and tools. For example, relatively few software test teams have experience in demonstrating software reliability through a continuous operation demonstration.

When improved requirements specification techniques are adopted along with a broad set of quality release criteria, an organization must also consider how to train and equip personnel to support the required demonstrations. Without this, the investment in new practices within product requirements and quality release criteria will be frustrating, unsuccessful, and a waste of project resources.

Quality Reporting

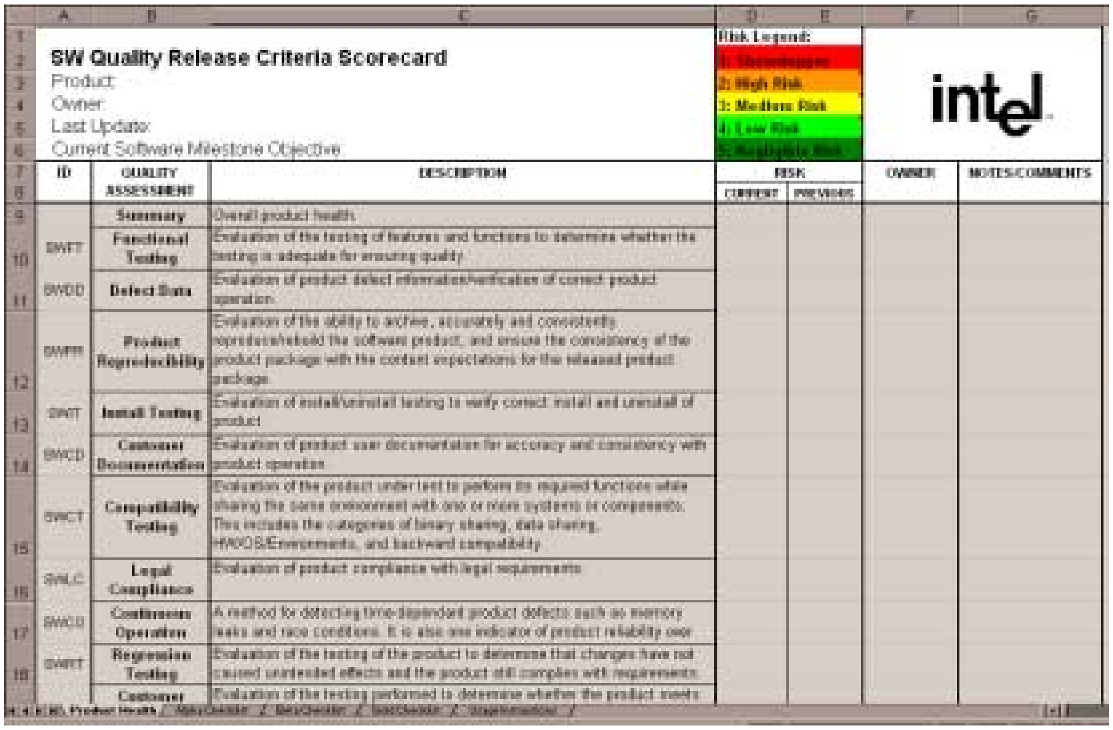
Organizations must consider how, how often, and to whom to report the QRC data. The data should be reported to a group chartered with managing the product development effort and monitoring risks to the alpha, beta, and gold milestones. This might be a project office, a project manager, risk manager, product planning council, or another similar organization. Typically, a cross-section of stakeholder groups (such as product quality, program/product management, corporate finance, and product marketing) should receive the release criteria risk data to make sound project-related decisions.

Quality reporting is often a new practice for an organization. Because of this newness and the nature of quality data, quality reporting can be a sensitive topic at first, so be sure that all parties understand the data reported. Quality data is prone to misinterpretation by those unfamiliar with data gathering practices and limitations. Misinterpretation or misuse of the quality data[2] can have lasting negative consequences for an organization. Educating the participants about the definition, collection practices, reporting formats, and allowed uses of the data can help overcome initial reluctance. Sometimes, placing allowed and prohibited uses of quality data into a company policy is also helpful, so long as compliance is monitored and there are consequences for violating the policy.

Quality Monitoring

The quality release criteria enable quality monitoring to begin as soon as the data can be generated. Exactly when this is depends on the nature of each criterion. For example, the product reproducibility criterion cannot be effectively measured until the development team generates a complete system build.

Because of the large number of quality release criteria, a quality dashboard made up of the assessment categories is useful during project review meetings to get a quick but complete picture of quality assessment status (Figure 3). Program managers and others with responsibilities related to product or risk management can monitor the progress towards alpha, beta, and gold milestone releases at a glance, while the details behind the dashboard are available on the sheets for each milestone if needed.



