Software Requirements Specification for The Nursery Project: The Pot-Pulator

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 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	unit	SI
m	length	metre
kg	mass	kilogram
s	time	second
°C	temperature	centigrade
J	energy	joule
W	power	watt ($\mathrm{W} = \mathrm{J\,s^{-1}}$)

[Only include the units that your SRS actually uses. —TPLT]

[Derived units, like newtons, pascal, etc, should show their derivation (the units they are derived from) if their constituent units are in the table of units (that is, if the units they are derived from are used in the document). For instance, the derivation of pascals as $\mathrm{Pa} = \mathrm{N\,m^{-2}}$ is shown if newtons and m are both in the table. The derivations of newtons would not be shown if kg and s are not both in the table. —TPLT]

[The symbol for units named after people use capital letters, but the name of the unit itself uses lower case. For instance, pascals use the symbol Pa, watts use the symbol W, teslas use the symbol T, newtons use the symbol N, etc. The one exception to this is degree Celsius. Details on writing metric units can be found on the NIST web-page. —TPLT]

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with the heat transfer literature and with existing documentation for solar water heating systems. The symbols are listed in alphabetical order.

symbol	unit	description
A_C	m^2	coil surface area
A_{in}	m^2	surface area over which heat is transferred in

[Use your problems actual symbols. The si package is a good idea to use for units. —TPLT]

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
ProgName	[put an expanded version of your program name here (as appropriate) —TPLT]
T	Theoretical Model

[Add any other abbreviations or acronyms that you add —TPLT]

1.4 Mathematical Notation

[This section is optional, but should be included for projects that make use of notation to convey mathematical information. For instance, if typographic conventions (like bold face font) are used to distinguish matrices, this should be stated here. If symbols are used to show mathematical operations, these should be summarized here. In some cases the easiest way to summarize the notation is to point to a text or other source that explains the notation. —TPLT]

[This section was added to the template because some students use very domain specific notation. This notation will not be readily understandable to people outside of your domain. It should be explained. —TPLT]

[This SRS template is based on Smith and Lai (2005); Smith et al. (2007). It will get you started. You should not modify the section headings, without first discussing the change with the course instructor. Modification means you are not following the template, which loses some of the advantage of a template, especially standardization. Although the bits shown below do not include type information, you may need to add this information for your problem. If you are unsure, please can ask the instructor. —TPLT]

[Feel free to change the appearance of the report by modifying the LaTeX commands. —TPLT]

[This template document assumes that a single program is being documented. If you are documenting a family of models, you should start with a commonality analysis. A separate template is provided for this. For program families you should look at Smith (2006); Smith et al. (2017). Single family member programs are often programs based on a single physical model. General purpose tools are usually documented as a family. Families of physical models also come up. —TPLT]

[The SRS is not generally written, or read, sequentially. The SRS is a reference document. It is generally read in an ad hoc order, as the need arises. For writing an SRS, and for reading one for the first time, the suggested order of sections is:

- Goal Statement
- Instance Models
- Requirements
- Introduction
- Specific System Description

—TPLT]

[Guiding principles for the SRS document:

- Do not repeat the same information at the same abstraction level. If information is repeated, the repetition should be at a different abstraction level. For instance, there will be overlap between the scope section and the assumptions, but the scope section will not go into as much detail as the assumptions section.

—TPLT]

[The template description comments should be disabled before submitting this document for grading. —TPLT]

[You can borrow any wording from the text given in the template. It is part of the template, and not considered an instance of academic integrity. Of course, you need to cite the source of the template. —TPLT]

[When the documentation is done, it should be possible to trace back to the source of every piece of information. Some information will come from external sources, like terminology. Other information will be derived, like General Definitions. —TPLT]

[An SRS document should have the following qualities: unambiguous, consistent, complete, validatable, abstract and traceable. —TPLT]

[The overall goal of the SRS is that someone that meets the Characteristics of the Intended Reader (Section 2.4) can learn, understand and verify the captured domain knowledge. They should not have to trust the authors of the SRS on any statements. They should be able to independently verify/derive every statement made. —TPLT]

2 Introduction

The current method of preparing pots and trays to be filled with soil and populated with seeds at Sheridan Nurseries is a process with little to no automation, requiring many manual labour hours. Each year, 250,000 pots need to be placed in trays, each one being placed by hand. Recently, the owners of these farms have been finding it increasingly more difficult to fill these roles with enough workers to run the operation smoothly and meet production demands.

