A New Requirements Template for Scientific Computing

Spencer Smith[a] and Lei Lai[a]

[a] Computing and Software Department, McMaster University

Abstract

This paper presents a new template for writing requirements specifications for scientific computing software, using the following characteristics to guide the design: i) only one user viewpoint needs to be considered for specifying the physical model; ii) assumptions can be used to distinguish between models and a strong coupling exist between the assumptions and the functional requirements; iii) a high potential exists for reuse of the functional requirements; iv) the hierarchical nature of goals, theoretical models and instanced models facilitates change management; and, v) documenting and validating continuous mathematics needs to be supported. One significant change from the existing templates is for the functional requirements, which are split into two main sections: "Problem Description" and "Solution Characteristics Specification." The template introduces a new traceability matrix to facilitate future modifications and it explicitly addresses nonfunctional requirements for the accuracy of the input data, the sensitivity of the model, the tolerance of the solution and the solution validation strategies.

Keywords: Domain-Specific Requirements Engineering, Combination of Templates, Scientific Computing

1 Introduction

Requirements analysis and documentation techniques can be used to improve the quality of scientific computing software, where scientific computing is defined as the use of computer tools to analyze or simulate mathematical models of real world systems of engineering or scientific importance so that we can better understand and predict the system's behaviour. Unfortunately requirements documentation in scientific computing is often nonexistent or incomplete because of a lack of an accepted systematic approach. To encourage a systematic approach one generally accepted technique from requirements engineering is to adopt and follow a requirements template. However, a survey of existing templates shows that the particular needs of scientific computing systems are not met by the current options. This paper addresses this shortcoming by presenting a new template that has been tailored to meet the special characteristics inherent in scientific computing software.

The first section below provides background information on scientific computing, the advantages of requirements documentation in this context, and a review of existing requirements templates. The next section explains that the existing templates do not address the specific characteristics of scientific software that distinguish it from other classes of applications. Section 4 presents the template itself by first showing the proposed table of contents and then explaining the intent of each of the sections. The new template combines features of existing templates as well as adding some new features. Sections 5 and 6 discuss future work and concluding remarks, respectively.

2 Background

The following five-step procedure is typically used to solve analysis problems in scientific computing: i) define the problem; ii) create a mathematical model by applying proper assumptions; iii) identify a computational method; iv) refine and implement a solution; and, v) validate the solution. If a requirements stage is made part of the development process for scientific software, then the first two steps above will correspond to the stage of requirements gathering, analysis and documentation. In the first step the problem should be clearly, completely and unambiguously defined. The second step involves using simplifying assumptions to develop a model of the real world. In scientific computing the model often consists of governing ordinary or partial differential equations together with boundary conditions and/or initial conditions and a set of closure equations.

Although the size and complexity of scientific computing problems are constantly growing, the approach to problem definition, model development and documentation is often done in an ad hoc manner. Moreover, the nonfunctional requirements that would provide information on such important decisions as required accuracy, precision, efficiency, etc., are typically neglected. Unfortunately requirements for scientific computing problems are rarely systematically gathered, analyzed and documented, even though the existence of a complete and consistent requirements document can lead to better decisions for improving such software qualities as reliability, usability, verifiability, maintainability, reusability and portability. One example of how requirements improve quality is that the reliability of software can only be accurately judged when the validation step (v in the above list) has an unambiguous statement of the behaviours and qualities that the software is required to satisfy.

Besides enabling software validation, a requirements document provides many other advantages during the lifecycle of a software project (IEEE, 1998; Sommerville and Sawyer, 1997). For instance, a software requirements specification (SRS) reflects the mutual understanding of the problem to be solved between the requirements analyst and the client and it provides a starting point for the software design phase. Other advantages that requirements documentation brings that are of particular value to scientific computing software include the following, as discussed in Smith et al. (2005): i) reducing ambiguity; ii) clearly identifying and documenting the range of model applicability; iii) clearly identifying and documenting the assumptions that simplify the real world for modelling purposes; iv) increasing confidence that all special cases have been considered; and, v) encouraging the analyst to scrutinize their problem in advance of designing the computational software.

