# Software Requirements Specification for Tandem: subtitle describing software

Team 11, Tandem
Justin Kwinecki
Zayed Umer
William Pol
Manav Patel
Musab Mondal

November 3, 2026

# 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** **2**
- 2.1 Purpose of Document . . . 2
- 2.2 Scope of Requirements . . . 2
- 2.3 Characteristics of Intended Reader . . . 3
- 2.4 Organization of Document . . . 3

**3 General System Description** **3**
- 3.1 System Context . . . 3
- 3.2 User Characteristics . . . 5
- 3.3 System Constraints . . . 5

**4 Specific System Description** **5**
- 4.1 Problem Description . . . 5
  - 4.1.1 Terminology and Definitions . . . 5
  - 4.1.2 Physical System Description . . . 6
  - 4.1.3 Goal Statements . . . 6
- 4.2 Solution Characteristics Specification . . . 6
  - 4.2.1 Types . . . 8
  - 4.2.2 Scope Decisions . . . 8
  - 4.2.3 Modelling Decisions . . . 8
  - 4.2.4 Assumptions . . . 8
  - 4.2.5 Theoretical Models . . . 8
  - 4.2.6 General Definitions . . . 10
  - 4.2.7 Data Definitions . . . 11
  - 4.2.8 Data Types . . . 12
  - 4.2.9 Instance Models . . . 13
  - 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 |
|---|---|---|
| Date 1 | 1.0 | Notes |
| Date 2 | 1.1 | Notes |

[This template is intended for use by CAS 741. For CAS 741 the template should be used exactly as given, except the Reflection Appendix can be deleted. For the capstone course it is a source of ideas, but shouldn't be followed exactly. The exception is the reflection appendix. All capstone SRS documents should have a reflection appendix. —TPLT]

# 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 |
|---|---|---|
| $A_C$ | $\mathrm{m^2}$ | coil surface area |
| $A_{\mathrm{in}}$ | $\mathrm{m^2}$ | 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 |
| Tandem | [put an expanded version of your program name here (as appropriate) —TPLT] |
| TM | 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 **???**. 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 **??**. 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 2.3) 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 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]

## 2.1 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 4.1). 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]

## 2.2 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 4.2.4). 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]

[The scope section is relevant for later determining typical values of inputs. The scope should make it clear what inputs are reasonable to expect. This is a distinction between scope and context (context is a later section). Scope affects the inputs while context affects how the software will be used. —TPLT]

### 2.3 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 3.2). 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]

### 2.4 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]

[Include a reference to the template (**???**) you are using in the documentation in the Organization of the Document section. —TPLT]

## 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. [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]

### 3.1 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 $\rightarrow$ Calculations $\rightarrow$ 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 responsibilities 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:
  - 
- Tandem Responsibilities:
  - Detect data type mismatch, such as a string of characters instead of a floating point number
  - 

[Identify in what context the software will typically be used. Is it for exploration? education? engineering work? scientific work?. Identify whether it will be used for mission-critical or safety-critical applications. —TPLT] [This additional context information is needed to determine how much effort should be devoted to the rationale section. If the application is safety-critical, the bar is higher. This is currently less structured, but analogous to, the idea to the Automotive Safety Integrity Levels (ASILs) that McSCert uses in their automotive hazard analyses. —TPLT]

## 3.2 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 2.3. As in Section 2.3, the user characteristics should be specific an unambiguous. For instance, "The end user of Tandem should have an understanding of undergraduate Level 1 Calculus and Physics." —TPLT]

## 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

Tandem 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 elements of the physical system do not have to correspond to an actual physical entity. They can be conceptual. This is particularly important when the documentation is for a numerical method. —TPLT]

The physical system of Tandem, 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 4.2.9). 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]

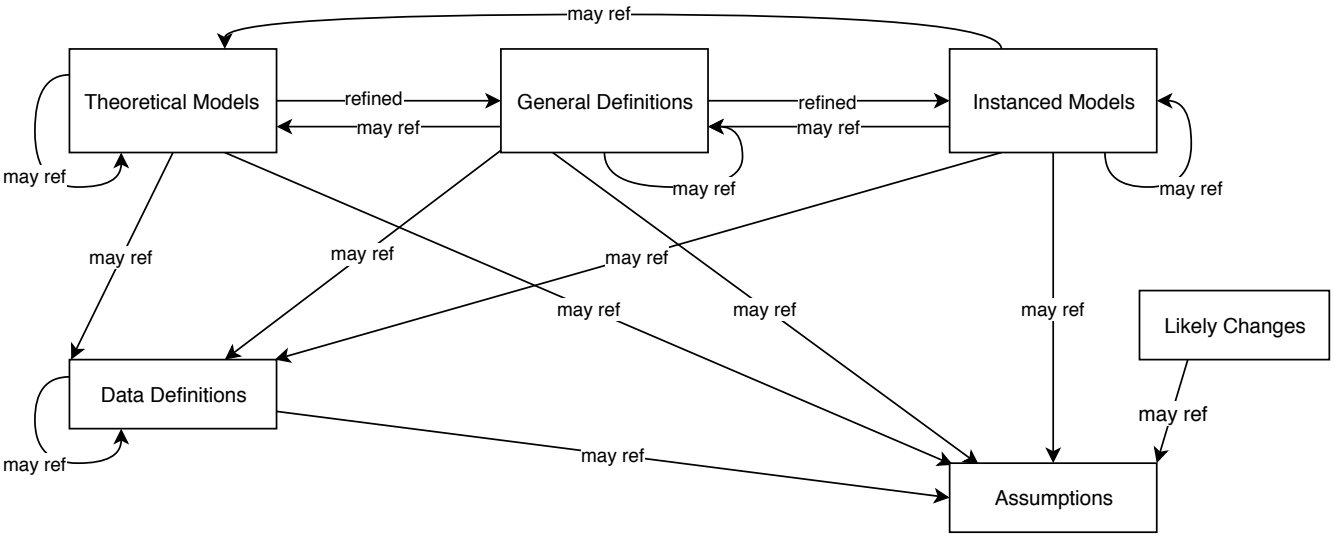
[This section presents the solution characteristics by successively refining models. It starts with the abstract/general Theoretical Models (TMs) and refines them to the concrete/specific Instance Models (IMs). If necessary there are intermediate refinements to General Definitions (GDs). All of these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

