Assurance Cases for Scientific Computing Software

Spencer Smith
Computing and Software Department,
McMaster University
1280 Main Street West
Hamilton, Ontario L8S 4K1
smiths@mcmaster.ca

Mojdeh Sayari Nejad
Computing and Software Department,
McMaster University
1280 Main Street West
Hamilton, Ontario L8S 4K1
sayarinm@mcmaster.ca

Alan Wassyng
Computing and Software Department,
McMaster University
1280 Main Street West
Hamilton, Ontario L8S 4K1
wassyng@mcmaster.ca

ABSTRACT

Assurance cases that provide an organized and explicit argument for correctness should be used for certifying Scientific Computing Software (SCS), especially when the software impacts health and safety. Assurance cases have already been effectively used for safety cases for real time systems. Their advantages for SCS include engaging domain experts, producing only necessary documentation, and providing evidence that can potentially be verified/replicated by a third party. This paper illustrates assurance cases for SCS through the correctness case for 3dfim+, an existing medical analysis software for analyzing activity in the brain by computing the correlation between the measured and an ideal brain signal. This example was partly chosen because of recent concerns about the validity of fMRI (Functional Magnetic Resonance Imaging) studies. No errors were found in the software outputs from 3dfim+. However, the example still justifies the value of assurance cases, since the existing documentation is shown to have ambiguities and omissions, such as an incompletely defined ranking function and missing details on the coordinate system convention adopted. In addition, a potential concern for the software itself is identified: running the software does not produce any warning about the necessity of using data that matches the parametric statistical model employed for the correlation calculations.

CCS CONCEPTS

• **Mathematics of computing** → *Mathematical software*; • **Software and its engineering** → *Requirements analysis*; *Software verification and validation*;

KEYWORDS

assurance cases, software quality, software requirements specification, medical imaging software

ACM Reference format:
Spencer Smith, Mojdeh Sayari Nejad, and Alan Wassyng. 2018. Assurance Cases for Scientific Computing Software. In *Proceedings of* 40^{th} *International Conference on Software Engineering, Gothenburg, Sweden, May 2018 (ICSE),* 12 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ICSE, May 2018, Gothenburg, Sweden

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Are we currently putting too much trust in the quality of Scientific Computing Software (SCS)? Given its role in such fields as nuclear safety analysis and medical imaging, SCS has a significant impact on health and safety related planning and decision making. Although SCS developers do excellent work, do we have enough checks and balances in place for confidence in correctness? The usual approach employed when correctness is critical is to impose requirements for official software certification, where the goal for certification is to: "...systematically determine, based on the principles of science, engineering and measurement theory, whether a software product satisfies accepted, well-defined and measurable criteria" [16, p. 12]. Unfortunately, two significant problems exist for SCS in completing a conventional certification exercise through an external body:

- The external body often does not have deep expertise on the physical problem the software simulates or analyses, or on the numerical techniques employed. This tends to lead the external body to request a large quantity of documentation, as shown in standards for SCS in the nuclear domain [4, 9, 10, 39], or in the medical domain [7].
- SCS developers tend to dislike documentation. As observed by Carver [5], scientists do not view rigid, process-heavy approaches, favourably. Moreover, they often consider reports for each stage of software development as counterproductive [31, p. 373].

A potential solution to these two problems is to have the SCS developers create an assurance case as they develop their software. Assurance case techniques have been developed and successfully applied for real time safety critical systems [30, 32, 43]. An assurance case presents an organized and explicit argument for correctness (or whatever other software quality is deemed important) through a series of sub-arguments and evidence. Putting the argument in the hands of the experts means that they will work to convince themselves, along with the regulators. They will use the expertise that the regulators do not have; they will be engaged. This engagement will hopefully help bridge the current chasm between software engineering and scientific computing [20], by motivating scientists toward documentation and correcting the problem of software engineers failing to meet scientists' expectations [33]. Significant documentation will still likely be necessary, but now the developers control the documentation. What is created will be relevant and necessary. More details on the current literature on assurance cases is given in Section 2.

Arguing in favour of assurance cases does not imply that SCS developers have not, or do not currently, treat correctness seriously. They have developed many successful theories, techniques, testing

procedures and review processes. In fact, an assurance case will likely use much of the same evidence that SCS developers currently use to convince themselves of the correctness of their software. The difference is that the argument is no longer ad hoc, or incompletely documented. The argument will now be explicitly presented for review by third parties; it is no longer about implicitly trusting the developer. The act of creating the assurance case may also lead the developer to discover subtle edge cases, which would not have been noticed with a less rigorous and systematic approach.

While the eventual goal is developing a template for assurance cases for any SCS, our initial approach is to learn by first building an assurance case for one particular example. Our case study focuses on 3dfim+, a medical image analysis software package that supports Functional Magnetic Resonance Imaging (fMRI). 3dfim+ was selected because it is a reasonably small (approximately 1700 lines of code) and easy to understand example of medical image analysis software. We targeted medical image analysis software, since some of the common fMRI statistical analyses data have not yet been validated [29] and because a recent study [11] has shown a potentially serious flaw in software commonly used to analyze fMRI data. More detail on 3dfim+ can be found in Section 3.

The scope of our work does not include redeveloping or reimplementing 3dfim+. Our goal is to build an assurance case for the existing software 3dfim+, by treating it as black box. We consider only the executable for 3dfim+ and the existing documentation and correspondingly produce new documentation and testing results, but not new code. Excluding the code makes the case study more realistic, since, if an assurance case exercise were to be conducted in industry, there would be little appetite for reimplementation. Considerable effort has already gone into writing medical image analysis; it is not feasible for the community to start over.

To argue for the correctness of 3dfim+, we developed an assurance case with the top goal of " Program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment." We also developed a Software Requirements Specification (SRS) document that contains all the necessary information and mathematical background needed to understand 3dfim+. This document can be used for validation and verification activities, and appears many times as evidence in our assurance case. The SRS was reviewed by a domain expert to provide further evidence. We also developed a test case to illustrate how the results from 3dfim+ can be checked to provide additional evidence of correctness. The full assurance case for 3dfim+ can be found in Nejad 2017 [25]. Excerpts from the full case are given in Section 4.

Besides providing a means to illustrate assurance cases for SCS, the 3dfim+ example provides an opportunity to justify their value for certification. Although no errors were found in the output of the existing software, the rigour of the proposed approach did lead to finding ambiguities and omissions in the existing documentation, and highlighted a potential concern when running the software itself. More importantly, the explicit arguments and artifacts provided through the assurance case provide evidence that can be independently judged for sufficiency. The specific critique for the existing implementation and documentation for 3dfim+ is given in Section 5.

