25th ANNUAL NATIONAL CONFERENCE ON MANAGING ENVIRONMENTAL QUALITY SYSTEMS

APRIL 24-27, 2006

Marriott Renaissance, Austin, Texas

Technical Papers

Best Practices in Information Quality Improvement

- L.Petterson, ORD's Scientific Office of the Future - 8:30 AM
- R.Wright, Application of Software QA Concepts and Procedures to Research Involving Software Development - 9:00 AM
- C. Mallory, The Emergent Data Steward - 9:30 AM

TECHNICAL SESSION:
Best Practices in Information Quality Improvement

ORD's Scientific Office of the Future

Lynne Petterson, U.S. EPA

EPA research is becoming increasingly collaborative with multi-discipline research teams located in geographically distributed sites. Scientific requirements necessitate the creation and implementation of "collaboratories" or virtual science laboratory/centers without walls, in which Agency researchers and their partners can conduct their science without regard to geographical location. As a first step in this vision, ORD is creating the Scientific Office of the Future (SoF). The SoF consists of dual processor scientific workstations with 64-bit architectures (with 32-bit backward compatibility) and multiple monitors. SoF workstations allow the Agency to take advantage of cost effective, high-end workstations to balance its scientific computing load across its administrative desktop systems, SoF workstations, and high performance computing (HPC) platforms.

In 2004, ORD tested the new architectures with applications in the following areas: Geospatial Analysis/Remote Sensing; Computational Methods including Computational Toxicology, Computational Chemistry, and 'Nomics; Modeling/Simulation, and; Statistics. Preliminary testing created excitment among researchers evaluating the SoF workstations. For example, a computational chemist found he could run 95% of his research on the SoF workstation, replacing 3 Silicon Graphics workstations and 20% of his allocation at EPA's National Environmental Sciences Computing Center (NESCC).

In 2005, ORD researchers and IT staff collaborated on identifying a single, standard configuration for the SoF workstation that could meet most scientific computing needs in ORD. The group also created an in-depth analysis demonstrating that it was more cost efficient to purchase the systems rather than to lease them.

With Management Council approval, 67 SoF workstations were acquired in November 2005 for an SoF Pilot that begins in March 2006. The purpose of the SoF Pilot is to determine whether recipients experience a reduction in processing time for their current scientific applications/models, increase the size and complexity of their scientific applications/models, and run jobs that were not possible on their current PCs/workstations. The SoF Pilot will also assess the ability of SoF researchers to reduce their reliance on the central HPC platform and save HPC cycles for large production jobs by moving appropriate, smaller jobs to SoF systems. A subset of the SoF workstations will be involved in a proof-of-concept that demonstrates cycle sharing and remote access to unused cycles.

A final goal of the SoF Pilot is to identify and track the costs needed to support the SoF workstations. This information is needed for senior management to decide on next steps.

Application of Software Quality Assurance Concepts and Procedures to Environmental Research Involving Software Development

Robert S. Wright, Mail Code E343-03
U.S. Environmental Protection Agency
National Risk Management Research Laboratory
Air Pollution Prevention and Control Division
Research Triangle Park, NC 27711

As EPA's environmental research expands into new areas that involve the development of software, quality assurance concepts and procedures that were originally developed for environmental data collection may not be appropriate. Fortunately, software quality assurance is a well-developed technical field in software engineering and its concepts and principles can be applied to software that is developed for environmental research. There are significant parallels between the two types of quality assurance and it should not be difficult to incorporate software quality assurance concepts and procedures into the EPA Quality System. This paper compares these two types of quality assurance and highlights their similarities and differences. Even readers who are not familiar with software quality assurance can use the concepts and procedures described in this paper to improve the quality of software developed for environmental research.

INTRODUCTION

EPA Order 5360.1 A2 establishes policy and program requirements for the mandatory Agency-wide quality system. Its scope includes the collection, evaluation, and use of environmental data as well as the design, construction, and operation of environmental technology. In the past, EPA's quality assurance activities have focused largely on environmental data collection. Increasingly, the uses of environmental data in databases and other information systems have become an area of concern. Quality assurance principles and procedures are needed for these information systems. The concepts and procedures that have been developed for environmental data collection do not apply well to information systems. Software quality assurance has developed in parallel with EPA's quality system and can be applied in those instances in which environmental research includes the development of software. This paper describes software quality assurance concepts and procedures that can be useful in those instances.

