SOFTWARE QUALITY ASSURANCE

OFFICE OF QUALITY ASSURANCE PROGRAMS (EH-31)

FEBRUARY 2005

This is the fourth in a series of newsletters designed to provide updates to the Quality Assurance community on Software Quality Assurance (SQA) activities to improve communications and support of field activities.

WHAT'S NEW?

Performance Degradation of the Advanced Mixed Waste Treatment Facility Fissile Tracking System Software

On September 1, 2004, a performance degradation was discovered in the Advanced Mixed Waste Treatment Facility Fissile Tracking System (FTS) software that might allow the transport of a fissile waste container, with potentially invalid assay results, past interlocks designed to prevent exceeding fissile mass control limits. The FTS software and interlocks are designated as a "safety significant system," in the facility Documented Safety Analysis. Failure of the interlocks to prevent the entry of a waste container with invalid assay results into a controlled area could potentially lead to inadvertently exceeding nuclear material safety limits.

Investigation revealed the vendor had not built the FTS software exactly as designed. Whereas the design would have required containers marked as a "failed assay" to maintain that status until administrative controls of expert assay reviews were completed, the software vendor built a loop into the software logic that could allow those results to be held in the database until two trailing calibration checks were subsequently passed. Passing of the calibration checks would then have designated those containers as "acceptable," allowing them to continue through the treatment processes. This anomaly could occur irrespective of any corrective maintenance that may have occurred on the assay systems, without regard to time span between the calibration checks, and without regard to the expert assay review process.

INSIDE THIS ISSUE

The root causes were identified as 1: The Software Quality Assurance Program was not implemented at a level of detail to assure changes to, and deviations from design requirements were identified, understood and corrected in a controlled manner; and 2: Acceptance testing of the FTS was not written or conducted in a manner that assured all parties involved understood the basis for expected system response and acceptable performance.

Source: William McQuiston *Performance Degradation of the Advanced Mixed Waste Treatment Facility Fissile Tracking System*, October 2004. A copy of the report is available on the SQA Knowledge Portal at http://www.eh.doe.gov/sqa/ under SQA Information Sharing and Lessons Learned.

SQA Work Activity #7, Software Safety and NQA-1-2000

This article is the first in a series that will address how the 10 SQA work activities in the draft DOE O 414.1C relate to NQA-1-2000. Draft DOE G 414.1-4 also provides additional detail for implementing the 10 work activities to meet the SQA requirements in the draft DOE O 414.1C. Work activity #7, **Software Safety** was chosen to be the first in this series because NQA-1-2000 contains the least requirements and guidance for Software Safety when compared to the 10 work activities.

The software safety work activity addresses the mitigation strategy for the software components that have potential safety consequences if a fault occurs, whereas the software design and implementation work activity addresses the architecture of the safety software application. NQA-1-2000 addresses the software safety work activity through Subpart 2.7, Section 402 *"Measures to mitigate the consequences of software problems shall be an integral part of the design."* Safety should be designed into a system, just as quality should be built into the system. Safe design of a system, in which software is a subcomponent, utilizes two primary approaches: (1) applying good engineering practices based upon industry proven methods; and (2) guiding design through the results of hazard analysis. The other 9 work activities address applying good engineering practices. NQA-1-2000 Subpart 4.1, Section 100 refers to ANSI/IEEE Standard 7-4.3.2 as providing requirements to address the engineering practices. This work activity, software safety, focuses on guiding the design through the hazard and accident analysis results. Fault avoidance and fault tolerance are key aspects in performing this work activity.

For safety systems, hazards and accident analyses are performed at the system level and then for any subcomponent of the system (including software) that potentially could have an adverse effect on safety. There are several hazard analysis techniques that may be applied to software components. Many of these techniques are performed as preliminary analyses and later updated as more information is known about the requirements and design structure. These techniques include failure mode and effects analysis (FMEA), fault-tree modeling, event-tree modeling, cause-consequence diagrams, hazard and operability (HAZOP) analysis, and interface analysis. Several good technical consensus standards and publications exist that describe these techniques in detail and assist in the proper implementation of the techniques.

The results of the hazards and accident analysis should control the design decisions for the software components that perform the safety functions. The software design should consider principles of simplicity, decoupling and isolation to eliminate the hazards. The safety features should be separate from non-safety modules, minimizing the impact of failure of one module on another. The interfaces between the modules need to be defined and tested thoroughly. Separation of the safety features also allows for more

(continued on page 3)

rigorous software development and verification practices to be applied to the safety components while providing the appropriate and cost effective level of SQA applied to the non-safety components. Software engineering safety design practices should include process flow analysis, data flow analysis, path analysis, interface analysis, and interrupt analysis during the design phase.