We aim to aid in this process by designing and implementing a machine that is able to fill trays with pots and prepare them for populating with soil and seeds. This will alleviate the reliance on manual labour and improve the overall efficiency of the farm.

This introduction will outline the document's purpose, the scope of requirements, characteristics of the intended reader, and the organization of document.

2.1 Purpose of Document

This Software Requirements Specification (SRS) document is to define the purpose of the Pot-Pulator. It will describe how the software will interact with the hardware, and how it is expected to perform, defining the executable requirements leading to the creation of a final product. This document should lay the important groundwork needed for every team member to understand the in-depth technical details of the product to be developed, in preparation for the design stage.

2.2 Scope of Requirements

The branches of the Pot-Pulator include a conveyer belt, tray allocation, pot dropping and verification. These four systems will be implemented to work together to create an efficient product. It will be reliable, easily configurable, affordable, and will greatly reduce the need for hands-on labour in relation to seed population.

The system implemented will be designed exclusively for Sheridan Nurseries, and for the pots and trays at their locations in a three-dimensional workspace. It is limited to usage on their round, 4-inch diameter pots and their respective trays. Considering the conditions of the greenhouses this machine will be used in, material degradation, water resistance, and pressure will all be negligible.

In order to maintain speed of the pots being placed in trays in a timely manor, one of the goals of the machine is to fill each tray with 10 pots every 30 seconds. This is the time it will take for the trays to be populated and move down the conveyer belt. With this goal in place, the machine will only need to be refilled with pots and trays every 15 minutes, greatly reducing the amount of hands-on labour from the workers at Sheridan Nurseries.
After the initial completion of the goal associated with the 4-inch pots, larger pots and trays may be considered as a future goal. There will always be the limitation of the requirement of a rim on the top of the pot, for the machine to easily grasp onto when lifting and dropping.

2.3 Behavioural Overview

The Pot-Pulator will have 4 branches working together to accomplish the task. First, the trays and pots will be manually filled into a section of the machine, ideally 30 trays and 300 pots. The first section of the machine will drop a tray, and the conveyer belt will move it to the pot dropping section.
The conveyer belt will stop once it senses the tray is in the correct position. The machine will drop pots until all 10 spots are filled. The conveyer belt will move the tray according to the needed positions for pot dropping, dispensing 2 pots into the allocated positions, moving forward 4 inches at a time until the process is completed.
The tray filled with pots will then move into a verification system, to verify the pots have been placed in the correct positions. The conveyer belt will then move it to another conveyer system, which will lead to the collection of completed trays. The only maintenance this machine needs is the refill of pots and trays every 15 minutes, and the collection of completed trays at the end of the final conveyer belt.

2.4 Characteristics of Intended Reader

This SRS document is written for the directly individuals involved with the development of the Pot-Pulator. This includes every member of the team; all being involved with the development of the project. It is useful as reference to the functionality of in-depth details regarding the project, and will be read, reviewed, and maintained as the project progresses by all team members.
In general, any users of this document should have an undergraduate level understanding of software and hardware system functionality, physics, and mathematics.

2.5 Organization of Document

The format of this document follows the standard IEEE template divided into multiple sections. The first two sections describe the system descriptions. First a general system description defining the system context, user characteristics and system constraints. It follows with a more specific system description, where the project is looked at more closely, outlining terminology, goal statements, models and constraints.

The next section is dedicated to Requirements. This section will define functional requirements and non-functional requirements with respect to the system features. Finally, the following sections review the likely and unlikely changes of the project. It will reference any specifications that may be subject to change when further advancing with the software and hardware requirements.

3 Project Constraints

This section identifies the interfaces between the system and its environment, describes the user characteristics, lists the system constraints, and provides assumptions relating to the system/environment.

3.1 System Context

The system will consist of a stack of pots, stack of trays, and the mechanism that will autonomously populate the trays with the pots. The trays will move along a conveyor system that operates at approximately 3 feet and will act as the operating workspace for the device. Once the trays are populated with pots, they are fed into a soil potting machine in preparation for planting.

3.2 User Responsibilities & Characteristics

The user will be responsible for refilling the stacks of trays and pots when the supply is depleted. The user will also be responsible for knowing how the entire system will operate within each state. For example, if a tray fails the validation step, the user must be capable of removing the defect and resume the operation of the machine.

