Software Requirements Specification for ProgName: subtitle describing software

Team #, Team Name
Student 1 name
Student 2 name
Student 3 name
Student 4 name

October 5, 2022

Contents

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

1 Introduction

[The introduction section is written to introduce the problem. It starts general and focuses on the problem domain. The general advice is to start with a paragraph or two that describes the problem, followed by a "roadmap" paragraph. A roadmap orients the reader by telling them what sub-sections to expect in the Introduction section. —TPLT]

1.1 Purpose of Document

[This section summarizes the purpose of the SRS document. It does not focus on the problem itself. The problem is described in the "Problem Description" section (Section **??**). The purpose is for the document in the context of the project itself, not in the context of this course. Although the "purpose" of the document is to get a grade, you should not mention this. Instead, "fake it" as if this is a real project. The purpose section will be similar between projects. The purpose of the document is the purpose of the SRS, including communication, planning for the design stage, etc. —TPLT]

1.2 Scope of Requirements

[Modelling the real world requires simplification. The full complexity of the actual physics, chemistry, biology is too much for existing models, and for existing computational solution techniques. Rather than say what is in the scope, it is usually easier to say what is not. You can think of it as the scope is initially everything, and then it is constrained to create the actual scope. For instance, the problem can be restricted to 2 dimensions, or it can ignore the effect of temperature (or pressure) on the material properties, etc. —TPLT]

[The scope section is related to the assumptions section (Section **??**). However, the scope and the assumptions are not at the same level of abstraction. The scope is at a high level. The focus is on the "big picture" assumptions. The assumptions section lists, and describes, all of the assumptions. —TPLT]

1.3 Characteristics of Intended Reader

[This section summarizes the skills and knowledge of the readers of the SRS. It does NOT have the same purpose as the "User Characteristics" section (Section **??**). The intended readers are the people that will read, review and maintain the SRS. They are the people that will conceivably design the software that is intended to meet the requirements. The user, on the other hand, is the person that uses the software that is built. They may never read this SRS document. Of course, the same person could be a "user" and an "intended reader." —TPLT]

[The intended reader characteristics should be written as unambiguously and as specifically as possible. Rather than say, the user should have an understanding of physics, say what kind of physics and at what level. For instance, is high school physics adequate, or should the reader have had a graduate course on advanced quantum mechanics? —TPLT]

1.4 Organization of Document

[This section provides a roadmap of the SRS document. It will help the reader orient themselves. It will provide direction that will help them select which sections they want to read, and in what order. This section will be similar between project. —TPLT]

2 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints. [This text can likely be borrowed verbatim. —TPLT]

[The purpose of this section is to provide general information about the system so the specific requirements in the next section will be easier to understand. The general system description section is designed to be changeable independent of changes to the functional requirements documented in the specific system description. The general system description provides a context for a family of related models. The general description can stay the same, while specific details are changed between family members. —TPLT]

2.1 System Context

[Your system context will include a figure that shows the abstract view of the software. Often in a scientific context, the program can be viewed abstractly following the design pattern of Inputs → Calculations → Outputs. The system context will therefore often follow this pattern. The user provides inputs, the system does the calculations, and then provides the outputs to the user. The figure should not show all of the inputs, just an abstract view of the main categories of inputs (like material properties, geometry, etc.). Likewise, the outputs should be presented from an abstract point of view. In some cases the diagram will show other external entities, besides the user. For instance, when the software product is a library, the user will be another software program, not an actual end user. If there are system constraints that the software must work with external libraries, these libraries can also be shown on the System Context diagram. They should only be named with a specific library name if this is required by the system constraint. —TPLT]

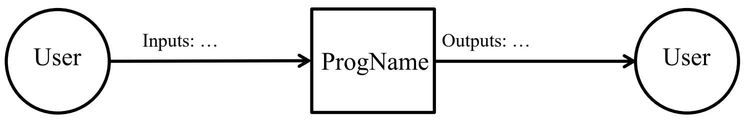

Figure 1: System Context

[For each of the entities in the system context diagram its responsibilities should be listed. Whenever possible the system should check for data quality, but for some cases the user will need to assume that responsibility. The list of responsibilites should be about the inputs and outputs only, and they should be abstract. Details should not be presented here. However, the information should not be so abstract as to just say "inputs" and "outputs". A summarizing phrase can be used to characterize the inputs. For instance, saying "material properties" provides some information, but it stays away from the detail of listing every required properties. —TPLT]

