Requirements Quality Metrics: The Basis of Informed Requirements Engineering Management *

Robert J. Halligan FIE Aust CPEng

Project Performance (Australia) Pty Ltd
2 Parkgate Drive
Ringwood North Vic 3134 Australia
Fax 61-3-9876-2664 email: rhalligan@ppi-int.com

Presented at the 1993 Complex Systems Engineering Synthesis and Assessment Technology Workshop (CSESAW '93), Calvados, MD, USA

Abstract

Available data demonstrates that defective requirements are a dominant cause of cost and schedule overrun in defense and aerospace programs. This paper presents a structured methodology for measuring the quality of requirements, individually and collectively. It is shown that requirements may be characterized by ten quality factors, each with an associated metric, and by an overall requirements quality metric. In addition, the requirements engineering process itself can be instrumented by means of five process-related metrics. The paper describes the author's experience with application of both types of metric to engineering decision-making. A tool which automates aspects of metrics collection is presented.

1. Introduction

Requirements engineering deals with the capture, analysis, expression and traceability of requirements. Requirements engineering may commence at the level of a broad statement of military need, and will continue through the definition of the system solution, right down to the lowest levels of specification of elements of that solution, for example, C, D and E specifications in the hardware world and minispecs in the software world.

Requirements engineering does not simply happen, it requires management. Classically, management is considered to comprise planning, organizing, staffing, monitoring and controlling. If we accept that "that which cannot be measured cannot be controlled", the role of requirements metrics is readily apparent.

But which metrics? Should we instrument the product (the requirements) or the process (the requirements engineering process) or both? How can requirements metrics be used to help the project team satisfy project success criteria? These and related issues are addressed below.

2. The State of the Requirements Art

Data from TRW developed in the early 1980s showed that, on a range of representative projects, 30 per cent of design problems requiring correction were due to erroneous or incomplete specifications. Another 24 per cent of errors were due to conscious deviation from product and process requirements. Other studies [1] have shown that the cost to correct an error typically increases by a factor of between 20 and 1000 over the life cycle of a system acquisition. System solutions which satisfy the contract, but not the need, are, unfortunately, commonplace.

Engineering practitioners have come to regard improved requirements engineering as one of the challenges of the 90's. The responses to this challenge have included:

- early, concurrent development of product and process requirements covering all product life cycle phases, from concept through to disposal [2];
- improved analysis of requirements by the use of operational requirements languages and associated tools, for example RDD [3]; and
- management of requirements through integration of text processing and relational database (or similar) support [4], resulting in improvements in requirements traceability and in the productivity of requirements analysis and flowdown activities.

This latter trend has brought with it a tendency, highly beneficial in the author's view, to manage all program requirements as a single set. Requirements may be readily allocated across all elements of the program, for example the prime mission products(s), project management, system engineering, test and evaluation, production, etc., and their interfaces. Within each of these program elements, requirements may be decomposed and allocated to lower-level elements, product interfaces and functional interfaces.

3. Users of Requirements

Since requirements define the product or process to be realized, it is axiomatic that the success of any program is closely linked to the adequacy of definition and communication of requirements:

- users rely on requirements as a precise expression of their need;
- the program office relies on requirements for eliciting offers;
- both the customer and the contractor rely on requirements as an expression of their agreement as to what is to be delivered; and
- the functional elements of the project organizations of both the customer and the contractor rely on requirements as an expression of what they are to deliver to their respective internal customers.

4. Requirements Quality

Requirements, to satisfy their users, must, in their expression, exhibit certain attributes. We refer to these attributes as requirements quality factors. The author has found that a set of ten requirements quality factors is necessary to adequately define the quality of requirements, individually and collectively.

Correctness refers to an absence of errors of fact in the statement of requirement.

Completeness requires that the requirement contain all of the information necessary, including constraints and conditions, to enable the requirement to be implemented such that the need will be satisfied.

