

Quality Management in Astronomical Software and Data Systems

Nicole M. Radziwill

*National Radio Astronomy Observatory, 520 Edgemont Rd.,
Charlottesville VA 22903*

Abstract. As the demand for more sophisticated facilities increases, the complexity of the technical and organizational challenges faced by operational space and ground-based telescopes also increases. In many organizations, funding tends not to be proportional to this trend, and steps must be taken to cultivate a lean environment both in development and operations to consistently do more with less. To facilitate this transition, an organization must be aware of how it can meet quality-related goals, such as reducing variation, improving productivity of people and systems, streamlining processes, ensuring compliance with requirements (scientific, organizational, project or regulatory), and increasing user satisfaction. Several organizations are already on this path. Quality-based techniques for the efficient, effective development of new telescope facilities and maintenance of existing facilities are described.

1. Introduction

Quality is an indicator of high performance, whether that performance is measured in terms of individuals, teams, products, or the entire organization. The ISO 8402 definition of quality is the "totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs." This definition is substantiated by the official glossary of the American Society for Quality (ASQ), which simultaneously notes that quality is the state of a product or service being free from deficiencies, which can be expressed as nonfunctional requirements or also implied. In the development of software and data management systems for astronomy, we must contend with both stated and implied needs. The stated need is typically to satisfy the needs of the current observer or PI on a project. The primary implied need is to satisfy the needs of the future researcher, who has retrieved his or her data or science data products from an archive and wishes to achieve scientific objectives which are distinct from the original observer's intent.

There are many aspects of the software engineering process that aim to build quality into software systems, or test failure modes out, prior to deployment. The requirements process, which includes identifying the customers, documenting their needs, developing product features that suit their needs, and then establishing processes to build or generate those products is one such example. At the same time, software engineering practices often fail to help us operate in the most efficient environment.

By shifting our focus from software engineering practices to *quality management that involves software development*, the next wave of process innovations for optimizing the scientific productivity of our telescopes could be realized. The remainder of this chapter covers some fundamental concepts in the discipline of quality management and how it can be applied to software development and data management. The information is intended to be *descriptive*, not *prescriptive*, but will provide a launching point to learn more about quality techniques and how they may be applied.

2. Important Terms and Concepts in Quality Management

Quality is maximizing customer satisfaction at the lowest cost. But quality can be accurately defined in many ways, including fitness for use, zero defects, and conformance to requirements. Despite the range of definitions, the goals underlying the pursuit of quality are the same: achieving conformity, reducing variation, eliminating waste and rework, eliminating non-value-adding activity, preventing human error, preventing defects, improving productivity, and increasing efficiency and effectiveness (Okes & Westcott 2003). According to the official glossary of the American Society for Quality (ASQ), quality management is defined as the "application of a quality management system to achieve maximum customer satisfaction at the lowest overall cost to the organization." This must, of course, be done in a way that aligns well with the goals of the organization - even if it's done for free, satisfying the wrong group of customers will not add value.

Quality management, as defined above, involves building and applying a quality management system. A quality management system is a *customized application of guiding principles into a collection of standards, policies, methodologies and tools enacted to satisfy quality goals*.

Several systems of guiding principles can be drawn from. Total Quality Management (TQM), for example, is based on the principle that everyone in the organization should be involved in continuous improvement. The criteria for the Malcolm Baldrige National Quality Award (MBNQA) program are often used to guide the development of a quality system. These include leadership, strategic planning, customer focus, measurement/analysis and knowledge management, human resource focus, and results.

Standards are rules, requirements or frameworks. They can be adhered to by individuals, groups, companies, industries, or governments, and are directed towards achieving specific goals. These goals may include: establishing a basis for communication, establishing a basis for shared meaning, ensuring conformity, establishing social order, ensuring personal and corporate responsibility, achieving consistency and order, and prescribing behavior. Standards are often put in place to help achieve quality goals, thus blurring the distinction between standards in general and standards specifically instituted to meet quality goals (such as the ISO 9000 series and QS 9000 for the automotive industry). Standards are typically characterized by slow cycles of variation, and are often changed or augmented only through formal processes. Policies condone or enforce standards for behavior, for conducting business, or for making decisions.