The characteristics of the user will ensure smooth operation and workplace safety. The user should be able to lift a sufficient weight and have a good understanding of basic functionality of the system.

3.3 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]

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models. [Add any project specific details that are relevant for the section overview. —TPLT]

4.1 Problem Description

ProgName is intended to solve ... [What problem does your program solve? The description here should be in the problem space, not the solution space. —TPLT]

4.1.1 Terminology and Definitions

[This section is expressed in words, not with equations. It provide the meaning of the different words and phrases used in the domain of the problem. The terminology is used to introduce concepts from the world outside of the mathematical model The terminology provides a real world connection to give the mathematical model meaning. —TPLT]

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

-

4.1.2 Physical System Description

[The purpose of this section is to clearly and unambiguously state the physical system that is to be modelled. Effective problem solving requires a logical and organized approach. The statements on the physical system to be studied should cover enough information to solve the problem. The physical description involves element identification, where elements are defined as independent and separable items of the physical system. Some example elements include acceleration due to gravity, the mass of an object, and the size and shape of an object. Each element should be identified and labelled, with their interesting properties specified clearly. The physical description can also include interactions of the elements, such as the following: i) the interactions between the elements and their physical environment; ii) the interactions between elements; and, iii) the initial or boundary conditions. —TPLT]

The physical system of ProgName, as shown in Figure ?, includes the following elements:

PS1:

PS2: ...

[A figure here makes sense for most SRS documents —TPLT]

4.1.3 Goal Statements

[The goal statements refine the "Problem Description" (Section 4.1). A goal is a functional objective the system under consideration should achieve. Goals provide criteria for sufficient completeness of a requirements specification and for requirements pertinence. Goals will be refined in Section "Instanced Models" (Section 4.2.6). Large and complex goals should be decomposed into smaller sub-goals. The goals are written abstractly, with a minimal amount of technical language. They should be understandable by non-domain experts. —TPLT]

Given the [inputs —TPLT], the goal statements are:

GS1: [One sentence description of the goal. There may be more than one. Each Goal should have a meaningful label. —TPLT]

4.2 Solution Characteristics Specification

[This section specifies the information in the solution domain of the system to be developed. This section is intended to express what is required in such a way that analysts and stakeholders get a clear picture, and the latter will accept it. The purpose of this section is to reduce the problem into one expressed in mathematical terms. Mathematical expertise is used to extract the essentials from the underlying physical description of the problem, and to collect and substantiate all physical data pertinent to the problem. —TPLT]

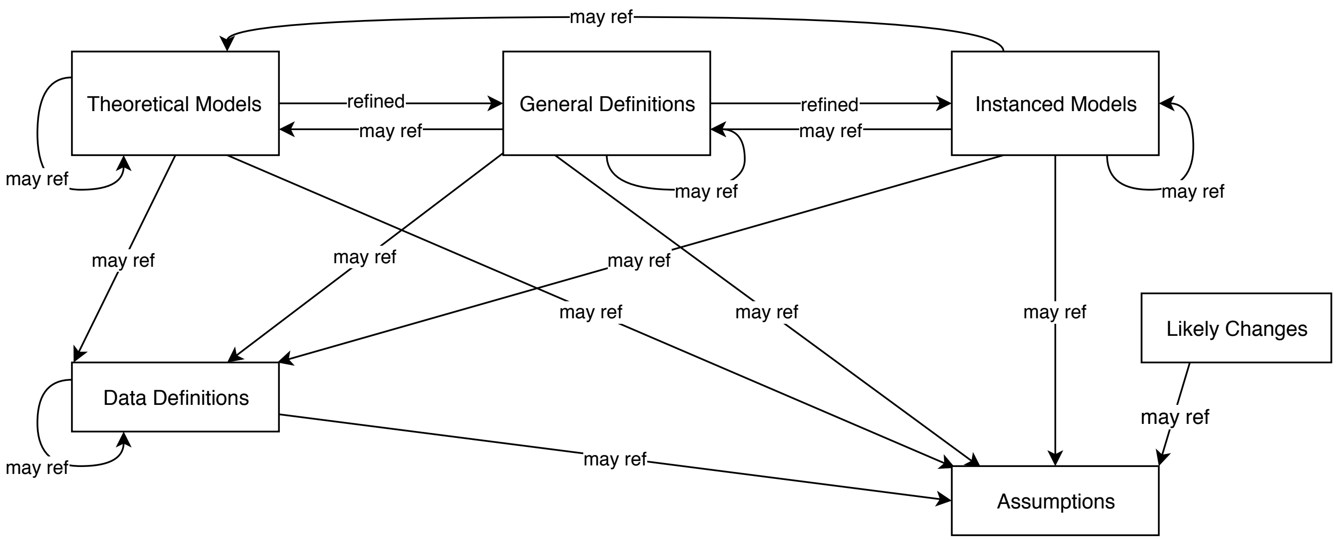
[This section presents the solution characteristics by successively refining models. It starts with the abstract/general Theoretical Models (TMs) and refines them to the concrete/specific Instance Models (IMs). If necessary there are intermediate refinements to General Definitions (GDs). All of these refinements can potentially use Assumptions (A) and Data Definitions (DD). TMs are refined to create new models, that are called GMs or IMs. DDs are not refined; they are just used. GDs and IMs are derived, or refined, from other models. DDs are not derived; they are just given. TMs are also just given, but they are refined, not used. If a potential DD includes a derivation, then that means it is refining other models, which would make it a GD or an IM. —TPLT]

