Software Assurance for Project Managers

Joel Wilf
Group Supervisor
Software Assurance and Assurance Research
Jet Propulsion Laboratory, California Institute of Technology

July 20, 2012

© 2012 California Institute of Technology. Government sponsorship acknowledged

Purpose

Answer Project Manager questions about SQA:

- What is SQA?
- What is the value of SQA?
- How is SQA implemented?
- How do I lower the cost?
- How does SQA Research help?
- Questions from the group

What is SQA?

- NASA Software Assurance:

  "The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures." – NASA STD 8739.8

- JPL SQA:

  "Throughout a project's life cycle, SQA independently checks software products and processes to make sure they meet JPL's standards of quality, reliability, and safety." – Informal

- JPL SQA ~ NASA Software Assurance
  - For quality, reliability, and safety
- Key points for JPL SQA:
  - Do throughout the life-cycle
  - Check both software products and processes
  - Independent => checker is not the doer
  - Objective => check against standards

What is not SQA?

- SQI
  - Process improvement – not assurance
  - JPL's SDR and SDSPs
  - JPL's CMMI certification
- IV&V
  - Assigned by & reports to NASA
  - Greater separation from project
  - Some areas overlap
- Testing Organizations
  - Test Orgs: run the V&V tests
  - SQA: ensures tests are traced, complete, performed as planned
- Everything so far refers to SQA's **Role**
- To fully understand, need SQA **Value**

What is the Value of SQA?

- Example SQA Success Story:

> While checking the requirements verification matrix for completeness, SQA discovered a missing test case: needed to verify solar array switching table.
>
> Once test case was developed and run, it revealed a critical defect that would have caused the entire solar array to be enabled, rather than selected segments.

- Note roles:
  - Engineers develops SW
  - Test team develops and run verification tests
  - SQA assures that the software has been completely tested
- SQA value is clear: it led to uncovering high-impact defect
- Question: would there still be value if SQA had assured that the testing was **complete**, with **no missing test cases**?

Value in More Confident Decisions

- Yes, value is in:
  - Reduction of uncertainty: project knows software is completely verified
  - Confidence in delivery decision, due to above
- SQA value proposition:

SQA provides independent, objective evidence that reduces the uncertainty about software attributes, in order to help projects make better decisions – in order to reduce the risk due to software