2 OVERVIEW OF ASSURANCE CASES

An assurance case is "[a] documented body of evidence that provides a convincing and valid argument that a specified set of critical claims about a system's properties are adequately justified for a given application in a given environment" [30, p. 5].

The idea of assurance cases (or safety cases) began after a number of serious accidents, starting with the Windscale Nuclear Accident in the late 1950s. This incident was the United Kingdom's most serious nuclear power accident [27] and was instrumental in the government setting up new safety regulations incorporating assurance cases. Although there had not previously been an ignorance of safety concerns, and safety standards and regulatory approaches had been applied as the norm, the previous approaches proved to be insufficient. They lacked interaction between regulators and developers, especially since the developers had more knowledge than the regulators about the safety of their products.

Assurance cases have been widely used in the European safety community for over 20 years to ensure system safety [22]. This methodology has been applied in industries such as aerospace, transportation, nuclear power, and defence [1]. Some other examples include the energy sector, such as oil and gas, aviation infrastructure, such as ground systems, aerospace vehicles, such as space vehicles and aircraft, railways, automobiles, and medical devices, such as pacemakers, and infusion pumps [30]. Also, there were some attempts to develop assurance cases for security sectors [3].

In North America, the medical domain is showing an increased interest in assurance cases. Safety cases are considered to have "the potential to support healthcare organizations in the implementation of structured and transparent systems for patient safety management" [8]. This potential is reflected in the Food and Drug Administration's (FDA's) strong recommendation that manufacturers submit a safety assurance case of any new infusion pumps [40].

Safety cases, and in general assurance cases, require a clearly articulated argument, supported by evidence. An assurance case consists of a Top Goal, which is the main proposition we want our software to satisfy; Sub-Goals, which provide the decompositions of the Top Goal; Strategy, which presents the rationale adopted while making arguments and choosing sub-goals; and, Evidence, which is the evidence supporting the argument. Figure 1 shows what an assurance case might look like, using Goal Structuring Notation (GSN) for goals, context and assumptions.

For our work we have chosen the popular Goal Structuring Notation (GSN), developed by Kelly [23] to make our arguments clear, easy to read and, hence, easy to challenge. To develop the assurance case, we used Astah (http://astah.net/) to create and edit our GSN arguments. A full overview of GSN, with examples, can be found in Sprigg's book [38].

Focusing on the assurance case from the start of a project can improve the efficiency of the development process. Scientific software, such as medical software, is often subject to standardization and regulatory approval. While applying such approvals and standards has had a beneficial effect on system quality, it does not provide good tracking of the development stages, as the compliance with the standards are mostly checked after the system development. Once a system is implemented, its documentations must be approved by the regulators. This process is lengthy and expensive. In contrast,

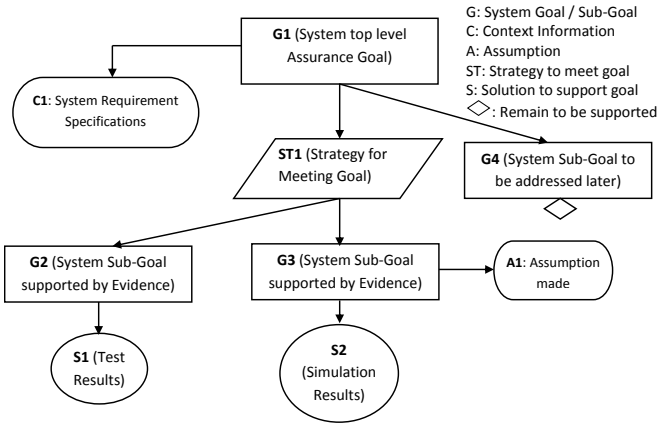

Figure 1: A basic GSN structure [13]

assurance case development usually occurs in parallel with the system construction, resulting in a traceable, detailed argument for the desired property. Moreover, assurance cases take a more direct, flexible and explicit approach. They are flexible enough to incorporate all existing assurance activities and artifacts in any step of the procedure. Therefore, developing an assurance case does not necessarily require much additional effort, and it potentially reduces costs, saves time and gives greater freedom in accommodating different standards.

3 OVERVIEW OF 3DFIM+

3dfim+ [42] is a tool in the Analysis of Functional NeuroImages (AFNI) package (https://afni.nimh.nih.gov/). 3dfim+ analyzes the activity of the brain by computing the correlation between an ideal signal and the measured brain signal. The ideal signal is defined by the user. For instance, the ideal signal could be a square wave, as shown in red in Figure 2. For ease of comparison, the corresponding value of the measured activity in Figure 2 is scaled between 0 and 1. This figures shows high correlation between ideal and measured signals. The correlation between the ideal signal and the measured activity can be calculated at each voxel (volume element) in the full 3D image of the brain.

Figure 2 shows the ideal signal versus the brain activity for one voxel in the full 3D image of the brain. This analysis is completed for every voxel. The results can be visualized using the tools in the AFNI. Figure 3 shows the AFNI environment, in which we can see the brain from different aspects with 2 parts shown in red and blue. These parts of the brain are those which are, respectively, the most positively correlated and the most negatively correlated to the ideal signal.

As mentioned in Section 2, assurance cases are usually developed in parallel with the system construction. Given that 3dfim+ already exists, this was not an option for our current case study. This means that we have to be particularly vigilante to avoid the bias problem. We do not want to prove correctness of 3dfim+ with a flawed argument simply because that is what we set out to do. Since we do not have a vested interest in the correctness of 3dfim+, this is likely less of a problem than it might generally be.

