System Verification and Validation Plan for Software Engineering

Team #10, Five of a Kind
Omar Abdelhamid
Daniel Maurer
Andrew Bovbel
Olivia Reich
Khalid Farag

October 27, 2025

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

[The intention of the VnV plan is to increase confidence in the software. However, this does not mean listing every verification and validation technique that has ever been devised. The VnV plan should also be a **feasible** plan. Execution of the plan should be possible with the time and team available. If the full plan cannot be completed during the time available, it can either be modified to "fake it", or a better solution is to add a section describing what work has been completed and what work is still planned for the future. —SS]

[The VnV plan is typically started after the requirements stage, but before the design stage. This means that the sections related to unit testing cannot initially be completed. The sections will be filled in after the design stage is complete. the final version of the VnV plan should have all sections filled in. —SS]

Contents

List of Tables

[Remove this section if it isn't needed —SS]

List of Figures

[Remove this section if it isn't needed —SS]

1 Symbols, Abbreviations, and Acronyms

symbol	description
T	Test

[symbols, abbreviations, or acronyms — you can simply reference the SRS (Author, 2019) tables, if appropriate —SS]

[Remove this section if it isn't needed —SS]

This document outlines the Verification and Validation (V&V) Plan for **VoxelMag Designer**, developed by Team 10 — *Five of a Kind*. It describes how the system will be tested to ensure that all functional, non-functional, and safety requirements defined in the Software Requirements Specification (SRS) are satisfied. The plan establishes the strategies, team roles, and procedures that will be used to verify correctness, performance, and usability of the software.

The document is organized into two main sections. The first section, **General Information**, provides background context including a summary of the system, testing objectives, project challenge level, and relevant supporting documents. The second section, **Plan**, outlines the structure of the V&V process, identifies team responsibilities, and describes the approach for verifying the SRS through structured reviews, stakeholder feedback, and continuous validation throughout development.

2 General Information

2.1 Summary

The software system being tested is called **VoxelMag Designer**, developed by Team 10 — *Five of a Kind*. VoxelMag Designer is a desktop-based CAD enhancement tool that allows researchers and engineers to assign magnetic properties to individual voxels within a 3D model. It is designed to streamline the process of magnetization planning for multi-material 3D printing workflows within a research laboratory environment.

Currently, magnetized objects are modeled voxel-by-voxel in COMSOL, requiring researchers to manually assign magnetization directions for each voxel and rebuild the model from scratch if an error occurs. VoxelMag Designer removes this bottleneck by enabling users to import pre-built CAD models (e.g., from AutoCAD), automatically slice them into voxel layers, and interactively select single or multiple voxels to assign or modify magnetization directions. The software also provides per-layer visualization, undo/redo functionality, and validation mechanisms to ensure complete and accurate voxel data before export.

The final output is a **Java-readable metadata file** that contains per-voxel magnetization information, which integrates directly with the custom 3D printer pipeline used in the lab. By automating voxel processing and

enabling efficient magnetization editing, VoxelMag Designer dramatically reduces the time and effort required to prepare print-ready models, ensuring a faster, more reliable, and user-friendly workflow for advanced magnetized 3D printing research.

2.2 Objectives

The primary objective of the verification and validation (V&V) process for **VoxelMag Designer** is to ensure that the system functions correctly, reliably, and efficiently according to the requirements defined in the Software Requirements Specification (SRS). The V&V activities will confirm that each system component performs as intended, meets stakeholder expectations, and aligns with the project's functional and non-functional goals.

