Project Title: System Verification and Validation Plan for The Nursery Project

Aaron Billones, billonea
Gillian Ford, fordg
Juan Moncada, moncadaj
Steven Ramundi, ramundis

November 2, 2022

Date	Version	Notes
2022-11-02	Juan Moncada, Aaron Billones, Steven Ramundi, Gillian Ford	Initial release

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 (**?**) 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]

2.3 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. —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 shoud 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 Plan

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

[Maybe create an SRS checklist? —SS]

3.3 Design Verification Plan

[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 Plan

[The verification and validation plan is an artifact that should also be verified. —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]

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]

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

4 System Test Description

4.1 Tests for Functional Requirements

The following section includes system test cases for functional requirements. The tests are designed in such a way to ensure that all the functional requirements are met. For reference of the functional requirements, please review the SRS document.

4.1.1 Pot-pulator Complete System Testing

[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 —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 Tray Dispenser Subsystem Testing

1. TDST-01: **Tray Stack Detection**

   Control: Static, Manual

   Initial State: No trays present in the stack. Trays present in the stack.

   Input: Sensor reads the status of tray stack.

   Output: Sends a signal/bit to microprocessor that tells the system there are/aren't trays present.

   Test Case Derivation: The observed signal/bit is the expected value. The subsystem does not operate when no trays are present.

   How test will be performed: All other sensors and subsystems will be switched off. All trays will be removed from the stack. The detection bit will be observed. Then trays will be placed in the stack, and the detection bit will be observed.

2. TDST-02: **Operation from Tray Stack Detection**

   Control: Dynamic, Manual

   Initial State: Some amount of trays in the stack.

   Input: Sensor reads the status of tray stack.

   Output: Subsystem operates or remains idle.

   Test Case Derivation: If no trays are present, the subsystem will not operate and remain ready in the idle state. Otherwise, operate normally.

   How test will be performed: All other sensors and subsystems will be switched off. Trays will be removed from the stack and operation will be observed. Trays will be put in the stack and operation will be observed.

3. TDST-03: **Tray from Stack to Conveyor**

   Control: Dynamic, Manual

   Initial State: There is a stack of trays beside the vacant conveyor with the subsystem in idle position.

   Input: Stack of trays.

   Output: One tray from the stack is placed onto the end of the conveyor and returns to idle position.

   Test Case Derivation: There is a tray in the correct designated position. The subsystem moves into the ready idle state to retrieve more trays.

   How test will be performed: All other sensors and subsystems will be switched off. The system will be manually activated to retrieve one tray from its stack. The success or failure will be observed.

4. TDST-04: **Verify Tray Status on Conveyor**

   Control: Dynamic, Manual

   Initial State: Tray put on conveyor.

   Input: Sensor reads the status of tray on conveyor.

   Output: Subsystem continues operation or stops.

Test Case Derivation: Subsystem continues operation (when successful) or stops (when tray is stuck/fails to move on conveyor).

How test will be performed: Trays will be fed onto the conveyor correctly. Results will be observed. then trays will be placed stuck on purpose. Results will be observed.

4.1.3 Pot Dispenser Subsystem Testing

The tests outlined below will cover all functional requirements outlined in the SRS pertaining to the pot dropping subsystem. They will cover functional requirements involving the ability of the pot dispenser to place pots into trays, confirm the presence of a tray below the pot dispenser, and cease operation and notify an operator once pot storage is empty.

1. PDTest-01: **Placing a pot into an empty tray**

   Control: Dynamic, Manual

   Initial State: Pot dispenser loaded with two pots

   Input: Simulated sensor input, two pot locations of tray directly below pot dispenser

   Output: Pot dispenser will dispense two pots into designated pot locations on tray

   Test Case Derivation: Pot dispenser will dispense pots into correctly positioned tray as it is prompted to

   How test will be performed: Tray will be manually placed directly below pot dispenser with pot locations directly below pot stack. Machine will be turned on. Once pots are dispensed, pot dispenser will queue next two pots and tray will be removed.

