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

2 General Information

2.1 Summary

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

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

[Introduce this section. You can provide a roadmap of the sections to come. —SS]

3.1 Verification and Validation Team

[Your teammates. Maybe your supervisor. You should do more than list names. You should say what each person's role is for the project's verification. A table is a good way to summarize this information. —SS]

3.2 SRS Verification

[List any approaches you intend to use for SRS verification. This may include ad hoc feedback from reviewers, like your classmates (like your primary reviewer), or you may plan for something more rigorous/systematic. —SS]

[If you have a supervisor for the project, you shouldn't just say they will read over the SRS. You should explain your structured approach to the review. Will you have a meeting? What will you present? What questions will you ask? Will you give them instructions for a task-based inspection? Will you use your issue tracker? —SS]

[Maybe create an SRS checklist? —SS]

3.3 Design Verification

[Plans for design verification —SS]

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

[Create a checklists? —SS]

3.4 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]

3.5 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]

3.6 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]

3.7 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

This section outlines the tests for verifying both functional and nonfunctional requirements of the system. It helps ensure that the system meets stakeholders' expectations. This section includes tests covering the essential aspects of the systems requirements.

4.1 Tests for Functional Requirements

The subsections below outline tests corresponding to functional requirements in the SRS. Each test is associated with a unique functional area, helping to confirm that the system meets the specified requirements.

4.1.1 Import Manager Tests

CAD File Import and Project Management Tests

1. test-FR-IM-1 Valid CAD File Import for New Project

   Type: Functional, Dynamic, Manual

   Covers: F211

   Initial State: No active project is loaded in the system, and the system is ready to create a new project.

   Input: A CAD file (STL file) is provided to the system through the import interface by the user.

   Output: The system creates the new project with the imported CAD file, initiates the voxel slicing process, and the imported model is ready for visualization.

   Test Case Derivation: The test case is derived from the functional requirement F211, which states that the Import Manager should allow a user to start a new project with the option to import an initial CAD file from their personal device that will be sliced.

   How test will be performed: Create a new project on the system and import a CAD file into the new project to verify that the system will correctly process the file and create a new project. The system should be able to handle the CAD file and initiate the voxel slicing process. This will be tested manually by the developer of the system.

2. test-FR-IM-2 Past Project File Import

   Type: Functional, Dynamic, Manual

   Covers: F212

   Initial State: No active project is loaded in the system, and the system is ready to open a past project.

   Input: A previously saved project file containing magnetization and material properties is provided to the system through the import interface by the user.

Output: The system loads the project containing all of the magnetization and material properties preserved. The model is ready for further editing and visualization.

Test Case Derivation: The test case is derived from the functional requirement F212, which states that the Import Manager should allow a user to import a past project file in order to reopen a project with all magnetization and material properties preserved.

How test will be performed: Open a previously saved project containing magnetization and material properties on the system. Verify that the system correctly loads the project and all of the magnetization and material properties are preserved. The model should be ready for further editing and visualization. This will be tested manually by the developer of the system.

3. test-FR-IM-3 Invalid File Format Handling

Type: Functional, Dynamic, Manual

Covers: F211

Initial State: No active project is loaded in the system, and the system is ready to import a file.

Input: A file with unsupported format (e.g. .txt, .jpg, .pdf) is provided to the system through the import interface by the user.

Output: The system rejects the file and displays an appropriate error message indicating the file format is unsupported, without creating a project or crashing.

Test Case Derivation: The test case is derived from the functional requirement F211, which states that the Import Manager should allow a user to start a new project with the option to import an initial CAD file from their personal device that will be sliced.

How test will be performed: Try to import a file with an unsupported format (e.g. .txt, .jpg, .pdf) into the system. Verify that the system correctly rejects the file and displays an appropriate error message indicating the file format is unsupported. This will be tested manually by the developer of the system.

Model Configuration and Slicing Tests

1. test-FR-IM-4 Voxel Size Configuration

   Type: Functional, Dynamic, Automated

   Covers: F213

   Initial State: A valid CAD model has been imported and the system is ready to configure the voxel dimensions.