Specifically, the objectives of V&V are to:

- **Correctness:** Verify that all functional requirements are implemented as specified, including accurate import, voxelization, visualization, editing, and export of CAD models.
- **Reliability:** Ensure the system maintains consistent behaviour under normal and high-load conditions, preventing crashes, data corruption, or voxel property loss.
- **Usability:** Validate that the interface allows users to intuitively view, select, and edit voxels across layers, and that feedback mechanisms (such as undo/redo, autosave, and validation prompts) function as expected.
- **Performance:** Confirm that voxel rendering, editing, and export operations meet timing thresholds and remain responsive on standard laboratory hardware.
- **Safety and Security:** Validate that safeguards identified in the Hazard Analysis (e.g., input file validation, export completeness, autosave recovery) are correctly implemented and meet the defined safety requirements.
- **Traceability:** Demonstrate that all SRS requirements are testable, measurable, and traceable through verification procedures and testing artifacts.

Overall, the V&V process aims to provide stakeholders with confidence that **VoxelMag Designer** achieves its intended purpose as a robust, user-friendly, and trustworthy software solution for voxel-based magnetization design.

2.3 Challenge Level and Extras

The challenge level for **VoxelMag Designer** is classified as **advanced**. The system integrates multiple complex subsystems, including 3D model import and voxelization, real-time voxel visualization and editing, magnetization assignment, and structured export to a custom printer pipeline. Each subsystem involves substantial technical and architectural considerations, such as handling large voxel datasets, maintaining UI synchronization, and ensuring data integrity across modules. These factors collectively establish the project as an advanced-level software system.

The extras included in this project are:

- **User Manual:** A comprehensive guide explaining how to operate the software, including detailed instructions, annotated screenshots, and workflow examples. The manual emphasizes ease of use and clarity to enable independent operation by research personnel.
- **Usability Report:** A formal documentation of the usability testing process, summarizing user feedback, identified issues, and recommendations for improving interface design and user experience. The report also outlines the methodology used during testing, including participant demographics and task-based evaluations.

These extras complement the technical objectives of the project by ensuring that the final product is not only functionally correct but also accessible, intuitive, and well-documented for its target users.

2.4 Relevant Documentation

Author (2019)

The documentation that supports the verification and validation (V&V) process for **VoxelMag Designer** includes the Software Requirements Specification (SRS), Hazard Analysis (HA), Development Plan (DP), and Problem

Statement and Goals. Each document serves a distinct purpose in guiding and supporting the testing strategy.

The **Software Requirements Specification (SRS)** defines all functional and non-functional requirements for the system. It provides the foundation for test case design, ensuring that every requirement can be verified through measurable criteria and traced throughout the V&V process.

The **Hazard Analysis (HA)** identifies potential system hazards, their causes, and mitigation strategies. It establishes the safety requirements (SCRs) that must be verified to ensure the system operates safely and consistently under expected laboratory conditions.

The **Development Plan (DP)** outlines the project's workflow, coding standards, continuous integration setup, and proof-of-concept demonstration plan. It informs the structure and scheduling of testing activities by defining the methods and tools used during development.

The **Problem Statement and Goals** document provides the original motivation, scope, and intended outcomes of the project. It ensures that the V&V process remains aligned with the project's high-level objectives and stakeholder expectations.

Finally, the project's **GitHub Repository** serves as the central hub for all implementation artifacts, testing scripts, and documentation updates.
https://github.com/OmarHassanAdelhamid/Five-of-a-Kind-capstone-project

Together, these documents ensure that all verification and validation activities are traceable, comprehensive, and consistent with the system's intended functionality and safety goals.

3 Plan

The **Plan** section outlines the structure and approach of the verification and validation (V&V) process for **VoxelMag Designer**. It describes the composition of the V&V team, their assigned roles, and responsibilities across different testing domains, ensuring that all system components are thoroughly verified and validated. This section also details the planned approach for verifying the Software Requirements Specification (SRS), including the review strategy, feedback mechanisms, and formal verification methods that will be employed to confirm that all requirements are correct, testable, and traceable throughout development.

The verification and validation (V&V) efforts for **VoxelMag Designer**

will be conducted by **Team 10 — *Five of a Kind***. Each member plays a specific role to ensure that all functional, non-functional, and safety requirements are properly verified and validated. The distribution of responsibilities ensures comprehensive coverage of testing, documentation, and review activities across all project domains.