- User Responsibilities:
  -

- ProgName Responsibilities:
  - Detect data type mismatch, such as a string of characters instead of a floating point number
  -

2.2 User Characteristics

[This section summarizes the knowledge/skills expected of the user. Measuring usability, which is often a required non-function requirement, requires knowledge of a typical user. As mentioned above, the user is a different role from the "intended reader," as given in Section **??**. As in Section **??**, the user characteristics should be specific an unambiguous. For instance, "The end user of ProgName should have an understanding of undergraduate Level 1 Calculus and Physics." —TPLT]

2.3 System Constraints

[System constraints differ from other type of requirements because they limit the developers' options in the system design and they identify how the eventual system must fit into the world. This is the only place in the SRS where design decisions can be specified. That is, the quality requirement for abstraction is relaxed here. However, system constraints should only be included if they are truly required. —TPLT]

3 Requirements

This section provides the functional requirements, the business tasks that the project is expected to complete, and the nonfunctional requirements, the qualities that the project is expected to exhibit. Requirements will be trackl through github issues, which shall be be updated and not marked completed until the requirement has been met in full b y the project.

Requirements that we should outline
take in a stack on 30 trays, take in a stack of 300 pots.
fill a tray of pots in 30 seconds
should be able to connect to the pre- existing conveyor system
must be be able to verify that trays have been filled with the proper amount of pots

3.1 Functional Requirements

fit criterion: basically a test case for all of the functional requirements should be a benchmark for product specifics
the following table will outline the requirements of the project.

requirement number	requirement catagory	requirement
1	tray dispensing	tray dispensing must be able to handle the autonomous lo
2	tray dispensing	tray dispensing must be able to recognize that there are
3	tray dispensing	If there are no trays to be dispensed, tray dispensing shou
4	tray dispensing	if there are

there should be a test case for each use case

R1: [Requirements for the inputs that are supplied by the user. This information has to be explicit. —TPLT]

R2: [It isn't always required, but often echoing the inputs as part of the output is a good idea. —TPLT]

R3: [Calculation related requirements. —TPLT]

R4: [Verification related requirements. —TPLT]

R5: [Output related requirements. —TPLT]

[Every IM should map to at least one requirement, but not every requirement has to map to a corresponding IM. —TPLT]

4 Nonfunctional Requirements

the systems nonfunctional requirements will be divided by categories for which the requirements fit into. The mention of subsystems throughout this section is a reference to the four subsystems that the project can be devided into; conveyor, tray dispensor, pot dispensor and verification.

4.1 Appearance Requirements

NFR1: All electrical equipment and electronics must be well covered and protected. The User must not have access to equipment.

NFR2: All wiring must be tucked away and not accessible to avoid potential woring failure.

NFR3: All moviung part must be covered and protected. moving parts should be covered to protect both the mechanism and the safety of the opperator.

4.2 Usability Requirements

NFR4: System must have tray and pot refill locations visible and accessible in order to provide visual verification of capacity status.

NFR5: System output (end of conveyor) must be clear and visible in order to provide visual verification and access to failed outputs.

4.3 Learning Requirments

NFR6: System must be simple to opperate, requiiring minimal training to opperate effectively ($< 1h$).

4.4 Accessibility Requirements

NFR7: System must have both audible and visible signal outputs for each system status.

Speed Requirements

NFR8: Conveyor system must not accelerate in a manner that would hsift the position of the tray. a shift in the position of the tray could result in a misalignement and potential error.

4.5 Safety Critical Requirements

NFR9: System must hgave emergency cut off. in case of any emergency this will trip off all power to system.

NFR10: System must be able to locate and identify failures within each independent subsystem.

4.6 Precision Requirements

NFR11: Conveyor must center trays before potting dispensor withing 1 cm of centered position, this allows the tray to enter the pot dispensing system within tolerance for a pot to be dropped in.

NFR12: Pot dispensor should dispence pots within a 0.5 cm radius of centered position.

4.7 Reliability Requirements

NFR13: System must be able to opperate under constant low and high frequency vibration (small amplitude).

4.8 Robustness and Fault Tolerance Requirements

NFR14: Components must withstand 250,000 cycles system should call for replacement parts after a full season of opperation.

4.9 Scalability or Extensibility Requirements

NFR15: System must fit into the existing assembly line and should take minimal effort to implement ($< 1d$).

4.10 Expected Physical Environement Requirements

NFR16: System must withstand operating in a room with high arial polution including dust, dirt and small amounts of water.