A methodology is a system of principles, rules and approaches to solving problems. Mature sets of guiding principles can be used as methodologies (in fact, XP or extreme programming falls into this category). Methodologies may promote the use of certain tools, classes of tools, or techniques, or may even encapsulate other methodologies. Six Sigma, Lean Manufacturing, PDCA (Plan-Do-Check-Act), and TQM are examples of methodologies.

A tool is an implement for performing or facilitating the accomplishment of a specific task or purpose, and can be associated with or used as part of one or more methodologies. Tague (2005) distinguishes tools from standards, methodologies and systems in that tools generate directed action. Concepts and standards generate no action in and of themselves; methodologies generate much action (that is, action that can be directed towards accomplishing many objectives.) Pareto diagrams, checksheets, control charts, failure mode effects analysis (FMEA) and value stream maps are examples of tools.

3. Quality Goals

There are a proliferation of tools that can be applied to increase quality levels in processes, products, and data, so it is important to consider goals when evaluating methodologies and tools. Different departments (e.g. software, electronics) and operational processes may have distinct quality goals, so it is important to resist establishing one or two quality goals - or all quality goals - unilaterally.

1. Achieving Conformity

According to the Random House Unabridged Dictionary, *conformity* is action taken in accord with prevailing standards, attitudes and practices. Achieving conformity is desirable when we are required to (e.g. if there are regulatory or safety requirements that we are obligated to meet) or if we want to meet strategic goals (e.g. adopting technical standards to achieve interoperability).

2. Reducing Variation

Variation exists in all processes. Stable processes are subject to random variation, unstable processes are influenced by variation that can be traced to an assignable root cause, and structural variation emerges when there are changes in patterns of usage. For example, random variation (noise) in telescope raw data is caused by weather effects, and assignable variation can be caused by different operators commanding the telescope. An example of structural variation would be seeing a sudden influx of bug reports associated with the software running a particular backend. If the backend has not been used in a while, these errors are due to structural variation. Quality objectives associated with the goal of reducing variation include minimizing the effects of noise, identifying root causes more accurately and/or more quickly, and appropriately factoring in or out the effects of patterns of usage.

3. Eliminating Non-Value-Adding (NVA) Activity

In any process, steps may be performed habitually. For example, in one software development project I was involved with, a coding standard had emerged over time that did not seem to fit the current needs of the project. After some prodding, I found that the standard was in place to ensure that

users would have easy access to the tape drives that stored their data. Since we had not used tape drives for a few years, this assumption was invalid and the step in the process of ensuring that the code complied to the standard could be removed. The point here is not to seek out ways in which people are wasting their time, but to question the assumptions underlying each step of a process to make sure it is part of that process for the right reasons, and that those assumptions have not become invalid.

4. **Reducing Waste and Rework**

According to Imai (1997) and English (2005), there are nine sources of waste (which often lead to necessary rework). These are: overproduction, holding too much inventory, handling repairs and rejects (bugs), unnecessary motion, too much or unnecessary processing, excessive transport, waiting, process failure caused by defective information, and wrong or sub-optimized decisions caused by defective information. In software and data management, we might produce and store far more monitor data than is ever used, transport our data using a 20-step process when a 5-step process would suffice, or make architecture and design decisions based on faulty or inaccurate assumptions. All of these things that be worked on to improve quality.