Name	Role	Responsibilities
Omar Abdelhamid	Lead Tester / Integration Coordinator	Oversees the overall testing strategy and ensures all verification activities align with the SRS. Manages issue tracking for validation tasks, coordinates integration testing, and co-leads structured review meetings with the supervisor.
Daniel Maurer	Functional Requirements Tester / Supervisor Liaison	Focuses on verifying functional aspects such as import, voxelization, and export. Coordinates communication with the supervisor, organizes feedback sessions, and ensures review comments are integrated into revisions.
Andrew Bovbel	Non-Functional Requirements Tester	Responsible for evaluating system performance, responsiveness, and reliability. Conducts stress and performance tests to ensure compliance with non-functional requirements (e.g., rendering latency, autosave frequency).
Olivia Reich	Usability and Documentation Tester	Leads usability testing, gathers user feedback, and evaluates interface intuitiveness. Reviews and refines the User Manual to ensure clarity and alignment with system behavior.
Khalid Farag	Safety and Validation Tester	Validates safety and security requirements derived from the Hazard Analysis. Maintains traceability between identified hazards, safety requirements (SCRs), and corresponding test cases.

Table 1: Team roles and responsibilities for verification and validation.

This structure ensures that all verification and validation efforts—ranging from requirement coverage and performance testing to usability evaluation and safety assurance—are systematically planned, executed, and reviewed in

collaboration with the supervisor.

3.1 SRS Verification

Verification of the Software Requirements Specification (SRS) for **VoxelMag Designer** will follow a structured, multi-stage process to ensure that all requirements are correct, complete, testable, and traceable. This process includes internal reviews, supervisor meetings, peer review feedback, and continuous verification. Each activity is designed to improve requirement quality and maintain traceability between the SRS and subsequent project deliverables.

3.1.1 Internal Review and Checklist Verification

The first stage of SRS verification will involve a detailed internal review conducted by all team members. Each member will focus on a particular category of requirements:

- **Omar Abdelhamid** and **Daniel Maurer** will verify that all functional requirements are clear, consistent, and feasible to implement.
- **Andrew Bovbel** will confirm that non-functional requirements, such as performance and reliability metrics, are measurable and verifiable.
- **Khalid Farag** will validate that safety and security requirements (SCRs) properly trace to hazards identified in the Hazard Analysis.
- **Olivia Reich** will review documentation-related and usability-focused requirements for clarity and completeness.

A collaborative **SRS Verification Checklist** will guide the review. The checklist will include the following key items:

1. Each requirement is uniquely identified and unambiguous.
2. Functional requirements are measurable and testable.
3. Non-functional requirements specify clear performance criteria.
4. Safety and security requirements trace back to identified hazards.

5. Each requirement includes a defined verification or validation method.

6. Requirements are free of contradictions or redundancies.

All findings and issues identified during the review will be documented as GitHub issues and assigned to the appropriate team member for correction prior to the next SRS revision.

3.1.2 Supervisor Review

Following internal review, the team will conduct a structured SRS review meeting with the project supervisor. **Omar Abdelhamid** and **Daniel Maurer** will organize and co-lead the meeting, preparing a summary document highlighting high-risk and complex requirements that need confirmation (e.g., voxel visualization performance, data export structure, and UI synchronization accuracy).

The review meeting will include:

- A walkthrough of the SRS to verify alignment between requirements and project scope.
- Discussion of any ambiguous, conflicting, or technically challenging requirements.
- Presentation of requirements identified as high-risk and their proposed validation methods.
- A feedback session for the supervisor to provide clarifications or suggest modifications.

Supervisor feedback will be recorded within the GitHub issue tracker. Each comment will be linked to the corresponding requirement ID and tracked until resolution. Follow-up actions will be reviewed in subsequent supervisor meetings.