When hazards related to software safety functions can not be eliminated, the hazard should be reduced and/or monitored through fault tolerant methods. Additionally, software can experience partial failures that can degrade the capabilities of the overall system that may not be immediately detectable by the system. In these instances, other design techniques, such as building fault detection and self-diagnostics into the software should be implemented. Using external monitors (safety bag) for the software safety functions, n-version programming, and Petri nets are examples of techniques that could be implemented. Self-diagnostics detect and report software faults and failures in a timely manner, and allow actions to be taken to avoid an impact on the system operating safety. Some of these techniques include memory functionality and integrity tests, such as checksums and watch dog timers for software processes, including operating system processes. Additionally, software control functions can be performed incrementally rather than in a single step reducing the potential that a single failure of a software component would cause an unsafe state.

References for these and other techniques are listed below and in the DOE G 414.1-4. For further information, please contact Debra Sparkman at (301) 903-6888 or Debra.Sparkman@eh.doe.gov.

1. Hermann, Debra, *Software Safety and Reliability*, IEEE Computer Society, 1999
2. Leveson, Nancy, *Safeware*, Publishers Addison Wesley, 1995.
3. SAE JA1003, *Surface Vehicle/Aerospace Recommended Practice-Software Reliability Program Implementation Guide, Risk Management*, Society of Automotive Engineers, January 2004.
4. Sparkman, Debra, *Techniques, Processes and Measures for Software Safety and Reliability*, Lawrence Livermore National Laboratory, UCRL-ID 108725, 1992.
5. IEC 61508 Part 7, *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, Overview of Techniques and Measures*, International Electrotechnical Commission, Geneva, Switzerland, 1998.
6. IEEE Std 7-4.3.2-2003, *IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations*, Institute of Electrical and Electronic Engineers, Inc., 2003.

Visual Sample Plan 3.0 Release

Visual Sample Plan (VSP) Version 3.0 has been released. This enhanced version may be downloaded from http://dqo.pnl.gov/vsp without charge. The VSP 3.0 Users Manual has also been completely updated and is available for downloading from the website. Elements of VSP are sponsored by several U.S. government agencies including Department of Energy (DOE), Environmental Protection Agency (EPA), Department of Defense (DoD), Technical Support Working Group (TSWG), and Department of Homeland Security (DHS). VSP is a statistically based data quality objective (DQO)-based software tool for developing optimal sampling strategies for contaminants in soil, sediments, water and on building surfaces and can also help determine the most probable locations of unexploded ordnance. With over 4,000 users, VSP is widely used by environmental scientists and engineers and accepted by regulatory agencies. Many new features and

(continued on page 4)

capabilities were added since Version 2.0. Some of these include:

- Collaborative Sampling for Tests of Hypothesis and Confidence Interval Estimation;
- Perimeter Sampling Using Composite Sampling Techniques;
- Improved Methods for Detecting Target Areas where Unexploded Ordnance is Most Likely;
- Numerous Mapping Improvements including annotations, 3D room viewing/drawing/specification, Shape file (.shp) import/export, background picture world files with transparent sample areas, user defined room parameters, room labels, sample area information dialog, sample area nesting, posting plots, infinite map zoom, and more;
- New Tools: Distance Measurer, Reset Design, Sample Labeler, Map Legend; and
- Improved Customizable Automatically Generated Reports and Online Help.

A comprehensive list of added capabilities and enhancements are shown in the release notes at http://dqo.pnl.gov/vsp.

The Department will be offering several VSP training sessions over the course of the year and is sponsoring a 2.5 Day VSP training course at two DOE sites in the Spring of 2005. One or two other offerings of this course are planned during the Summer of 2005. Discussions are also in progress with the EPA and others interested in VSP training. If you would like to attend or sponsor any future VSP training courses, please send an email response to vsp.training@pnl.gov.

For further information, please contact Brent Pulsipher, Pacific Northwest National Laboratory 509-375-3989, brent.pulsipher@pnl.gov or George Detsis at (301) 903-1488, George.Detsis@eh.doe.gov.

DID YOU KNOW?

Test your knowledge of SQA with the following **"did you know"** facts:

Did you know that in order to ensure the maintainability of the requirements, the requirement specification should be evaluated to ensure that each requirement is stated in only one place? If the requirements that are stated in more than one place in the specification or duplicated in multiple documents, it becomes difficult to maintain the internal consistency of the requirement specification.