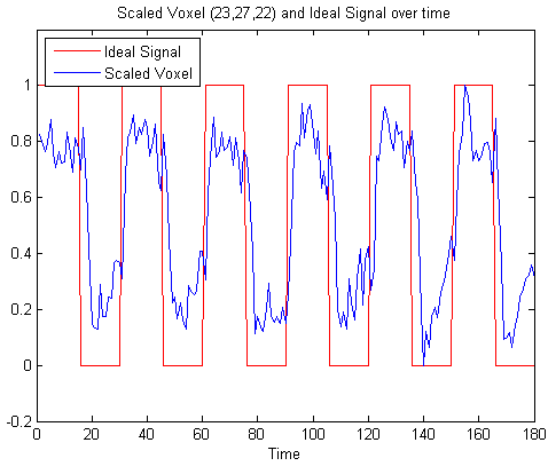

Figure 2: Ideal signal versus activity of the voxel at position (23,27,22) over time

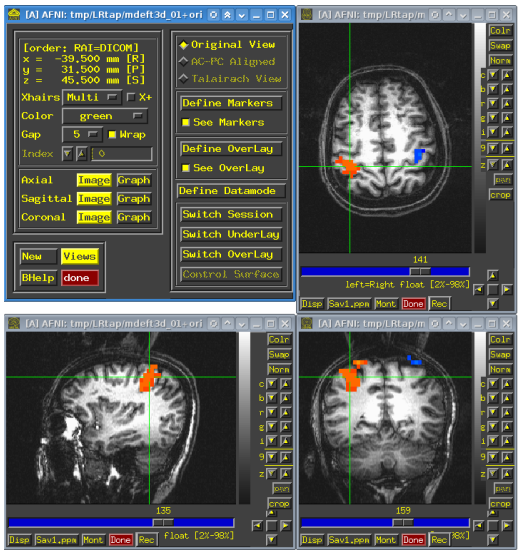

Figure 3: AFNI environment and visualizing the active parts of the brain (from https://commons.wikimedia.org/wiki/File:AFNI_screenshot.png)

4 ASSURANCE CASE FOR 3DFIM+

We have used the guidance provided in "General Principles of Software Validation; Final Guidance for Industry and FDA Staff" [6] to develop our assurance case. This guide outlines generally recognized validation principles that are FDA acceptable for the medical software validation. It was prepared by the International Medical Device Regulators Forum (IMDRF) in an attempt to provide globally harmonized principles concerning medical device software. The general principles document includes software, like 3dfim+, that is itself considered a medical device.

The presentation of the assurance case starts with an overview of the GSN arguments. This is followed by summaries and excerpts

from the evidence used to support the argument. The evidence includes the Software Requirements Specification (SRS), Test Cases, and Domain Expert Review.

4.1 Assurance Case

Our assurance case consists of many sub-claims, which means it cannot legibly by represented on a single page. Therefore, we will only include a representative subset of the argument, which has been split to separately show the sub-structures. The full assurance case can be found in Nejad 2017 [25].

We have to label all parts of the assurance case structure, i.e. all goals, evidence, and contexts, so that our arguments can be discussed and reviewed unambiguously. There are a number of strategies to do this [38, p. 32–33]. For the ease of navigation, we prefer a hierarchical scheme; top goals in each sub-structure are labeled with a word or a letter but without a number (for example G) and then their sub-goals are labeled as G.1, G.2, ... and the sub-goals of G.1 and G.2 are labeled, respectively, as G.1.1, G.1.2, ... and G.2.1 and G.2.2, ... and so on. The evidence is labeled in a similar way. Contexts, strategies, evidence and justifications are labeled alphabetically if more than one context, strategy or justification is used for an argument; for example, C_Ga, C_Gb, C_Gc, ... for contexts and S_Ga, S_Gb, S_Gc for strategies of the Goal G and so on.

When splitting a goal into its sub-goals, the rationale behind the choice of sub-goals might be obvious to the reader or might require further explanation. In a case the rationale is not clear, we explain it using strategies.

We have defined our top goal as "Program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment." The truth of a claim depends on its context; therefore, we must be explicit about what we mean by each term in our goal statement. We could include the details with the goal statement itself, but then it would be too long and would lose its focus. The solution is to declare the context separately. We have defined each term in the top goal in several contexts. We have also made an assumption that must be considered. The assumption and contexts are shown in Figure 4.

As previously done for medical device assurance cases [43], we have divided the top goal into four sub-goals, as shown in Figure 5. The first sub-goal (GR) argues for the quality of the documentation of the requirements. To make an overall argument for correctness, we need a specification to judge correctness against. The second sub-goal (GD) says that the design complies with the requirements and the third proposes that the implementation also complies with the requirements. The fourth sub-goal (GA) claims that the inputs to 3dfim+ will satisfy the operational assumptions because we need valid input to make an argument for the correctness of the output.

The top level of the assurance case in Figure 5 does not imply that up-front requirements are needed. This is fortunate, since scientists have the view that requirements are impossible to determine up-front, since they believe that details can only emerge as the work progresses [5, 34]. The assurance case needs requirements, but they can come out of the development process in any way appropriate for the developers. That is, the documentation can be "faked" like it is part of a rational design process [28].

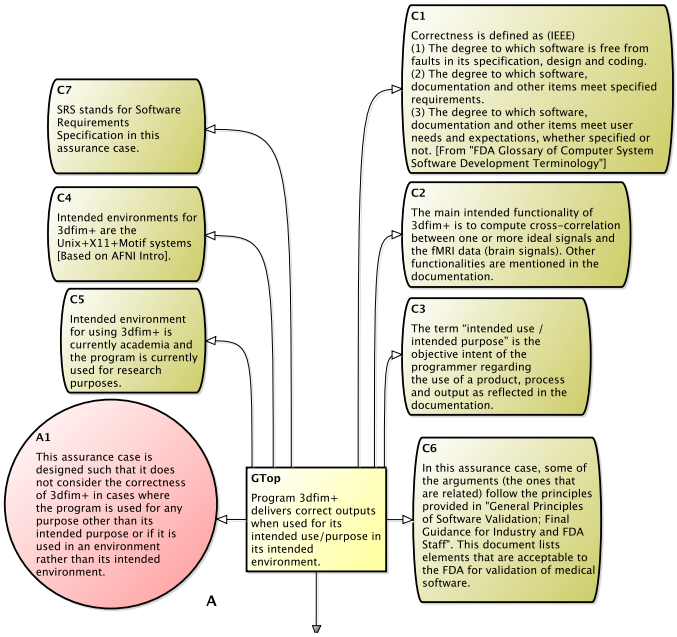

Figure 4: Contexts and Assumption in Top Goal

The main focus in our assurance case is arguing for GR (quality requirements documentation). The decomposition of GR into its sub-goals is shown in Figure 6. This decomposition is based on the IEEE standard 830-1993 [2]. This standard states that good documentation of requirements should be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable and traceable. Using the IEEE resource increases confidence in the argument and makes it more compelling. Hence, our sub-goals address correctness, unambiguity, completeness, consistency, verifiability, modifiability and traceability of the requirements documentation. "Ranked for importance and/or stability" is excluded from the sub-goals in Figure 6 because our domain is SCS. For 3dfim+ to work, all of the requirements are considered to be of equal importance. This is shown as justification J_GRb in Figure 6.