Given the advantages that requirements documentation can provide, it is surprising that a literature review on the application of software engineering methodologies to scientific computing problems shows little on this topic. The research usually focuses on the design and implementation of the software and does not address how to improve the quality from the requirements level. One exception to neglecting the requirements phase is a requirements analysis of data parallel applications (Gerlach, 2002). Another exception documents the requirements of models of physical phenomena (Kreyman and Parnas, 2002) using tabular expressions. However, this approach to documenting physical phenomena does not entirely match the needs of scientific computing because Kreyman and Parnas (2002) use ideas that were first developed for real-time embedded systems, which have different characteristics than scientific computation systems. Moreover, Kreyman and Parnas (2002) allow the numerical methods, which are essentially implementation decisions, to be encompassed into the requirements documentation, which contradicts with the principle that requirements should address "What" is required, but not "How" to achieve it.

The scientific computing literature apparently does not provide many details on a methodology for requirements documentation, so the next step is to determine what ideas can be borrowed from the field of requirements engineering. One idea from requirements engineering that satisfies the current goal of encouraging a systematic procedure for requirements documentation is the common practise using of a requirements template, where the template provides a frame of reference, identifies needed information, and suggests an order of presentation for the requirements. An advantage of using requirements templates, as discussed in Sommerville and Sawyer (1997), is increasing the productivity and adequacy of an SRS, because a well-organized format for the document acts as a checklist for writers of the SRS, which reduces the chances of omitting information. The template also has an important advantage for scientific computing in that it facilitates the communications among various SRS users, which in the scientific context will be researchers, software developers, physical modellers, computational scientist, etc.

A template seems ideal for documenting scientific computing problems, but it is difficult to directly adopt an existing template. Although there are several requirements templates that are designed for general purposes and contain good advice on how to write requirements and how to avoid problems (Heninger, 1980; NASA, 1989; Robertson and Robertson, 1999; IEEE, 2000), these templates are not usually used without modifications that depend on the application. Most templates to date focus on business applications and real-time systems. Although the existing templates provide an excellent starting point, they do not address all of the issues of importance for scientific computation problems, as discussed in the next section.

3 Why a New Template?

Differences between the characteristics of scientific computing software and other types of systems suggest that a new template needs to be developed that is specific to scientific software. These characteristics are enumerated below. Although some of these characteristics can also occur in other classes of software, they are considered to be important enough in scientific computing to guide the design of the new requirements template.

3.1 One User Viewpoint for the Physical Model

The specification of the mathematical model in scientific computing software allows for an important simplification over some other classes of software because it does not have the variety of viewpoints for the functional requirements that are evident in many other software systems. Unlike banking software, or a library database, which have many different stakeholders and other software/hardware systems, there is only one user viewpoint for the physical model in scientific software. (Although the functional requirements for the physical model have one viewpoint, the nonfunctional requirements will certainly change depending on the user viewpoint.) Only one viewpoint is necessary for the physical model because there is only one correct statement of the problem physics. The one user viewpoint allows application of the principle of separation of concerns so as to focus on making the documentation of the mathematical model self-contained. As mentioned in Section 3.3 below, the computation may be embedded in a larger system context, but in the context of the five steps outlined at the beginning of Section 2, the physical model need only be considered from one viewpoint. Although there is considerable complexity inside scientific software, from the external viewpoint the model is relatively simple. Although differing in the details, the solution of a scientific computing problem can be effectively abstracted following the simple sequential scenario: input information, perform calculations, output the results. With only one viewpoint to consider, the functional requirements do not have to specify many partial viewpoints, or the complex relationships associated with concurrency.

3.2 Assumptions Distinguish Models

To build any model of the real-world it is necessary to introduce simplifying assumptions, such as assuming small distances, angles, or forces may be neglected in favour of larger distances, angles or forces. In scientific computation it is often the differing assumptions that distinguish one piece of work from another. For instance, Smith and Stolle (2003) summarize research on polymer film casting by distinguishing papers based on their simplifying assumptions, such as whether the problem is modelled as 1D, 2D or 3D, or whether the problem is isothermal or nonisothermal, or whether the polymer is assumed viscous or viscoelastic. Often the quality of the model depends on how reasonable the simplifying assumptions are. Assumptions also play a crucial role in scientific computing because a strong coupling exists between the assumptions and the physical model. Given the importance of assumptions, the SRS template should clearly show how they will be documented and justified. Other researchers cannot judge a model, reproduce its results, or improve on it, without knowing the assumptions that were made and the reasoning behind those assumptions.