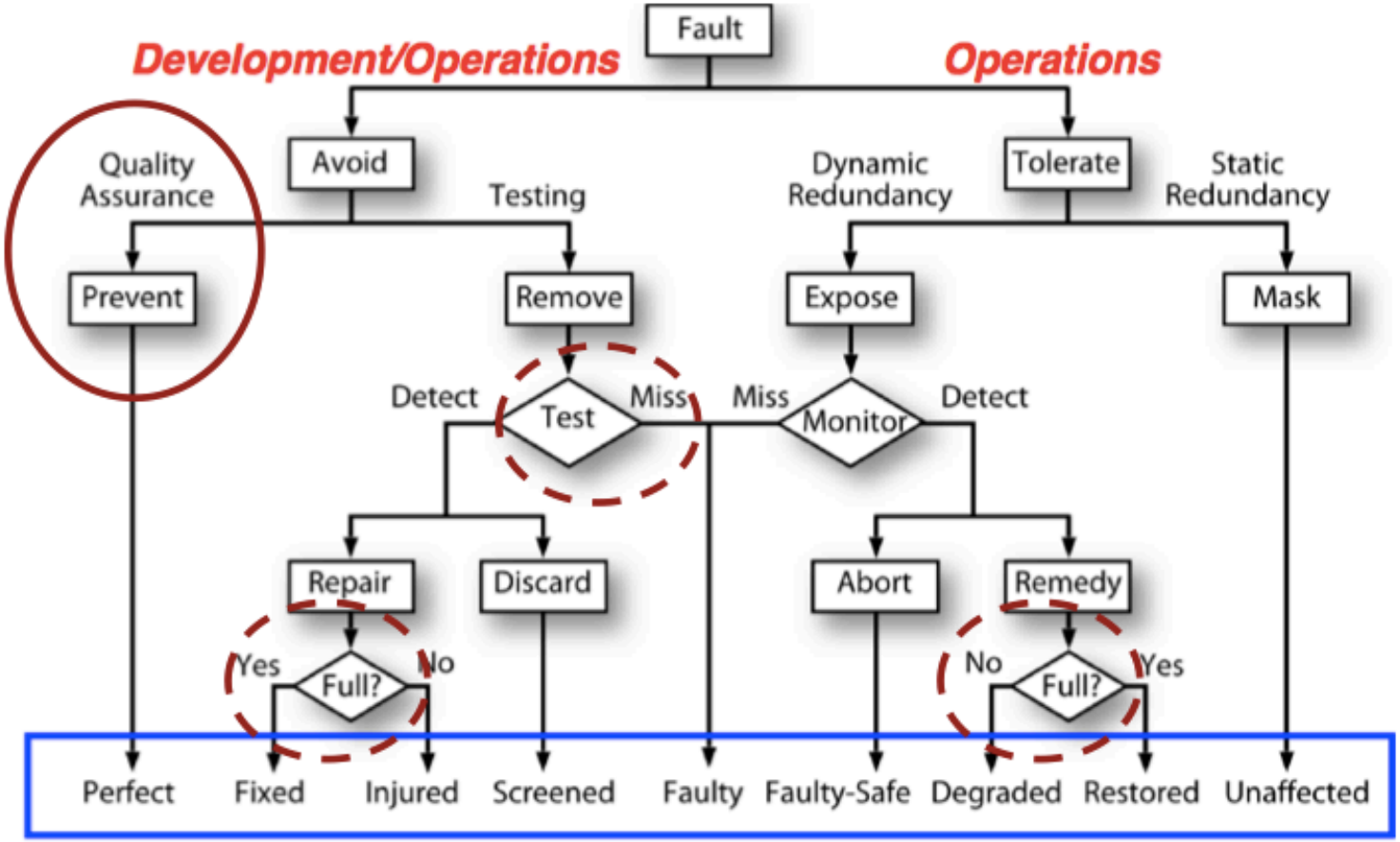
- SQA is an independent "checker"
- Checks reduce uncertainty about software attributes
  - Completeness, readiness, compliance, reliability, safety, security, etc.
  - Note: "reduce uncertainty" == "increase confidence"
- The checks should support project decision-making
  - Gateway reviews, resource allocation, risk mitigation
  - Most decisions at SW level – but some at PM level

Goal: reduce risk from bad decisions due to software uncertainty

What Increases Value?

- Higher value when the software is…
  - More critical (higher risk of wrong decision)
  - More uncertainty in state
  - More complex => more uncertainty
  - Bigger => more complexity
- Higher value when SQA…
  - Focuses on decisions, uncertainty, risk
  - Improves tools/techniques for efficiency
  - Reduces more uncertainty (effective)
  - Reduces uncertainty earlier in life-cycle
- Value can be hard to see
  - In prevention (bad things didn't happen)
  - In cleaning things up (just part of decision process)