The arguments for consistency, completeness, and correctness were combined together in goal G_3C. These qualities were grouped because, according to some publications, such as " The Three Cs of Requirements: consistency, completeness, and correctness" [44], there is an important relationship between completeness, consistency and correctness for software requirements. Improving one of these three qualities may diminish the others. From another perspective, correctness is a combination of consistency and completeness. So it is important to consider these 3 qualities together. The argument for completeness is partially based on the argument for the readiness of a business plan from Spriggs [38, p. 30]. Due to space limitations, the full details of this argument are not included here.

A sample expansion for the sub-goal of modifiability from Figure 6) is shown in Figure 7. Modifiability is a quality attribute of the software architecture that relates to "the cost of change and refers to the ease with which a software system can accommodate

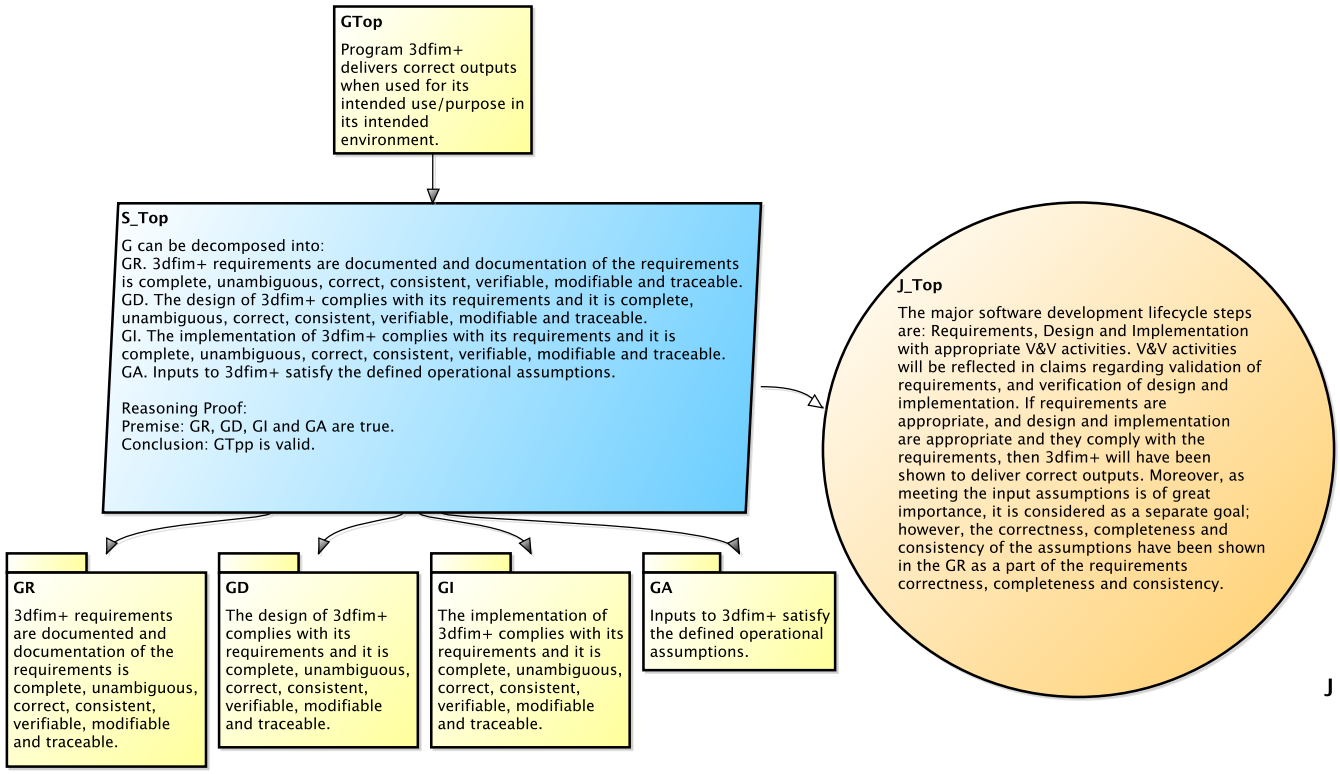

Figure 5: Top Goal of the assurance case and its sub-goals

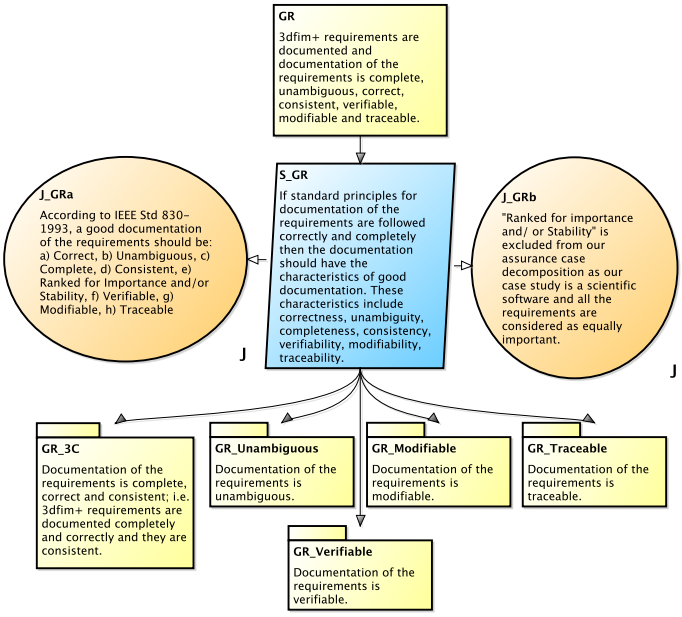

Figure 6: GR decomposition

changes" [26]. Modifiability generally requires a requirement documentation to have a coherent and easy-to-use organization with a table of contents, an index, and explicit cross-referencing. Moreover, requirements should not be redundant and they must be expressed separately. As for the other qualities, the argument for modifiability makes use of the generic evidence template (Figure 8) discussed below.