2. PDTest-02: **Tray sensing (PDR2)**

   Control: Dynamic, Manual

   Initial State: Mounted sensor with no object being sensed

   Input: Manual placement of trays in front of sensor

Output: Sensor will output a signal when the presence of a tray is sensed

Test Case Derivation: Sensor will recognize that a tray is beneath the pot dispenser

How test will be performed: Tray will be manually placed directly in front of the mounted sensor. Signal output from sensor will be analyzed to determine sensor is aware of tray presence. Tray will then be moved forward and output from sensor will be analyzed to confirm sensor is aware that tray is moving. Tray will then be moved forward out of view of sensor and output from sensor will be analyzed to confirm snesor is aware that tray is no longer present.

3. PDTest-03: **Ability to dispense 4" diameter pots (PDR 3)**

   Control: Static, Manual

   Initial State: Pot dispenser mechanism loaded with one pot

   Input: Single pot

   Output: Single pot

   Test Case Derivation: Pot dispenser mechanism will dispense one 4" diameter pot

   How test will be performed: All specifications of pot dispenser will ensure that a 4" diameter pot is able to be dispensed. Measurements and reviews will be conducted by another member of the group any time a change is made to the dispenser during design and build phases. During build phase, test will be conducted on both pot dispensers.

4. PDTest-04: **Ability to store/dispense multiple pots (PDR4)**

   Control: Dynamic, Manual

   Initial State: Pot dispenser loaded with pots

   Input: Ten pots, simulated sensor input

   Output: Pot dispenser will dispense two pots, reload with two pots from stack, dispense two pots, etc. until pot storage is empty

Test Case Derivation: Pot dispenser will complete 5 cycles of dispensing, storing and dispensing 10 pots in total

How test will be performed: Pot dispenser will be loaded with 10 pots, 5 per side. Sensor input will be simulated to indicate presence of tray. Pot dispenser will complete 5 cycles of dispensing, at which point pot storage will be spent.

5. PDTest-05: **Pot storage sensing (PDR5, PDR6)**

   Control: Dynamic, Manual

   Initial State: Pot dispenser with no pots in storage

   Input: N/A

   Output: Pot storage sensor will output a signal when no trays are detected in pot storage

   Test Case Derivation: Sensor will recognize that no pots are sensed in pot storage

   How test will be performed: All pots will be removed from pot storage. Signal output from sensor will be analyzed to confirm sensor is aware that pot storage is empty.

4.1.4 Conveyor Subsystem Testing

4.1.5 Verification Subsystem Testing

1. VST-01: **Verify Correct Number of Pots in Tray**

   Control: Dynamic, Manual

   Initial State: One tray filled with some pots placed on the conveyor.

   Input: Tray filled with a number of pots.

   Output: Returns a count of the number of pots in the tray.

   Test Case Derivation: The count read by the subsystem matches the actual number of pots in the tray.

   How test will be performed: All other sensors and subsystems will be switched off. The subsystem will be manually activated to count the number of pots in the given tray as it moves on the conveyor. The

success or failure will be observed.

2. VST-02: **Operation from Verification Status**

   Control: Dynamic, Manual

   Initial State: Tray has completed counting the number of pots in the tray and deemed it success or fail.

   Input: Status bit for success or fail of the pot verification step.

   Output: Signal to tell the system to continue/stop operation based on status bit.

   Test Case Derivation: The subsystem should signal the main processsor to turn off other subsystems when there is a problem in verifying the number of pots (ie. $actual \neq target$).

   How test will be performed: All other sensors and subsystems will be switched off. The subsystem will be manually activated to count the number of pots in the given tray as it moves on the conveyor. The success or failure will send a status bit to the main processor. The status bit will be observed.

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]

[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]

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

The following table outlines all of the system tests and how they relate to the relevent requirements. The requirements can be referenced in the SRS document.

Table 1: Corresponding Test IDs and Requirements

Test ID	**Supporting Requirements**
TDST-01	TDR3, TDR5
TDST-02	TDR4, TDR5
TDST-03	TDR2
TDST-04	TDR2
VST-01	VR1
VST-02	VR2

5 Unit Test Description

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS] [This section should not be filled in until after the MIS (detailed design document) has been completed. —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]

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

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

1. 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 etc. You should look to identify at least one item for each team member.

2. 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?