Consistency requires that a requirement not be in conflict with any other requirement, nor with any element of its own structure.

Clarity requires that the requirement be readily understandable without semantic analysis.

Non-Ambiguity requires that there be only one semantic interpretation of the requirement.

Connectivity refers to the property whereby all of the terms within the requirement are adequately linked to other requirements and to word and term definitions, so causing the individual requirement to properly relate to the other requirements as a set.

Singularity refers to the attribute whereby a requirement cannot sensibly be expressed as two or more requirements having different subjects, verbs and/or objects.

Testability refers to the existence of a finite and objective process with which to verify that the requirement has been satisfied.

Modifiability requires that:

a. necessary changes to a requirement can be made completely and consistently; and
b. the same requirement is specified only once.

Feasibility requires that a requirement be able to be satisfied:

a. within natural physical constraints;
b. within the state-of-the-art as it applies to the project; and
c. within all other absolute constraints applying to the project.

5. A Requirements Structural Model

Requirements are most commonly expressed as natural language statements, although graphical and formal mathematical requirements languages are widely used.

For the natural language type of expression, requirements quality metrics may be developed through the parsing of each requirement statement into the elements of a structural model of a sound requirement, a template. A template found to be suitable for English language requirement statements is illustrated in figure 1 [after 5]. Figure 1 also shows an example requirement parsed into the template.

Original Requirements:

In the Combat Zone, an HQ Switch, which is identical to a trunk node switch, shall be given two (2) independent links to at least two (2) other nodes in the network.

Element	Text
1 Actor	an HQ Switch,
2 Conditions for Action	In the Combat Zone
3 Action	shall be given
4 Object of action	two (2) independent links
5 Constraints of Action	
6 Refinement of Object	
7 Refinement of Action	to at least two (2) other nodes in the network.
8 Other	which is identical to a trunk node switch,

Figure 1 - Requirement Structural Template

Elements of the template are defined, generally in accordance with Fuji [5], as below:

Actor. This is the subject of the sentence - the thing being specified. Examples are: "the system", "the interface", "the function", ...

Conditions of Action. This defines the conditions under which the action takes place, for example "upon receipt of a message", "in high resolution mode", "power having been applied", ...

Action. This is a verb - the action to be taken by the actor (subject). Examples are "shall calculate", "shall display", "shall fly", ...

Object of Action. This is a noun, and is the thing acted upon. Examples are: "the message", "the input signal", ...

Constraints of Action. This qualifies the action, for example "at a resolution of 400 x 1000 pixels", "within limits imposed by vehicle speed", ...

Refinement/Source of Object. These qualify the object, for example (refinement): "of flash priority", for example (source): "from DISCON".

Refinement/Destination of Action. These further qualify the action, and may be additional to Constraints of Action. Examples are "in accordance with IEEE 802.11g", "to DISCON".

Other. This element collects non-requirements material.

6. Requirements Quality Metrics

A strong requirement will have each applicable element of the requirement, and the requirement overall, satisfying each of the quality factors described earlier. This ideal provides a basis for the development of requirements quality metrics.

Figure 2 illustrates the construction of a set of metrics based on parsing of a requirement into the template.

These metrics are defined below.

IRQ Individual Requirement Quality

This metric for a single requirement is a number between 0 and 1, 1 representing a "perfect" requirement and zero representing a totally defective requirement. The metric is constructed from the parsed version of the requirement by:

a. determining which of the possible seven elements of the structure are applicable and assigning a value of 1 to each applicable element (most requirements have 5-7 applicable elements);
b. assessing each element of the parsed requirement against the quality factor criteria, and scoring each applicable element as 1 (satisfactory) or 0 (unsatisfactory). An element may be unsatisfactory because it is missing, or because it is defective in some other way. A variant on the approach is to permit individual element scores between the limits of 1 and 0, although it is doubtful whether this refinement offers any significant benefit;
c. calculating the metric by dividing the sum of the applicable element values into the sum of the element scores.