3.3 High Potential for Reuse

A template for scientific software should support the reuse of functional requirements, where the functional requirements correspond to the problem description and the mathematical model. The functional requirements for scientific software have a high potential for reuse because the laws on which they are based are almost universally accepted and slow to change. An example of a requirement that is unlikely to change is that a body at rest shall satisfy force equilibrium. The stable functional requirements should be separated from the nonfunctional requirements, which will change depending on the context in which the software is to be used. For instance, educational software and a safety critical real-time system may both have identical functional requirements based on the same mathematical model, but the educational software will emphasize nonfunctional requirements for ease of use and portability, while the scientific software that is employed as part of a safety critical control system will likely emphasize accuracy and speed of calculation over all other considerations. The documentation on the mathematical model of the scientific computation should be self-contained so that it can be embedded in a larger system, no matter the context of the other system. If a library of SRS's for scientific software can be built, then people designing a large project will be able to choose the appropriate models from the library and incorporate them into their project.

3.4 Hierarchical Nature Facilitates Change

In software engineering, one of the most important design principles is to design for change. The preceding section described how the document should be designed to allow for changes in the nonfunctional requirements; the functional requirements themselves should also be designed for likely changes. In the case of the functional requirements the most likely changes are adding or subtracting goals, modifying the theoretical model, or modifying the instanced model. In this decomposition goals are defined as the objectives that the software should achieve; theoretical models are the mathematical equations used to solve the problem described in the problem domain; and, instanced models, which refine the theoretical models, express the problem in sufficient detail so that it can be solved. The sequence from goals to theoretical and instanced models corresponds to a decreasing level of abstraction. As an example, the abstract goal might be to solve for unknown forces and the associated theoretical model that is selected may be to use the principle of equilibrium. The theoretical model will be refined further to an instanced model when decisions are made about a specific coordinate system, sign convention, etc. Ideally the requirements template should support reuse of the abstract goals with different theoretical models. For instance, the theoretical model of having the forces of an unmoving body sum to zero could be changed to instead use the principle of virtual work. Similarly a specific theoretical model that was refined into an instanced model using a Cartesian coordinate system could be refined into a different instanced model using a polar coordinate system. In both cases the "higher-level" requirements could be reused. It should also be possible to reuse "lower-level" requirements in a different context, such as using the same theoretical model to satisfy different goal statements. For instance, the same theoretical model of equilibrium could be employed to solve problems where the goal is to find internal forces in a beam or internal stresses inside a glacier.

3.5 Continuous Mathematics Poses a Challenge

Scientific software differs from most other software by nature of the fact that the quantities of interest are continuous, as opposed to discrete. The variables that are to be solved for are usually continuous in nature, such as velocity, displacement, temperature, pressure, concentrations, etc. Unfortunately, the formal techniques developed for specifying requirements generally focus on discrete mathematics, which mean the techniques will have to be modified for scientific software requirements specification. Moreover, the SRS template should assist with the challenge of validating the numerical results because most scientific computing problems cannot be solved exactly. When it comes to validation the question of how much error can be tolerated must be addressed. Moreover, since the true solution is often unknown, the specification must address the requirements that must be satisfied to consider a result correct.

4 The New Template

The proposed template is organized in a hierarchical format with nine main sections, as presented in the table of contents shown in Figure 1. The majority of the initial sections are inspired by the IEEE Standard 830 (IEEE, 1998), while the subsection "Nonfunctional Requirements" in the section "Specific System Description" and the section "Other System Issues" mainly come from the Volere Requirements Specification Template (Robertson And Robertson, 1999). The section "General System Description" comes from both of the previously mentioned sources. The subsections "Problem Descriptions" and "Solution Characteristics Specification" in the section "Specific System Description" are unique propositions of the current work. The template also introduces a systematic approach to manage changes in a scientific computing SRS through a newly defined traceability matrix, which is documented in SRS Section 6. A description of the important sections of the new SRS template are provided in the subsections that follow. Further details on the template presented in this section can be found in Lai (2004).