   Input: User specifies custom voxel dimensions (e.g. 0.1mm x 0.1mm x 0.1mm) by entering the dimensions into the system through the import interface by the user.

   Output: The system applies the specified voxel dimensions and successfully slices the model into voxels of the requested dimensions.

   Test Case Derivation: The test case is derived from the functional requirement F213, which states that the Import Manager should allow a user to configure the voxel dimensions into which the model will be sliced.

   How test will be performed: Configure the voxel dimensions for the imported model by entering the dimensions into the system through the import interface by the user. Verify that the system correctly applies the specified voxel dimensions and successfully slices the model into voxels of the requested dimensions. This will be tested automatically by the developer of the system by running a script that configures the voxel dimensions and verifies that the system correctly slices the model into voxels of the requested dimensions.

2. test-FR-IM-5 Model Scaling Functionality

   Type: Functional, Dynamic, Manual

   Covers: F214

   Initial State: A valid CAD model has been imported and the system is ready to scale the model.

   Input: User modifies the model dimensions to the desired printing size by entering the new dimensions into the system through the import interface by the user.

Output: The model is scaled to the desired printing size, maintaining its geometric integrity and proportions.

Test Case Derivation: The test case is derived from the functional requirement F214, which states that the Import Manager should allow a user to scale the model to the desired printing size before being sliced into voxels.

How test will be performed: Modify the model dimensions to the desired printing size by entering the new dimensions into the system through the import interface by the user. The developer will then manually inspect the rendered 3D model within the application. Verification will involve comparing the visual representation of the sliced model against the original CAD model and the expected scaling of the model to confirm that model integrity and accuracy are preserved as specified in F214

3. test-FR-IM-6 Partial Voxel Resolution

   Type: Functional, Dynamic, Manual

   Covers: F215

   Initial State: A CAD model with complex geometry has been imported and the system is ready to resolve the partial voxels.

   Input: A model containing surfaces that intersect voxel boundaries is imported by the user through the import interface.

   Output: The system correctly resolves partial voxels and preserves model integrity and accuracy.

   Test Case Derivation: The test case is derived from the functional requirement F215, which states that the Import Manager shall resolve partial voxels and interpret them in the best way that preserves the integrity and accuracy of the provided model.

   How test will be performed: Import a model containing surfaces that intersect voxel boundaries. The developer will then manually inspect the rendered 3D model within the application. Verification will involve comparing the visual representation of the sliced model against the original CAD model and the expected resolution of partial voxels to

confirm that model integrity and accuracy are preserved as specified in F215

4. test-FR-IM-7 Model Division for Display

   Type: Functional, Dynamic, Manual

   Covers: F216

   Initial State: A large CAD model has been imported and the system is ready to divide the model into manageable display sections.

   Input: A model containing more than MAX_DISPLAY voxels is imported by the user through the import interface.

   Output: The system automatically partitions the model into manageable display sections that do not exceed the MAX_DISPLAY threshold, enabling smooth visualization.

   Test Case Derivation: The test case is derived from the functional requirement F216, which states that the Import Manager shall partition the model into user-manageable display sections that do not surpass the MAX_DISPLAY threshold.

   How test will be performed: Import a model containing more than MAX_DISPLAY voxels. The developer will then manually inspect the rendered 3D model within the application. Verification will involve comparing the visual representation of the sliced model against the original CAD model and the expected partitioning of the model to confirm that model integrity and accuracy are preserved as specified in F216

4.1.2 Visualization Manager Tests

3D Model Rendering and Display Tests

1. test-FR-VM-1 3D Model Rendition with Clear Voxel Visualization

   Type: Functional, Dynamic, Manual

   Covers: F221

   Initial State: A CAD model has been imported and sliced into voxels by the Import Manager and the system is ready to render the model.

Input: Sliced voxel data from Import Manager is provided to the Visualization Manager.

Output: The system renders a 3D model with clear visualization of individual voxels, maintaining geometric accuracy and providing distinct voxel boundaries.

