Best Practices in Software Assurance

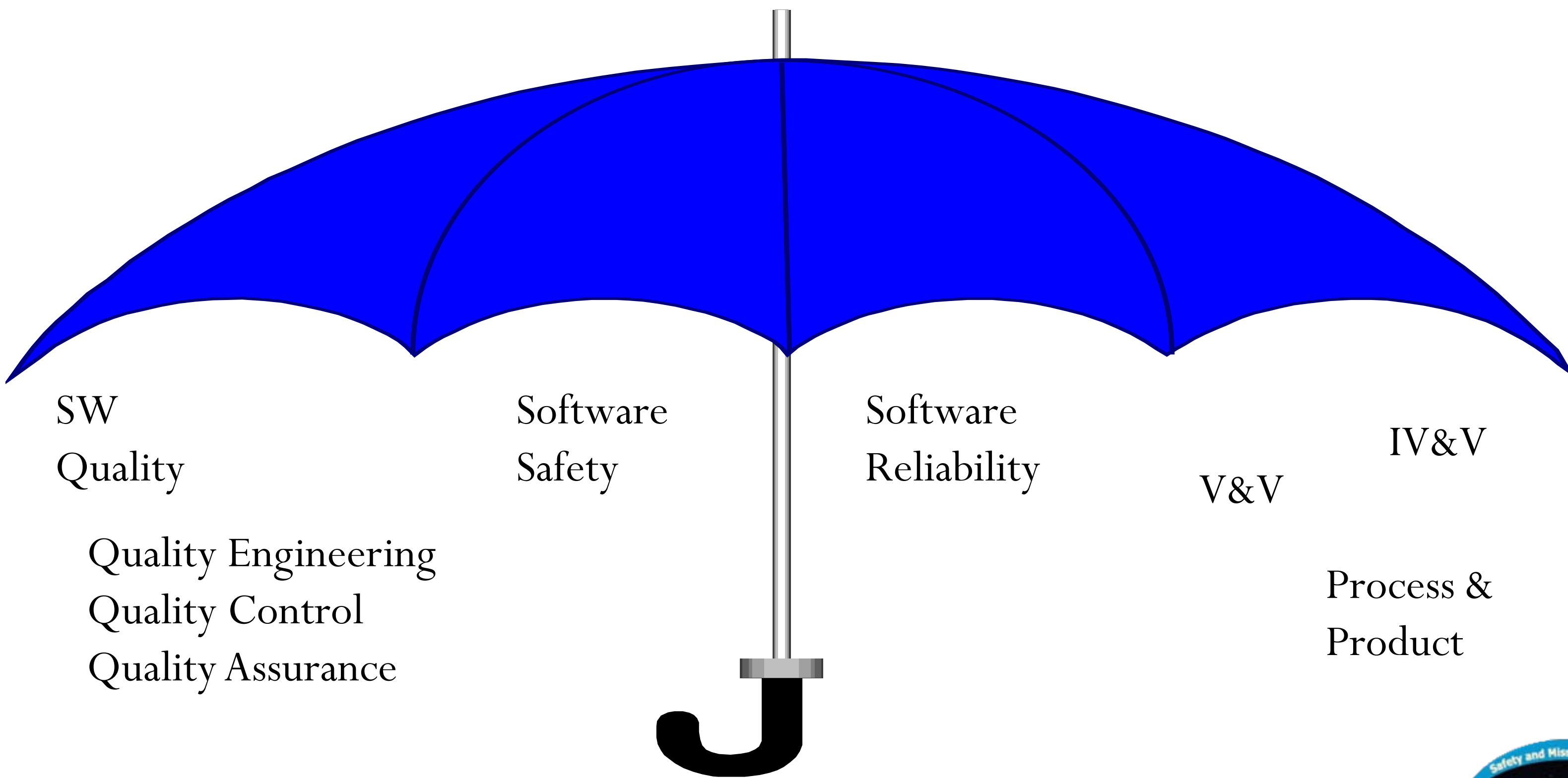
Risha George
Software Safety Lead, GSFC

NASA GSFC Supply Chain Management Conference

October 21, 2010

Software Assurance Overview

Software Assurance is an umbrella risk identification and mitigation strategy for safety and mission assurance of all NASA's software