[The above makes a distinction between "refined" and "used." A model is refined to another model if it is changed by the refinement. When we change a general 3D equation to a 2D equation, we are making a refinement, by applying the assumption that the third dimension does not matter. If we use a definition, like the definition of density, we aren't refining, or changing that definition, we are just using it. —TPLT]

[The same information can be a TM in one problem and a DD in another. It is about how the information is used. In one problem the definition of acceleration can be a TM, in another it would be a DD. —TPLT]

[There is repetition between the information given in the different chunks (TM, GDs etc) with other information in the document. For instance, the meaning of the symbols, the units etc are repeated. This is so that the chunks can stand on their own when being read by a reviewer/user. It also facilitates reuse of the models in a different context. —TPLT]

[The relationships between the parts of the document are show in the following figure. In this diagram "may ref" has the same role as "uses" above. The figure adds "Likely Changes," which are able to reference (use) Assumptions. —TPLT]



The instance models that govern ProgName are presented in Subsection 4.2.6. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Assumptions

[The assumptions are a refinement of the scope. The scope is general, where the assumptions are specific. All assumptions should be listed, even those that domain experts know so well that they are rarely (if ever) written down. —TPLT] [The document should not take for granted that the reader knows which assumptions have been made. In the case of unusual assumptions, it is recommended that the documentation either include, or point to, an explanation and justification for the assumption. —TPLT]

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [T], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: [Short description of each assumption. Each assumption should have a meaningful label. Use cross-references to identify the appropriate traceability to T, GD, DD etc., using commands like dref, ddref etc. Each assumption should be atomic - that is, there should not be an explicit (or implicit) "and" in the text of an assumption. —TPLT]

4.2.2 Theoretical Models

[Theoretical models are sets of abstract mathematical equations or axioms for solving the problem described in Section "Physical System Description" (Section 4.1.2). Examples of theoretical models are physical laws, constitutive equations, relevant conversion factors, etc. —TPLT]

This section focuses on the general equations and laws that ProgName is based on. [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

RefName: **T:COE**

Label: Conservation of thermal energy

Equation: $-\nabla \cdot \mathbf{q} + g = \rho C \frac{\partial T}{\partial t}$

Description: The above equation gives the conservation of energy for transient heat transfer in a material of specific heat capacity C ($\mathrm{J\,kg^{-1}\,{}^{\circ}C^{-1}}$) and density ρ ($\mathrm{kg\,m^{-3}}$), where $\mathbf{q}$ is the thermal flux vector ($\mathrm{W\,m^{-2}}$), g is the volumetric heat generation ($\mathrm{W\,m^{-3}}$), T is the temperature (°C), t is time (s), and ∇ is the gradient operator. For this equation to apply, other forms of energy, such as mechanical energy, are assumed to be negligible in the system (A**??**). In general, the material properties (ρ and C) depend on temperature.

Notes: None.

