DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

SAND2025-06890R

Sandia National Laboratories

Software Quality Assurance Plan

X-Ray ToolKit (XTK) & Related Software

May 2025

U.S. DEPARTMENT OF ENERGY

National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

CHANGE HISTORY

Version	Date	Originator(s)	Description
1.0	05/25/2025	Bryce Eldridge (06617)	Initial Release

APPROVALS

Department Manager

Name (Org #) _____ Date _____

Project Lead

Name (Org #) _____ Date _____

CONTENTS

LIST OF TABLES

1. INTRODUCTION

This Software Quality Assurance Plan (SQAP) sets forth the process, methods, standards, and procedures that will be used to perform the Software Quality Assurance functions for the X-Ray ToolKit (XTK) program and related software.

This document aligns with the following policies and orders:

- Sandia policy IT008
- SSQAP policy SS-R89727
- DOE O 414.1D, Quality Assurance for Software Development.

2. SOFTWARE RISK/PRACTICE LEVEL DETERMINATION

2.1. Identify software intended operating environment, function, and use.

The X-Ray ToolKit (XTK) software is used on Windows and Android devices to acquire and process radiographs (x-ray images) for security and Explosive Ordnance Disposal (EOD) operations.
XTK provides a wide variety of image processing tools for enhancing, combining, and measuring using radiographs.

XTK also provides a common interface for integration of commercial x-ray acquisition equipment.

More information is available at https://xraytoolkit.sandia.gov

ID #	Operating Environment	Description
OE-1	Host Computer System	The host computer system for the XTK software can be provided by a variety of sources, including the x-ray hardware vendor or the end user directly. XTK software requires Windows 8, 10, or 11 to run.
OE-2	X-Ray Acquisition Software	If used for acquisition, XTK can interface with a variety of commercial x-ray imaging software systems to capture images in a consistent way. These commercial products are responsible for control of the hardware in a safe way and for adhering to the requirements in the XTK interface.
OE-3	X-Ray Acquisition Hardware	If used for acquisition, a commercial x-ray hardware package is used in conjunction with the software. This package usually consists of a detector panel or scanner, a portable x-ray source, and any required cables or interface boxes. **These are commercial products and any quality assurance associated with this equipment is outside the scope of the XTK software.**

Table 1 - Software Operating Environment

2.2. Consequence of Software Failure

	Risk Categories/Consequences				
Application Area	**Environment, Safety & Health**	**Performance**	**Public & Customer Confidence**	**Safeguards & Security**	**Financial**
System Operation	C0 - None	C2 – A system failure could impact the availability of needed information	C2 - Loss of public/customer confidence	C0 - None	C1 – A system failure could cause lost time or training/exercise disruption
Consequence of Failure Level (Max)	**C2**				

Table 2 - Defined Risk Categories and Consequences

Rationale Statements for Consequence Level(s):

Environment, Safety & Health: There are no identifiable environment, safety, & health risks or consequences to a software failure. All ES&H risks are controlled with external procedures not specific to this software (i.e. any RGD or x-ray acquisition hardware is subject to separate controls related to RGDs and x-ray imaging regardless of what software is used).

Performance: The software is used primarily for image processing and analysis to inform decisions made by security or EOD teams. A failure of the software to perform it's functions correctly could result in a loss of important diagnostic information used in making decisions.

Public & Customer Confidence: The software is widely used throughout the world, and a failure could cause significant loss of public or customer confidence in the system.

Safeguards & Security: The software is not responsible for controlling access to any information or property, so there are no safeguard & security consequences to a software failure. All IT safeguards are applied and managed for the host computer system that runs the software.

Financial: The software is covered by an End User License Agreement provided by Sandia Legal that mitigates any risk of financial consequences (i.e. lawsuits) associated with a software failure. However, a software failure could result in training or exercise disruption and lost time or delays.

2.3. Likelihood of Failure

Likelihood of Failure Level: **Low (*L1)***

Rationale:

- The software's requirements are relatively stable which decreases the likelihood of failure. General software requirements are well understood and are not expected to change (low).
- The wide variety of host computing systems and commercial x-ray interfaces used by the software leads to slightly higher likelihood of unexpected behavior in isolated conditions (low).
- The project's organizational complexity is low which decreases likelihood of failure. (low)
- The project's testing and staged release process decrease the likelihood of a failure in a production environment (low).
- Overall, the likelihood of software failure is low (L1) due to the general program stability and existing testing and release processes.

2.4. Project Risk Level and Associated Practice Level

Project Risk Level Determination: **Low (L)**
Project Practice Level Determination: **(P1)**

Rationale:

The highest risk paring is (C2,L1) which corresponds to a Risk Level of RL=L.