4.11 Maitenance Requirements

NFR17: System must be built for ease of maitenance. high wear parts should be easily accessible.

4.12 Requirements for Interacting with Adjacent Systems

NFR18: system must opperate at same speed as adjacent systems.

NFR19: System must opperate at the same conveyor height as the current system to maintain continuity between systems.

4.13 Supportability Requirements

NFR20: System documentation must be available to troubleshoot, diagnose and fix common issues and replace high wear parts.

4.14 Compliance Requirements

NFR21: System must follow electronics system safety requirements.

5 Likely Changes

LC1: [Give the likely changes, with a reference to the related assumption (aref), as appropriate. —TPLT]

6 Unlikely Changes

LC2: [Give the unlikely changes. The design can assume that the changes listed will not occur. —TPLT]

7 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table **??** shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table **??** shows the dependencies of instance models, requirements, and data constraints on each other. Table **??** shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

[You will have to modify these tables for your problem. —TPLT]

[The traceability matrix is not generally symmetric. If GD1 uses A1, that means that GD1's derivation or presentation requires invocation of A1. A1 does not use GD1. A1 is "used by" GD1. —TPLT]

[The traceability matrix is challenging to maintain manually. Please do your best. In the future tools (like Drasil) will make this much easier. —TPLT]

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at

	T??	T??	T??	GD??	GD??	DD??	DD??	DD??	DD??	IM??	IM??	IM??	IM
T??													
T??			X										
T??													
GD??													
GD??	X												
DD??				X									
DD??				X									
DD??													
DD??								X					
IM??					X	X	X				X		
IM??					X		X		X	X			X
IM??		X											
IM??		X	X				X	X	X		X		

Table 1: Traceability Matrix Showing the Connections Between Items of Different Sections

	IM??	IM??	IM??	IM??	??	R??	R??
IM??		X				X	X
IM??	X			X		X	X
IM??						X	X
IM??		X				X	X
R??							
R??						X	
R??					X		
R??	X	X				X	X
R??	X						
R??		X					
R??			X				
R??				X			
R??			X	X			
R??		X					
R??		X					

Table 2: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??
T??	X																		
T??																			
T??																			
GD??		X																	
GD??			X	X	X	X													
DD??							X	X	X										
DD??			X	X						X									
DD??																			
DD??																			
IM??											X	X		X	X	X			X
IM??												X	X			X	X	X	
IM??														X					X
IM??													X					X	
LC??				X															
LC??								X											
LC??									X										
LC??											X								
LC??												X							
LC??															X				

Table 3: Traceability Matrix Showing the Connections Between Assumptions and Other Items

the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed. Figure **??** shows the dependencies of theoretical models, general definitions, data definitions, instance models, likely changes, and assumptions on each other. Figure **??** shows the dependencies of instance models, requirements, and data constraints on each other.

8 Development Plan

[This section is optional. It is used to explain the plan for developing the software. In particular, this section gives a list of the order in which the requirements will be implemented. In the context of a course this is where you can indicate which requirements will be implemented as part of the course, and which will be "faked" as future work. This section can be organized as a prioritized list of requirements, or it could should the requirements that will be implemented for "phase 1", "phase 2", etc. —TPLT]

9 Values of Auxiliary Constants

[Show the values of the symbolic parameters introduced in the report. —TPLT]

[The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —TPLT]

[The value of FRACTION, for the Maintainability NFR would be given here. —TPLT]

[The following is not part of the template, just some things to consider when filing in the template. —TPLT]
[Grammar, flow and LaTeXadvice:

- For Mac users `*.DS_Store` should be in `.gitignore`
- LaTeX and formatting rules
  - Variables are italic, everything else not, includes subscripts (link to document)
    - Conventions
    - Watch out for implied multiplication
  - Use BibTeX
  - Use cross-referencing
- Grammar and writing rules
  - Acronyms expanded on first usage (not just in table of acronyms)
  - "In order to" should be "to"

—TPLT]
[Advice on using the template:

- Difference between physical and software constraints
- Properties of a correct solution means *additional* properties, not a restating of the requirements (may be "not applicable" for your problem). If you have a table of output constraints, then these are properties of a correct solution.
- Assumptions have to be invoked somewhere
- "Referenced by" implies that there is an explicit reference
- Think of traceability matrix, list of assumption invocations and list of reference by fields as automatically generatable
- If you say the format of the output (plot, table etc), then your requirement could be more abstract

—TPLT]

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 this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, 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?