5. **Preventing Human Error**

Substantial expertise is required to run a telescope control system, observing software, or data management system, making human error at any stage of operations a possibility. There are many actions that can reduce the likelihood of introducing human error into the system, which can lead to unproductive and lengthy troubleshooting sessions. Documenting commonly used procedures ("Standard Operating Procedures" or SOPs), for example, can help to reduce or prevent human error in many functional areas. These include: conducting regression tests, managing the production control process (where code that is ready to be deployed is promoted to the operations environment), and establishing and auditing procedures for operators and scientific support. Unit tests and test-first development are among the best means to preventing human error in code, because the developer is required to think about the objectives of each module as it is constructed.

6. **Preventing Defects, Improving Accuracy, and Increasing Integrity**

Software development methodologies and the discipline of software reliability are strongly based upon achieving these three goals. Requirements and design processes are intended to prevent defects before they are introduced into the system. The Rayleigh Model was developed to track the process of defect removal as the development process is underway, to serve as the basis for continuous improvement in defect removal processes. For an excellent survey of software reliability, which covers all of these examples and many more, see Kan (2003).

7. **Improving Productivity**

Productivity is all about volume, and being able to produce more designs, more code, or more data. Both people and systems can be made more productive. Dynamic scheduling automation is one approach to improving the productivity of a telescope. Directly improving software performance by leveraging computing hardware or platforms more effectively, parallelizing

code, and utilizing modern computing resources such as the Grid are all means of improving productivity. Productivity is typically measured in terms of speed or throughput enhancements.

8. Increasing Efficiency and Effectiveness

Efficiency is being able to produce more (or more quickly) with fewer inputs to the production process, in essence, productivity that considers inputs in addition to outputs. Effectiveness is being able to produce the results that are needed. A combination of efficiency and effectiveness is the goal, because rapid results are useless unless they are on target. We gain efficiency by building our skills, becoming more aware of developments outside our workgroups, and enacting training programs, documentation or mentorship to help new staff members become productive more quickly.

The above list of quality goals is comprehensive but not exhaustive. For example, using quality principles to drive innovation is currently an area of active research.

4. Quality Systems

A quality system can be implemented to help achieve one or more of the quality goals previously described. Criteria for the Malcolm Baldrige National Quality Award (MBNQA) program administered by NIST, and ISO 9000, are two examples of guidance for quality systems implementation. The first example provides conceptual assistance only, while the second is an international standard for what elements (e.g. a management policy) must be included in a quality system for certification. Certification is often desired when one organization seeks to ensure that a supplier has met minimum standards for achieving quality in the production of a component product or part.

All approaches to developing a quality system share common factors: they are driven by operational processes, they include measurement (specifically metrics to evaluate products and processes), and they provide a clear mechanism for benchmarking to best practices, auditing, and continuous improvement. In short, by committing to a quality management system, an organization also commits to *not reinventing the wheel*.

5. The Process-Oriented Approach

The approach to defining requirements and specifications that technical professionals in astronomy typically follow is functionally driven, beginning with a set of user requirements or use cases. As defined by Jacobson et al. (1999), "use cases are the functions a system provides to add value to its users". This approach puts the software system at the center of the analysis activity, instead of the outcome that the organization is trying to achieve utilizing the software.

The solution to this gap is to shift the focus from the software systems to the operational processes that deliver value to the user, and embark on process-oriented development. The use case approach, in the absence of process-oriented development, will artificially limit continuous improvement efforts by masking the root causes of inefficient processes. This is because use cases focus on the

interactions that actors (either people or systems) have with software tools, not the complete flow of the process itself, along with the value that process is intended to deliver.

Members of the data management community in astronomy, with representation from both space-based and ground-based disciplines, have already started to recognize the need for a process-driven approach. In a recent publication from the IAU meeting in Prague, it was noted that the most critical challenges in data management such as long-term curation, version control and intellectual property standards "require a shift in focus away from technological tools towards a suite of data management processes." (Norris et al. 2006) This shift requires a complete reorientation of our approach to design and development.