3.1.3 Peer Review and Class Feedback

In addition to the supervisor review, the SRS will undergo peer evaluation during the class-wide review process. The primary reviewer team will assess the document's clarity, consistency, and completeness. Their feedback will be formally logged and categorized according to:

- Readability and clarity of requirement descriptions.
- Consistency of terminology and format across sections.
- Logical organization of requirements by subsystem or feature.
- Appropriateness of fit criteria and testability.

All peer feedback will be integrated systematically into the SRS through a traceability matrix, ensuring that each comment maps to a specific requirement and its revision status.

3.1.4 Continuous Verification and Traceability Maintenance

Throughout development, the team will maintain continuous verification of the SRS using GitHub issues, milestones, and project boards. Each requirement will be linked to corresponding design elements, test cases, and validation results. Regular bi-weekly internal reviews will verify that any new or modified requirements remain traceable and verifiable.

This iterative approach ensures that the SRS remains accurate, actionable, and aligned with the evolving implementation. Any changes introduced after supervisor or peer feedback will trigger updates to both the traceability matrix and the V&V Plan to maintain consistency across documents.

Summary: This structured, multi-level verification process guarantees that the SRS for **VoxelMag Designer** is validated both internally and externally. Through continuous review, stakeholder feedback, and traceability enforcement, the team ensures that all requirements are precise, measurable, and fully verifiable before system testing begins.

3.2 Design Verification

[Plans for design verification —SS]
[The review will include reviews by your classmates —SS]
[Create a checklists? —SS]

3.3 Verification and Validation Plan Verification

[The verification and validation plan is an artifact that should also be verified. Techniques for this include review and mutation testing. —SS]

[The review will include reviews by your classmates —SS]

[Create a checklists? —SS]

3.4 Implementation Verification

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

[The final class presentation in CAS 741 could be used as a code walkthrough. There is also a possibility of using the final presentation (in CAS741) for a partial usability survey. —SS]

3.5 Automated Testing and Verification Tools

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[If you have already done this in the development plan, you can point to that document. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

3.6 Software Validation

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

[You might want to use review sessions with the stakeholder to check that the requirements document captures the right requirements. Maybe task based inspection? —SS]

[For those capstone teams with an external supervisor, the Rev 0 demo should be used as an opportunity to validate the requirements. You should

plan on demonstrating your project to your supervisor shortly after the scheduled Rev 0 demo. The feedback from your supervisor will be very useful for improving your project. —SS]

[For teams without an external supervisor, user testing can serve the same purpose as a Rev 0 demo for the supervisor. —SS]

[This section might reference back to the SRS verification section. —SS]

4 System Tests

[There should be text between all headings, even if it is just a roadmap of the contents of the subsections. —SS]

4.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. —SS]

4.1.1 Area of Testing1

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Title for Test

1. test-id1

   Control: Manual versus Automatic

   Initial State:

   Input:

   Output: [The expected result for the given inputs. Output is not how you are going to return the results of the test. The output is the expected result. —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

   Control: Manual versus Automatic

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

4.1.2 Area of Testing2

...

4.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. Not all projects will necessarily have nonfunctional requirements related to accuracy. —SS]

[For some nonfunctional tests, you won't be setting a target threshold for passing the test, but rather describing the experiment you will do to measure the quality for different inputs. For instance, you could measure speed versus the problem size. The output of the test isn't pass/fail, but rather a summary table or graph. —SS]

[Tests related to usability could include conducting a usability test and survey. The survey will be in the Appendix. —SS]

[Static tests, review, inspections, and walkthroughs, will not follow the format for the tests given below. —SS]

[If you introduce static tests in your plan, you need to provide details. How will they be done? In cases like code (or document) walkthroughs, who will be involved? Be specific. —SS]

4.2.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

4.2.2 Area of Testing2

...

4.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

5 Unit Test Description

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, you code needs to be well-documented, with meaningful names for all of the tests. —SS]

5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

5.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

Author Author. System requirements specification. https://github.com/..., 2019.

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]

Appendix — Reflection

[This section is not required for CAS 741 —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage, Valgrind etc. You should look to identify at least one item for each team member.

4. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?