The associated Practice Level is P1.

The Risk Level and Practice Level are primarily driven by the potential for performance failure or loss of public & customer confidence.

2.5. Practice Level Adjustment

The project implements practices at the P1 level, with additions from the P2 level where deemed valuable.

3. SAFETY SOFTWARE DETERMINATION

To determine if the software is "safety software", the following six questions are used:

1. Could the inadvertent response of the software to stimuli result in an accident?
    a. No, mechanical interlocks and other safety procedures are in place to prevent any accident being caused by the software. The software is used primarily for image analysis and as such does not directly interact with any hardware that does not provide separate safety controls.
2. Could the failure of the software to respond when required result in an accident?
    a. No
3. Could the software provide a response out-of-sequence that would result in an accident?
    a. No
4. Could the software provide a response that, when used in combination with other responses, results in an accident?
    a. No
5. Is the software intended to be used to mitigate the result of an accident?
    a. No
6. Is the software intended to be used to recover from the results of an accident?
    a. No

The X-Ray ToolKit software is determined to not be safety software.

4. PRACTICE ACTIVITIES

4.1. Project Planning [PP]

1. **Plan for project resources, needed knowledge and skills, stakeholder involvement, and data management.**
    - **Resources Needed**: Development team (software engineers, QA testers), project manager, UX/UI designer.
    - **Skills Required**: Proficiency in GitLab, continuous integration (CI) practices, software testing, and user experience design.
    - **Stakeholders**: Project team, management, end-users, and customer support.
2. **Review plans that affect the project.**
    - **Existing Plans**: Review project timelines, resource allocation plans, and risk management strategies.
3. **Identify project risks.**
    - **Potential Risks**: Delays in development, integration issues, user adoption challenges, and potential security vulnerabilities.
4. **Reconcile work and resource levels.**
    - **Alignment Strategy**: Regular team meetings to assess workload and resource allocation; use GitLab for tracking resource assignments.

4.2. Risk Management [RK]

1. **Risk Identification**
    - The project team will identify potential risks associated with the software development process, including risks related to functionality, performance, security, and safety. This will involve brainstorming sessions, review of historical data from similar projects, and consultation with Subject Matter Experts (SMEs).
2. **Risk Assessment**
    - Each identified risk will be assessed based on its likelihood of occurrence and potential impact on the project. Risks will be categorized into low, moderate, and high-risk levels to prioritize management efforts.
3. **Risk Mitigation Strategies**
    - For high-risk items identified during the assessment phase, the project team will implement more extensive testing protocols. This may include:

- **Increased Test Coverage**: Ensuring that high-risk components undergo rigorous testing, including unit tests, integration tests, and system tests, to validate their functionality and performance under various conditions.
- **Stress Testing**: Conducting stress tests to evaluate how high-risk components perform under extreme conditions or loads, ensuring that they can handle unexpected scenarios without failure.
- **User Acceptance Testing (UAT)**: Engaging end-users and SMEs in UAT specifically focused on high-risk features to gather feedback and identify any issues before full deployment.

4. **Risk Monitoring**
    - The project team will continuously monitor identified risks throughout the software development lifecycle. Risk reviews will be incorporated as part of the internal technical review process and planning for new features or changes.

4.3. Requirements Management [RM]

1. **Maintain bidirectional traceability of requirements.**
    - **Documentation**: Use GitLab to link requirements to specific issues and commits.
2. **Identify inconsistencies between project work and requirements.**
    - **Process**: Conduct internal technical reviews to determine if project work meets requirements.

4.4. Requirements Development [RD]

1. **Allocate product component requirements.**
    - **Allocation Method**: Assign requirements to team members via GitLab issues.
2. **Identify interface requirements.**
    - **Interfaces**: Work with commercial or integration and development partners to understand and develop interface requirements.
3. **Establish operational concepts and scenarios, and a definition of required functionality.**
    - **Method**: Use cases are collected through customer interaction, support help desk software, feedback mechanisms in the distribution system (feature wishlist), or sponsor requests. Desired requirements are documented in GitLab with a specific release in mind or a "future" tag.
4. **Analyze requirements to achieve balance**

- **Balancing Method**: Prioritize requirements based on user feedback and project goals, as well as currently available development resources.

5. **Validate requirements.**
    - **Validation Process**: Review requirements with stakeholders and conduct usability testing.

4.5. Technical Solution [TS]

1. **Develop alternative solutions and selection criteria.**
    - **Alternatives**: Internal technical discussion of implementation alternatives.
    - **Selection Criteria**: Usability, community support, potential impact, ease of integration, and low risk of breaking existing functionality.
2. **Select product component solutions.**
    - **Evaluation Method**: Conduct proof-of-concept implementations for shortlisted solutions. Use of RFC system for formal customer feedback on proposed changes or designs.
3. **Prototype integration**
    - **Method:** Use prototyping and wireframe tools to design changes to the software and evaluate how they will fit in or modify existing behavior or user interface systems.