The content of the documentation of the requirements must be reviewed and verified by domain experts. Spriggs [38, p. 37] gives a decomposition for this argument. We have developed a similar decomposition in our template modules, called GenericEvidence, as shown Figure 8. GenericEvidence is a generic argument. The generic argument is often called a "pattern". "A pattern in this context is an argument that applies to a class of things, which you can use as the basis of an argument for a specific instance" [38, p. 103]. We have developed this module to re-use it for several arguments in our assurance case. We have an argument that a particular quality of the requirements documentation has been met; the main evidence items are the acceptance report and the addressed comments submitted by the reviewers. If we want to ensure that another quality has been met, we would not want to start our argument again from scratch. It would be better to use the same module (sub-structure), but bring in a new evaluation, comments and sections in the report as evidence. In that case, we could just have the name of the quality in the module, but publish the argument stating exactly which quality is reviewed. For instance, for the sake of completeness, we verified that all statements made in the original documentation are reflected in the new documentation. This comparison is mentioned as GenericEvidence.3 in Figure 8.

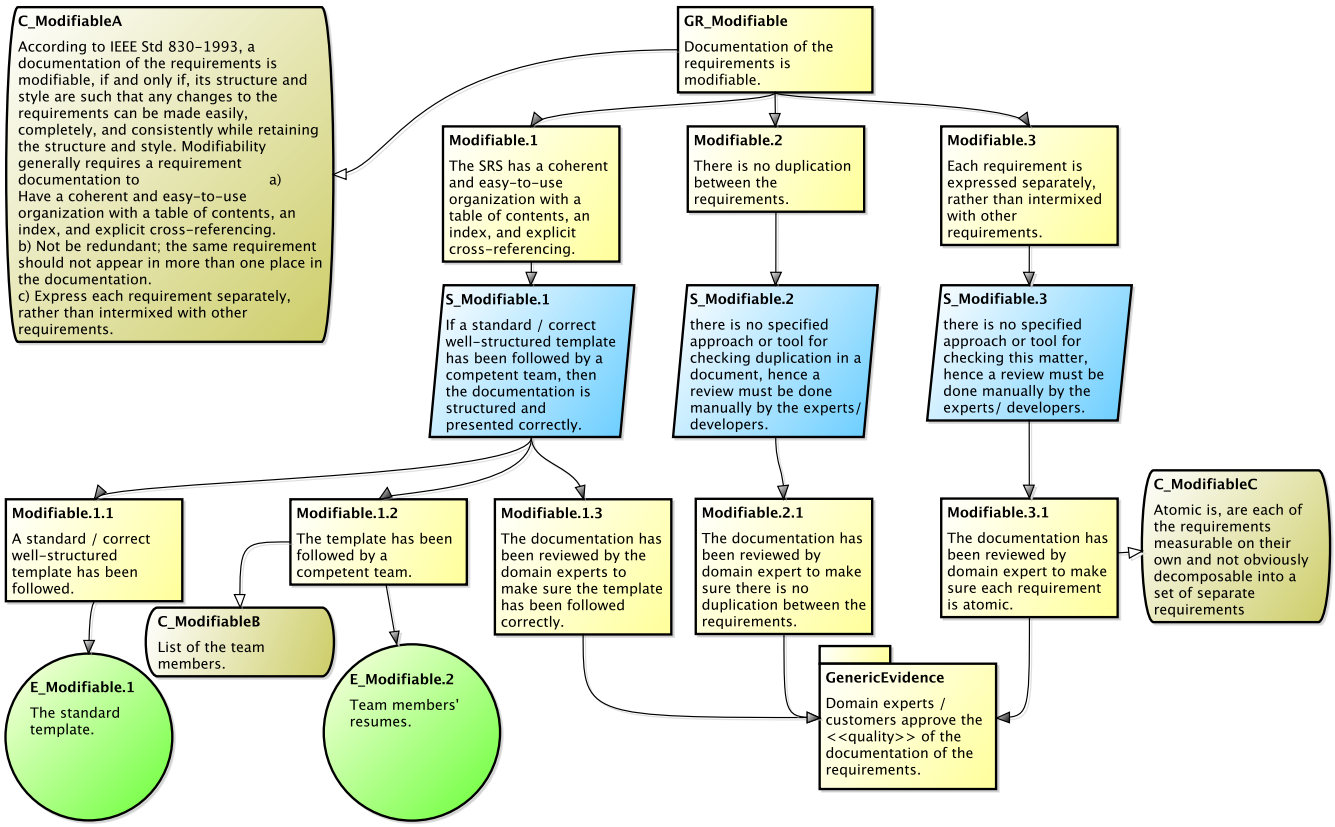

Figure 7: Argument for modifiability of documentation of requirements

E_GenericEvidence.1 in Figure 5 mentions the acceptance criteria for reviewers' resumes. This information is included in the assurance case to mitigate against the bias problem mentioned in Section 2. Reviewers need to be qualified, and we should say what qualified means before we start looking for a reviewer. When we document the assurance case, we verify that the experts satisfy these criteria. If not, we have to make an argument why they should still be considered experts. This draws attention to the fact that there is something "unusual" here that may typically be overlooked.

In Figure 5 we defined GA as "Input(s) to 3dfim+ satisfies the defined operational assumptions." Achieving this goal relies on the software to check if the input is valid as well as the user to make sure the input they give to the program is valid. The user has some responsibilities for the use of 3dfim+, in the same sense that an automobile driver has responsibilities to operate their vehicle safely. This argument for GA is shown in Figure 9. This argument makes explicit that the user has responsibility for validating the input. The software can do automated checks, like verify that the measured activities are positive, but the software can never tell if 3dfim+ is the right tool for the job. For instance, the statistical model for 3dfim+ is parametric, if a non-parametric model would be more appropriate, the user will have to select another tool. Although not currently part of 3dfim+, we added a warning message, as part of the assurance case, that users be explicitly reminded of their responsibilities while running the software.

4.2 Software Requirements Specification

Having a Software Requirements Specification (SRS) is critical for software validation [6]. The requirements mentioned in the assurance case for goal GR (Figure 6) are documented in the SRS. This document is necessary to verify correctness, since it provides a specification against which correctness can be judged. As a consequence, the SRS is mentioned in the sub-goals and evidence for several goals. For instance, in Figure 7, for modifiability, the SRS is mentioned in goal Modifiable.1. Figure 8 for generic evidence references the SRS in E_GenericEvidence.4, by calling for a requirements acceptance report. Goal GA (Figure 9) for the operational assumptions imposes several requirements on the SRS, such as in E_GA.2 where mention is made of SRS content related to assumptions and data constraints.

Due to space limitations, only some representative excerpts from the SRS for 3dfim+ can be reproduced here. The excerpts selected are intended to give an overall feel for the document and to highlight some areas where the rigour of the assurance case provides benefits for the documentation and software quality.

