# Software Requirements Specification for VDisp

Emil Soleymani, Dr. Spencer Smith

July 4, 2022

# Contents

# Revision History

| Date | Version | Notes |
|---|---|---|
| May 24, 2022 | 1.0 | Initial Draft |

# 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 | meter |
| kg | mass | kilogram |
| s | time | second |
| rad | angle | radians |
| $\mathrm{N{=}kg\,m\,s^{-2}}$ | force | newton |
| $\mathrm{Pa{=}N\,m^{-2}}$ | pressure | pascal |
| *Add imperial* units | ... | .. |

## 1.2 Table of Symbols

| Symbol | Description | Units |
|---|---|---|
| $e$ | Void ratio | ? |
| $K_0$ | Coefficient of lateral earth pressure | ? |
| $\sigma_v$ | Vertical stress | meter |
| $\sigma'_v$ | Effective stress | meter |
| $\gamma_w$ | Unit weight of water | $\mathrm{N\,m^{-3}}$ |
| $\phi_s$ | Plane strain angle | rad |

*TODO: Add all constants, arrange in alphabetical order, add caption for table?*

## 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 |
| *VDisp* | Software Settlement Analysis Software |
| T | Theoretical Model |

*Should we add MG to this? Or MIS?*

## 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.

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.

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.

Feel free to change the appearance of the report by modifying the LaTeX commands.

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.

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

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.

The template description comments should be disabled before submitting this document for grading.

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.

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.

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

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.

# 2 Introduction

Our cities and towns are in constant need of new infrastructure due to the expanding and evolving nature of our societies. With architects planning new buildings every day, there is a never ending need for experienced geotechnical engineers to perform the complex task of soil settlement analysis prior to laying down the foundation. The *VDisp* software aims to provide undergraduate Civil Engineering students the tools they need to make the simple analyses expected from them in their studies, while also providing them with visualizations and real-time interaction as a means of educational enhancement.

The following section provides an overview of the Software Requirements Specification (SRS) for the soil settlement analysis software. The developed program will be referred to as *VDisp*. 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 primary purpose of this document is to record the requirements of the *VDisp* software. Goals, assumptions, theoretical models, definitions, and other model derivation information are specified, allowing the reader to fully understand and verify the purpose and scientific basis of *VDisp*. Except for system constraints, this SRS will remain abstract, describing what problem the software is solving, but not how to solve it.

This document will be used as a starting point for subsequent development phases, including writing the design specification and the software verification and validation plan. The design document will show how the requirements are to be realized, including decisions on the numerical algorithms and programming environment. The verification and validation plan will show the steps that will be used to increase confidence in the software documentation and the implementation. Although the SRS fits in a series of documents that follow the so-called waterfall model, the actual development process is not constrained in any way. Even when the waterfall model is not followed, as Parnas and Clements point out parnasClements1986(*TODO: add citation*), the most logical way to present the documentation is still to "fake" a rational design process.

## 2.2 Scope of Requirements

The scope of the requirements includes soil settlement analysis of rectangular slab foundations and long strip footing. This entire document is written assuming homogeneous*(is this* *how you say same material throughout layer?)* soil layers and is restricted to one-dimensional analysis *(for now)*.

## 2.3 Characteristics of Intended Reader

*TODO*

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."

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?

## 2.4 Organization of Document

*TODO*

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.

# 3 General System Description

*TODO*

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.

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.

## 3.1 System Context

*TODO*

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

*TODO*

Figure 1: System Context

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

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

## 3.2 User Characteristics

*TODO*

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 *VDisp* should have an understanding of undergraduate Level 1 Calculus and Physics."

## 3.3 System Constraints

*TODO*

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. In the context of CAS 741, you often will may not have any system constraints.

# 4 Specific System Description

*TODO*

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.

## 4.1 Problem Description

*TODO*

*VDisp* is intended to solve ... What problem does your program solve? The description here should be in the problem space, not the solution space.

### 4.1.1 Terminology and Definitions

*TODO*

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.

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

*TODO*

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.

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

PS1:

PS2: ...

A figure here makes sense for most SRS documents

### 4.1.3 Goal Statements

*TODO*

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.
Given the inputs, the goal statements are:

GS:FindDisp: : Given the soil properties, load, and type of footing, find the vertical displacement of the footing.

## 4.2 Solution Characteristics Specification

*TODO*

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.

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.

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.

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.

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.

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.

The instance models that govern *VDisp* 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

*TODO*

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. 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.

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.

A:SLH: The soil mass is homogeneous, with consistent soil properties throughout. (RefBy: LC:Calculate-Inhomogeneous-Soil-Layers.)

A:Isotropic: The material properties are independent of direction. (RefBy: ?.)