Source: http://www.efunda.com/formulae/heat_transfer/conduction/overview_cond.cfm

Ref. By: GD**??**

Preconditions for T:COE: None

Derivation for T:COE: Not Applicable

[“Ref. By” is used repeatedly with the different types of information. This stands for Referenced By. It means that the models, definitions and assumptions listed reference the current model, definition or assumption. This information is given for traceability. Ref. By provides a pointer in the opposite direction to what we commonly do. You still need to have a reference in the other direction pointing to the current model, definition or assumption. As an example, if T1 is referenced by G2, that means that G2 will explicitly include a reference to T1. —TPLT]

4.2.3 General Definitions

[General Definitions (GDs) are a refinement of one or more TMs, and/or of other GDs. The GDs are less abstract than the TMs. Generally the reduction in abstraction is possible through invoking (using/referencing) Assumptions. For instance, the TM could be Newton’s

Law of Cooling stated abstracting. The GD could take the general law and apply it to get a 1D equation. —TPLT]

This section collects the laws and equations that will be used in building the instance models.

[Some projects may not have any content for this section, but the section heading should be kept. —TPLT] [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Number	GD1
Label	**Newton's law of cooling**
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q(t) = h\Delta T(t)$
Description	Newton's law of cooling describes convective cooling from a surface. The law is stated as: the rate of heat loss from a body is proportional to the difference in temperatures between the body and its surroundings. $q(t)$ is the thermal flux ($\mathrm{W\,m^{-2}}$). h is the heat transfer coefficient, assumed independent of T (A**??**) ($\mathrm{W\,m^{-2}\,{}^\circ C^{-1}}$). $\Delta T(t) = T(t) - T_{\text{env}}(t)$ is the time-dependent thermal gradient between the environment and the object (°C).
Source	Citation here
Ref. By	DD1, DD**??**

Detailed derivation of simplified rate of change of temperature

[This may be necessary when the necessary information does not fit in the description field. —TPLT] [Derivations are important for justifying a given GD. You want it to be clear where the equation came from. —TPLT]

4.2.4 Data Definitions

[The Data Definitions are definitions of symbols and equations that are given for the problem. They are not derived; they are simply used by other models. For instance, if a problem depends on density, there may be a data definition for the equation defining density. The DDs are given information that you can use in your other modules. —TPLT]

[All Data Definitions should be used (referenced) by at least one other model. —TPLT]

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Number	DD1
Label	**Heat flux out of coil**
Symbol	q_C
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q_C(t) = h_C(T_C - T_W(t))$, over area A_C
Description	T_C is the temperature of the coil (°C). T_W is the temperature of the water (°C). The heat flux out of the coil, q_C ($\mathrm{W\,m^{-2}}$), is found by assuming that Newton's Law of Cooling applies (A**??**). This law (GD1) is used on the surface of the coil, which has area A_C ($\mathrm{m^2}$) and heat transfer coefficient h_C ($\mathrm{W\,m^{-2}\,{}^\circ C^{-1}}$). This equation assumes that the temperature of the coil is constant over time (A**??**) and that it does not vary along the length of the coil (A**??**).
Sources	Citation here
Ref. By	IM1

4.2.5 Data Types

[This section is optional. In many scientific computing programs it isn't necessary, since the inputs and outpus are straightforward types, like reals, integers, and sequences of reals and integers. However, for some problems it is very helpful to capture the type information. —TPLT]

[The data types are not derived; they are simply stated and used by other models. —TPLT]

[All data types must be used by at least one of the models. —TPLT]

[For the mathematical notation for expressing types, the recommendation is to use the notation of Hoffman and Strooper (1995). —TPLT]

This section collects and defines all the data types needed to document the models. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Type Name	Name for Type
Type Def	mathematical definition of the type
Description	description here
Sources	Citation here, if the type is borrowed from another source

4.2.6 Instance Models

[The motivation for this section is to reduce the problem defined in "Physical System Description" (Section 4.1.2) to one expressed in mathematical terms. The IMs are built by refining the TMs and/or GDs. This section should remain abstract. The SRS should specify the requirements without considering the implementation. —TPLT]

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.4 to replace the abstract symbols in the models identified in Sections 4.2.2 and 4.2.3.