4.2.1 SRS Template. Writing an SRS generally starts with a template, which provides guidelines and rules for documenting the requirements. The assurance case supports the need for a template

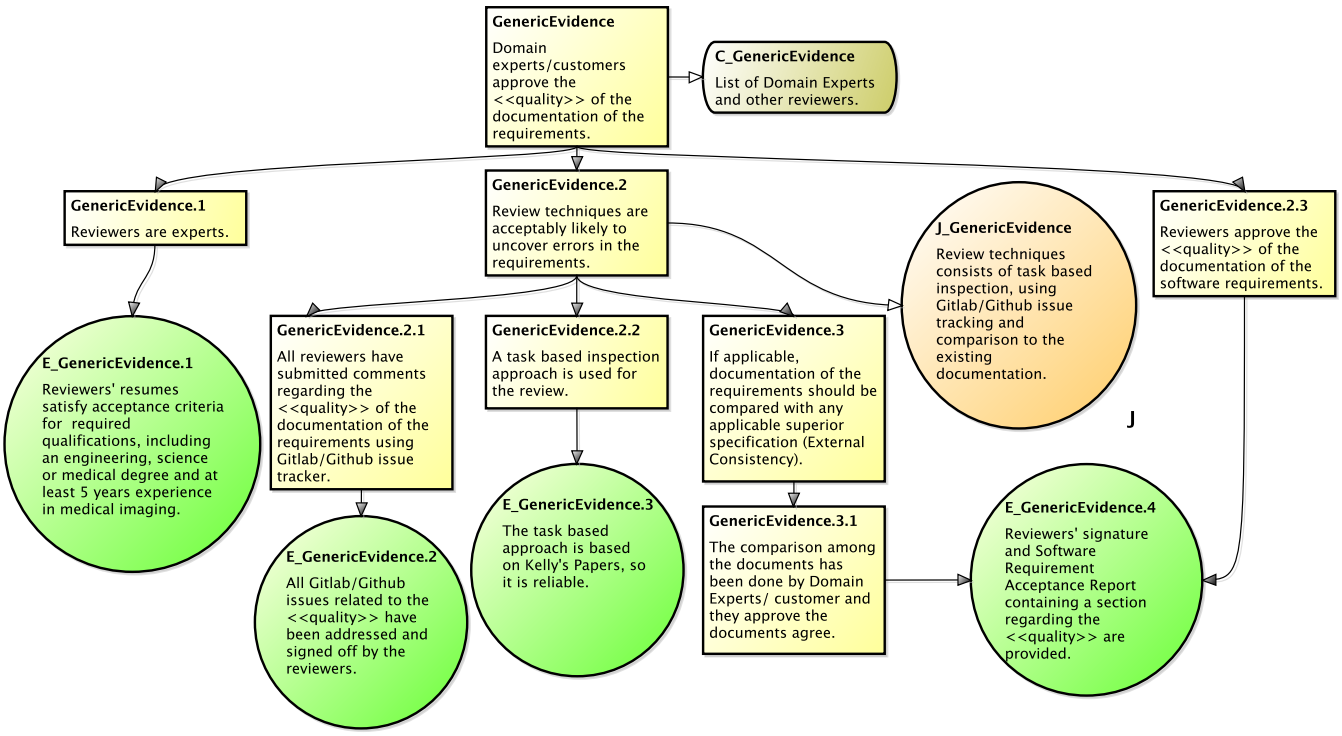

Figure 8: Generic evidence module used as a pattern in our assurance case

through the modifiability goal (Figure 7) Modifiable.1.1: "A standard-/correct well-structured template has been followed." Several existing templates contain suggestions on how to avoid complications and how to achieve qualities such as verifiability, maintainability and reusability [12, 18, 24]. However, no template is universally accepted. For the current purpose, a good starting point is a template specifically designed for scientific software [36, 37], as illustrated in the table of contents below. The recommended template is suitable for science, because of its hierarchical structure, which decomposes abstract goals to concrete instance models, through the support of data definitions, assumptions and terminology. The document's structure facilitates its maintenance and reuse [36], by using separation of concerns, abstraction and traceability.

(1) Reference Material
  (a) Table of Units
  (b) Table of Notations
  (c) Table of Symbols
  (d) Abbreviations and Acronyms
(2) Introduction
  (a) Purpose of Document
  (b) Scope of Requirements
  (c) Organization of Document
(3) General System Description
  (a) System Context
  (b) User Characteristics
  (c) System Constraints
(4) Specific System Description
  (a) Problem Description
    (i) Background
    (ii) Terminology Definition
    (iii) Coordinate Systems
    (iv) Physical System Description
    (v) Goal Statements
  (b) Solution Characteristics Specification
    (i) Assumptions
    (ii) Theoretical Models
    (iii) Data Definitions
    (iv) Instance Models
    (v) Data Constraints
    (vi) Properties of a Correct Solution
(5) Requirements
  (a) Functional Requirements
  (b) Non-functional Requirements
(6) Other System Issues
(7) Traceability Matrix
(8) Likely Changes

4.2.2 Goals. The high level objectives of the software are documented in the goals (Section 4.a.v of the SRS template shown in Section 4.2.1). A sample goal for 3dfim+ is:

G1: Estimate the Pearson correlation coefficients between the (best) ideal time series and the fMRI time series at each voxel over time.

4.2.3 Assumptions. An assumption (Section 4.b.i of the SRS template) highlights a simplification made for the purpose of the

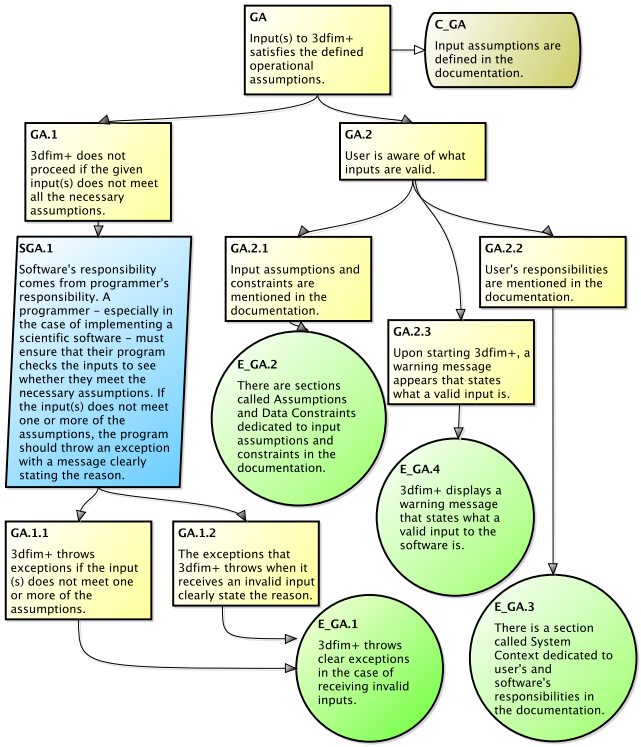

Figure 9: Argument for inputs satisfying the defined operational assumptions

mathematical modelling. Sample assumptions for 3dfim+ are given below:

A1: The variables should be either of type interval or ratio.
A2: There is a linear relationship between the two variables.
A3: The variables are bivariately normally distributed.