6. Integrating Quality Management into Astronomical Software Operations

The essence of integrating quality management into software operations and data management is to combine the best aspects of *agility and accountability*. Agility means we can effectively respond to new information and insights about our problem. Accountability means we can provide an audit trail for decisions that have been made, which helps us to understand why some solutions are chosen while others are discarded in favor of new ideas.

1. Identify and Articulate Operational Processes

There are three classes of processes common to telescope operations: manufacturing raw data, producing science data products, and distributing those data products to data curation facilities and the scientific community. Sometimes, distribution to researchers is accomplished by way of the data archives. Processes are often described by flowcharts, which graphically illustrate the options available and decisions that are made as we progress from one step to another. It is important to step outside the lens of software development when describing a process. In many cases, the process can be executed without any software at all, and automation is only a means of making the process more efficient or effective.

2. Map Processes to Quality Goals

The reason it is first important to understand the operational processes that constitute an end to end system is that different processes may be associated with different quality goals. For example, it may be appropriate to achieve conformity in technical standards but not with the customized hands-on help that is often provided to visiting observers. It may be very useful to eliminate waste and rework in data flow processes, but be critical to generating new ideas to allow dead-end prototyping to occur in some cases. After processes are identified and documented, determine which quality goals are significant for each.

3. Map Quality Goals to Methodologies and Tools

As described above, methodologies and tools must be selected to appropriately support quality goals for different processes. Applying the wrong approach can lead to waste, rework, and the insertion of non-value-adding activities to a process, so this step is extremely important. A summary of standards and methodologies to investigate for various quality goals is

Quality Goals	Standards and Methodologies to Consider
Achieve Conformity	ISO 9000
Reduce Variation	Six Sigma for Software Lean Manufacturing Statistical Process Control (SPC) Simulation
Eliminate NVA Activity	Auditing Balanced Scorecard Benchmarking Kaizen
Reduce Waste & Rework	Financial Analysis/Cost of Quality (CoQ) Kaizen Lean Manufacturing
Prevent Human Error	Design for Manufacturability/Maintenance (DFM) ISO 9000 Quality Function Deployment (QFD) Total Quality Management (TQM)
Prevent Defects, Improve Accuracy, Increasing Integrity	Design for Six Sigma (DFSS) PDCA(Plan-Do-Check-Act)/Deming Cycle Six Sigma for Software
Improve Productivity	Baldrige Criteria Benchmarking Business Process Reengineering (BPR) Kaizen Quality Function Deployment (QFD) Total Productive Maintenance (TPM) Total Quality Management (TQM)
Increase Efficiency & Effectiveness	Quality Function Deployment (QFD) DMAIC (for existing products) DMADV (for new product development) Kaizen Taguchi Methods/Robust Design Total Quality Management (TQM)
Drive Innovation	TRIZ (Theory of Inventive Problem Solving)

Figure 1. A non-exhaustive list of standards and methodologies that can be employed to satisfy quality goals.

presented in Figure 1. The tool selection process can be more extensive, but detailed references are available to aid in selection (e.g. Tague 2005).

4. Define Data Quality Objectives

Operational processes, particularly those in software-intensive environments, often produce raw data and data products derived from raw data. Not all data products are equal, however. Figure 2 shows a breakdown of the different data types, which we have found to be relatively consistent between organizations. In much the same way that different processes can have different quality goals, unique data quality attributes will often be critical for the various data types.

An outline of data quality attributes is presented in Figure 3. Check to ensure that the methodologies and tools selected will also support meeting the data quality attributes for the appropriate data type.