A:Saturated: The soil properties are independent of dry or saturated conditions, with the exception of unit weight. (RefBy: ?.)

A:2DPlane: The domain is a 2D Euclidean plane. (RefBy: TM:Equilibrium)

A:HalfPlane: The domain is half of a 2D Euclidean plane, as divided by a straight line. (RefBy: ?.)

A:Terzaghi: When we apply stress to a porous material, the stress is opposed by the pressure in the fluid that fills the pores of the material (Wikipedia, 2022c). (RefBy: ?.)

A:Continuum: The underlying molecular structure of matter is not considered and gaps and empty spaces within a material particle are ignored. The material is assumed to be continuous. (**?**, p. 33–34), (**?**, p. 1-2) (RefBy: ?.)

#### 4.2.2 Theoretical Models

*TODO*

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.

This section focuses on the general equations and laws that *VDisp* is based on. Modify the examples below for your problem, and add additional models as appropriate.

**RefName:** **BT:NormStress**

**Label:** Total Normal Stress

**Equation:** $\sigma = \frac{F_n}{A}$

**Description:**
$\sigma$ is the total normal stress (Pa)
$F_{\text{n}}$ is the total normal force (N)
$A$ is the cross-sectional area ($\text{m}^2$)

**Notes:** None.

**Source:** Fredlund and J.Krahn (1977)

**Ref. By:** BT:EffStress

**Preconditions for BT:NormStress:**

- list preconditions here

**Derivation for BT:NormStress:** Not Applicable

**RefName:** **BT:EffStress**

**Label:** Effective Stress

---

**Equation:** $\sigma' = \sigma - u$

**Description:**
$\sigma$ is the total normal stress on the soil mass (Pa).
$\sigma'$ is the effective normal stress provided by the soil skeleton (Pa).
$u$ is the pore pressure from the water within the soil (Pa).

**Notes:** According to Terzaghi's principle (A:Terzaghi) the changes caused by stress in a porous medium are a result of changes to the effective stress. The total stress $\sigma$ is defined in BT:NormStress.

**Source:** Fredlund and J.Krahn (1977)

**Ref. By:** text

**Preconditions for BT:EffStress:**

- A:Terzaghi

**Derivation for BT:EffStress:** Not Applicable

**RefName:** **TM:Equilibrium**

**Label:** Equilibrium

---

**Equation:** $\sum F_x = 0, \sum F_y = 0, \sum M = 0$

**Description:**
$\sum F_x$ is the sum of the x-components of all the forces (N)
$\sum F_y$ is the sum of the y-components of all the forces (N)
$M$ is the moment (N m)

**Notes:** For a body in static equilibrium, the net forces and moments acting on the body will cancel out. Assuming a 2D problem, the sum of the x-components of all the forces, $\sum F_x$, and the sum of the y-components of all the forces, $\sum F_y$, will be equal to 0. All forces and their distance from the chosen point of rotation will create a net moment equal to 0.

**Source:** Fredlund and J.Krahn (1977)

**Ref. By:** ?.

**Preconditions for TM:Equilibrium:**

- A:2DPlane

**Derivation for TM:Equilibrium:** Not Applicable

**RefName:** **GD:Weight**

**Label:** Weight

**Equation:** $W = V\gamma$

**Description:**
$W$ is the weight (N)
$V$ is the volume ($\text{m}^3$)
$\gamma$ is the specific weight ($\frac{\text{N}}{\text{m}^3}$)

**Notes:** None.

**Source:** Wikipedia (2022d)

**Ref. By:** ?.

**Preconditions for GD:Weight:**

- add preconditions

**Derivation for GD:Weight:** Not Applicable

**RefName:** **GD:hsPressure**

**Label:** Hydrostatic Pressure

**Equation:** $p = \gamma h$

**Description:**
$p$ is the pressure (Pa)
$\gamma$ is the specific weight ($\frac{\mathrm{N}}{\mathrm{m}^3}$)
$h$ is the height (m)

**Notes:** This equation is derived from Bernoulli's equation for a slow moving fluid through a porous material.

**Source:** Wikipedia (2022a)

**Ref. By:** ?.

**Preconditions for GD:hsPressure:**

- add preconditions

**Derivation for GD:hsPressure:** Not Applicable

**RefName:** **TM:saintVenant**

**Label:** Saint-Venant's Principle

---

**Equation:** N/A

**Description:** N/A

**Notes:** The difference between the effects of two different but statically equivalent loads becomes very small at sufficiently large distances from the load

**Source:** Wikipedia (2022b)

**Ref. By:** ?.

**Preconditions for TM:saintVenant:**

- add preconditions

**Derivation for TM:saintVenant:** 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.

### 4.2.3 Refined Theories (RT)

*TODO*

Refined theories are a refinement of one or more other theories. The refined theories are less abstract than the background theories. Generally the reduction in abstraction is possible through invoking (using/referencing) Assumptions. For instance, the RT could be Newton's Law of Cooling stated abstracting.