4.2.4 Theoretical Models. The theoretical models are sets of governing equations or axioms that are used to model the problem described in the problem definition section (SRS Section 4.b.ii). Traceability exists between the theoretical model and the other components of the documentation. For instance, the description for the T1 (Pearson Correlation Coefficient), which is given below, references the definition for mean (DD1) and several assumptions, including the three listed above.

Number	**T1**
Label	**Calculating Pearson Correlation Coefficient**
Equation	$\rho(A,B) = \frac{\sum_{i=1}^{n}(a_i-\bar{a})(b_i-\bar{b})}{[\sum_{i=1}^{n}(a_i-\bar{a})^2 \sum_{i=1}^{n}(b_i-\bar{b})^2]^{\frac{1}{2}}}$
Description	The equation calculates Pearson correlation coefficients ρ applied to two datasets $A : \mathbb{R}^n$ and $B : \mathbb{R}^n$ both of size n. $\bar{a}$ and $\bar{b}$ are sample means (DD1) of A and B, respectively. ρ is the Pearson correlation coefficient between A and B. Assumptions A1–A5 must hold when calculating this correlation.

4.2.5 Coordinate Convention. Section 4.a.iii of the SRS documents describes the coordinate system for the fMRI images. This information is necessary to make the requirements unambiguous, since there are several choices for coordinate system for medical images. As an example, the SRS defines the Anatomical Coordinate System, which describes the standard anatomical position of a human being using 3 orthogonal planes: axial/transverse (plane parallel to the ground that separates the body into head (superior) and tail (inferior) positions), coronal/frontal (plane perpendicular to the ground that divides the body into front (anterior) and back (posterior) positions), and sagittal/median (plane that divides the body into right and left positions. 3dfim+ uses NIfTI data files that store voxels from right to left to create rows, rows from anterior to posterior to create slices and slices from superior to inferior to create volumes.

4.2.6 Rank Function. Calculating the Spearman and Quadrant correlation coefficients [42] requires the use of the rank function. The rank function is defined in the SRS as a data definition. The details of the description for the rank function in the SRS are given below.

The rank of data points is determined by sorting them in an ascending order and assigning a value according to their position in the sorted list. If ties exist, the average of all of the tied positions is calculated as the rank. Mathematically, the rank of element a in dataset A is defined as follows:

$\text{rank}(a, A) : \mathbb{R} \times \mathbb{R}^n \to \mathbb{N}$
$\text{rank}(a, A) \equiv \text{avg}(\text{indexSet}(a, \text{sort}(A)))$

$\text{indexSet}(a, B) : \mathbb{R} \times \mathbb{R}^n \to \text{ set of } \mathbb{N}$
$\text{indexSet}(a, B) \equiv \{j : \mathbb{N} | j \in [1..|B|] \wedge B_j = a : j\}$

$\text{sort}(A) : \mathbb{R}^n \to \mathbb{R}^n$
$\text{sort}(A) \equiv B : \mathbb{R}^n, \text{ such that}$
$\forall(a : \mathbb{R} | a \in A : \exists(b : \mathbb{R} | b \in B : b = a) \wedge \text{count}(a, A) = \text{count}(b, B)) \wedge \forall(i : \mathbb{N} | i \in [1..|A| - 1] : B_i \leq B_{i+1})$

$\text{count}(a, A) : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{N}$
$\text{count}(a, A) : +(x : \mathbb{N} | x \in A \wedge x = a : 1)$

$\text{avg}(C) : \text{ set of } \mathbb{N} \rightarrow \mathbb{R}$
$\text{avg}(C) \equiv +(x : \mathbb{N} | x \in C : x)/|C|$

The above equations use the Gries and Schneider notation [15, p. 143] for set building and evaluation of an operator applied over a set of values. Specifically, the expression $(*x : X | R : P)$ means application of the operator $*$ to the values P for all x of type X for which range R is true. In the above equations, the $*$ operators include $\forall$, $\exists$ and + are used.

4.3 Test Cases

To verify the implementation of 3dfim+, we developed test cases based on the functional requirements documented in the SRS. The results of the test cases are used as evidence for goal GI (Figure 5), which argues that the implementation matches the SRS. Since our case study is for scientific software, verification through testing is challenging. The source of the challenge is that SCS differs from most other software because the quantities of interest are continuous, as opposed to discrete. As shown for the calculation of the Pearson correlation coefficient (Section 4.2.4), the inputs and outputs are continuously valued real variables. Validating the requirements is difficult because there are an infinite number of potential input values, many of which cannot be represented as floating point numbers. In general for SCS, the correct value for the output variable is unknown. That is, SCS problems typically lack a test oracle [21]. Fortunately for 3dfim+, the correlation calculations are based on finite sets of real numbers, so constructing a pseudo oracle using Matlab was relatively straightforward.

We developed one test case per each functional requirement, to compare their results with the results of 3dfim+. As an example, we had a test case to check the correctness of the Pearson Correlation Coefficient, which is one of the main functionalities of 3dfim+. We used our Matlab pseudo oracle and AFNI to visualize the results and obtain the indices of voxels. Our input consisted of 180 frames of 64×64×28 images. In this test case, we found the minimum and the maximum Pearson correlation coefficients and their locations. For this test and others, we achieved the same results for both 3dfim+ and our independently developed Matlab script.

However, the testing was not without its challenges. Agreement between the Matlab pseudo oracle and 3dfim+ took a considerable amount of time to achieve, because the coordinate systems conventions for Matlab and AFNI are different. Since this information was not documented in the original 3dfim+ manual, we were unaware of this subtly. The coordinate system description for 3dfim+ was added to the SRS (as described in Section 4.2.5) after our struggles with achieving test case agreement.