IQF1-IQF10 Individual Quality Metrics

Ten individual (requirement) quality metrics correspond to the ten requirement quality factors, as follows:

IQF1	Correctness
IQF2	Completeness
IQF3	Consistency
IQF4	Clarity
IQF5	Non-Ambiguity
IQF6	Connectivity
IQF7	Singularity
IQF8	Testability
IQF9	Modifiability
IQF10	Feasibility

Structural Element	Applicability	Score		Metric Name	Metric Value
Actor	1	0	Correctness	IQF1	0
Conditions of Action	1	0	Completeness	IQF2	0
Action	1	0	Consistency	IQF3	1
Refinement of Action	0	0	Clarity	IQF4	1
Object of Action	1	0	Non-ambiguity	IQF5	0
Refinement/Source of Object	1	0	Connectivity	IQF6	0
Refinement/Destination of Action	1	1	Singularity	IQF7	1
TOTAL	**6**	**1**	Testability	IQF8	0
Metric IRQ1	**0.17**		Modifiability	IQF9	1
Omission Ratio	**5.00**		Feasibility	IQF10	1

Figure 2 - Construction of Requirement Quality Metrics

These metrics assume, for an individual requirement, a value of 1 or 0 depending on whether the requirement overall has a defect of that type (0) or not (1). Again, scoring between these range limits may be used if desired.

The metrics for individual requirements rarely directly serve a useful purpose. It is necessary to aggregate individual requirements metrics to form metrics for groups of requirements in order to serve our objective of control. In making this transposition to aggregate metrics, we have consistently found the need to adjust completeness to allow for requirements which are missing altogether, not just incomplete in the sense of missing a condition or a refinement.

Requirements which have been omitted may be accounted for by estimating an *omission ratio* for each requirement that *is* present. The omission ratio is the number of new requirements that would be created if all possible areas of omission suggested by the requirement that *is* present were pursued to resolution. The omission ratio must be constructed such as to support aggregation of requirements having different omission ratios.

The quality metrics for sets of requirements correspond to, and are produced from, the individual metrics, as follows (for *n* requirements):

RQ Requirements Quality

$$RQ = \frac{\sum IRQ}{n}$$

QF1 Correctness

$$QF1 = \frac{\sum IQF1}{n}$$

QF2 Completeness

$$QF2 = \frac{\sum QF1}{n} - \frac{\sum omissionratio}{n}$$

Note that completeness may have a negative value.

QF3 to QF10 are derived as for QF1.

7. Application of Requirements Quality Metrics

A metric is only of value if it assists in decision making.

Areas of application of the metrics described above are summarized in Table 1.

Metrics should only be used where they contribute positively to the degree of satisfaction of project goals, including cost goals.

Metric	Application
RQ Requirements Quality	• estimation of requirements-related bidding risk/opportunity (depending on the type of contract) • estimation of requirements-related contract risk/opportunity • determination of the skills and level of resources required for requirements analysis • measurement of the quality of the product of requirements analysis, in relation to decisions such as: a. termination of formal requirements analysis; b. whether the project is ready for System Requirements Review (SRR), Software Specification Reviews (SSR) and other requirements reviews; c. whether system requirements are sufficiently mature for establishment of the functional baseline; d. whether CI requirements are sufficiently mature for establishment of the allocated baseline; • assessment of the specification writing skill levels of project team members • estimation of requirements-related subcontract risk/opportunity • use as a Technical Performance Measurement (TPM) parameter
QF1-QF10 Requirements Quality Factors	• identification of aspects of requirements which are unsatisfactory • identification of requirements-related skills in which training of project personnel is needed • use as a TPM parameter

Table 1 - Application of Requirements Quality Metrics

8. Typical Values of Requirements Quality Metrics

Our experience in use of the metrics suggests the typical relationships between values of the metrics and requirements quality shown in Table 2.