Value can be Hard to See

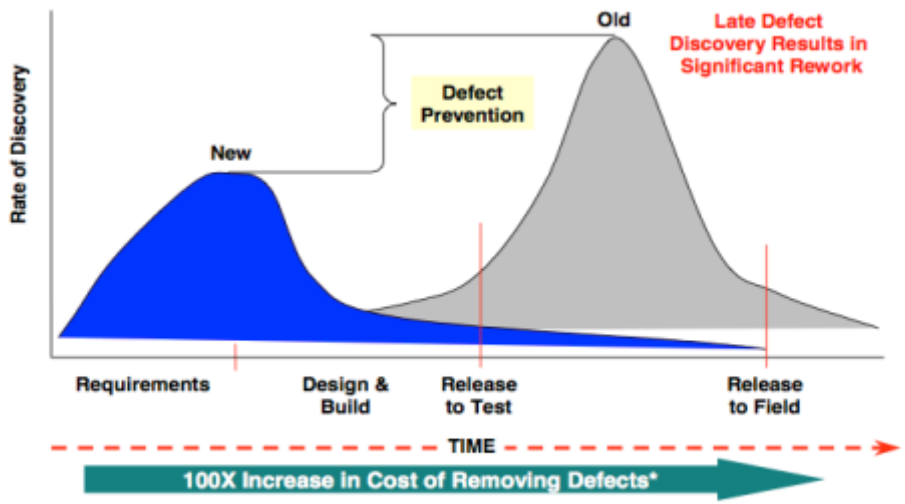


Value is in prevention (——) or embedded in decision (– – –)

Can Value be Measured?

- Measuring the value quantitatively is a work in progress:
  - Software cost models (COCOMO) include SQA as a positive factor
  - Capers Jones analysis of project defects: One of the determining factors in achieving 90% and better success in removing defects is "Active SQA > 3% of development budget"
  - JPL SQA gathering data to quantify SQA effectiveness in reducing opportunity loss from bad decisions, and structural equation models that that measure SQA influence on positive project outcomes

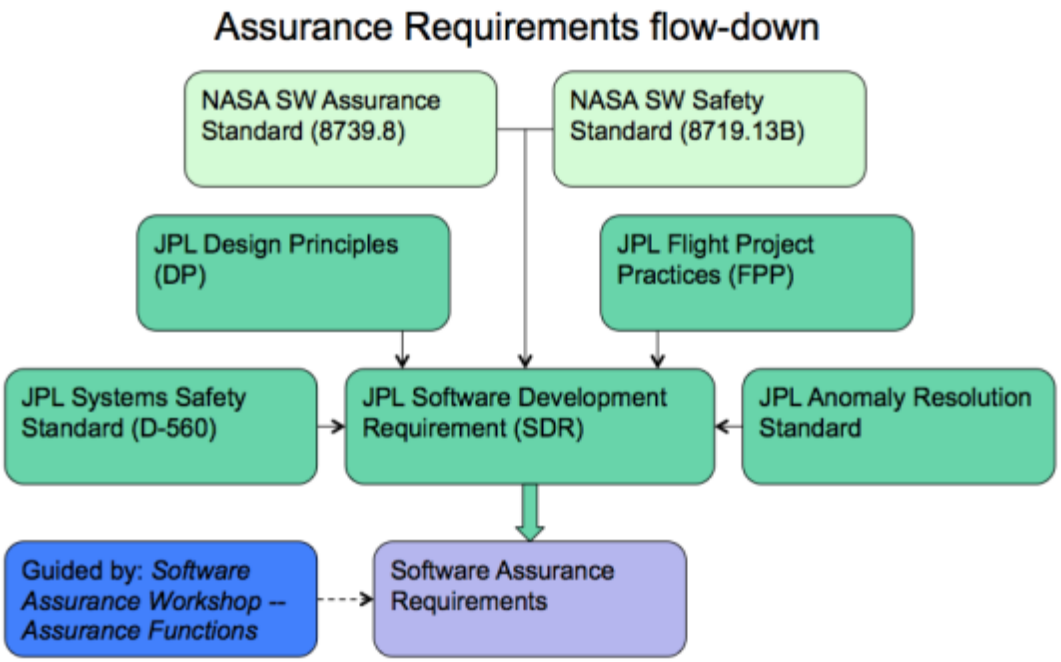
- Value of early defect prevention has been recognized often in the literature (e.g., Barry Boehm's chart below showing benefit of doing requirements assessment in addition to a testing regime):



Source*: Boehm, Barry. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, Inc. 1981.
Boehm, Basil. "Software Management." *IEEE Computer*, January 2001.

How to Implement for Value?

