System Verification and Validation Plan for CVT Simulator

Team #17, Baja Dynamics
Grace McKenna
Travis Wing
Cameron Dunn
Kai Arseneau

October 30, 2024

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

List of Figures

1 Symbols, Abbreviations, and Acronyms

symbol	description
T	Test

[symbols, abbreviations, or acronyms — you can simply reference the SRS (Dunn et al., 2024) 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

The software that will be tested is the CVT Simulator, this is a system that simulates the Continuously Variable Transmission (CVT) system of a Baja vehicle. The system has three components, a user interface which will allow the user to input the parameters of the CVT system and visualize the output of the simulation. A mathematical component that will take the input parameters and simulate the CVT system. There is also a 3D model component that will display the some models of the CVT system. Each of these components covered will be verified and validated to ensure that the system is functioning as intended.

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

The challenge level of this project is general.

The extras for this project are the following:
Validation Report

- The system will undergo validation testing to verify that accuracy of the simulation.
- The validation report will be included in the final documentation.

Usability Testing

- The system will undergo usability testing by members of the McMaster Baja team to ensure that the user interface is intuitive and easy to use.
- We will compile the results of the usability testing into a report that will be included in the final documentation.

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]

Software Requirements Specification Dunn et al. (2024)

This document is relevant because it it list all the requirements that the system should have and specifies how they should be achieved. Requirements serve an important role in the verification process as the metric for determining if the system is functioning as intended. When we do the verification of the system we will compare the system's capabilities and outputs to the requirements from the SRS document. This will tell us what parts of the system are working as intended and which need to be changed or updated. For validation, the specification serves and equivalent role as the metric for determining if the system is implemented correctly. The specifications will be compared with the implementations of the system to see if they correspond. If they do not, then the system will need to be updated so the the implementations match with the specifications. Overall, the SRS document is important for verification and validation as it provides the metrics for determining their success.

Module Guide

The Module Guide will serve as a reference for the components of the system. Each module listed in the MG will have a corresponding test case in the VnV plan.

Module Interface Specification

The Module Interface Specification serves as a reference for the functions and methods of the system. Each function and method listed in the MIS will have a corresponding test case in the VnV plan.

Verification and Validation Report
The Verification and Validation Report will be referenced to ensure that all the components of the system have been verified and validated.

[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]

This section outlines the overall plans for the verification and validation of the system. This includes our approaches, techniques and tools that will be used. Our approaches our outlined and each person's role is specified. There is also a plan listed for both the verification and validation along with an implementation specification. Any tools that we plan to use to automate the process are also listed in this section. Overall this section details how we plan to verify and validate the system.

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]

Name	Role	Responsibilities
Kai Arseneau	Data Verification Lead	Analyzes data outputs to ensure trends align with expected performance and validates consistency across simulation runs.
Travis Wing	GUI Verification Lead	Ensures user interface functionality and usability, verifying that GUI elements accurately reflect simulation data and respond as intended.
Cameron Dunn	Experimental Data Comparison Lead	Compares simulation outputs to real-world data and scripts, verifying the simulation's accuracy and adjusting model parameters as needed.
Grace McKenna	Quality Assurance Lead	Conducts comprehensive testing of simulation processes and components, identifying and documenting issues, and ensuring overall software reliability.
Ariel Wolle	Team Lead	Supports the project by overseeing resources and approving initiatives that facilitate enhanced data acquisition or other verification efforts.
Daksh Mathur	Data Acquisition Lead	Acts as the primary contact for real-world data acquisition, providing essential data for validation and supporting data needs beyond the project group.
Benjamin Walde	Mechanical Advisor	Collaborates with the team to validate mathematical models within the simulation, ensuring mechanical accuracy and guiding technical correctness.
Duncan Shearer and Leonardo Ansari	Primary User	Utilizes the simulation tool to tune CVTs, providing feedback on tool usability and functionality to ensure it meets real-world tuning needs.
Dr. Spencer Smith	Project Advisor / Supervisor	Reviews and provides final feedback, ensuring verification meets project standards and advising on methodology improvements.

Table 1: Verification and Validation Team for CVT Simulation Project

3.2 SRS Verification Plan

[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]

Do a quick intro for this section, stating that all issues with the SRS will be brought onto the github issue tracker. For all below sections, there is also a checklist that can be used as follows:

- Review the SRS document
- Check the theoretical models
- Check the data definitions
- Check the instance models

This checklist is to be used by the team as well as our supervisor.

Review Meetings

This section discusses sitting down with Ben and Ariel to review the SRS document, but specifically the theoretical models, data definitions and instance models. Alumni as well?

Practicality check on the requirements to ensure they are all feasible and measureable metrics. Assumptions check to ensure they will not have a massive impact on our simulations behaviour.

Team Wide Review

Talk here about sharing the document to the team as a whole, include a checklist and receive issues on github.

Us review

Perform unit analysis / dimension analysis inspections at each stage of mathematical models, ensuring consistency.

Traceability matrix will be verified and use to make sure requirements map logically to our goals. Further using our asumptions in this will be sure they are all valid assumptions and are used somewhere.

3.3 Design Verification Plan

[Plans for design verification —SS]

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

[Create a checklists? —SS]

To ensure that the CVT system is functional and accurate, a design verification plan will be conducted on the system. The components of the system that will be tested are the graphical user interface, data output and visualization and the mathematical model. The design verification plan will include, reviews from fellow classmates, stakeholder reviews and specification testing based on the functional and nonfunctional requirements of the system. The system will receive Stakeholder review providing us with relevant feedback on the functionality and interface of our design. The system will undergo reviews from fellow classmates, providing our team with quality feedback from an outside perspective. A checklist has been designed below to which fellow classmates and stakeholders can use to verify the system.

1. For each functional requirement there is a at least one corresponding system test.

2. For each non functional requirement there is a at least one corresponding system test.

3. The system tests contain tests for boundary cases for user inputs, empty inputs and edge cases.

4. Each instance model (IM) within the SRS document shall correspond to a unit test.

5. Each function written in our mathematical python model shall correspond to at least one unit test.

6. There exists system tests for each component: user interface, mathematical model, data visualization and 3D model.

3.4 Verification and Validation Plan Verification Plan

[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 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

[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 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

[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

Cameron Dunn, Grace McKenna, Kai Arseneau, and Travis Wing. System requirements specification. https://github.com/gr812b/CVT-Simulator/blob/develop/docs/SRS/SRS.pdf, 2024.

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?