This section collects the laws and equations that will be used in building the final theories.

### 4.2.4 Data Definitions

*TODO*

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.

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

### 4.2.5 Data Types

*TODO*

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.

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

All data types must be used by at least one of the models.

For the mathematical notation for expressing types, the recommendation is to use the notation of **?**.

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.

| 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 Final Theories

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 FTs are built by refining the RTs and BTs. This section should remain abstract. The SRS should specify the requirements without considering the implementation.

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 are solved by referencing your final theories.

### 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

Table 2: Specification Parameter Values

| Var | Value |
|---|---|
| $L_{\text{min}}$ | 0.1 m |

#### 4.2.8 Properties of a Correct Solution

A correct solution must exhibit fill in the details. 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

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.

# 5 Requirements

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

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.

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

R3: Calculation related requirements.

R4: Verification related requirements.

R5: Output related requirements.

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

## 5.2 Nonfunctional Requirements

List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. 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. 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. 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.

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 *VDisp* 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.

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.

NFR3: **Maintainability** The effort required to make any of the likely changes listed for *VDisp* 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.

NFR4: **Portability** This NFR is easier to write than the others. The systems that *VDisp* 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.

- Other NFRs that might be discussed include verifiability, understandability and reusability.

# 6 Likely Changes

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

# 7 Unlikely Changes

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

# 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.

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.

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

| | T?? | T?? | T?? | GD?? | GD?? | DD?? | DD?? | DD?? | DD?? | IM?? | IM?? | IM?? | IM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T?? | | | | | | | | | | | | | |
| T?? | | | X | | | | | | | | | | |
| T?? | | | | | | | | | | | | | |
| GD?? | | | | | | | | | | | | | |
| GD?? | X | | | | | | | | | | | | |
| DD?? | | | | X | | | | | | | | | |
| DD?? | | | | X | | | | | | | | | |
| DD?? | | | | | | | | | | | | | |
| DD?? | | | | | | | | X | | | | | |
| IM?? | | | | | X | X | X | | | | X | | |
| IM?? | | | | | X | | X | | X | X | | | X |
| IM?? | | X | | | | | | | | | | | |
| IM?? | | X | X | | | | X | X | X | | X | | |

Table 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. In the context of a course like CAS 741, 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.

| | IM?? | IM?? | IM?? | IM?? | 4.2.7 | 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 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?? | | | | | | | | | | | | | | | | | | | |
| GD?? | | X | | | | | | | | | | | | | | | | | |
| GD?? | | | X | X | X | X | | | | | | | | | | | | | |
| DD?? | | | | | | | X | X | X | | | | | | | | | | |
| DD?? | | | X | X | | | | | | X | | | | | | | | | |
| DD?? | | | | | | | | | | | | | | | | | | | |
| DD?? | | | | | | | | | | | | | | | | | | | |
| IM?? | | | | | | | | | | | X | X | | X | X | X | | | X |
| IM?? | | | | | | | | | | | | X | X | | | X | X | X | |
| IM?? | | | | | | | | | | | | | | X | | | | | X |
| IM?? | | | | | | | | | | | | | X | | | | | X | |
| LC?? | | | | X | | | | | | | | | | | | | | | |
| LC?? | | | | | | | | X | | | | | | | | | | | |
| LC?? | | | | | | | | | X | | | | | | | | | | |
| LC?? | | | | | | | | | | | X | | | | | | | | |
| LC?? | | | | | | | | | | | | X | | | | | | | |
| LC?? | | | | | | | | | | | | | | | X | | | | |

Table 6: 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.

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

The value of FRACTION, for the Maintainability NFR would be given here.

The following is not part of the template, just some things to consider when filing in the template.
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"

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

# References


D.G. Fredlund and J.Krahn. Comparison of slope stability methods of analysis. *Can. Geotech. J.*, 4(14):429–439, April 1977.

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. ISSN 1573-1340. doi: 10.1007/s11155-006-9020-7. URL https://doi.org/10.1007/s11155-006-9020-7.

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.

Wikipedia. Pressure. https://en.wikipedia.org/wiki/Pressure, June 2022a.

Wikipedia. Saint-venant's principle. https://en.wikipedia.org/wiki/Saint-Venant%27s_principle, July 2022b.

Wikipedia. Terzaghi's principle. https://en.wikipedia.org/wiki/Terzaghi%27s_principle, June 2022c.

Wikipedia. Weight. https://en.wikipedia.org/wiki/Weight, June 2022d.