Test Case Derivation: The test case is derived from the functional requirement F221, which states that the Visualization Manager shall recreate a 3D model with clear visualization of sliced voxels.

How test will be performed: Render a 3D model with clear visualization of sliced voxels by providing the Visualization Manager with sliced voxel data from the Import Manager. Verify that the system correctly renders the model and displays the individual voxels with proper boundaries and geometric accuracy. This will be tested manually by the developer of the system by providing the system with a CAD model and verifying that the system correctly renders the model and displays the individual voxels with proper boundaries and geometric accuracy.

2. test-FR-VM-2 Partition Navigation Functionality

   Type: Functional, Dynamic, Manual

   Covers: F222

   Initial State: A large model has been divided into multiple display partitions and the system is ready to navigate between the partitions.

   Input: User navigates between model partitions by clicking on the different display sections of the model.

   Output: The system provides smooth navigation between partitions, allowing users to access all model sections without performance degradation.

   Test Case Derivation: The test case is derived from the functional requirement F222, which states that the Visualization Manager shall provide users with simplistic navigation across user-manageable display sections to ensure access to all model partitions.

   How test will be performed: Navigate between model partitions by clicking on the different display sections of the model. Verify that the

system correctly provides smooth navigation between partitions, allowing users to access all model sections without performance degradation. This will be tested manually by the developer of the system.

3. test-FR-VM-3 Multi-Perspective 3D Model Interaction

   Type: Functional, Dynamic, Manual

   Covers: F223

   Initial State: A 3D model is rendered and displayed and the system is ready to interact with the model from different perspectives.

   Input: User rotates and zooms the model to view it from different perspectives using intuitive interface controls.

   Output: The system correctly provides intuitive interface controls that allow seamless navigation across multiple perspectives of the 3D model with responsive interaction.

   Test Case Derivation: The test case is derived from the functional requirement F223, which states that the Visualization Manager shall provide users with an intuitive interface that permits seamless navigation across multiple perspectives of the 3D model.

   How test will be performed: Rotate and zoom the model to view it from different perspectives using intuitive interface controls. Verify that the system correctly provides intuitive interface controls that allow seamless navigation across multiple perspectives of the 3D model with responsive interaction. This will be tested manually by the developer of the system.

Layer Focus and Voxel Selection Tests

1. test-FR-VM-4 Layer Isolation Functionality

   Type: Functional, Dynamic, Manual

   Covers: F224

   Initial State: A 3D model with multiple layers is displayed and the system is ready to isolate a specific layer.

Input: User selects a specific layer to focus on by clicking on the layer in the display.

Output: The system isolates the selected layer, rendering all other voxels irrelevant or transparent, facilitating property assignment on the focused layer.

Test Case Derivation: The test case is derived from the functional requirement F224, which states that the Visualization Manager shall allow users to isolate a specific layer to facilitate property assignments, rendering all other voxels irrelevant while that layer is in focus.

How test will be performed: Select a specific layer to focus on by clicking on the layer in the display. Verify that the system correctly isolates the selected layer and renders all other voxels irrelevant while that layer is in focus. This will be tested manually by the developer of the system.

2. test-FR-VM-5 Voxel Selection Highlighting

   Type: Functional, Dynamic, Manual

   Covers: F225

   Initial State: A specific layer is in focus and displayed and the system is ready to highlight the selected voxels.

   Input: User selects individual voxels or groups of voxels within the focused layer by clicking on the voxels in the display.

   Output: The system correctly provides clear visual feedback showing which voxels are currently selected using distinct highlighting.

   Test Case Derivation: The test case is derived from the functional requirement F225, which states that the Visualization Manager shall provide users with visualization that showcases which voxels are currently selected within a specific layer.

   How test will be performed: Select individual voxels or groups of voxels within the focused layer by clicking on the voxels in the display. Verify that the system correctly provides clear visual feedback showing which voxels are currently selected using distinct highlighting. This will be tested manually by the developer of the system.