- Start with SQA requirements that represent best practices
- JPL Software Development Requirement (SDR)
- JPL Software Assurance Workshop (2005)
  - Chris Jones, Jan Chodas, Richard Brace, and others
  - Recommended project Engineering vs. Assurance functions
  - These guide selection of Assurance requirements
- SQA requirements flow-down shown opposite



Tailor SQA to fit Project

- Tailor SQA activities to fit project SW life-cycle, for example:

Concept Def. & Planning*

Assure SW classification, process tailoring, inherited software, management plans, contractor oversight

Requirements Def. & Analysis

Assure requirements, check trace between requirements levels, check verifiability

Architectural Design

Assure architecture, early assurance of nonfunctional requirements ("-ilities")

Cycle 1: Detailed Design, Code, Debug, Test & Delivery

Module reviews, static code analysis

SRCR: Assure deliveries to I&T, ATLO, flight

Cycle n: Detailed Design, Code, Debug, Test & Delivery

P/FR closures, P/FR burn-down

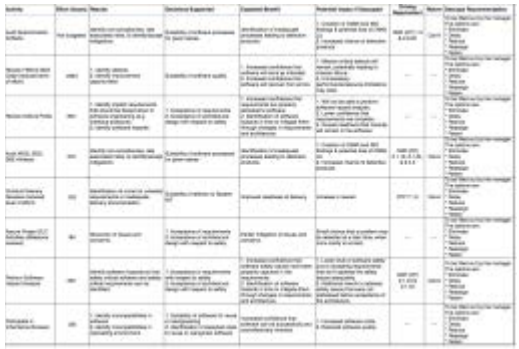
* Model shown is Iterative life-cycle from SQI's "**Software Life-Cycles and Milestones**"

- In every phase, SQA activities should help reduce both programmatic (cost, schedule, resources) and mission/technical risks

Focus on Value in Activities

- Example: SQA Activities – with notes to qualitatively understand the value

Activity	Results	Decision Supported	Evidence
Assess Software L4 Requirements	Concurrence on requirements readiness, Findings	Requirements ready to be baselined?	Requirements in DOORS, SDSP Requirements Checklist
Run Static Code Analysis	ODASA report, project bug report, dispositions	Is code ready for build level testing?	ODASA output, developer comments
SRCR	Signed SRCR form, JIRA Findings, Actions	Is software release ready for delivery to I&T/ATLO?	Software package documents (RDD, etc.)
Close P/FRs	Signed P/FR, information added to P/FR in PRS system, JIRA Findings	Can the P/FR be signed off as completely fixed and documented?	P/FR, testbed activity reports, developer and tester comments



- We use a more detailed table like the one above for descope decisions. Columns include: Activity, Effort, Results, Decisions Supported, Expected Benefit, Potential Impact if Descoped. Driving Requirement, Waiver, Descope Recommendation

How to Lower Cost of SQA?

- Tailor and prioritize
  - High risk, uncertainty, value
  - SQA Findings risk-rated
  - Okay to write waivers
- Sampling for process audits
- Use trusted supplier records
- Use IV&V inputs
- Embedded SQA <- Important
  - Assess **prior** to gateway
  - Preliminary Findings fixed
  - Result is “clean” as possible
- SQA continuous improvement
  - Processes, templates
  - Increasing use of data/metrics
  - Infuse advanced tools

SQA Cost Model

Instructions: Evaluate each item, estimate for the project, update the text to reflect rationale and estimates, then copy the number into right hand columns. Modify this template as needed to reflect project needs and complete a supportable estimate. Provide zeros where tasks are no bid. Maintain links to Summary BOE sheet - hours on this sheet will transfer to summary sheet.

Key Terms Used:		
	Participate:	to be a contributing member with defined roles and responsibilities
	Perform:	to lead or conduct a prescribed activity
	Assess:	to evaluate processes/products and provide assessment results and recommendations
	Review:	to extract for informational purposes and use as a potential input into SQA activity planning