1. Reference Material: a) Table of Contents b) Table of Symbols c) Abbreviations and Acronyms d) Index of Requirements

2. Introduction: a) Purpose of the Document b) Scope of the Software Product c) Organization of the Document

3. General System Description: a) System Context b) User Characteristics c) System Constraints

4. Specific System Description:

(a) Problem Description: i) Background Overview, ii) Terminology Definition, iii) Physical System Description, iv) Goal Statements

(b) Solution Characteristics Specification: i) Assumptions, ii) Theoretical Models, iii) Data Definitions, iv) Instanced Models, v) Data Constraints, vi) System Behaviour

(c) Nonfunctional Requirements: i) Accuracy of Input Data, ii) Sensitivity of the Model, iii) Tolerance of Solution, iv) Solution Validation Strategies, v) Look and Feel Requirements, vi) Usability Requirements, vii) Performance Requirements, viii) Maintainability Requirements, ix) Portability Requirements, x) Security Requirements

5. Other System Issues: a) Open Issues b) Off-the-Shelf Solutions c) New Problems, d) Waiting Room

6. Traceability Matrix

7. List of Possible Changes in the Requirements

8. Values of Auxiliary Constants

9. References

Figure 1: Proposed Requirements Template

4.1 Reference Material (SRS Section 1)

The requirements document should be organized as a reference as well as being a specification of the system. The information in the SRS is recorded in this section in a form that allows easy reference throughout the project. The information includes a table of symbols to provide a quick reference for the symbols specifically defined in the SRS. A table of symbols is invaluable in scientific computing due to the variety of symbols used and the fact that in a different scientific context the same symbol may have a different meaning. As an example, the same symbol σ is used to represent conductivity, stress, the Stefan-Boltzmann constant for radiative heat transfer, the standard deviation, etc.

4.2 General System Description (SRS Section 3)

The purpose of this section of the template is to provide general information about the system so the specific requirements in the next section will be easier to understand. Following the design principle that the mathematical model should be easy to reuse, as discussed in Section 3.3, 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 accounts for the specification details that change from one system context to another.

The general system description should include an overview of the system context that defines the boundaries between the product to be built and the people, organizations, other products and pieces of technology that have a direct interface with the product, such as system interfaces (which describe adjacent system), user interfaces, and software interfaces. The system context subsection also places the product into perspective with other related products. If the product is independent and totally self-contained, this should be stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to the functionality of the software and should identify interfaces between that system and the software. Stakeholder characteristics are also summarized within the general system description so as to facilitate consideration on how the system should be designed to conform to the features of the stakeholders. In the case of scientific software the mathematical model is documented from a single viewpoint, as discussed in Section 3.1, but in the larger system context there may be more than one stakeholder and multiple viewpoints. It is important to document the stakeholder and user characteristics, as the information in this subsection will affect the way the product is designed; for example, it could be referenced to determine the usability requirements of the product. This section should not be used to state specific requirements but rather to provide the reasons why certain specific requirements are later specified in SRS Section 4.

System constraints are also identified in the general system description section. They differ from other type of requirements in the sense that 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.

4.3 Problem Description (SRS Section 4.a)

The problem description is part of the "Specific System Description" (SRS Section 4) of the report, where the specific system description includes all of the SRS software requirements in sufficient detail to enable design and testing of a system that will satisfy the requirements. The order of the contents of the first two sections of the specific system description are motivated by common scientific and engineering practises, which typically systematically proceed from the general to the more specific. Thus, the problem to be solved is first described, and then the characteristics that a solution to the problem must satisfy are specified. In the problem description section information on the problem domain of the physical system is given, where the information includes the concepts that populate the area in which the users carry out their activities, and in which they have a problem and expect a solution.