SYSTEMATIC PLANNING

The development of software begins with the systems analysis and design process, which is analogous to the systematic project planning process (see Table 1). This process is a direct application of the Shewhart cycle (plan-do-check-act). In broad terms, problems and solutions are identified, goals are established, software quality criteria (metrics) are set to gauge performance, software development is implemented, and progress is assessed during development and at its completion. The software is documented during the analysis phase in software requirements, which describe the purpose and desired functions of the software, and in functional specifications, which are a formal description of the software and which is the blueprint for developing the software. Based on these documents, the design phase establishes the proposed structure of the software, which is documented in a high-level, architectural design for the overall system structure and in a detailed design that includes the design of specific program details. During this phase, testing procedures are developed to determine if quality metrics are being attained. During the development phase, testing and corrective action occur as needed. Additionally, the software is assessed in informal and formal reviews. A software quality plan is prepared to document how quality assurance activities support the development and to answer questions such as: what are the quality metrics?; what testing and assessment will be done?; and how are uncovered problems corrected? The plan may contain the following sections: purpose; reference documents; management; documentation; standards, practices, conventions, and metrics; assessments; software configuration management; problem reporting and corrective action; tools, techniques, and methodologies; code control; media control; supplier control; records collection, maintenance, and retention; and testing methods.

THE GRADED APPROACH

Some software quality assurance procedures may not be appropriate for a specific software development project. As is the case for environmental data collection projects, a graded approach can be used to apply an appropriate level of software quality assurance for a project. Argonne National Laboratory's Decision and Information Sciences Division (ANL DISD) has three quality levels for its software development projects (see Table 2). The quality assurance procedures that must be followed at each level are dependent on factors such as criticality, external impact, development effort, security impact, and cost of failure.

QUALITY METRICS

Quality metrics, such as reliability, usability, maintainability, and adaptability, are more appropriate for software development projects than are the data quality indicators applicable to environmental measurements, such as precision, bias, and representativeness (see Table 3). Quality metrics can be divided into (1) process metrics which are used to improve the software development and maintenance process, (2) product metrics which describe the characteristics of the software itself, and (3) project metrics which describe the project and its execution (Ginac, 1998). There should be a correlation between the product metrics and the software requirements.

SOFTWARE TESTING

Software testing procedures fill the same role in software development projects as quality control checks do in environmental data collection projects. Various manual and automated techniques are available to test software inputs and outputs (black box testing) or to test the internal structure of software (white box testing) at various stages of its development. One model of software testing is a "V" in which software requirements, functional specifications, architectural design, and detailed design move down the left side while unit testing, integration testing, system testing, and acceptance testing move up the right side. The expected output or result of each stage of the testing is defined in the documentation that is opposite it on the "V." Testing should not be done by the individual who prepared the software and, where feasible or necessary, an organization should not test its own software (Myers et al., 2004). Independent verification and validation of software is a quite formal testing process that reserved for important, large, and complex projects, such as in the aerospace industry, for which it is critical that the software perform successfully. It is performed by an organization not involved in developing the software. The purpose of independent verification and validation is to ensure that the software design, implementation, and documentation meet requirements. Verification addresses "Am I building the product right?" and validation addresses "Am I building the right product?" The expected benefits are increased objectivity, earlier detection of errors, reduced effort and costs of removing detected errors, enhanced operational correctness, and a more consistent testing. It establishes traceability between the software and the requirements.

SOFTWARE DOCUMENTATION

The documentation that is needed for a software development project should be defined in the design phase. It may be embedded in the software itself or it may be in hard-copy or on-line documents. The documentation requirements for ANL DISD projects include a software quality assurance plan, a configuration management plan, a test plan, and test documentation/results (see Table 4), which parallel EPA's documentation requirements. In addition to development-related documents, other documents that may be written for the software include code documentation, user documentation, and guides for installation, operation, and maintenance.

SOFTWARE ASSESSMENT

The assessment function can be performed by peer reviews that are performed at various stages of the development process and with various degrees of formality as follows (Wiegers, 2002):

- Inspection is the most systematic and rigorous of the assessments and it is the software industry's best practice. It has procedures that are similar to a technical systems audit. A group of people, including a moderator and a recorder, conduct an inspection to review a document, such as a functional specification or test plan. The goal is to find problems with the document, not to fix them. A formal report of the inspection will be prepared.
- Team review is planned and structured, but less formal and rigorous than inspections
- Walkthrough is an informal review in which the software author describes the software to a group of peers and solicits formal comments. The author takes the dominant role.
- Pair-programming is when two developers work on the same software simultaneously at the same workstation. The synergy of two focused minds creates superior software.

