

Software Requirements Specification for 2D Localizer

Aliyah Jimoh

February 5, 2025

Contents

1	Reference Material	iv
1.1	Table of Units	iv
1.2	Table of Symbols	iv
1.3	Abbreviations and Acronyms	v
1.4	Mathematical Notation	v
2	Introduction	1
2.1	Purpose of Document	1
2.2	Scope of Requirements	1
2.3	Characteristics of Intended Reader	1
2.4	Organization of Document	2
3	General System Description	2
3.1	System Context	2
3.2	User Characteristics	3
3.3	System Constraints	3
4	Specific System Description	3
4.1	Problem Description	3
4.1.1	Terminology and Definitions	3
4.1.2	Physical System Description	3
4.1.3	Goal Statements	4
4.2	Solution Characteristics Specification	4
4.2.1	Types	5
4.2.2	Scope Decisions	5
4.2.3	Modelling Decisions	5
4.2.4	Assumptions	5
4.2.5	Theoretical Models	6
4.2.6	General Definitions	8
4.2.7	Data Definitions	9
4.2.8	Data Types	10
4.2.9	Instance Models	11
4.2.10	Input Data Constraints	14
4.2.11	Properties of a Correct Solution	15
5	Requirements	15
5.1	Functional Requirements	16
5.2	Nonfunctional Requirements	16
5.3	Rationale	17
6	Likely Changes	17

7	Unlikely Changes	17
8	Traceability Matrices and Graphs	17
9	Development Plan	18
10	Values of Auxiliary Constants	21

Revision History

Date	Version	Notes
2025/02/05	1.0	Initial Draft
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
s	time	second
°	angle	degrees

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The symbols are listed in alphabetical order.

[Remeber to organize them in order —TPLT]

symbol	unit	description
i	—	Index for robot’s poses
j	—	Index for number of beacons
η_j	—	Sensor noise
n	—	Index for number of FMs
S	—	Set of beacon coordinates
$g_j(x_i)$	m	Noisy range measurement of the robot’s i th taken from j th beacon
A	m ²	Area of the 2D environment
F	—	Coordinates of FMs
σ_j^2	m ²	Noise variance for j th beacon
\tilde{d}_j	m	Measured distance with noise
$C(x_i)$	m	Cost Function
I	m ⁻²	Total Fisher Information Matrix
θ	deg	Robot angle
T_{robot}	m ²	surface area over which heat is transferred in
\tilde{D}	m ²	coil surface area

d_j	m^2	surface area over which heat is transferred in
T_{env}	m^2	coil surface area
A_{in}	m^2	surface area over which heat is transferred in

1.3 Abbreviations and Acronyms

Abbreviation/Acronym	Definition
2D	Two-Dimensional
A	Assumption
CRLB	Cramér-Rao Lower Bound
DD	Data Definition
FIM	Fisher Information Matrix
FM	Fiducial Marker
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
Localizer	2D Localization Solution
PS	Physical System Description
R	Requirement
SE(2)	Special Euclidean Group in 2D
SRS	Software Requirements Specification
TM	Theoretical Model

1.4 Mathematical Notation

Throughout this document, there will be typographic conventions as well as mathematical operators that are used to distinguish different variables and operations.

Variable	Definition	Description
A	Matrix	Bold capital letter
a	Vector	Bold lowercase letter
<i>a</i>	Scalar	Lowercase letter

2 Introduction

Mobile robots have been used to traverse areas with various hazards, help collect data whether through its sensors or obtaining samples, and overall complete challenging tasks. Due to their autonomy, there is no need to constantly monitor them as they have their own means with interacting with the environment. However, it could raise some concern when there is no reliable method to track their movements, especially if they are placed in a vastly large area or places that may be difficult to access once they are operational. Risks such as having difficulty retrieving them if there is a malfunction keeping them from returning or possible collisions in the area or with other robots are reasons one would want to find a way to locate them as they carry out their tasks. The program being documented, 2D Localizer, proposes to solve this problem by developing a 2D localization solution that can implement various sensors to accurately localize the robots as they traverse the map provided.

The following section provides an overview of the Software Requirement Specification (SRS) for 2D Localizer. This section explains the purpose of this document, the scope of the requirements, the characteristics of the intended reader, and the organization of the document.