The goals [reference your goals —TPLT] are solved by [reference your instance models —TPLT]. [other details, with cross-references where appropriate. —TPLT] [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

Number	IM1
Label	**Energy balance on water to find T_W**
Input	m_W, C_W, h_C, A_C, h_P, A_P, t_{final}, T_C, T_{init}, $T_P(t)$ from IM**??** The input is constrained so that $T_{\text{init}} \leq T_C$ (A**??**)
Output	$T_W(t)$, $0 \leq t \leq t_{\text{final}}$, such that $\frac{dT_W}{dt} = \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))]$, $T_W(0) = T_P(0) = T_{\text{init}}$ (A**??**) and $T_P(t)$ from IM**??**
Description	T_W is the water temperature (°C). T_P is the PCM temperature (°C). T_C is the coil temperature (°C). $\tau_W = \frac{m_W C_W}{h_C A_C}$ is a constant (s). $\eta = \frac{h_P A_P}{h_C A_C}$ is a constant (dimensionless). The above equation applies as long as the water is in liquid form, $0 < T_W < 100^o$C, where 0^oC and 100^oC are the melting and boiling points of water, respectively (A**??**, A**??**).
Sources	Citation here
Ref. By	IM**??**

Derivation of ...

[The derivation shows how the IM is derived from the TMs/GDs. In cases where the derivation cannot be described under the Description field, it will be necessary to include this subsection. —TPLT]

4.2.7 Input Data Constraints

Table 1 shows the data constraints on the input output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The column for software constraints restricts the range of inputs to reasonable values. The software constraints will be helpful in the design stage for picking suitable algorithms. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise.

The specification parameters in Table 1 are listed in Table 2.

Table 1: Input Variables

Var	**Physical Constraints**	**Software Constraints**	**Typical Value**	**Uncertainty**
L	$L > 0$	$L_{\text{min}} \leq L \leq L_{\text{max}}$	1.5 m	10%

(*) [you might need to add some notes or clarifications —TPLT]

Table 2: Specification Parameter Values

Var	**Value**
L_{min}	0.1 m

4.2.8 Properties of a Correct Solution

A correct solution must exhibit [fill in the details —TPLT]. [These properties are in addition to the stated requirements. There is no need to repeat the requirements here. These additional properties may not exist for every problem. Examples include conservation laws (like conservation of energy or mass) and known constraints on outputs, which are usually summarized in tabular form. A sample table is shown in Table 3 —TPLT]

Table 3: Output Variables

Var	**Physical Constraints**
T_W	$T_{\text{init}} \leq T_W \leq T_C$ (by A**??**)

[This section is not for test cases or techniques for verification and validation. Those topics will be addressed in the Verification and Validation plan. —TPLT]

5 Functional Requirements

This section provides the functional requirements of the machine, seperated into tray dispensing, pot dispensing, conveyor, and verification requierements.

5.1 Tray Dispensing Functional Requirements

TDR1: The tray dispenser must be able to place a standard tray from the loaded stack onto the conveyor belt.

TDR2: The tray dispenser must be able to store multiple trays at a time.

TDR3: The tray dispenser must be able to take a maximum of 30 trays from the user, such that $n_{trays}(t) \leq 30$

This will allow the machine to meet the goal of reloading trays every 15 minutes.

TDR4: The tray dispenser must dispense trays at a rate of 1 tray every 30 seconds, such that $t_{cycle} = 30\ seconds\ \&\ n_{trays}\,(t) = 300 - t/3;\ t\ \varepsilon\ \{30, 60, 90, ..., 900\}$

THIS IS NFR

This will allow the machine to meet the output goal required.

TDR5: The tray dispenser must stop operating once it has run out of trays.

TDR6: The tray dispenser must be able to recognize when trays need to be reloaded and notify an operator.

TDR7: The tray dispenser must stop when an error arises (verification failed, pot dispenser malfunction/empty, conveyor malfunction).

TDR8: The tray dispenser must be able to reset after it has been stopped due to error or needs to reload

5.2 Pot Dispensing Functional Requirements

PDR1: The pot dispenser must be able to place a pot into an empty place in a tray.

PDR2: The pot dispenser must be able to dispense a 4" diameter pot.

PDR3: The pot dispenser must be able to store multiple pots at a time.

PDR4: The pot dispenser must stop operating once it has run out of pots.

PDR5: The pot dispenser must be able to recognize when pots need to be reloaded and notify an operator.

PDR6: The pot dispenser must stop when an error arises (verification failed, tray dispenser malfunction/empty, conveyor malfunction).

PDR7: The pot dispenser must be able to reset after it has been stopped due to an error or needs to reload.

5.3 Conveyor Requirements

CR1: The conveyor must be able to move a tray from the tray dispensing station to the verification station.

CR2: The conveyor must be able to stop when a tray is being placed on the conveyor, or when a pot is being placed in a tray.