- Peer-deskcheck is a detailed self-review of software by the programmer to find errors.
- Passaround is when a programmer solicits informal comments from peers about software.
- Ad hoc review is a spur-of-the-moment review within the software team.

Table 1. Comparison of Systematic Planning Process and Software Analysis Process

Systematic Planning Process (after EPA, 2000)		Software Analysis Process (after Satzinger et al., 2004)
Step 1: identify project manager and staff		Step 1: research and understand problem
Step 2: identify project schedule, resources, milestones, and requirements		Step 2: verify that the benefits of solving the problem outweigh the costs of the solution
Step 3: describe the project goals and objectives		Step 3: define the requirements for solving the problem
Step 4: identify the types of data needed		Step 4: develop a set of possible solutions (alternatives)
Step 5: identify constraints to data collection		Step 5: decide which solution is best and make a recommendation
Step 6: determine the needed data quality		Step 6: define details of chosen solution
Step 7: describe how, when, and where the data will be obtained		Step 7: implement the solution
Step 8: specify QA and QC activities to assess the performance criteria		Step 8: monitor to make sure that you obtain the desired results
Step 9: describe methods for data analysis, evaluation, and assessment against the intended use of the data		

Table 2. Argonne National Laboratory Graded Approach to Software Quality Assurance

Quality level	Level C+ (low)	Level B (medium)	Level A (high)
Consequence of failure	Negligible	Moderate to severe	Unacceptable, major losses
External impact and visibility	Few external users, proof of concept	Limited distribution, prototype or beta	Wide distribution and visibility
Complexity and technical risk	Few modules, moderate complexity	Several modules and libraries	Many complex components
Development effort (person-years)	Less than 1	1 to 2.5	More than 2.5
Customization	Minimal	Moderate	Significant
Security impact and proprietary impact	None	Moderate	Significant
Cost of failure	Loss less than $100k	Loss $100k to $1m	Loss more than $1m

REFERENCES

Argonne National Laboratory, 2003. Appendix B to Software Quality Assurance (Argonne National Laboratory, Decision and Information Sciences Division)

R.H. Dunn, 1990. Software Quality Concepts and Plans (Prentice Hall)

R.H. Dunn and R.S. Ullman, 1994. TQM for Computer Software, Second Edition (McGraw-Hill)

G.M. Gelston et al., 1998. An Approach to Ensuring Quality in Environmental Software (Pacific Northwest National Laboratory Publication No. PNNL-11880)

T. Gilb and D. Graham, 1993. Software Inspection (Addison-Wesley)

F.P. Ginac, 1998. Customer Oriented Software Quality Assurance (Prentice Hall)

D. Ince, 1994. An Introduction to Software Quality Assurance and its Implementation (McGraw-Hill)

W.E. Lewis, 2000. Software Testing and Continuous Quality Improvement (Auerbach, 2000)

M. Luna, Systems Analysis and Design I (York University) (http://www.math.yorku.ca/Who/Guests/mluna/2010/ITEC-2010-2.html)

G.J. Myers et al., 2004. The Art of Software Testing, Second Edition (John Wiley & Sons)

J.W. Satzinger et al., 2004. Systems Analysis and Design in a Changing World, Third Edition (Course Technology)

U.S. Environmental Protection Agency, 2000. Guidance for the Data Quality Objectives Process- EPA QA/G-4 (EPA Publication No. EPA/600/R-96/055,)

U.S. Environmental Protection Agency, 2002. Guidance for Quality Assurance Project Plans- EPA QA/G-5 (EPA Publication No. EPA/240/R-02/009)

K.E. Wiegers, 2002. Peer Reviews in Software (Addison-Wesley)

The Emergent Data Steward

Cheri Mallory, Data Quality Consultant, Firstlogic, Inc.

Who in your agency knows enough about the data to support enterprise data management goals? In any data management effort, there must be recognition and acknowledgment of the data steward, the person who understands the complexities and abstractions of its data.

In this presentation, participants will hear how the data steward's responsibilities are moving from a single application to enterprise, collaborative change management and how agencies should leverage this growth going forward.

As agencies pursue new goals, such as master data management (MDM), data migrations and compliance issues, one thing becomes very clear. These are more than technology issues. These efforts will not succeed without an in-depth personal understanding of data.

From data management and data quality perspectives, agencies need to understand

- who the data steward is, both past and present,
- what are the boundaries (if any) of this newly evolving role, and
- how to leverage this growth going forward.

Attendees that join this session will learn the critical activities every federal agency and its supporting business partners must be doing now to improve their ability to manage enterprise data as valued assets.