SW Quality Release Criteria Scorecard
Product:
Owner:
Last Update:
Current Software Milestone Objective:

Risk Legend:
🟥 1: Showstopper
🟧 2: High Risk
🟨 3: Medium Risk
🟩 4: Low Risk
🟩 5: Negligible Risk

ID	QUALITY ASSESSMENT	DESCRIPTION	RISK		OWNER	NOTES/COMMENTS
			CURRENT	PREVIOUS		
	Summary	Overall product health.				
SWFT	Functional Testing	Evaluation of the testing of features and functions to determine whether the testing is adequate for ensuring quality				
SWDD	Defect Data	Evaluation of product defect information/verification of correct product operation				
SWPR	Product Reproducibility	Evaluation of the ability to archive, accurately and consistently reproduce/rebuild the software product, and ensure the consistency of the product package with the content expectations for the released product package				
SWIT	Install Testing	Evaluation of install/uninstall testing to verify correct install and uninstall of product				
SWCD	Customer Documentation	Evaluation of product user documentation for accuracy and consistency with product operation				
SWCT	Compatibility Testing	Evaluation of the product under test to perform its required functions while sharing the same environment with one or more systems or components. This includes the categories of binary sharing, data sharing, HW/OS/Environments, and backward compatibility				
SWLC	Legal Compliance	Evaluation of product compliance with legal requirements				
SWCO	Continuous Operation	A method for detecting time-dependent product defects such as memory leaks and race conditions. It is also one indicator of product reliability over				
SWRT	Regression Testing	Evaluation of the testing of the product to determine that changes have not caused unintended effects and the product still complies with requirements				
	Customer	Evaluation of the testing performed to determine whether the product meets				

Product Health / AlphaChecklist / BetaChecklist / GoldChecklist / StageInformation /

Figure 3. Example Quality Dashboard.

[2] Improper use includes employee performance appraisals, compensation reviews, or promotion evaluations. Misinterpretation includes incorrectly reading graphs, misinterpreting statistical results such as confidence intervals or significance levels, overlooking sampling problems, confusing association with causality, or violating distributional assumptions.

Putting It Together

Near the beginning of a project, a team starts with the standard set of QRC assessments and the quality requirement categories found in the organization's requirements document template. From these, the team can form a complete framework of product quality based on the particular characteristics of the product they must produce.

The quality requirements both influence and are influenced by the quality release criteria (Figure 4). Quality requirements provide target values used to demonstrate the status of the QRC, but the QRC also influence the categories of quality requirements chosen by a team for specification. For example, the continuous operation release criterion influences the product's requirements specification, forcing a team to specify product reliability as one of the requirements for the product. At the same time, the reliability requirement establishes the target values for the continuous operation demonstration that feeds data to the quality release criterion.

The concepts and practices presented above all work in concert to help ensure that product quality is not a matter of chance. By taking a holistic view, product quality will not be a matter of chance, but rather of conscious decisions based on data. This risk-driven, data-driven approach will get problems out in the open early, while there is still time to correct them. The result will be a reduction in rework, improved schedule predictability, shortened time to market, and improved product quality.

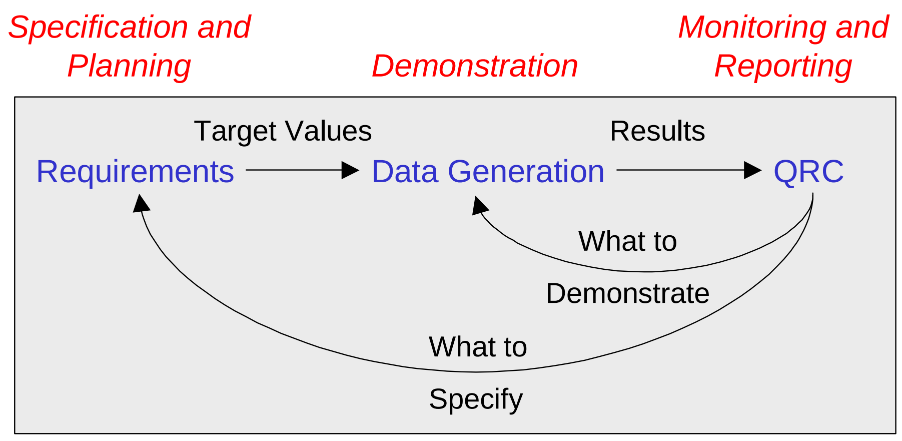


Figure 4. Relationship Between Requirements and Quality Release Criteria.