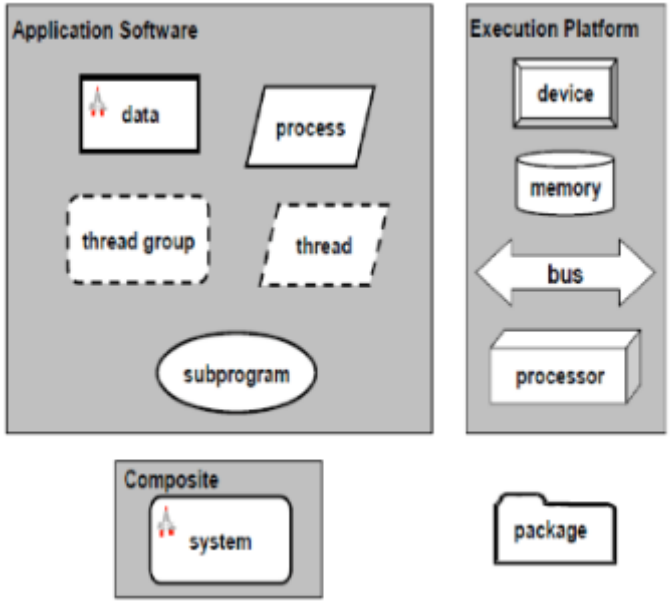
	# of Actions	Hours Per Action	Total HOURS
1. Software Quality Assurance Planning/Managing:			
A - Perform: Write, Release, and maintain Project Software Quality Program Plan: 40 hours (FPP 7.1.4; 7.3.6).	1	40	40
B- Perform: Develop schedule for Software Quality Assurance Activities: 16 hours.	1	16	16
C - Perform: Maintain schedule for Software Quality Assurance Activities: Reviewed and updated at 2 hours per quarter. (assume 6 months and 0.5 hr per month)	16	2	32
D - Perform: Produce Software Assurance Work Agreements (Phases A, B, C, D, and E) (FPP 5.2.10; 5.18.5.2; 6.17.1) (SDR 2.2) - 8 hours per WA. Assume two: Phase B and Phases C/D	2	8	16
E - Perform: Monthly Status reports to SQA/Quality Management: 2 hour per report.	48	2	96
F - Perform: Initial Cost Estimation in Phase A and updates as required when project re-baselines due to funding changes or modified scope. Covered by line management	0	2	0
Total Software Quality Assurance Planning/Management Hours:			200
Assumptions: <enter assumptions for boe - if more space needed insert rows>			

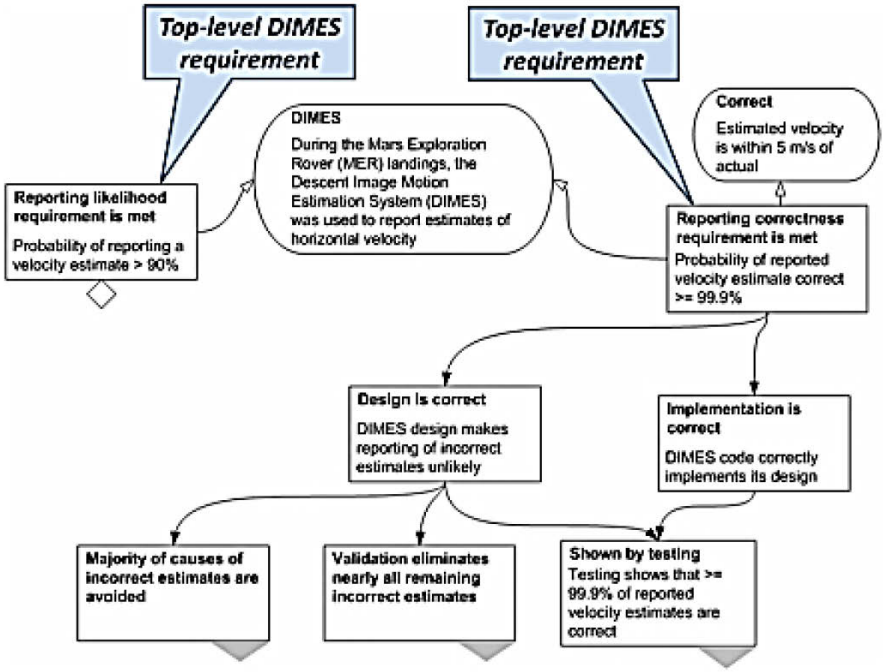
	# of Actions	Hours Per Action	Total HOURS
2. SDSP Tailoring and SMP Compliance:			
A1 - Participate and Review: FPP and SDR Waivers generated by project: Assume 60 hours total	1	60	60
A2 - Assess ESA standards for SEIS and HP3 instruments: 80 hours	1	80	80
B1 - Participate: SDSP Tailoring by project: For Robotic Arm and Camera software development and Oversite process: 60 hours	1	60	60
B2 - Assess: SMP Compliance (FPP 6.11.3.2; 6.11.3.3; 6.18.5; 7.3.9) (SDR 1.4; 2.1): # of Actions is based on the number of assessed pages. The Hours Per Action is based on assessing 5 pages per hour. Assume in-house project PSMP, Robotic Armm and Camera SMP - 150 pages (Contractor SMPs under Supplier Assurance)	150	0.2	30
C - Perform: Assure software identification and classification per DocID 57653 - Spacecraft, GDS, and three instruments	5	2	10
Total SDSP Tailoring and SMP Compliance Hours:			240