4.3.1 Terminology Definition

This section is motivated by a need to clarify the engineering concepts in the SRS and to serve as a reference aid. The contents consist of a list of engineering concepts and their exact meaning in the SRS, including some related conventions that will be used in the SRS. For example, definitions are given for the sign conventions that are used. This section should provide enough information to allow understanding of the later Sections "Theoretical Model" (SRS Section 4.a.ii) and "Data Definitions" (SRS Section 4.a.iii). The "Terminology Definition" section is necessary in scientific computing for an unambiguous SRS because terminology often has subtly different meanings, even in very similar contexts. As an example, in engineering mechanics the stress in a bar under uniaxial extension can either be expressed as the force divided by the original cross-sectional area (engineering stress), or the force divided by the current cross-sectional area (true stress). When large deformations occur, the distinction between these two stress measures becomes very important; therefore, if the convention for how stress is measured is not clearly stated, the results of the scientific analysis will likely be incorrect.

4.3.2 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. This portion of the SRS must clearly state what is unknown and what is known. The inclusion of a diagram or sketch will help the writers organize their thoughts and it will help them communicate the solution process to the reader. An example physical system for analysis of structural mechanics problems would be a beam. The properties of a beam would include its shape and the number of materials used to build it. Interactions between the beam and the physical environment would include external forces applied to the beam. The beam would also have associated boundary conditions that define how it is supported.

4.3.3 Goal Statements

The motivation of this section of the SRS is to capture the goals in the requirements process. As mentioned earlier, 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. For an SRS that solves beam problems an example goal would be to solve for the deflection of the beam, given the beam's properties and the applied forces. Goals will be refined in Section "Instanced Models" (SRS Section 4.b.vi). Large and complex goals should be decomposed into smaller sub-goals.

4.4 Solution Characteristics Specification (SRS Section 4.b)

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 is different from the requirements specification of business applications because scientific software development focuses on efficient mathematical models rather than on the complex interaction of objects.

4.4.1 Assumptions

Section 3.2 emphasized the importance of assumptions to scientific computing. Therefore, an entire section of the SRS template is devoted to listing and labelling the assumptions. 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. For instance, for software to solve for the deflection in a beam an assumption may be made that the weight of the beam can be neglected, with the justification that the applied forces in practise are generally larger than a beam's self-weight.

4.4.2 Theoretical Model

The assumptions in the SRS provide a means to bridge from the goal statements to the theoretical models, where theoretical models are sets of abstract mathematical equations or axioms for solving the problem described in Section "Physical System Description" (SRS Section 4.a.iii). Examples of theoretical models are physical laws, constitutive equations, relevant conversion factors, etc. This section is separate from goals and from the instanced model so that each portion of the document can be changed independently, as discussed in Section 3.4. Each theoretical model should be described from the following aspects: i) an introduction to the theory, which can take the form of a summary of the appropriate equations, together with a reference, or in the case of more complex problems, a more detailed derivation may need to be shown; ii) the reasons for choosing the model; iii) the rules and conventions of the model; and, iv) the limitations of the theory, which help identify the important physical data needed to solve the problem. An example of the limitation of a theory would be a theory for projectile motion that ignores the drag on the object caused by air friction.

4.4.3 Data Definitions

This section collects and substantiates all physical data needed to build the instanced model. For instance, if the models in Section "Instanced Model" (SRS Section 4.b.iv) needs a coordinate system, then this section should define it. The dimension system, as well as the dimension of each quantity can also be declared. In general, it is better to list the dimensions without reference to specific units, in support of the principle that it should be easy to reuse the mathematical model (Section 3.3). For instance, it is better to specify that the "MLtT" (Mass Length time Temperature) system is being used rather than to lock the specification into the units kg, m, s, etc. A more general description allows easy conversion of units, for instance between SI (Système International d'Unités) and imperial.

The diagram or sketch of the physical system in Section "Physical System Description" (SRS Section 4.a.iii) can be labelled with the defined data for quick reference. One technique that can be beneficially employed at this stage is to use a table summarizing data definitions along with textual descriptions. This approach has the advantage of ensuring a uniform structure for the same type of data objects. For example, the definitions of all applied forces could follow the same structure consisting of "symbol," "point of application," "magnitude," "direction," and "assumed positive direction."