CURRENT ACTIVITIES

Software Inspections a.k.a. Formal In-Process Technical Reviews

Software Inspections, a.k.a. Formal In-Process Technical Reviews, have been around for thirty-five years. Yet many software practitioners don't know or use this practice. It is seen as too "low tech" to be worth much. It's not as glamorous as automated specification-based testing, but it's worth every penny invested. Many practitioners don't like the idea of submitting their documents and code to others for review, though none of them has evidence their work is error-free. Admit it, we make errors. Software professionals know this, and they know that making better software means finding and removing the defects in a product before the user finds them. Inspecting software requirements, software and test design documents, code, and test cases always improves those work products. Not just any review approach will be cost effective, however.

The software inspection process is a defined process with improvable guidance and tangible results, providing efficient and effective reviews of your software work products. It has an extensive history of improving software as well as improving projects. Some software development organizations at Sandia National Laboratories (SNL) always schedule software inspections into their project plan because they have measured a higher quality product at less cost than any other improvement they have made in their processes. The Software Quality Engineering department at SNL has trained quite a few organizations, inside and outside SNL, in this classic technique.

The inspection process is independent of development paradigm, language, application domain, or computing environment. It works because it accounts for human cognitive ability – our ability to use natural language, tables, and figures in our documents to describe and explain complex systems, and to read and understand those descriptions and explanations. The universality of these abilities has been incorporated into quantitative guidelines that let you determine how good an inspection is before you use the inspection results to determine how good the inspected product is. There are few places in software development where you can get such measurable evaluations.

Software inspections don't remove the need to actually test executable code; they often find many code defects with less effort than it would take for testing to find them. An intelligent and balanced use of inspections and tests is one of the quickest ways to improve the quality of delivered software. Numerous books, papers, and web pages provide excellent information about software inspections. Some good starting points to learn more about this technique are listed below.

Walkthroughs, Inspections, and Technical Reviews, Daniel P. Freedman, Gerald M. Weinberg (Dorset House Publishing, 1982, ISBN: 0-932633-19-6)

Software Quality, Capers Jones (International Thomson Computer Press, 1997, ISBN: 1-85032-867-6)

Software Inspection, Tom Gilb, Dorothy Graham (Addison-Wesley, 1993)

The WWW Formal Technical Review Archive, http://www2.ics.hawaii.edu/~johnson/FTR/

Bibliography, Reviews and Inspections, http://www2.umassd.edu/SWPI/TechnicalReview/review.html

(continued on page 6)

DOE Directives (continued from page 5)

Article submitted by Dwayne Knirk, Sandia National Laboratories, Software Quality Engineering Department, (505) 844-7183, dlknirk@sandia.gov.

NNSA Safety Software Assessment Conducted at TA-V

A safety software assessment was performed on a representative sample of safety software at the Sandia National Laboratories (SNL) Research Reactor and Experimental Programs (RREP) organization in Technical Area V (TA-V). The assessment was conducted to meet commitments made in the Department's implementation plan for Defense Nuclear Facilities Safety Board Recommendation 2002-1, *Quality Assurance for Safety Software at Department of Energy Defense Nuclear Facilities*. The assessment was performed in accordance with Criteria and Review Approach Document (CRAD) 4.2.3.1, Criteria and Guidelines for the Assessment of Safety System Software and Firmware at Defense Nuclear Facilities, and CRAD 4.2.4.1, Assessment Criteria and Guidelines for Determining the Adequacy of Software Used in the Safety Analysis and Design of Defense Nuclear Facilities.

The assessment team included team leads from NNSA/SSO and team members from DOE/HQ EH, NNSA/SC, and SNL. The team held preliminary meetings to establish guidelines for the conduct of the assessment. The on-site portion of the assessment was conducted over the period March 9-16, 2004. A final briefing of the assessment results was held on March 16, and the contractor was very receptive to the issues identified by the briefing.

Safety software includes both safety system software and safety analysis and design software. The safety software selected for this assessment included:

1. Safety System Software Annular Core Research Reactor (ACRR) Rod Control and Reactor Console (RC/RC) Software
2. Safety Analysis and Design Software Critical Heat Flux Ratio (CHFR) Software - Design/Analysis Software
   Monte Carlo N-Particle Transport Code (MCNP) - Design/Analysis Software

The software reviewed ranged from externally developed "COTS-type" software (MCNP) to in-house developed codes (CHFR and ACRR). A final report was developed, reviewed for factual accuracy, and submitted for distribution to Dr. Everet Beckner, Deputy Administrator for Defense Programs, National Nuclear Security Administration.

DID YOU KNOW?