9. Requirements Process Metrics

Table 1 indicated the application of requirements *quality* metrics. We have also found it beneficial to use, for engineering management purposes, requirements *process* metrics, derived for requirements analysis tasks such as system requirements analysis, software requirements analysis for CSCIs and hardware requirements analysis for HWCIs.

Useful metrics include:

RSTA *Percent Started.*

This metric indicates the percentage of source requirements currently under development, the "work in progress".

RTBD *Percent "To be Determined".*

This metric indicates the percentage of requirements containing TBDs, i.e., requirements for which the resolution of incompleteness is beyond the resources of the analyst and which have been referred to other individuals, organizations or phases for resolution of missing information.

RCOM *Percent Completed.*

This metric indicates the analyst's view of that analysis of the source requirement has been completed.

RAPP *Percent Approved.*

This metric indicates the percent of source requirements for which the results of analysis (child requirements) have been approved for incorporation in the destination document.

In addition, the need to control the process of formally decomposing and allocating requirements of an element in the system hierarchy to its subordinate elements has led to an additional metric:

RALL *Percent Allocated.*

This metric indicates the percent of parent requirements of an element at one level of the WBS for which the corresponding child requirements have been allocated to the applicable lower level elements.

All of the above process metrics provide data for earned value measurement within project cost/schedule control systems. In addition, RTBD has proved to be a useful parameter for incorporation into a Technical Performance Measurement (TPM) program [2].

Metric	Very poor set of requirements, requiring substantial development	Fair set of requirements, may just be suitable for purposes of solicitation, depending on the SOW and type of contract envisaged	Requirements at SRR suitable for carrying forward into development	Requirements suitable for Critical Development
RQ-	0.01-0.3	0.3-0.7	0.85-0.99	0.99+
QF1-Correctness	0.9	0.98	0.99	0.99+
QF2-Completeness	-5	0	0.95	0.99+
QF3-Consistency	0.9	0.97	0.99	0.99+
QF4-Clarity	0.9	0.97	0.99	0.99+
QF5-Non-Ambiguity	0.3	0.7	0.9	0.98+
QF6-Connectivity	0.3	0.9	0.99	0.99+
QF7-Singularity	0.1	0.3	0.99+	1
QF8-Testablity	0.1	0.7	0.99	0.99+
QF9-Modifiability	0.1	0.5	0.99	0.99+
QF10-Feasibility	0.95	0.99	0.99+	0.99+

Table 2 - Typical Values of Requirements Quality Metrics

10. Computer Support to Metrics Generation

Requirements management benefits substantially from the use of computer based tools which facilitate, in particular, efficient text handling, rigorous requirements allocation and the creation and maintenance of peer and parent-child relationships for requirements traceability purposes. Metrics prove to be most easily calculated where a CASE environment is in use for those other aspects of requirements management.

One CASE tool for requirements management with which the author has experience is Document Director ReqMgr, produced by Bruce G. Jackson & Associates, Inc. A prototype software package which automates storage of metrics-related requirements quality and process data and which progressively builds up requirements quality and process metrics has been built for use with Document Director ReqMgr.

Proprietary tools known to the author are also being utilized in a similar way by other organizations.

11. Conclusions

Numerous best practice standards (ISO 9001, Software Engineering Institute criteria, MIL-STD-499B) emphasize a closed loop process as a key to effective technical management. The metrics described in this paper are a means of implementing closed loop control over the requirements engineering process.

The cost of implementing these metrics within a suitable, existing CASE environment appears to be around two percent of the cost of the total requirements engineering effort. The engineering manager must decide whether the resulting payoff will exceed this cost. Sufficient data to conclusively answer this question has not yet been developed by the author, nor has it been identified from other sources.

Assessment of the cost-effectiveness of the use of requirements metrics must therefore, for the present, be subjective. It is the author's assessment that requirements metrics, developed on a sampling basis, used within a suitable CASE environment, provide considerable leverage in satisfying the goals of complex systems development.