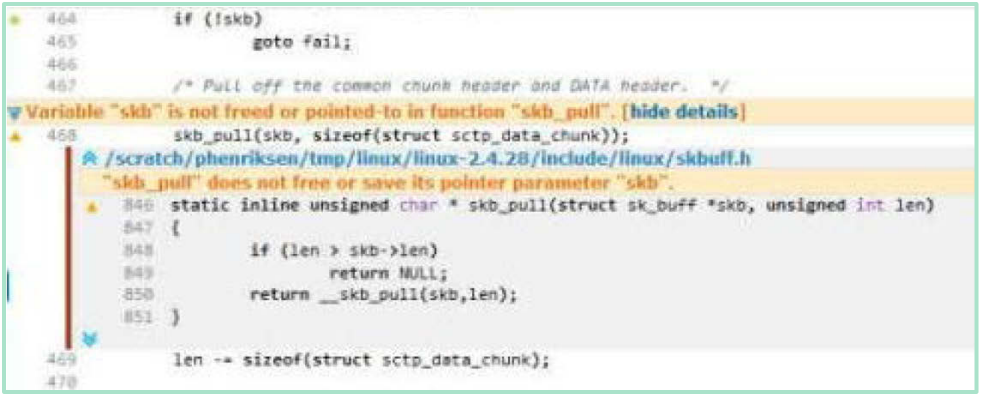
How does SQA Research help?

- JPL Assurance researchers work closely with practitioners
  - Assurance Technology Program Office (ATPO)
  - Laboratory for Reliable Software (LARS)
  - NASA Software Assurance Research Program (SARP) sponsors
- Research areas include:
  - Modeling with the Architecture Analysis and Design Language (AADL)
    - Modeling the run-time system in AADL enables analyzing system properties such as latency and resource utilization at the design phase, prior to coding
  - Static code analysis
    - Static code analyzers can read software source code and find patterns indicating possible defects, prior to the testing phase
  - Software safety cases
    - Systematic way of communicating the evidence that a system/subsystem or function is safe
  - Assurance for security – cyber security
    - Tools and techniques for reducing weaknesses and vulnerabilities that can be exploited by cyber attacks

Some examples







Discussion

Your questions and comments welcome…

Backup Slides

JPL Definitions

Software - Computer programs, procedures, rules, and associated documentation and data pertaining to the development and operation of a computer system. Software also includes:

- COTS (commercial off-the-shelf), GOTS (government off-the-shelf), MOTS (modified off-the-shelf),
- embedded software,
- reuse, heritage, legacy,
- auto generated code,
- firmware (restricted to symbolic logic and associated data loaded into programmable logic devices to be executed on an embedded processor), and
- open source software components.

NASA Definitions (1/2)

Software Product Quality - A measure of software that combines the characteristics of low defect rates and high user satisfaction.

Software Assurance - The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures. For NASA this includes the disciplines of Software Quality, Software Safety, Software Reliability, Software Verification and Validation, and IV&V.