What do we do

- The planned and systematic set of activities that ensure conformance of software life cycle processes and products to requirements, standards, and procedures.
- Assure that software products are of high quality and operate safely
- Assists in risk mitigation by minimizing defects and preventing problems
- Assures that software meets its specified requirements, conform to standards, are consistent, complete, correct, safe, and reliable and satisfy customer needs.
- Assures that all software processes are appropriate and implemented according to plan, meet any required standards, and quality requirements.

Software Assurance Disciplines

- **Software Quality**: planned and systematic set of activities to assure quality is built into the software. Assures that the standards, processes, and procedures are appropriate for the project and are correctly implemented.

- **Software Safety**: a systematic approach to identifying, analyzing, and tracking software mitigation and control of hazards and hazardous functions to ensure safer software operation within a system.

- **Software Reliability**: defines the requirements for software controlled system fault/failure detection, isolation, and recovery; reviews the software development processes and products for software error prevention and/or reduced functionality states

- **Verification and Validation**: verification ensures "you built it right" and validation ensures "you built the right thing"

- **Independent Verification and Validation**: Verification and validation performed by an organization that is technically, managerially, and financially independent.

Early Involvement

- Software assurance activities begin during the concept/initiation phase of the development process and proceed throughout maintenance.
- The goal is to build safety, reliability, and quality into the software product.
- Software assurance partners with engineering, early in the project, to build the highest quality software.
- Software assurance personnel must assure that the right requirements are in place from the beginning.
- Don't wait until CDR or even PDR to add software assurance to a given project.

We need a plan

- Document your software assurance activities in a Software Assurance Plan.
- Software Assurance Plans are typically due at the System Requirements Review (SRR).
- Most organizations have a template for Software Assurance Plans or they follow IEEE-STD-730-2002.
- Don't just write the plan and let it collect dust on the shelf or in your hard drive. Revisit the Software Assurance Plan as the project progresses.
- Get buy-in from the project and your organization on your approach. This is your communication tool.

Tailor Software Assurance Efforts

- Software assurance is a balancing activity that must be tailored as appropriate for each project
- Software assurance activities should be tailored based on risk
- Determine software classification and safety criticality upfront to help scope efforts
- Software assurance engineers must make trade-offs, based on their experience and the software risks on a project.

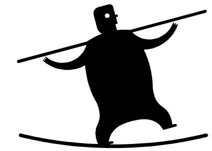
Not a checklist!

- Software assurance is not only about using a checklist to perform process and product audits

- Checklists are important because they provide objective evidence by which a process or product may be evaluated BUT…

- If our main focus becomes a checklist, then we are missing the mark

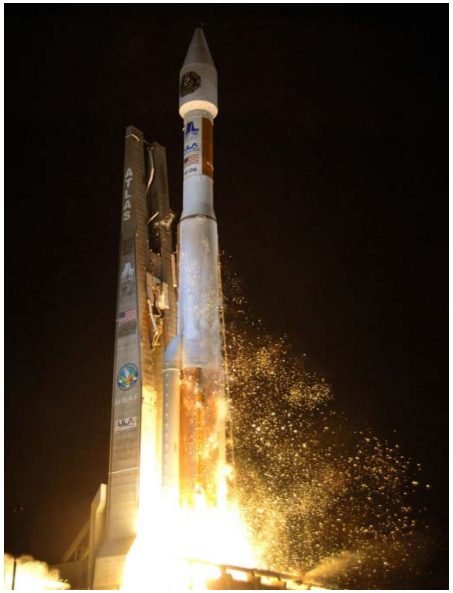
Focus on Reducing Risk



- Identify, address and eliminate software risk items before they become threats to success or major sources of rework.
- Software assurance activities should be driven by risk (includes safety, reliability and quality).
- Need a communication path to management for Software Assurance engineers to raise issues and concerns that they have on a project
- Get involved in the project’s formal risk management process.
- Software assurance engineers can rely on Safety & Mission Assurance technical authority, if necessary.