4.4.4 Instanced Model

The motivation for this section is to reduce the problem defined in "Physical System Description" (SRS Section 4.a.iii) to one expressed in mathematical terms. This section uses the concrete symbols defined in Section "Data Definitions" (SRS Section 4.b.iii) to replace the abstract symbols in the models identified in Section "Theoretical Model" (SRS Section 4.b.ii). At this stage it is important to avoid assuming or applying a numerical algorithms in the SRS. The SRS should specify the requirements without considering the implementation.

4.4.5 Data Constraints

The motivation for this section is to clarify the environment and the system limitations imposed on admissible data. These constraints are specified to maintain the validity of the models defined in Section "Instanced Model" (SRS Section 4.b.iv). Example data constraints include the following: i) possible units of the data; ii) value constraints imposed by the physics of the problem and from the system (for instance, all lengths must be greater than zero); and, iii) other properties such as whether the data is input or output, or whether its value is known or unknown. As for Section "Data Definitions" (SRS Section 4.b.iii), a table is suggested for the purpose of listing the data constraints in a uniform format. Listing the constraints on the data is necessary to address the challenge of solution validation, which is one of the template's design principles that were discussed in Section 3.5. As an example, the force F_{ax} applied to a beam can be constrained so that it is not unrealistically different in magnitude compared to the other forces acting on the beam, which are given in the set S^F:

$$(\min_f \leq |F_{ax}| \leq \max_f) \wedge [(|F_{ax}| \neq 0) \Rightarrow \forall (FF \mid FF \in S_F \bullet FF \neq 0 \wedge \frac{\max\{|F_{ax}|, |FF|\}}{\min\{|F_{ax}|, |FF|\}} \leq 10^{r_f})]$$

where $\min_f$ and $\max_f$ are the system constraints for the minimum and maximum magnitude forces, and r_f is a positive integer that is the maximum exponent of base 10 for the ratio between the magnitudes of the largest and smallest forces. A constraint such as this is valuable in the subsequent design stage because it identifies whether the software will be required to handle the potentially difficulty situation of performing calculations with values of widely different magnitudes.

4.4.6 System Behaviour

The purpose of this section is to give a detailed model of the system's dynamic functionalities based on the information in Sections "Data Constraints" (SRS Section 4.b.v) and "Instanced Model" (SRS Section 4.b.iv). The undesired situations, such as user errors, should also be documented here. Responses to undesired situations should be stated in the SRS instead of being left for the programmer to later invent. This section is a technical refinement for the Section "Goal Statements" (SRS Section 4.a.iv), since the technical concerns, such as assumptions and models, are clarified by previous sections. The system behaviour specification reflects a dynamic process of first receiving the input data, then applying the model, and finally obtaining the results. The system behaviour should be specified in a way that all the system goals are satisfied and the data constraints are considered. The content in this portion of the document should be formal enough for design and testing, with a suggested technique being the use of tabular expressions to specify partial specifications of the system functionalities so that the specifications are verifiable on domain coverage, disjoint domain, and definedness (Lai, 2004).

4.5 Nonfunctional Requirements (SRS Section 4.c)

This section specifies system requirements that consider the quality and behaviour of the system as a whole. This section is separate from the functional requirements to facilitate the potential independent change of these two portions of the SRS, as discussed in Section 3.3. Several of the sections are borrowed from the Volere template (Robertson and Robertson, 1999), such as sections for look and feel, usability, performance, maintainability, portability and security requirements and a section for other system issues (SRS Section 5). The sections detailed below are unique to the new template.

4.5.1 Accuracy of Input Data

This section indicates the error associated with the input data. The contents of this section can be specified by illustrating each input data with their possible sources of measurement error and error range. The value of specifying the accuracy of the input is that these values can be used to judge the acceptability of the errors in the output.

4.5.2 Sensitivity of the Model

An inaccurate solution is not necessarily due to an ill-conceived algorithm, but may be inherent in the problem being solved. Even with exact computation, the solution to the problem may be highly sensitive to perturbations in the input data, such as when an unstable ordinary differential equations is solved. The cause of any potential propagated data error, which reflects the sensitivity of the problem, should be studied since it is a factor that can lead to inaccurate numerical results. Sensitivity of the problem is sometimes specified by calculating the condition number. In general, the condition number varies with the input, and in practise one usually does not know the exact condition number, or it is very expensive to compute. Thus, analysts often must content themselves with a rough estimate or upper bound for the maximum condition number over some domain of inputs. When the condition number is unavailable other techniques, such as linear perturbation analysis, may be used.