2.1 Purpose of Document

The main purpose of this document is to describe the requirements needed to run the localizer. Information such as the constraints, assumptions and theoretical models used will be provided to help readers get a better understanding of the purpose and computations of 2D Localizer. Therefore, it can be used as a reference guide on how to plan and set up the requirements needed by the user to get the desired and accurate results.

2.2 Scope of Requirements

The scope of the requirements includes the robot analyzed being in a controlled environment to help with potential lighting problems for vision sensors used. The sensors used on the robot and environment are specified for modelling purposes. Some models can include a sum through sensors having independent measurements. The noise from the sensors will be considered zero-mean Gaussian for simplicity in calculations.

2.3 Characteristics of Intended Reader

Reviewers of this documentation should have taken a graduate course on linear algebra, estimation theory or matrix computation. They should also have some knowledge in undergraduate statistics and probability.

2.4 Organization of Document

The organization of this document follows the template for an SRS for scientific computing software proposed by [Smith and Lai \(2005\)](#), [Smith et al. \(2007\)](#), and [Smith and Koothoor \(2016\)](#). The presentation follows the standard pattern of presenting goals, theories, definitions, and assumptions. For readers that would like a more bottom up approach, they can start reading the data definitions and trace back to find any additional information they require.

The goal statements are refined to the theoretical models and the theoretical models to the instance models. The data definitions are used to support the definitions of the different models.

3 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.

3.1 System Context

The system context is displayed in [Figure 1](#) below. The circles represent the external aspects related to the software which are the users. The rectangle represents the software system being used (2D Localizer) and the arrows explain what information is being passed between the user and the software.

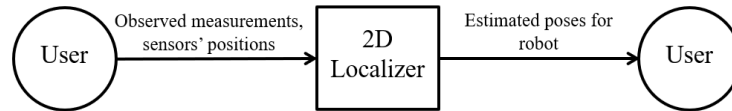


Figure 1: System Context

- User Responsibilities:
 - Provide inputs including the coordinates of each environmental sensor and the measurements taken from them.
 - Evaluate the inputs to ensure all are their respective types
- ProgName Responsibilities:
 - Detect data type mismatch, such as a string of characters instead of a floating point number
 - Estimate the position and orientation of the robot

3.2 User Characteristics

The end user of 2D Localizer should have some familiarity with types of range sensors along with how to read and collect their data. They should also have basic experience with programming.

3.3 System Constraints

There are no system constraints.

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.

4.1 Problem Description

2D Localizer is intended to help keep track of mobile robots while carrying out their tasks.

4.1.1 Terminology and Definitions

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:

- **Pose:** Position and orientation of the robot.
- **Localization:** Determining where an object is with respect to its environment.
- **Fiducial Markers:** Markers placed around the environment for the robot to determine its location.
- **Beacon:** A range sensor

4.1.2 Physical System Description

The physical system of the localizer includes the following elements:

PS1: The mobile robot

PS2: The beacons placed in the environment

PS3: The camera sensors on the mobile robot

PS4: The fiducial markers in the environment

4.1.3 Goal Statements

Given the imported 2D map, coordinates of all sensors and fiducial markers in the environment, and the noisy measurements of the sensors, the goal statements are:

GS1: Calculate the estimated pose and error of the robot throughout its trajectory from both sensors (environment and robot).

GS2: Display a visual representation of the robot traversing the 2-D map with the sensors placed.

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]