CR3: The conveyor must be able to move pots at a speed sufficient to meet the goal of one cycle per 30 seconds NFR

CR4: The conveyor material must have a high enough friction such that the trays do not slide while in contact with the belt.

CR5: The conveyor must stop when an error arises (verification failed, pot dispenser malfunction/empty, tray dispenser malfunction/empty).

5.4 Verification Requirements

VR1: The verification must check that only one tray is being output by the conveyer belt, and the tray is filled with exactly 10 pots, such that $n_{traysout} = 1$ & $n_{potsout} = 10$

VR2: The verification must notify all other systems if verification failes, such that $n_{traysout} \neq 1$ *or* $n_{potsout} \neq 10$

6 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 dispenser, pot dispenser and verification.

6.1 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.

6.2 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.

6.3 Learning Requirments

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

6.4 Accessibility Requirements

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

6.5 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.

NFR9: The pot dispenser must dispense pots at a rate of 10 pots every 30 seconds, such that $t_{cycle} = 30\ seconds\ \&\ n_{pots}\,(t) = 300 - t/3;\ t\ \varepsilon\ \{30, 60, 90, ..., 900\}$ This will allow the machine to meet the output goal required.

6.6 Safety Critical Requirements

NFR10: System must hgave emergency cut off. in case of any emergency this will trip off all power to system.

NFR11: System must be able to locate and identify failures within each independent subsystem.

6.7 Precision Requirements

NFR12: 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.

6.8 Capacity Requirements

NFR13: The pot dispenser must be able to take a maximum of 30 pots from the user, such that $n_{pots}(t) \leq 300$ This will allow the machine to meet the output goal required.

NFR14: Pot dispensor should dispence pots within a 0.5 cm radius of centered position.

6.9 Reliability Requirements

NFR15: System must be able to opperate under constant low and high frequency vibration (small amplitude).

6.10 Robustness and Fault Tolerance Requirements

NFR16: Components must withstand 250,000 cycles system should call for replacement parts after a full season of opperation.

6.11 Scalability or Extensibility Requirements

NFR17: System must fit into the existing assembly line and should take minimal effort to implement ($< 1d$).

6.12 Expected Physical Environement Requirements

NFR18: System must withstand operating in a room with high arial polution including dust, dirt and small amounts of water.

6.13 Maitenance Requirements

NFR19: System must be built for ease of maitenance. high wear parts should be easily accessible.

6.14 Requirements for Interacting with Adjacent Systems

NFR20: system must opperate at same speed as adjacent systems.

NFR21: System must opperate at the same conveyor height as the current system to maintain continuity between systems.

6.15 Supportability Requirements

NFR22: System documentation must be available to troubleshoot, diagnose and fix common issues and replace high wear parts.

6.16 Compliance Requirements

NFR23: System must follow electronics system safety requirements.

7 Likely Changes

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

8 Unlikely Changes

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

9 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 4 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 5 shows the dependencies of instance models, requirements, and data constraints on each other. Table 6 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 the head of that arrow. Therefore, if a component is changed, the components that it points

	T??	T??	T??	GD1	GD??	DD1	DD??	DD??	DD??	IM1	IM??	IM??	IM??
T??													
T??			X										
T??													
GD1													
GD??	X												
DD1				X									
DD??				X									
DD??													
DD??								X					
IM1					X	X	X				X		
IM??					X		X		X	X			X
IM??		X											
IM??		X	X				X	X	X		X		

Table 4: Traceability Matrix Showing the Connections Between Items of Different Sections

	IM1	IM??	IM??	IM??	4.2.7	R??	R??
IM1		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 5: 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??																			
GD1		X																	
GD??			X	X	X	X													
DD1							X	X	X										
DD??			X	X						X									
DD??																			
DD??																			
IM1											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 6: Traceability Matrix Showing the Connections Between Assumptions and Other Items

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.

10 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]

11 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]

References

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

W. Spencer Smith. Systematic development of requirements documentation for general purpose scientific computing software. In *Proceedings of the 14th IEEE International Requirements Engineering Conference, RE 2006*, pages 209–218, Minneapolis / St. Paul, Minnesota, 2006. URL http://www.ifi.unizh.ch/req/events/RE06/.

W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Ȧgerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13(1):83–107, February 2007.

W. Spencer Smith, John McCutchan, and Jacques Carette. Commonality analysis for a family of material models. Technical Report CAS-17-01-SS, McMaster University, Department of Computing and Software, 2017.

[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?