4.6. Verification [VE]

1. **Conduct internal technical review:** Internal technical reviews are held for major release candidates or testing releases. Release notes are examined in detail to ensure the software implementation meets the requirements.
2. **Comprehensive unit testing:** A full unit test suite is executed automatically with every GitLab commit to verify that all features are correctly implemented.

4.7. Validation [VA]

1. **Validation is conducted for high-risk software components of XTK.**
    - **Grid-Aim:** Known images with known solutions are used as part of a regression testing package to ensure that the correct solutions are generated using a wide variety of input conditions. No software is distributed that does not pass this regression testing.
    - **Products**: The final build of X-Ray ToolKit for beta testing.
2. **Establish the validation environment, procedures, and criteria.**
    - **Environment**: An automated build/staging environment is used for regression testing.

4.8. Deployment [DE]

1. **Assemble product components.**
    - **Components**: Compile code, package and sign binaries, and prepare documentation. All of this is an automated process that happens using the GitLab CI/CD platform.
2. **Package and deliver the product or product component.**
    - **Packaging Method**: Use versioning and tagging in GitLab for releases.
    - **Release Process**:
        1. **Beta Releases**: Made available to a select group of users for beta testing and feedback. This package can change frequently as needed.
        2. **Early Access Releases**: Generally available to all registered users, with the caveat that the released package may change more often.
        3. **Stable Release**: Early access package is promoted to stable release after being used for a defined period of time or number of downloads with no further reported issues. Stable releases are made no more often than once per year, and no changes are made except in the rare case of some critical issue being discovered.
3. **Communication**
    - **Release Announcement:** Releases are announced via the XTK website, by sending an email to all registered users, and by posting in the relevant DOE/NNSA Teams channels.

4. **Package and Dependency Scanning**
    - **Method:** Dependency packages are scanned for cybersecurity issues and known vulnerabilities both by the development tool (Visual Studio) and also as part of the CNARS approval process for installation on Sandia classified networks. Any issues are resolved before releasing software to the end user community.
5. **Publication Review**
    - **Method:** Any updated documentation or training materials are submitted through the Information Release (IR) process before distribution.

4.9. Life Cycle Support [LS]

1. **Develop product support documentation.**
    - **Support Documentation**: All training materials and documentation (tutorials, quick start guides, videos, etc.) are updated as needed for each major stable release.
    - **Integration Package**: The sample plugin integration package, which includes automatically generated API documentation and sample code, is updated for each major stable release and as needed for beta testing.

4.10. Configuration Management [CM]

1. **Configuration management system**
    - **Records**: All configuration items, source code, software product artifacts are managed using the GitLab source control system.
2. **Track change requests**
    - **Method**: Change requests are documented and tracked from initial creation to implementation and release using the GitLab issue management system.
3. **Versioning**
    - **Method:** Releases are numbered using the industry standard semantic versioning system (MAJOR.MINOR.PATCH), details available here: https://semver.org/

4.11. Product Integration [PI]

1. **Manage commercial x-ray integration interfaces.**
    - **Management Method**: Third party commercial vendors and select end user representatives are granted access to the beta testing releases to evaluate their product integrations for any issues. Gitlab is used to track and resolve these issues before each major release.

4.12. Stakeholder Involvement [SI]

1. **Manage stakeholder involvement.**
    - **Regular Communication**: Regular updates and feedback sessions with stakeholders as needed.
    - **Request for Comments**: For additional formality in some programs, a "request for comments" (RFC) system is used to solicit stakeholder feedback and buy in for important changes.
    - **Quarterly Status Reports:** Status reports are sent quarterly to stakeholders that include information about current releases, planned work, current projects and sponsors, software download and activity summaries, and a support ticket summary from the osTicket help desk system.

4.13. Training [TR]

1. **Deliver training.**
    - **Training Provided**: Conduct onboarding sessions for new team members on GitLab and CI practices.
2. **Assess training effectiveness.**
    - **Assessment Method**: Gather feedback from participants and track performance improvements.

4.14. Problem Reporting and Corrective Actions [PR]

1. **Analyze issues.**
    - **Issue Analysis**: Use osTicket help desk software to categorize and prioritize reported issues.
2. **Take corrective action.**
    - **Corrective Actions**: Assign issues to team members via Gitlab for resolution based on priority.
3. **Manage corrective action.**
    - **Tracking Method**: Use GitLab to track the status of corrective actions. Update release notes for published versions.