Software Quality - The discipline of software quality is a planned and systematic set of activities to ensure quality is built into the software. It consists of software quality assurance, software quality control, and software quality engineering.

Software Quality Assurance - The function of software quality that assures that the standards, processes, and procedures are appropriate for the project and are correctly implemented.

Software Quality Control - The function of software quality that checks that the project follows its standards, processes, and procedures, and that the project produces the required internal and external (deliverable) products.

Software Quality Engineering - The function of software quality that assures that quality is built into the software by performing analyses, trade studies, and investigations on the requirements, design, code, and verification processes and results to assure that reliability, maintainability, and other quality factors are met.

NASA Definitions (2/2)

Software Reliability - The discipline of software assurance that

- (1) defines the requirements for software controlled system fault/failure detection, isolation, and recovery;
- (2) reviews the software development processes and products for software error prevention and/or reduced functionality states; and
- (3) defines the process for measuring and analyzing defects and defines/derives the reliability and maintainability factors.

Software Safety - The discipline of software assurance that is a systematic approach to identifying, analyzing, tracking, mitigating, and controlling software hazards and hazardous functions (data and commands) to ensure safe operation within a system.

Software Validation - Confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled.

Software Verification - Confirmation by examination and provision of objective evidence that specified requirements have been fulfilled.

Software IV&V - Verification and validation performed by an organization that is technically, managerially, and financially independent. IV&V, as a part of software assurance, plays a role in the overall NASA software risk mitigation strategy applied throughout the life cycle, to improve the safety and quality of software.

NASA IV&V and SQA

IV&V is one of five disciplines identified in NASA's definition of software assurance

- The others are software quality, safety, reliability, and V&V*
- JPL's Flight Project Practices state that:

"Software IV&V is an independent activity and is not used in place of, or as a substitute for, project software V&V or software quality assurance (SQA)."

The goals and methods of SQA and NASA IV&V are complementary.

IV&V	SQA
Product-focused	Balance of product and process assessment
Fully independent, with independent funding	Objective feedback, with project funding
Team is separate from development team	Team is embedded in development team
Performed on specific products delivered to IV&V by the project	Performed interactively and concurrently with development
Mandatory for projects that meet specific criteria (MSL, Juno, and GRAIL at JPL)	Available to any JPL project or task that includes software
Both should be performed across the full software development life-cycle	

Top Ten Potential Sources of Risk on Software Tasks*

Risk Sources	Potential Mitigations
Personnel shortfalls	Staffing with top talent, job matching, team building, key personnel agreements, cross training
Unrealistic schedules and budgets	Detailed multi-source cost and schedule estimation, design to cost, incremental development, software reuse, requirements scrubbing.
Developing the wrong functions and properties	Organization analysis, mission analysis, operations-concept formulation, user surveys and user participation, prototyping, early users' manuals, off-nominal performance analysis, quality-factor analysis.
Developing the wrong user interface	Prototyping, scenarios, task analysis, user participation.
Gold-plating	Requirements scrubbing, prototyping, cost-benefit analysis, designing to cost.
Continuing stream of requirements changes	High change threshold, information hiding, incremental development (deferring changes to later increments).
Shortfalls in externally furnished components	Benchmarking, inspections, reference checking, compatibility analysis.
Shortfalls in externally performed tasks	Reference checking, pre-award audits, award-fee contracts, competitive design or prototyping, team-building.
Real-time performance shortfalls	Simulation, benchmarking, modeling, prototyping, instrumentation, tuning.
Straining computer-science capabilities	Technical analysis, cost-benefit analysis, prototyping, reference checking.