[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] [If it helps with the organization and understandability, the assumptions can be presented as sub sections. The following sub-sections are options: background theory assumptions, helper theory assumptions, generic theory assumptions, problem specific assumptions, and rationale assumptions —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 [TM], 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 TM, 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]

### 4.2.5 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]

[Optionally the theory section could be divided into subsections to provide more structure and improve understandability and reusability. Potential subsections include the following: Context theories, background theories, helper theories, generic theories, problem specific theories, final theories and rationale theories. —TPLT]

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

**RefName:** **TM: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 $\rho$ ($\mathrm{kg\,m^{-3}}$), 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 $\nabla$ 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 ($\rho$ and $C$) depend on temperature.

**Notes:** None.

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

**Ref. By:** GD**??**

**Preconditions for TM:COE:** None

**Derivation for TM: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 TM1 is referenced by GD2, that means that GD2 will explicitly include a reference to TM1. —TPLT]

### 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 | **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.<br>$q(t)$ is the thermal flux ($\mathrm{W\,m^{-2}}$).<br>$h$ is the heat transfer coefficient, assumed independent of $T$ (A**??**) ($\mathrm{W\,m^{-2}\,{}^{\circ}C^{-1}}$).<br>$\Delta T(t) = T(t) - T_{\mathrm{env}}(t)$ 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]

### 4.2.7 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 | **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 $h_C$ ($\mathrm{W\,m^{-2}\,°C^{-1}}$). 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 |

### 4.2.8 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 **?**. —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 | **Energy balance on water to find $T_W$** |
| Input | $m_W$, $C_W$, $h_C$, $A_C$, $h_P$, $A_P$, $t_{\text{final}}$, $T_C$, $T_{\text{init}}$, $T_P(t)$ from IM**??**<br>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<br>$\frac{dT_W}{dt} = \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))]$,<br>$T_W(0) = T_P(0) = T_{\text{init}}$ (A**??**) and $T_P(t)$ from IM**??** |
| Description | $T_W$ is the water temperature (°C).<br>$T_P$ is the PCM temperature (°C).<br>$T_C$ is the coil temperature (°C).<br>$\tau_W = \frac{m_W C_W}{h_C A_C}$ is a constant (s).<br>$\eta = \frac{h_P A_P}{h_C A_C}$ is a constant (dimensionless).<br>The above equation applies as long as the water is in liquid form, $0 < T_W < 100^o$C, where $0^o$C and $100^o$C 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]

#### 4.2.10 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 |
|---|---|
| $L_{\text{min}}$ | 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 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 Requirements

[The requirements refine the goal statement. They will make heavy use of references to the instance models. —TPLT]

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 now 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 Tandem 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 Tandem 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 Tandem 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 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]

| | TM?? | TM?? | TM?? | GD1 | GD?? | DD1 | DD?? | DD?? | DD?? | IM1 | IM?? | IM?? |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TM?? | | | | | | | | | | | | |
| TM?? | | | X | | | | | | | | | |
| TM?? | | | | | | | | | | | | |
| GD1 | | | | | | | | | | | | |
| GD?? | X | | | | | | | | | | | |
| DD1 | | | | X | | | | | | | | |
| DD?? | | | | X | | | | | | | | |
| DD?? | | | | | | | | | | | | |
| DD?? | | | | | | | | X | | | | |
| IM1 | | | | | X | X | X | | | | X | |
| IM?? | | | | | 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

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.

| | IM1 | IM?? | IM?? | IM?? | 4.2.10 | R?? | R?? |
|---|---|---|---|---|---|---|---|
| IM1 | | 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 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?? |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TM?? | X | | | | | | | | | | | | | | | | | | |
| TM?? | | | | | | | | | | | | | | | | | | | |
| TM?? | | | | | | | | | | | | | | | | | | | |
| 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

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]

# 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]

[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

[Not required for CAS 741 —SS]

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

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

5. 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.

6. 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?