4.5.3 Tolerance of the Solution

The motivation of this section is to establish a criterion for the correct solution, using the principle that it is better to specify what the correct answer should be, as provided in "System Behaviour" (SRS Section 4.b.vi), and then specifying what the allowed tolerances are, rather than to try and specify both simultaneously. Another motivation for this section is to develop an understanding of what level of solution accuracy is required. Solution accuracy refers to the closeness of a computed solution to the true solution of the problem under consideration. The accuracy of numerical solutions against the model depends on input data error, numerical error, computer round-off error, programming error, sensitivity of the problems, and stability of numerical algorithms. As a general rule, the accuracy of the final computation of the output need not be greater than the accuracy of the input data.

One possibility is that the tolerance can be specified by tolerance functions. For example, the tolerance for the calculation of force equilibrium in 1D might be: $|\Sigma F_i|/\sqrt{\Sigma F_i^2} \leq \varepsilon$ where F_i is the ith force component and ε is the allowed error.

4.5.4 Solution Validation Strategies

As observed in Section 3.5, it is difficult to validate scientific computing software. The purpose of this section of the SRS is to capture the experts insight on how to validate the system results. Four potential evaluation strategies are: i) solve the problem by different techniques, such as electronic spreadsheet, graphical solution, etc.; ii) substitute the calculated results back into the original governing equations to calculate the residual error; iii) partially validate the problem by validating simpler subsets of it for which the solution is known; and, iv) check that the governing equations are satisfied, boundary conditions are satisfied, energy is conserved, mass is conserved, etc.

4.6 Traceability Matrix (SRS Section 6)

The traceability matrix was introduced into the SRS template because of the design principle that the goals, theoretical models and instanced models should be simple to manage, as discussed in Section 3.4. The traceability matrix was designed based on the portions of the document that are most likely to change, such as the assumptions, which as discussed in Section 3.2 are tightly coupled with the functional requirements. By showing the relationship between different items in the SRS, the traceability matrix shows how a change in one functional requirements impacts the other functional requirements. The traceability matrix will often be sparse, so that significant portions of the SRS can be reused for different, but related, scientific computing problems.

The entities that were identified as most likely to be reused include the following: "Physical System Descriptions," "Goal Statements," "Data Definitions," "Assumptions," "Theoretical Models" and "Instanced Models." These sections of the SRS were selected for tracing because they represent the essential information about the model. A sample traceability matrix is provided in Table 1. In this table the prefixes G, T, A, M, and PS followed by a number correspond to the goals, theoretical model, assumptions, instanced model, and physical system, respectively. The symbol *x* stands for one of the data definitions. An example of two traceability matrices can be found in Lai (2004) for the case of modelling a rigid beam and a deformable beam. A comparison of the two traceability matrices shows what portions of the mathematical model for the rigid beam can be obtained simply by removing sections from the model of the deformable beam. If an assumption is added that the beam behaves as a rigid body, then goals related to solving for shear, bending moment and deflection can be removed. The removal of these goals means that the associated theoretical models and instanced models are also removed. For the traceability matrix to work effectively it is important that independent assumptions are used when deriving the mathematical model of the physical system.

