System Verification and Validation Plan for FBP CT Image Reconstruction

Qianlin Chen

February 14, 2025

roman

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

1 Symbols, Abbreviations, and Acronyms **4**

2 General Information **1**
- 2.1 Summary 1
- 2.2 Objectives 1
- 2.3 Challenge Level and Extras 1
- 2.4 Relevant Documentation 2

3 Plan **2**
- 3.1 Verification and Validation Team 2
- 3.2 SRS Verification Plan 3
- 3.3 Design Verification Plan 4
- 3.4 Verification and Validation Plan Verification Plan 4
- 3.5 Implementation Verification Plan 4
- 3.6 Automated Testing and Verification Tools 5
- 3.7 Software Validation Plan 5

4 System Tests **6**
- 4.1 Tests for Functional Requirements 6
  - 4.1.1 Input Tests 6
  - 4.1.2 8
- 4.2 Tests for Nonfunctional Requirements 8
  - 4.2.1 Area of Testing1 8
  - 4.2.2 Area of Testing2 9
- 4.3 Traceability Between Test Cases and Requirements 9

5 Unit Test Description **9**
- 5.1 Unit Testing Scope 10
- 5.2 Tests for Functional Requirements 10
  - 5.2.1 Module 1 10
  - 5.2.2 Module 2 11
- 5.3 Tests for Nonfunctional Requirements 11
  - 5.3.1 Module ? 11
  - 5.3.2 Module ? 12
- 5.4 Traceability Between Test Cases and Modules 12

6 Appendix **13**
6.1 Symbolic Parameters 13
6.2 Usability Survey Questions? 13

List of Tables

1 Verification and validation team 3

[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 ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

2 General Information

2.1 Summary

This document reviews the verification and validation plan for Filtered Back Projection (FBP) CT Image Reconstruction. The FBP method is used to reconstruct cross-sectional images from projection data collected in CT scans. The accuracy and performance of the reconstruction process are evaluated based on predefined validation criteria. The system's usability, maintainability, and reliability are also assessed using user inputs and experimental results.

2.2 Objectives

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: "build confidence in the software correctness," "demonstrate adequate usability." etc. You won't list all of the qualities, just those that are most important. —SS]

[You should also list the objectives that are out of scope. You don't have the resources to do everything, so what will you be leaving out. For instance, if you are not going to verify the quality of usability, state this. It is also worthwhile to justify why the objectives are left out. —SS]

[The objectives are important because they highlight that you are aware of limitations in your resources for verification and validation. You can't do everything, so what are you going to prioritize? As an example, if your system depends on an external library, you can explicitly state that you will assume that external library has already been verified by its implementation team. —SS]

2.3 Challenge Level and Extras

[State the challenge level (advanced, general, basic) for your project. Your challenge level should exactly match what is included in your problem statement. This should be the challenge level agreed on between you and the course instructor. You can use a pull request to update your challenge level (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

[Summarize the extras (if any) that were tackled by this project. Extras can include usability testing, code walkthroughs, user documentation, formal proof, GenderMag personas, Design Thinking, etc. Extras should have already been approved by the course instructor as included in your problem statement. You can use a pull request to update your extras (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

2.4 Relevant Documentation

[Reference relevant documentation. This will definitely include your SRS and your other project documents (design documents, like MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. You can create BibTeX entries for your documents and within those entries include a hyperlink to the documents. —SS]

Author (2019)

[Don't just list the other documents. You should explain why they are relevant and how they relate to your VnV efforts. —SS]

3 Plan

This section describes the verification and validation plan for the FBP CT Image Reconstruction.The planning starts with the verification and validation team in Section 3.1, followed by the SRS verification plan (Section 3.2), design verification plan (Section 3.3), verification and validation verification plan (Section 3.4), implementation verification plan (Section 3.5), Automated testing and verification tools (Section 3.6), and Software validation plan (Section 3.7).

3.1 Verification and Validation Team

Name	Document	Role	Description
Dr. Spencer Smith	All	Instructor/ Reviewer	Review the documents, design and documentation style.
Qianlin Chen	All	Author	Create and manage all documentation, develop the VnV plan, conduct VnV testing, and verify the implementation.
Xunzhou Ye	All	Domain Expert Reviewer	Review all the documents.