*From "Software risk management: principles and practices", Boehm, B.W.; Software, IEEE Volume 8, Issue 1, Jan. 1991 Page(s): 32 - 41

SQA Requirement Examples

- Examples of SQA requirements:
- For SQA Planning:
  - *"All software shall be classified in accordance with the classifications described in the Software Development Requirement (SDR), with the concurrence of the designated SQA representative (SDR 2.1.11-2.1.12)."*
  - The first step in understanding the level of process rigor and assurance to apply to software is to determine its classification (e.g., Class A – human rated, Class B – mission critical, Class C – mission support) and whether it is safety critical
- For Product and Process Assurance:
  - *"At designated points in the development cycle, selected software work products shall be evaluated to verify compliance with JPL institutional requirements and standards adopted by the project plan (FPP 7.3.9, SDR 2.4.5)."*
  - This requirement is written at a high level, to allow flexibility for the project to select which products to check
- For Delivery Assurance:
  - *"Prior to delivery, there shall be an independent verification that all software requirements identified for this delivery have been met, that all approved changes have been implemented, and that all defects designated for resolution prior to delivery have been resolved (SDR 3.7.1)."*

SQA Requirement Examples

- For Delivery Assurance (continued):
  - You may have recognized these checks as typically performed as part of a Software Review and Certification Record (SRCR)
- For Contractor Quality Assurance:
  - *"Supplier's software development processes shall be audited by SQA against the supplier's approved software plan, in accord with the requirements in Software Development (SDR 2.2.22)."*
  - This is to ensure that the supplier is following JPL standards in their software development and reduce the potential for unhappy surprises when they deliver. Again, selection of processes to audit is left to the project
- For Reliability Assurance:
  - *"Software P/FRs shall be reviewed and signed by SQA prior to closure (Anomaly Resolution Appendix A, paragraph 21-B)."*
  - SQA takes on the role of the "Software Reliability Engineer" in assuring the closure of P/FRs.
- Given the SQA Requirements, some with a lot of flexibility for tailoring – the idea is to implement for the right Value/Cost for the project

Examples of SQA benefits

Software Concept Definition and Planning - Improved software plans and processes that will deliver quality software with *no wasted effort*

- ICCA – SQA reviewed the SMP, and determined that some elements of the S/W should be classified as Class C, instead of Class A as proposed by the Project. This saved the Project money, and enabled them to focus on the more critical Class A elements of the S/W.

Software Requirements Definition and Analysis - More *complete and correct software requirements*

- Aquarius – The SQE discovered a discrepancy between the command dictionary and the behavior of the instrument FSW. This led to a clarification of the command dictionary, and improved verification of the software.

Detailed Design, Code, Debug, Test & Delivery - Increased confidence that the *software is safe and reliable*

- JUNO FSW – With the PSSE, instituted an informal SRCR prior to PSR, which led to the discovery that one of the instrument commands had not been verified. When the test was added to the test procedure, the command failed. Saved valuable testing time in the LMA Systems Testing Lab.

SQA's benefit should be a reduction in the severity and frequency of issues realized by Projects, and increased confidence in the delivered product – not last minute mission-saving heroics

Software Assurance Workshop

Software Assurance workshop was held in Nov. 2005

- Convened by Richard Brace, Jan Chodas, and Chris Jones
- Included representatives from Mission Assurance, Engineering, Project/Program Management, and Line Management
- Intended to further define the role of the SQA group in performing software assurance

SQA improvements to implement the workshop findings

- Rewrote the SQA charter to reflect the workshop conclusions
- Hired new Deputy Section Manager to focus on improving SQA
- Recruited Senior Software/SQA experts to support this activity
- Focused SQA research on improving the operations

Created an Organizational Software Assurance Management Plan that plans and tracks implementation of the workshop findings

Software Assurance Workshop Results

November 2005

Core Software Assurance Functions	Engineering	Assurance
Assurance Planning/Managing	Perform	Perform
PPQA: Process and Product Assessments, Reporting		Perform
IV&V Technical Interface	Perform	
Safety and Hazard Analysis		Perform
Design of Safety-critical Software	Perform	
Software Classification	Perform	Assure
Subcontract/Supplier Assurance		Perform
Subcontract/Supplier Engineering Insight and Oversight	Perform	
Requirements Traceability	Perform	Assure
Software Reliability Analysis	Perform	Assure
Contribute to Institutional Standards	Perform	Perform
Risk Management	Perform	Assure
Participate in Project Software Change Control Board	Perform	Perform