5. Enact a Continuous Improvement Model for Software Development and Data Management

Our understanding of problems and solutions is emergent; as we learn more about the problem domain and the environment of the problem, our

Domain	Data Type	NASA/EOS Terminology	Description
Instrument	Device monitoring data	N/A (Level-1)	Produced by instrumentation, typically not preprocessed, and typically not stored as part of raw data products. Useful for preparing trending data to detect emerging instrument failures, and providing operational responses to other failures to dynamically improve the potential for producing quality data products.
Instrument	Raw data	Level 0	Produced by instrumentation. May be subject to limited preprocessing in firmware (for example, autocorrelation spectrometers).
Instrument	Calibrated data	Level 1	Produced by removing instrumental and environmental effects. EOS breaks this down into Levels 1A (raw data appended with annotations and calibration information) and Level 1B (raw data processed to calibrated data).
Science	Derived data	Level 2	Produced by combining calibrated data with other calibrated data, or with other derived data, according to processes, techniques, or algorithms. Scientific analysis can take place at this level or any higher level.
Science	Assimilated data	Level 3	Produced by gridding, resampling, and/or changing the frame of reference for derived data.
Science	Model data	Level 4	Produced by applying one or more mathematical, physical, or stochastic models to collections of assimilated and derived data products.

©2006 ASQ

Figure 2. Summary and description of scientific data types, from Radziwill (2006).

perspective on the solution space will evolve as well. Many of us have recognized the shortcomings of the traditional requirements-test-design-deploy waterfall model, and have attempted to integrate prototyping techniques or agile methods (such as pair programming) into our work. (Note, however that s strongly prescriptive requirements and specification process may still be necessary to ensure that contractual or regulatory obligations are satisfied.) The major implication of adopting a continuous improvement model is that an organization will become more cognizant - and tolerant - of the natural process of requirements evolution and solution discovery. The Capability Maturity Model for Integration (CMMI) is the most recognized continuous improvement model being applied today, but much like ISO 9000 quality systems, the approach falls short when the role of continuous improvement is considered. Defining and adhering to processes alone is the order of business until Level 5, the pinnacle of maturity, is reached. Alistair Cockburn, one of the creators of the Agile movement, recognized that traditional approaches to software engineering were not able to explain why some projects succeed and others fail, or what factors are critical to project success, and they were also unable to provide prescriptive guidance and advice. Through his explorations, he has come to view software development as a "series of resource-limited, goal-directed cooperative games of invention and communication." A game is defined as an activity that consists of moves towards or away from a target, can be finite or infinite,

<p>Data Models</p> <ul style="list-style-type: none"> • Clarity of definition—unambiguous labels, distinctive and representative naming of objects, attributes, and relations. • Comprehensiveness—suitability of model for use by current applications, and adaptability of model to future uses. • Flexibility—ability of data model to be adapted to future usage scenarios. • Robustness—ease of adaptation of the model to future usage scenarios. • Essentialness—not including extraneous information; including fundamental information from which other quantities can be easily derived. • Attribute granularity—selecting the appropriate number of objects to represent a concept. • Precision of Domains—allocating appropriate precision (ex. data types, sizes) to a data attribute. • Homogeneity—not hiding business rules or the simplifying assumptions used by applications in the data. • Naturalness—not “overloading” data values with metadata or supplemental descriptive characteristics. • Identifiability—ensuring that there is a way to distinguish between similar entities (ex. using primary keys). • Obtainability—is it legal, feasible, and/or appropriate to collect, persist, and use a particular data value? • Relevance—are all the stored attributes useful by applications, or are there plans to use them? • Simplicity—the model does not encourage complications in the applications that will use the data. • Semantic consistency—ensure that the meanings and names of objects within a dataset are consistent, that one term, not several, are used to refer to the same physical concept. • Structural consistency—ensure that ideas are uniformly referenced, including use of standard units. <p>Data Values</p> <ul style="list-style-type: none"> • Accuracy—the level to which stored data agree with accepted sources of “correct” information. • Null values—the absence of information can provide valuable insight into obtainability/relevance problems. • Completeness—how well the expectation that certain fields will be appropriately populated is met. • Consistency—data values in one set being logically consistent with those in another related set. • Currency/Timeliness—degree to which information represents the up-to-date state of what is being modeled. <p>Information Domains</p> <ul style="list-style-type: none"> • Enterprise agreement of usage—communicating in terms of nomenclature to which all have conformed. • Stewardship—ensuring that responsibility for maintaining integrity and currency of attributes is assigned. • Ubiquity—encouraging sharing of data resources and standardization of data use across applications. <p>Data Presentation</p> <ul style="list-style-type: none"> • Appropriateness—how well the format and content of data satisfies the users’ needs. • Correct interpretation—how comprehensive the provided information is, so that users can make accurate inferences. • Flexibility—adaptability of system to changes in represented information or requirements for the use of that information. • Format precision—ensuring that the stored, displayed, and leveraged data instances contain the required granularity of information for the application for which they are used. • Portability—how well the capability to move applications from one platform to another has been preserved. • Representation consistency—whether instances of data effectively implement the consistency demanded by the model. • Representation of null values—representing all null or missing values equivalently. • Use of storage—ensuring that the storage of the data effectively uses the media upon which it resides. <p>Information Policy</p> <ul style="list-style-type: none"> • Accessibility—degree of ease of access, and breadth of access, to information. • Metadata—ensuring that metadata are not only defined but also meet required dimensions of data quality. • Privacy and security—ensuring that an approach is in place to display information selectively, and also to protect the integrity of the data and metadata themselves. • Redundancy—being cognizant of where and why repetitive data are useful, and managing the inflow of data accordingly. • Unit cost—understanding the costs associated with persisting and maintaining information, and being parsimonious where possible, so that the total cost of ownership of data is reduced or minimized.
--