Test your knowledge of SQA with the following **"did you know"** facts:

Did you know that the criteria for accepting software from an outsourced software supplier should be established and agreed upon as part of the contract with that supplier? Acceptance criteria should be established and agreed upon in writing by both the supplier and customer using the contract process. This criterion should include any acceptance testing, supplier assessments or reviews, or proof of completing software development activities such as requirements traceability to supplier test results. The contract should also contain any requirements for the supplier to conform to industry, DOE, or other government standards, orders, procedures or guidance and any accreditations or certifications required such as ISO 9000 or SEI CMMi.

(continued on page 7)

NNSA Safety Software (continued from page 6)

M. Hamilton, M. Ortega, S. Sen, N. Morley, D. Peercy, D. Talley, and T. Vanderbeek, *"Assessment of Safety System Software and Safety Analysis and Design Software at Sandia National Laboratories Technical Area-V (TA-V),"* National Nuclear Security Administration Report, May 26, 2004.

Submitted by David Peercy, Sandia National Laboratories, Albuquerque, NM (505) 844-7965, depeerc@sandia.gov and Mark Hamilton, NNSA/ Sandia Site Office (SSO), Albuquerque, NM (505) 845-4045, mhamilton@doeal.gov

DID YOU KNOW?

Test your knowledge of SQA with the following **"did you know"** facts:

Did you know that a second party quality system audit is conducted by a company (the customer) to determine if a prospective supplier has the capability to deliver a product with the required functionality and reliability? A second party audit is initiated by and conducted by the customer during the vendor selection process to evaluate the quality management systems of the supplier in delivering acceptable products or services. If the customer contracts with another company to conduct this audit, the audit is referred to as a third party quality system audit. A process audit would focus solely on a single process being performed and would not provide adequate information that the supplier had in place the overall quality management system to deliver acceptable products or services.

FUTURE ACTIVITIES

Safety Software Database Pilot

The Office of Quality Assurance Programs (EH-31) with the assistance of the Office of Information Management (EH-33) is developing a Safety Software Database as part of the web-based SQA Knowledge Portal. The purpose of the database is to catalog safety software being used throughout DOE facilities and allow for the sharing of SQA information, lessons learned and problem reporting.

The database will initially be populated using code survey data collected in commitment 4.2.1.5 of the SQA Implementation Plan. Site/Laboratory and Facility information will be imported from the Safety Basis Information System (SBIS) that is maintained by EH-33. DOE facilities will access the database to update site specific safety software codes being used, the version of the code, its application or how it is

(continued on page 8)

being used, the software type, DOE contact information for the code, etc. This will allow EH-31 to develop reports and share information and problem reporting with all users of specific codes.

This effort will also be evaluated by the EFCOG Safety Analysis Working Group. The goal is to develop a structure to facilitate the transfer of information between code users, developers and safety analysts. The database will also help facilitate training and enhance problem reporting and sharing of lessons learned.

Anyone wishing to participate in an upcoming pilot of the Safety Software Database should contact Chip Lagdon at (301) 903-4218 or Chip.Lagdon@eh.doe.gov.

Interest in Additional ASQ Software Quality Engineer Courses

Last year, twenty-eight individuals attended the American Society for Quality (ASQ) Software Quality Engineer courses that were presented in Germantown, Maryland. This one-week course satisfies nine of the twelve competencies in the Safety Software Quality Assurance Functional Area Qualification Standard. The Office of Quality Assurance Programs (EH-31) is soliciting interest in conducting additional courses this year. If you are interested, please contact Chip Lagdon at (301) 903-4218 Chip.Lagdon@eh.doe.gov or Rick Martinez at (240) 686-3059 rmartinez@pec1.net.

Defense Nuclear Facilities Safety Board Briefing

The sixth briefing to the Defense Nuclear Facilities Safety Board on the status of the SQA Implementation Plan will be conducted sometime in February 2005. At this briefing, the Office of Environment, Safety and Health, Office of Environmental Management and National Nuclear Security Administration representatives will provide updates on progress made in completing the SQA Implementation Plan commitments. For further information, please contact Chip Lagdon at (301) 903-4218 or Chip.Lagdon@eh.doe.gov.

Newsletter Articles Needed

If anyone has an interest in writing an article for this periodic newsletter, please contact Chip Lagdon at (301) 903-4218 or Chip.Lagdon@eh.doe.gov. Please share any activities that your site is doing with respect to software quality assurance (SQA) that may help other sites or provide useful lessons learned. As we continue to verify status of SQA in the Department, field input is critical in fostering an environment that promotes continuous sharing.