Material and Magnetization Tracking Tests

1. test-FR-VM-6 Material Assignment Visual Tracking

   Type: Functional, Dynamic, Manual

   Covers: F226

   Initial State: A model with unassigned materials is displayed and the system is ready to track the material assignment.

   Input: User assigns material IDs to various voxels by clicking on the voxels in the display and assigning the material IDs.

   Output: The system updates the voxel colors to indicate material assignment completeness, providing clear visual feedback about the assignment status.

   Test Case Derivation: The test case is derived from the functional requirement F226, which states that the Visualization Manager shall integrate easy tracking of voxels that have been assigned material IDs by adjusting the colour of the voxel to indicate assignment completeness.

   How test will be performed: Assign material IDs to various voxels by clicking on the voxels in the display and assigning the material IDs. Verify that the system correctly updates the voxel colors to indicate material assignment completeness, providing clear visual feedback about assignment status. This will be tested manually by the developer of the system.

2. test-FR-VM-7 Magnetization Assignment Visual Tracking

   Type: Functional, Dynamic, Manual

   Covers: F227

   Initial State: A model with unassigned magnetization vectors is displayed and the system is ready to track the magnetization assignment.

   Input: User assigns magnetization vectors to various voxels by clicking on the voxels in the display and assigning the magnetization vectors.

   Output: The system updates the voxel colors to indicate magnetization assignment completeness, providing clear visual feedback about the assignment status.

Test Case Derivation: The test case is derived from the functional requirement F227, which states that the Visualization Manager shall integrate easy tracking of voxels that have been assigned magnetization vectors by adjusting the colour of the voxel to indicate assignment completeness.

How test will be performed: Assign magnetization vectors to various voxels by clicking on the voxels in the display and assigning the magnetization vectors. Verify that the system correctly updates the voxel colors to indicate magnetization assignment completeness, providing clear visual feedback about the assignment status. This will be tested manually by the developer of the system.

4.1.3 Integration Tests Between Import and Visualization Managers

Data Flow and Communication Tests

1. test-FR-INT-1 Import to Visualization Data Transfer

   Type: Functional, Dynamic, Automated

   Covers: F211, F221

   Initial State: No active project is loaded in the system, and the system is ready to import a file.

   Input: A valid CAD file is imported by the user through the import interface.

   Output: The Visualization Manager receives complete voxel data and successfully renders the 3D model without data loss or corruption.

   Test Case Derivation: The test case is derived from the functional requirement F211 and F221, which states that the Import Manager should allow a user to import a CAD file from their personal device that will be sliced and the Visualization Manager should successfully render the 3D model without data loss or corruption.

   How test will be performed: Import a valid CAD file. Verify that the system correctly imports the CAD file and the Visualization Manager successfully renders the 3D model without data loss or corruption. This will be tested automatically by the developer of the system by running

a script that imports a CAD file and verifies that the system correctly imports the CAD file and the Visualization Manager successfully renders the 3D model without data loss or corruption.

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.

- **OPTIMAL_VOXEL_SIZE** = 5.5nm

  This was derived using the optimal voxel size specified by the stakeholder.

- **MAX_VOXELS** = 13,996,800,000 voxels

  This considers the maximum number of voxels contained within the entire model.

- **MAX_LAYER_DISPLAY** = 518,400 voxels

  This was derived assuming 960 voxels x 540 voxels per layer. It considers the maximum number of voxels contained within a single layer for a single display window.

- **MAX_DISPLAY** = 103,680,000 voxels

  This was derived assuming 960 voxels x 540 voxels per layer. It considers the maximum number of voxels that will be displayed at a given time.

- **MIN_RATE** = 500,000 voxels per second

  This considers how fast voxel data will be generated during file import and export.

- **MAX_EDIT_LATENCY** = 1 second

  This considers how quickly the system should respond to model modifications performed by the user.

- **MAX_VIEW_LATENCY** = 500 ms

  This considers how quickly the system should respond to changes in model view orientation by the user.

- **MAX_INTERACTIONS** = 5

  This considers the maximum number of steps (e.g. clicks, enter details into a text box) a user should take to complete a part of a task.

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?