Example

Suppose that you are the Quality Manager for a new product. The product is a device similar to a television set-top box in appearance, and its primary function is to capture movies broadcast on sidebands of standard TV broadcasts for possible later viewing. The device chooses to capture a movie for the user based on a proprietary algorithm that combines several dimensions, including end user preferences for movie type, the viewer's age, and actors. When a user decides to watch a captured movie, a charge is incurred via a back-channel telephone link to a billing service.

The potential market for the product is large, since it does not rely on proprietary networks or equipment for transmission or billing. The user base has a low tolerance for failure based on experience with television and household appliances. Competition exists, so prices for viewing a movie will lead to small margins. The product will use the “shaver and blades” model, in which the product itself will be inexpensive and most of the revenue will be derived from movie viewing fees.

The project team’s risk analysis shows that the large potential user base, low margins, and user intolerance for failure lead to a significant risk to profitability: support calls. The team estimates that recouping the cost of a single user support call will require a full year of average product usage. A defect in any portion of the product, its documentation, or its usability could have disastrous financial effects.

As quality manager, you want to help reduce this risk during design and development of the product. Several factors might lead to a support call, including installation and setup, daily use, product reliability, and fault tolerance and recovery. By ensuring that the product is simple to install, quick to set up, easy to use, reliable, and self-correcting for as many errors as possible, support calls will be minimized. These issues trace into the QRC assessments as shown in Table 2.

Table 2. Support Call Issues Mapped to QRC Assessments.

Issue Leading to Support Call	**QRC Assessments**
Ease of Use	Ease of Use
Ease of Installation	Install Testing Customer Documentation
Reliability	Continuous Operation
Fault Tolerance and Recoverability	Functional Testing Defect Data

Let’s examine two of the concerns listed above in more detail: reliability and ease of installation.

Reliability
Several factors govern reliability target for the example product. First, there are many types of failure, each possibly having a different target value. The reason for this is that a user base has differing tolerances for different types of failure, based on characteristics such as the effects of the failure, the user’s innovativeness, and his experience with technology [Rogers95]. For the class of failures defined as *software defects leading to media playback failure* but no damage to hardware, suppose that market research shows the user base to have a tolerance of one failure per 100 uses. Given a projected average usage duration of 2.5 hours, this translates to a target of 250 hours of failure-free operation (or, alternatively, 100 consecutive uses). Building in an engineering margin of 100 hours to the plan, a traditional reliability requirement might state that the system Mean Time to Failure (MTTF) be at least 350 hours, measured at 95% statistical confidence. However, a good reliability requirement is made up of much more than just the target value, and this is where Planguage is so effective. The Planguage requirement looks like this:

TAG: Reliability.Playback
GIST: The reliability of media playback.
SCALE: Mean Time to Failure for playback-related failures only, using a Representative Operational Profile with randomization.
METER: A Probability Ratio Sequential Test with α=10%, β=10%, and a discrimination ratio of 2.0
PLAN [Gold]: At least 350 hours
MUST [Gold]: At least 250 hours
MUST [Beta]: At least 75 hours
Representative Operational Profile: DEFINED: An operational profile that is likely to occur during use of the system after deployment.

This requirement is verifiable and measurable: the measurement method and targets are both specified. Planguage allows the specification of goals for different milestones (beta and gold). It also allows separation of planned levels of success from “must have” minimums needed to avoid financial, political, or other failure. In this case, a minimum reliability of 75 hours is required before the beta milestone is reached and the product is tested by a broad cross-section of the user base. This level is the minimum judged to protect brand equity, allowing earliest possible product use without damaging the reputation of the company. So, if you can’t demonstrate greater than a 75 hour MTTF, you do not pass the beta quality gate.

The continuous operation quality release criterion captures the reliability requirement and its associated targets:

CO1: The product is able to achieve the specified continuous failure free operation target defined in the Product Requirements Document or Software Quality Plan.

Once product development progresses to the point that we can measure reliability, this criterion will help determine whether the product meets the reliability requirement at all three milestones.

Ease of Installation

Targets for ease of installation must be based on some benchmark value that represents what the typical user would expect (and tolerate) as setup time without calling a technical support line for help. By examining devices such as CD or DVD players, VCRs, or value segment PCs, a reasonable target would be less than 30 minutes, with fewer than 10% of users requiring phone support during the process. Building in some margin, and recognizing that an ideal value might be closer to 10 minutes or less in all cases, we can write the requirement for installation time in Planguage as:

TAG: Setup.Time
GIST: The time needed for a user to set up the system.
SCALE: Mean Time from opening the box to completion of the enclosed instruction set.
METER: Usability testing with at least 25 focus group subjects in each of the top 3 product demographic groups.
PLAN [Gold]: No more than 20 minutes at least 80% of the time
MUST [Gold]: No more than 30 minutes at least 80% of the time
MUST [Beta]: No more than 45 minutes at least 80% of the time
WISH: No more than 10 minutes 100% of the time

The targets in this requirement allow progress towards increasingly stringent quality goals.