Table 1: Sample Traceability Matrix

Phy. Sys./ Goal	Data/ Model	Assumptions										Model	
		A1	A2	…	A4	…	A8	A9	A10	…	A14	**M1**	…
G1	**T1**	√		…		…	√	√		…		√	…
G2	**T2**	√		…		…	√	√		…			…
G3	**T3**	√		…		…		√	√	…			…
	M1		√	…		…				…		√	…
PS1	*x*			…		…			√	…		√	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…

5 Future Work

The proposed template has been used to document the engineering mechanics problem of solving for the internal forces, moments, shears and deflections of a statically determinant beam (Lai, 2004; Smith et al., 2005). Although this case study demonstrates the usefulness of the template, there is a need for additional case studies that select scientific computing problems from fields other than engineering mechanics and that select problems for which no closed-form solutions exist. Besides looking at special purpose scientific software that is built to solve one scientific computing problem, it would be beneficial to consider the specification of the requirements for general purpose software that is intended to solve a family of related scientific problems. For instance, case studies should be investigated for a system to solve linear algebra problems, or a system to generate finite element meshes. Ideally an empirical study should be conducted to determine the relative efficacy of the new template versus ad hoc procedures. Additional future work will include an investigation as to whether the process of gathering, analyzing and documenting requirements for scientific software differs from the process for other types of applications.

6 Concluding Remarks

The design principles identified in Section 3 were used to guide the structure and content of the proposed template. For instance, the first principle, which states that the functional requirements can be considered from only one user viewpoint, was used to design the template's simple structure. As there is only one viewpoint, the documentation can follow a simple framework that allows a refinement from general goals to specific statements. Also, the single viewpoint principle allows the template to separate the one viewpoint functional requirements from the, potentially many viewpoint, nonfunctional requirements. The second design principle is that the modelling assumptions play a crucial role in the documentation, which is reflected in the fact that the template has a section devoted to assumptions and a traceability matrix that can be used to track changes in assumptions. Due to the third principle, which requires that the mathematical model be easy to reuse, the template separates the general system description and nonfunctional requirements from the specification of the mathematical model, so that changes can be made to the context of the software without changing the model's specification. The third principle is also responsible for the template section on auxiliary constants and the suggestion that generic units (m, L, t and T) be used in place of specific units (m, kg, s, etc.). The template separates goals, theoretical models and instanced models because of the fourth principle, which says that their hierarchical nature means that changes to these entities should be easy to manage. The separation allows the individual sections to be considered in isolation, and it allows the influence of changes to the functional requirements to be tracked, via the traceability matrix. The final principle states that documenting and validation of continuous mathematics should be supported by the template. This goal was realized by explicitly introducing sections where data constraints and solution validation strategies must be documented.

Acknowledgements

The financial support of the Natural Sciences and Engineering Research Council (NSERC) is gratefully acknowledged. The authors would also like to thank Dr. Ridha Khedri and Dr. Jacques Carette for their valuable advice.

References

GERLACH J (2002) *Domain Engineering and Generic Programming for Parallel Scientific Computing*. PhD Thesis, Von der Fakultät IV - Elektrotechnik und Informatik der Technischen Universität, Berlin.

HENINGER KL (1980) Specifying software requirement for complex system: New techniques and their application. *IEEE Transactions on Software Engineering*, 6(1), 2–13.

IEEE 1998 *Recommended practice for software requirements specifications*, *IEEE Std. 830*. IEEE.

KREYMAN K and PARNAS DL (2002) On documenting the requirements for computer programs based on models of physical phenomena. SQRL Report 1, Software Quality Research Laboratory, McMaster University.

LAI L (2004) *Requirements documentation for engineering mechanics software: Guidelines, template and a case study*. MASc Thesis, McMaster University, Hamilton, Ontario, Canada.

NASA (1989) *Software requirements DID, SMAP-DID-P200-SW*, Release 4.3. Technical Report, National Aeronautics and Space Agency.

ROBERTSON S and ROBERTSON J (1999) Volere Requirements Specification Template. In *Mastering the Requirements Process*, p 353–391, ACM Press/Addison-Wesley Publishing Co, New York, NY, USA.

SMITH WS, LAI L and KHEDRI R (2005) Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, Accepted.

SMITH WS and STOLLE DFE (2003) Numerical simulation of film casting using an updated Lagrangian finite element algorithm. *Polymer Engineering and Science*, 43(5), 1105–1122.

SOMMERVILLE I and SAWYER P (1997) *Requirement Engineering: A Good Practice Guide*. John Wiley & Sons Ltd.

THAYER RH and DORFMAN (Editors) (2000) *IEEE Recommended Practice for Software Requirements Specifications,* 2nd edition. IEEE Computer Society, Washington, DC, USA.