Greatest leverage is obtained where sampling techniques are used in metric development. Such sampling may focus on, say, every nth requirement, or on areas of perceived risk.

References

[1] B.W. Boehm, *"Software Engineering Economics"*, Prentice-Hall, Englewood Cliffs, N.J., 1981

[2] MIL-STD-499B Draft, *"Systems Engineering"*, 6 May 1992

[3] M. Alford, *"Strengthening the Systems/Software Engineering Interface for Real-Time Systems"*, Proceedings of the Second Annual International Symposium of the National Council on Systems Engineering, Seattle, 1992

[4] M.B. Pinkerton and F.R. Fogle, *"Requirements Management/Traceability: A Case Study - NASA's National Launch System"*, ibid

[5] R. Fuji course notes on *Independent Verification and Validation*, 1989

ADDENDUM A

Addendum to the above paper: Estimating the Percentage of Completeness of Requirements

The Omission Ratio provides one of three ways of estimating the difference between the number of requirements present in a requirements specification (or database) versus the number needed to adequately specify the requirements in the circumstances, i.e. in practice the "% completeness" of requirements. I place "% completeness" of requirements in parentheses because a set of requirements can never be literally complete. For example, if we want a car, there are requirements of which we are conscious. We also have a requirement that when speed of the car exceeds 30mph, the car not spray the driver with an oily yellow dye. If it did, we would regard that as unacceptable. There is an endless number of requirements on the car that exist that we could spend the rest of our lives assembling.

So when we talk about "the completeness of requirements" we mean the number of valid requirements that we already *have* compared with the number of valid requirements that there *needs to be* there in a given set of circumstances. Those circumstances may be acquisition circumstances or development circumstances. The criterion for adequacy of a set of requirements is that "if this set of requirements is satisfied, the risk arising from residual defects in the requirements (omission and other defects) will be low to very low". Low level of risk is usually taken as expected loss (loss x probability of that loss occurring) below a couple of percent. Very low risk is expected loss well below 1% - it I had to put a number to it, I would say below 0.3%.

The Omission Ratio relies on the information content in requirements that are there providing clues as to requirements that need to be there, but are not there. A simple example would be a requirement on a system to consume no more than 500W of mains power, but no requirement to have a Mains Power Interface, and no requirements on such an interface. In using the Omission Ratio approach, one would estimate the number of related but missing requirements as (most likely) 4. By contrast, for a mass limit requirement on a cellphone, I would (most likely) estimate the Omission Ratio as 0. The end result is, say for a sample size of 30 requirements, and a sum of the omission ratios also of 30, a Completeness estimate of 50%. This method is quite good for fairly high initial completeness, say 50% or more.

These examples also illustrate that the accuracy of estimation of Omission Ratio increases with the degree of application domain knowledge of the analyst (as does the estimate of the quality of what is already there).

The second method of estimating Completeness relies on norms of relationships between types of requirements (Primary Requirement Type, Primary Actor). The most sensitive relationship is the relationship between the percentage of Functional versus External Interface. You expect these percentages to be similar within about 10%, with Functional on the higher side. The overall percentage of Other Qualities is also a useful indicator. You expect to see figures in the range 5-10% for very small systems (say up to 100 requirements), falling to about 1% for large systems (thousands of requirements).

The third method is to use "should be" numbers based on history. There is a history of how many requirements it takes to adequately specify for development various things, for example:

Business Jets	2500-4500
Emergency Communication Buoys	100-200 depending on sophistication
Heart Pacemakers	800
High Capacity Passenger Aircraft	8000-10000
High Speed Trains	4500.

These methods may be used in combination to further improve the accuracy of estimating. None of the methods is very accurate in absolute terms, but I have found all to be sufficiently accurate to support sound decision making. This, after all, is their purpose.

A second Requirement Quality Metric is developed, with the estimated percentage of missing requirements factored in at zero. This second metric is used as the overall indicator of requirements quality for most applications of the metric.