We can write a separate requirement to cover the proportion of the users requiring a support call to complete the installation:

TAG: Setup.Calls
GIST: The likelihood that technical support is required to complete a setup.
SCALE: Proportion of users having to ask for help at least once during setup.
METER: Usability testing with at least 25 focus group subjects in each of the top 3 product demographic groups.
PLAN [Gold]: No more than 5%
MUST [Gold]: No more than 10%
MUST [Beta]: No more than 20%
NOTE: Relates to Setup.DocQuality

The need for a call to technical support during installation relates to the quality of the user documentation, so we need an additional requirement for documentation quality:

TAG: Setup.DocQuality

GIST: The quality of the installation guide.
SCALE: Proportion of test subjects able to locate the solution to Common Problems in the installation guide.
METER: Common Problems tested on at least 20 focus group subjects in the primary product demographic group.
PLAN[Gold]: At least 95%
MUST[Gold]: At least 80%
MUST[Beta]: At least 60%
Common Problems: DEFINED: The top 3 problems located in installation testing for Setup.Calls

The related quality release criteria for these requirements are:

SWIT1: End-user installation documentation or configuration guide exists, includes install and uninstall procedures, and has been verified for accuracy by testing.
SWCD1: Customer documentation is checked for technical accuracy and consistency against the product.

Monitoring the release criteria associated with reliability, install time, support calls, and end user installation documentation as the product development effort progresses provides a risk-based assessment of product health in the areas needed to minimize support costs. More generally, the summarized release criteria constitute a product health dashboard that can be updated and reviewed at regular intervals throughout the project.

References

Chung00	Chung, Lawrence, Nixon, Brian, et al, *Non-Functional Requirements in Software Engineering*, Klewer Academic Publishers, 2000
Gatlin00	Gatlin, Manny, and Hannon, Gregory, *Managing Software Product Quality*, presented at the Pacific Northwest Software Quality Conference, 2000
Gilb01	Gilb, Tom, *Competitive Engineering*, Addison Wesley 2001 (forthcoming)
Gilb97b	Gilb, Tom, *Quantifying the Qualitative*, available at http://www.result-planning.com
Gilb97a	Gilb, Tom, *Requirements-Driven Management: A Planning Language*, Crosstalk, June 1997
Gilb88	Gilb, Tom, *Principles of Software Engineering Management*, Addison Wesley 1988
Rogers95	Rogers, Everett M., *Diffusion of Innovations 4th Ed.*, The Free Press 1995
Simmons01	Simmons, Erik, *Quantifying Quality Requirements Using Planguage*, presented at Quality Week 2001
Wilson96	Wilson, William, Rosenberg, Linda, and Hyatt, Lawrence, *Automated Quality Analysis of Natural Language Requirements Specifications*, available at http://satc.gsfc.nasa.gov/support.

Appendix A: A Brief Description of Planguage

Planguage (PLANning lanGUAGE) was created by Tom Gilb [Gilb01, Gilb97a, Gilb97b]. Planguage is a keyword-driven language that can be used in requirements specifications, design documents, plans, and other places where qualitative statements are common[3]. Its primary benefits are that it quantifies qualitative statements, improves communication about complex ideas, prevents omissions within quality requirements, and is easy to learn.

Planguage has a rich set of keywords. The commonly used keywords are given in Table 3.

Table 3. Planguage Keywords.

TAG	A unique, persistent identifier
GIST	A short, simple description of the concept contained in the Planguage statement
STAKEHOLDER	A party materially affected by the requirement
SCALE	The scale of measure used to quantify the statement
METER	The process or device used to establish location on a SCALE
MUST	The minimum level required to avoid failure
PLAN	The level at which good success can be claimed
STRETCH	A stretch goal if everything goes perfectly
WISH	A desirable level of achievement that may not be attainable through available means
PAST	An expression of previous results for comparison
TREND	An historical range or extrapolation of data
RECORD	The best-known achievement
DEFINED	The official definition of a term
AUTHORITY	The person, group, or level of authorization

Besides keywords, Planguage also offers several convenient and useful sets of symbols:

- Fuzzy concepts requiring more details are marked using angle brackets: <fuzzy concept>
- Qualifiers, which are used to modify other keywords, are contained within square brackets: [when, which, …]
- A collection of objects is indicated by placing the items in braces: {item1, item2, …}
- The source for a statement is indicated by an arrow: Statement ← source

[3] The term Planguage is also used as the name of some programming languages for parallel processors, but that use is not related to its use in this paper.