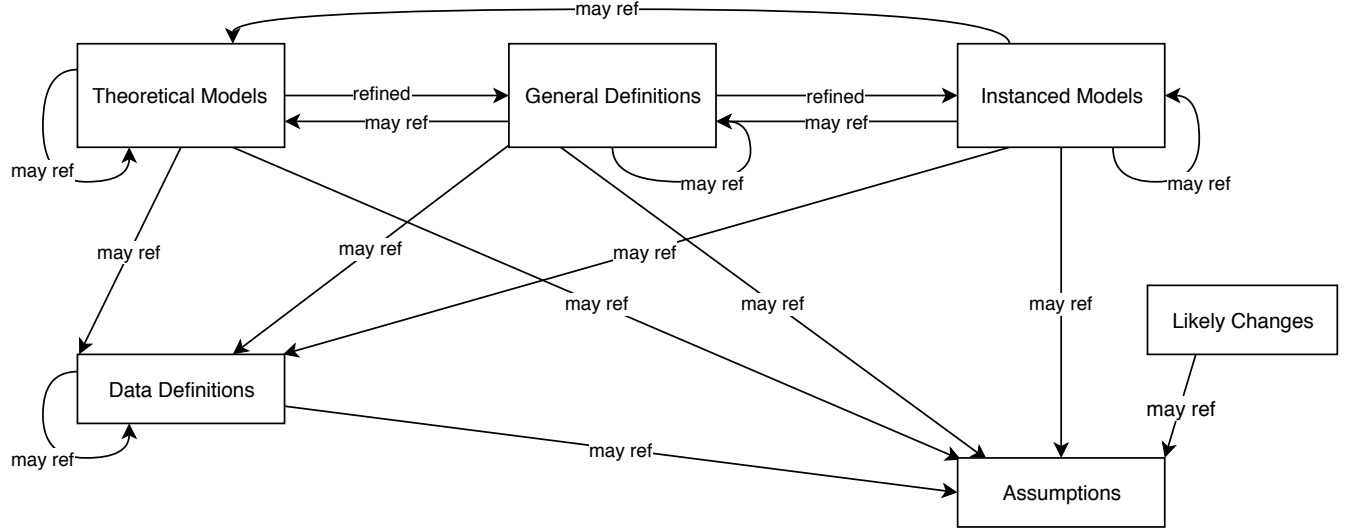
[This section presents the solution characteristics by successively refining models. It starts with the abstract/general Theoretical Models (TMs) and refines them to the concrete/specific Instance Models (IMs). If necessary there are intermediate refinements to General Definitions (GDs). All of these 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 4.2.9. 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 Types

[This section is optional. Defining types can make the document easier to understand. — TPLT]

4.2.2 Scope Decisions

[This section is optional. — TPLT]

4.2.3 Modelling Decisions

[This section is optional. — TPLT]

4.2.4 Assumptions

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 [TM], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1 : Robot uses camera sensors while the environment uses beacons and fiducial markers

A2 : Localizer is used in a controlled environment (i.e., indoors)

A1 : Each sensor has independent measurements

A1 : Sensor noise is zero-mean Gaussian

4.2.5 Theoretical Models

This section focuses on the general equations and laws that 2D Localizer is based on.

Number	TM1
Label	Noisy Range Measurement
Equation	$g_j(x) = \ \mathbf{x}_i - \mathbf{a}_j\ + \eta_j$
Description	The equation above gives the noisy range measurement g_j (m) of the beacons placed in the environment (A1) where \mathbf{x}_i is the position of the robot (m), \mathbf{a}_j is the position of the jth beacon placed (m), and η_j is the noise from the jth beacon (m) (A4).
Source	Citation here
Ref. By	DD??, DD??
Preconditions	
Derivation	

Number	TM2
Label	Cost Function
Equation	$C(x) = \sum_{j=1}^N \frac{1}{\sigma_j^2} \left(\tilde{d}_j - \ \mathbf{x}_i - \mathbf{a}_j\ \right)^2$
Description	<p>The above equation gives the conservation of energy for transient heat transfer in a material of specific heat capacity C ($\text{J kg}^{-1} \text{ }^\circ\text{C}^{-1}$) and density ρ (kg m^{-3}), where \mathbf{q} is the thermal flux vector (W m^{-2}), g is the volumetric heat generation (W m^{-3}), T is the temperature ($^\circ\text{C}$), 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.</p>
Source	http://www.efunda.com/formulae/heat_transfer/conduction/overview_cond.cfm
Ref. By	DD??, DD??
Preconditions	
Derivation	

Number	TM3
Label	Fisher Information Matrix
Equation	$\mathbf{I}_j \cong \frac{1}{\sigma_j^2} \left(\frac{\hat{x} - a_j}{\ \hat{x} - a_j\ } \right) \left(\frac{\hat{x} - a_j}{\ \hat{x} - a_j\ } \right)^T$
Description	<p>This equation gives the MLE which estimates the best position \hat{x} (m) to align based on the range measurements given.</p>
Source	Citation here
Ref. By	DD??, DD??
Preconditions	
Derivation	

Number	TM4
Label	Cramér-Rao Lower Bound
Equation	$Var(\hat{\theta}) \geq \frac{1}{I(\theta)}$
Description	This equation gives the MLE which estimates the best position \hat{x} (m) to align based on the range measurements given.
Source	Citation here
Ref. By	DD??, DD??
Preconditions	GD??
Derivation	