Figure 3. Summary and description of scientific data types, adapted from Loshin (2001).

competitive or cooperative, and can be terminated either arbitrarily or by achieving a goal, such as completing a mission.

Refactoring (the process of continually refining code) is also an important tool for continuous improvement, but should be used in the context of a continuous improvement model so that the act of refactoring is tied to higher level goals.

7. Conclusions

Using quality management techniques as the basis for software and data management in astronomy is a promising avenue for improvements in a resource-constrained environment where challenges are ever increasing. A process for integrating quality management into operations is to identify and articulate operational processes, map those processes to one or more quality goals, define data quality objectives for the information products produced by each process, ensure that the data quality objectives align with the quality goals of the process, and select methodologies and tools for each process that fit the goals. The addition of a continuous improvement approach to iteratively revisit the operational processes, which can be applied symmetrically to new development and operations, provides the feedback to make the model for a quality management system complete.

References

- Cockburn, A. C. 2004, The End of Software Engineering and the Start of Economic-Cooperative Gaming. *ComSIS*, 1(1), pp. 1-32.
- Jacobson, I. Booch, G. & Rumbaugh, J. 1999, *The Unified Software Development Process*. Addison-Wesley: Reading, MA.
- Kan, S. 2003, *Metrics and Models in Software Quality Engineering*, 2nd Ed. Addison-Wesley: Reading, MA.
- Loshin, D. 2001, *Enterprise Knowledge Management: The Data Quality Approach*. Morgan Kaufmann: San Diego.
- Norris, R., Andernach, H., Eichhorn, G., Genova, F., Griffin, E., Hanisch, R., Kembhavi, A., Kennicutt, R. & Richards, A. 2006, *Astronomical Data Management*. Highlights of Astronomy, Vol. 14, XXXVIth IAU General Assembly, Prague, August 2006.
- Okes, D. & Westcott, R. T. 2001, *The Certified Quality Manager Handbook*, 2nd Ed. ASQ Press: Milwaukee, WI.
- Radziwill, N. M. 2006, Foundations for Quality Management of Scientific Data Products. *Quality Management Journal*, 13(2), pp. 7-21.
- Sullivan, K. (Ed.) 2003, *Preliminary Report: NSF Workshop on the Science of Design*.
- Tague, N. 2005, *Quality Toolbox*. ASQ Press: Milwaukee, WI.