Table 1: Verification and validation team

3.2 SRS Verification Plan

The SRS document for FBP CT Image Reconstructionwill be verified using a structured review process. The verification plan consists of the following steps:

Initial Review

Assigned reviewers, instructor (Dr. Smith) and domain expert (Xunzhou). will conduct a manual review of the SRS document. In this step, a structured SRS Checklist will be used to systematically evaluate key aspects such as requirement completeness, consistency, feasibility, and traceability.

Issue Tracking

eviewers will provide feedback by documenting identified issues in an issue tracker (GitHub Issues). Each issue will be assigned to the document owner for later revision.

Revision

The document owner will address reported issues by making necessary modifications to the SRS. Any unresolved or disputed concerns will be discussed among the reviewers.

Final Documentation
After revisions, a final review will be conducted to confirm that all identified issues have been adequately addressed.

3.3 Design Verification Plan

The design documentation, including the Module Guide (MG) and Module Interface Specification (MIS), will be reviewed to ensure accuracy and completeness. The verification process will involve a static analysis approach, where assigned reviewers will inspect the documents to confirm that they align with system requirements and architectural design principles. Reviewers will provide feedback through an issue-tracking system, allowing the document owner to address concerns systematically.
A structured MG Checklist and MIS Checklist will be used to ensure consistency, correctness, and traceability. The finalized documents will be reviewed again to validate that all identified issues have been resolved.

3.4 Verification and Validation Plan Verification Plan

The review process for Verification and Validation (VnV) Plan will involve assigned team members in Table 1 conducting a structured inspection to verify that the plan aligns with project requirements and verification strategies. Identified issues will be documented in an issue-tracking system for resolution.
A VnV Checklist will be used to assess key aspects of the plan, including coverage of testing methodologies and traceability of requirements. The final review will confirm that all necessary corrections have been implemented before approval.

3.5 Implementation Verification Plan

[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]

3.6 Automated Testing and Verification Tools

To ensure the correctness and efficiency of the FBP CT Image Reconstruction, the following automated testing and verification tools will be utilized:

Unit Testing
: pytest will be used to test individual functions, such as backprojection and filtering operations, to verify that they produce the expected output given test input data.

Performance Analysis
: cProfill will analyze execution time and identify performance bottlenecks in computationally intensive functions like fourier transforms and interpolation.

Static Analysis
: flake8 will enforce python code standards, ensuring maintainability and readability.

Test Coverage
: coverage.py will measure test coverage to ensure that all critical functions are tested and validated against various datasets.

Visualization
: matplotlib will be used to visualize sinograms and reconstructed images, allowing for manual verification of reconstruction accuracy.

Continuous Integration (CI) tools are unnecessary for this project since it does not require automated builds, tests, or deployments like a web application. Additionally, visual verification is crucial for image reconstruction, and CI automation cannot replace the need for manual inspection of reconstructed images.

3.7 Software Validation Plan

[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

The functional requirements are divided into input and output tests.
R1 correspond to input-related tests, ensuring that the system correctly processes input images, projection data, and transformations.
R2 and R3 correspond to output-related tests, verifying the correctness of Fourier filtering, back-projection, and reconstructed attenuation values.
While R3 operate on transformed projection data, it should be considered output-related because they primarily contribute to intermediate and final outputs of the reconstruction process, rather than defining new inputs.

[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 Input Tests

[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]

Test for Input - Intensity Values

1. test-input-intensity-id1

   Control: Automatic

   Initial State: System is ready to accept a 2-D grayscale image for Radon Transform.

   Input: A 2-D NumPy array (M by N matrix), where:

   - Number of row (M): Number of detector positions
   - Number of column (N): Number of projection angles.
   - Value in the matrix: X-ray intensity measurements, normalized between 0 and 1.

   Output: A valid 2-D matrix array (M by N matrix), where:

   - Number of row (M): Number of detector positions
   - Number of column (N): Number of projection angles.
   - Value in the matrix: Represent integrated projection data, between 0 and 1.

   Test Case Derivation: This test ensures that the system correctly accepts a valid grayscale 2D input image, applies the Radon Transform, and produces a valid sinogram matrix.
   The system should return an output that maintains consistency in shape and correctly reflects the values after applying the log transform.

   How test will be performed: The test will be automated using PyTest.

4.1.2

...

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?