Good communication is key

- A good software assurance engineer must possess good communication skills.
- Essentially, we are evaluating software products and processes; must be as diplomatic as possible.
- Must win the trust and respect of the software system developers



- In order to make a difference, your ideas and suggestions must be accepted by the project team.
- Building relationships with the project team members is the best way to ensure that your voice is heard.

Maintaining Independence

- Independence implies performing product quality evaluations by an outside organization (NASA governance model)

- Need for independence arises because developer may have a biased expectation of what a product should be; could miss anomalies; could fail to perform certain checks

- Notion of independence is applied to reduce errors resulting from extensive familiarity with the product being evaluated

- Important to maintain independence while still being a team player

It's not ALL about Quality

- Safety and Reliability are critical aspects of software assurance as well as Quality
- A system safety process usually contains the following elements:
  - Planning
  - Identifying and characterizing the hazards
  - Assessing and prioritizing risks and making risk decisions
  - Reducing risks to acceptable levels through valid controls
  - Verifying that risks are reduced
  - Tracking hazards, risks, and problems
- Software safety efforts should include each element of a good system safety process
- Qualitative reliability analysis should also include software as part of the system (e.g. Fault Tree Analysis, Failure Modes and Effects Analysis)

Identifying safety-critical software

- Don’t be afraid to declare software as safety-critical ☺

- How do I know if software is safety-critical?
  - Resides in a safety-critical system (as determined by a hazard analysis) AND at least one of the following apply:
    - 1) Causes or contributes to a hazard.
    - 2) Provides control or mitigation for hazards.
    - 3) Controls safety-critical functions.
    - 4) Processes safety-critical commands or data.
    - 5) Detects and reports, or takes corrective action, if the system reaches a specific hazardous state.
    - 6) Mitigates damage if a hazard occurs.
    - 7) Resides on the same system (processor) as safety-critical software.
  - Processes data or analyzes trends that lead directly to safety decisions.
  - Provides full or partial verification or validation of safety-critical systems, including hardware or software subsystems.

- Software Safety efforts should be based on risk. (See the NASA Software Safety Guidebook, NASA-GB-8719.13 for details)

- Software Assurance engineers must maintain close communication with System Safety engineers in order to make this determination. They should be the liaison between System Safety and the Software Engineers.

Ariane 5 Explosion

Software Safety generic requirements

NPR 7150.2, NASA Software Engineering Requirements, section 2.2.12:

- Safety-critical software is initialized, at first start and at restarts, to a known safe state.
- Safety-critical software safely transitions between all predefined known states.
- Termination performed by software of safety critical functions is performed to a known safe state.
- Operator overrides of safety-critical software functions require at least two independent actions by an operator.
- Operator overrides of safety-critical software functions require at least two independent actions by an operator.
- Safety-critical software rejects commands received out of sequence, when execution of those commands out of sequence can cause a hazard.
- Safety-critical software detects inadvertent memory modification and recovers to a known safe state.

Software Safety generic requirements

- Safety-critical software performs integrity checks on inputs and outputs to/from the software system.
- Safety-critical software performs prerequisite checks prior to the execution of safety-critical software commands.
- No single software event or action is allowed to initiate an identified hazard.
- Safety-critical software responds to an off nominal condition within the time needed to prevent a hazardous event.
- Software provides error handling of safety-critical functions.
- Safety-critical software has the capability to place the system into a safe state.
- Safety-critical elements (requirements, design elements, code components, and interfaces) are uniquely identified as safety-critical.
- Incorporate requirements in the coding methods, standards, and/or criteria to clearly identify safety-critical code and data within source code comments.

References

- NASA Software Assurance Standard, NASA-STD-8739
- NASA Software Safety Standard, NASA-STD-8719.13B
- NASA Software Safety Guidebook, NASA-GB-8719.13
- NASA Software Engineering Requirements, NPR 7150.2