Number	TM5
Label	SE(2) Transformation
Equation	$T_{robot} = \begin{bmatrix} R(\theta) & t \\ 0 & 1 \end{bmatrix}$
Description	This equation gives the MLE which estimates the best position \hat{x} (m) to align based on the range measurements given.
Source	Citation here
Ref. By	DD??, DD??
Preconditions	
Derivation	

4.2.6 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	Probability Density Function
SI Units	m^{-1}
Equation	$p_j(\tilde{d}_j x) = \mathcal{N}(\ x - a_j\ , \sigma_j^2)$
Description	<p>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.</p> <p>$q(t)$ is the thermal flux (W m^{-2}).</p> <p>h is the heat transfer coefficient, assumed independent of T (A??) ($\text{W m}^{-2} \text{ }^\circ\text{C}^{-1}$).</p> <p>$\Delta T(t) = T(t) - T_{\text{env}}(t)$ is the time-dependent thermal gradient between the environment and the object ($^\circ\text{C}$).</p>
Source	Citation here
Ref. By	DD??, 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]

4.2.7 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	Actual Range Measurement
Symbol	\tilde{d}_j
SI Units	m
Equation	$\tilde{d}_j = \ \mathbf{x}_i - \mathbf{a}_j\ + \eta_j$
Description	T_C is the temperature of the coil ($^{\circ}\text{C}$). T_W is the temperature of the water ($^{\circ}\text{C}$). The heat flux out of the coil, q_C (W m^{-2}), is found by assuming that Newton’s Law of Cooling applies (A??). This law (GD??) is used on the surface of the coil, which has area A_C (m^2) and heat transfer coefficient h_C ($\text{W m}^{-2} ^{\circ}\text{C}^{-1}$). 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	TM??

4.2.8 Data Types

[This section is optional. In many scientific computing programs it isn’t necessary, since the inputs and output 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

4.2.9 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 4.2.7 to replace the abstract symbols in the models identified in Sections 4.2.5 and 4.2.6.

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	Full Likelihood Function
Input	\tilde{D}, N
Output	$p(\tilde{D} x)$
Equation	$p\left(\tilde{D} x\right)=\prod_{j=1}^N p\left(\tilde{d}_j x\right)$
Description	<p>T_W is the water temperature ($^{\circ}\text{C}$).</p> <p>T_P is the PCM temperature ($^{\circ}\text{C}$).</p> <p>T_C is the coil temperature ($^{\circ}\text{C}$).</p> <p>$\tau_W = \frac{m_W C_W}{h_C A_C}$ is a constant (s).</p> <p>$\eta = \frac{h_P A_P}{h_C A_C}$ is a constant (dimensionless).</p> <p>The above equation applies as long as the water is in liquid form, $0 < T_W < 100^{\circ}\text{C}$, where 0°C and 100°C are the melting and boiling points of water, respectively (A??, A??).</p>
Sources	Citation here
Ref. By	IM??

Number	IM2
Label	Maximum Likelihood Estimation
Input	$p(\tilde{D} x)$
Output	\hat{x}
Equation	$\hat{x} = \arg \min p(\tilde{D} x)$
Description	<p>T_W is the water temperature ($^{\circ}\text{C}$).</p> <p>T_P is the PCM temperature ($^{\circ}\text{C}$).</p> <p>T_C is the coil temperature ($^{\circ}\text{C}$).</p> <p>$\tau_W = \frac{m_W C_W}{h_C A_C}$ is a constant (s).</p> <p>$\eta = \frac{h_P A_P}{h_C A_C}$ is a constant (dimensionless).</p> <p>This equation gives the MLE which estimates the best position \hat{x} (m) to align based on the range measurements given.</p>
Sources	Citation here
Ref. By	IM??

Number	IM3
Label	Maximum Likelihood Estimation
Input	$p(\tilde{D} x)$
Output	\hat{x}
Equation	$\hat{x} = \arg \min p(\tilde{D} x)$
Description	<p>T_W is the water temperature (°C).</p> <p>T_P is the PCM temperature (°C).</p> <p>T_C is the coil temperature (°C).</p> <p>$\tau_W = \frac{m_W C_W}{h_C A_C}$ is a constant (s).</p> <p>$\eta = \frac{h_P A_P}{h_C A_C}$ is a constant (dimensionless).</p> <p>The above equation applies as long as the water is in liquid form, $0 < T_W < 100^\circ\text{C}$, where 0°C and 100°C are the melting and boiling points of water, respectively (A??, A??).</p>
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]

4.2.10 Input Data Constraints

Table 5 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 5 are listed in Table 7.

Table 5: Input Variables

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

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

Table 7: Specification Parameter Values

Var	Value
L_{\min}	0.1 m

4.2.11 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 9 —TPLT]

Table 9: 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 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

- 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]

5.2 Nonfunctional Requirements

[List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. —TPLT] [The goal is for the nonfunctional requirements to be unambiguous, abstract and verifiable. This isn't easy to show succinctly, so a good strategy may be to give a "high level" view of the requirement, but allow for the details to be covered in the Verification and Validation document. —TPLT] [An absolute requirement on a quality of the system is rarely needed. For instance, an accuracy of 0.0101 % is likely fine, even if the requirement is for 0.01 % accuracy. Therefore, the emphasis will often be more on describing how well the quality is achieved, through experimentation, and possibly theory, rather than meeting some bar that was defined a priori. —TPLT] [You do not need an entry for correctness in your NFRs. The purpose of the SRS is to record the requirements that need to be satisfied for correctness. Any statement of correctness would just be redundant. Rather than discuss correctness, you can characterize how far away from the correct (true) solution you are allowed to be. This is discussed under accuracy. —TPLT]

- NFR1: **Accuracy** [Characterize the accuracy by giving the context/use for the software. Maybe something like, "The accuracy of the computed solutions should meet the level needed for <engineering or scientific application>. The level of accuracy achieved by ProgName shall be described following the procedure given in Section X of the Verification and Validation Plan." A link to the VnV plan would be a nice extra. —TPLT]
- NFR2: **Usability** [Characterize the usability by giving the context/use for the software. You should likely reference the user characteristics section. The level of usability achieved by the software shall be described following the procedure given in Section X of the Verification and Validation Plan. A link to the VnV plan would be a nice extra. —TPLT]

- NFR3: **Maintainability** [The effort required to make any of the likely changes listed for ProgName should be less than FRACTION of the original development time. FRACTION is then a symbolic constant that can be defined at the end of the report. —TPLT]
- NFR4: **Portability** [This NFR is easier to write than the others. The systems that ProgName should run on should be listed here. When possible the specific versions of the potential operating environments should be given. To make the NFR verifiable a statement could be made that the tests from a given section of the VnV plan can be successfully run on all of the possible operating environments. —TPLT]
- Other NFRs that might be discussed include verifiability, understandability and reusability.

5.3 Rationale

[Provide a rationale for the decisions made in the documentation. Rationale should be provided for scope decisions, modelling decisions, assumptions and typical values. —TPLT]

6 Likely Changes

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

7 Unlikely Changes

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

8 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 11 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 12 shows the dependencies of instance models, requirements, and data constraints on each other. Table 13 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]

	TM??	TM??	TM??	GD??	GD??	DD??	DD??	DD??	DD??	IM??	IM??	IM??
TM??												
TM??			X									
TM??												
GD??												
GD??	X											
DD??				X								
DD??				X								
DD??												
DD??								X				
IM??					X	X	X				X	
IM??					X		X		X	X		
IM??		X										
IM??		X	X				X	X	X		X	

Table 11: Traceability Matrix Showing the Connections Between Items of Different Sections

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 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.

9 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]

	IM??	IM??	IM??	IM??	4.2.10	R??	R??
IM??		X				X	X
IM??	X			X		X	X
IM??						X	X
IM??		X				X	X
R??							
R??						X	
R??					X		
R2	X	X				X	X
R??	X						
R??		X					
R??			X				
R??				X			
R4			X	X			
R??		X					
R??		X					

Table 12: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	A1	A2	A3	A4
TM1	X			
TM2				
TM3				
TM4				
TM5				
GD1		X		
DD1				
IM2				
IM3				
LC??				
LC??				
LC??				
LC??				
LC??				
LC??				

Table 13: Traceability Matrix Showing the Connections Between Assumptions and Other Items

10 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 Nirmitha Koothoor. A document-driven method for certifying scientific computing software for use in nuclear safety analysis. *Nuclear Engineering and Technology*, 48(2):404–418, April 2016. ISSN 1738-5733. doi: <http://dx.doi.org/10.1016/j.net.2015.11.008>. URL <http://www.sciencedirect.com/science/article/pii/S1738573315002582>.
- W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Ågerfalk, 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 L^AT_EX advice:

- For Mac users *.DS_Store should be in .gitignore
- L^AT_EX 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]