In the case of 3dfim+ a pseudo oracle was available. For other SCS software, other techniques may be needed for the verification that the implementation matches the requirements. Where appropriate, use can be made of the Method of Manufactured Solutions [31] and metamorphic testing [19]. For testing purposes, the slower, but guaranteed correct, interval arithmetic [17] can be used to ensure that calculated answers lie within the guaranteed bounds. Verification tests can also include plans for convergence studies. The discretization used in the numerical algorithm should be decreased (usually halved) and the change in the solution assessed. Although not used in the current example, verification can also use non-testing techniques, such as code walkthroughs, code inspections, and correctness proofs etc. [14, 41].

4.4 Domain Expert Review

An important component of evidence for our assurance case is the domain experts review. Review of the SRS is important to reach a common understanding between the software engineers and scientists. As mentioned in the introduction (Section 1), building an assurance case facilitates bridging the gap between software engineers and scientists.

Domain expert review appears in our assurance case as "Domain experts/customers approve the «quality» of the documentation of the requirements." This corresponds to GenericEvidence.2.3 in Figure 8. To ensure our SRS is of high quality, a task-based inspection approach was used [23]. For the review process we assigned a set of tasks asking questions about each section of the SRS. We used Github (https://github.com/) issue tracking for assigning the tasks and for discussion. Two sample review questions are reproduced below.

Q7: Please let us know if all symbols in Theoretical Model T1 (from Section 4.2.4) are defined. Is enough information provided that you could calculate the Pearson correlation coefficient if you are given datasets A and B.

Q10: Please let us know if Data Definition DD4 (Rank Function) (from Section 4.2.6) is explained clearly or needs any additional information. Please let us know if the notation we are using for this function is clear and understandable.

A domain expert that completed the review has a degree in engineering and over 10 years experience in medical imaging. He therefore meets the acceptance criteria given in E_GenericEvidence.1 in Figure 5. The reviewer went through all the assigned tasks and provided answers/suggestions. For the most part, the SRS did not need to be modified as a result of the expert review. However, some of the symbols in the SRS, such as $\mathbb{N}$ for the set of natural numbers, were clarified as a result of the discussion with the expert reviewer.

5 PROPOSED CHANGES TO 3DFIM+

The assurance case not only provides an explicit argument for correctness that can be verified/replicated by third parties. The process of developing and documenting the assurance case also provides an opportunity to improve the software and its documentation. Although several potential improvements were noticed for 3dfim+ during the course of this research, this is not a criticism of the original 3dfim+ software and its documentation. The goal of the assurance case is to provide certifiable software, but the original software did not have this goal. It was written for researchers, not for clinicians. The users for 3dfim+ and readers of its documentation are likely to be domain experts. However, even for the existing audience for 3dfim+, there will likely be some novices, since nobody starts out as an expert. The improvements noted below would likely interest new users, since the new documentation is more complete and less ambiguous than the original. This benefit of improving

software and its documentation was previously observed during the course of retroactively writing an SRS for nuclear safety analysis software [35].

Given the different audience that was envisioned, the original documentation would not satisfy the GR goal (Figure 6) for high quality requirements. The existing documentation is not fully complete, unambiguous, correct, consistent, verifiable, modifiable or traceable. One of the main ambiguities is through the absence of documentation on the coordinate system (Section 4.2.5). As mentioned previously, the absence of any specification related to the coordinate system meant that comparing the 3dfim+ results to an independent calculation of the correlation was difficult. Additional investigation was necessary to find the necessary details so that both results were expressed in the same coordinate system. Another ambiguity, due to incomplete documentation, was for the definition of the rank function (Section 4.2.6). In the original documentation the specific definition of the rank function is not given. When there are ties in the data, some means has to be found to determine the rank of each entry. For instance, all ties could be given the same rank, and then a gap could be left in the ranking numbers. Alternatively, ties could get the same rank, but no gap could be included before listing the next ranking number. Five different ranking algorithms can be found on https://en.wikipedia.org/wiki/Ranking. The one actually used by 3dfim+ gives the same ranking number to all ties, with the ranking number being equal to the mean of what they would have under ordinal ranking. This fact was not determined from the documentation, but by investigating the C code for 3dfim+. It is the specification for this algorithm that is given in Section 4.2.6.

Although the software for 3dfim+ did not show any errors in its output, some subtle concerns were raised by considering the assurance case GA for satisfying the operational assumptions (Figure 9). As shown in GA.1 3dfim+ should not proceed if the input does not match the necessary assumptions. However, the actual software does not actually check the input data. GA.2 ("User is aware of that inputs are valid") is also not considered for 3dfim+. The input assumptions are not made explicit in the documentation and the user is not warned that it is their responsibility to provide valid data. As mentioned previously, the user has the responsibility of determining whether the statistical model used by 3dfim+ provides the right tool for them.

6 FUTURE WORK

Based on the current work and our review of past work on assurance cases, we have identified a number of directions for the future development of assurance cases, as follows:

- Additional Examples to Create a Template: Additional examples of documentation for SCS should be conducted to determine an ideal template for SCS assurance cases. Much of the current work can be reused in other projects, but determining the specifics will require investigating different SCS problems, especially outside of the medical imaging domain.
- Work on a New SCS Project from the Start: The current work produced the assurance case, SRS and test plan a posteriori. As mentioned in Section 2, assurance cases work best when they are used from the start of a project. A particular benefit for research purposes will be a likely increase in the workload that can be assigned to domain reviewers, since they will likely have a greater vested interest in the success of the project than when it is a purely academic exercise.
- Tool Support Improvement: Currently, there is no tool that provides an abstraction of goals and sub-goals to handle the complexity of the assurance case structure. For instance, it would be nice to hide the details of a goal (or a context, justification, evidence or assumption) and only show the title. This would improve readability. Details could be revealed via clicking on the goals, or context etc.
- Publishing Examples of Practical Assurance Cases: Currently, many existing assurance cases are not released due to proprietary rights. The more presentations on adoption of assurance cases and case studies, the better resources we have to learn about assurance cases.
- Extension to Other Areas: Assurance cases can be used in other areas that require assurance. Assurance cases have been used for safety, and now for SCS. It would be potentially valuable to extend the idea to other area, like networking.
- Adding Formality to Assurance Cases: The means of expressing confidence in assurance cases and the top-level claims may benefit from further formality and rigour. Adding formality could justify the completeness and consistency of claim decomposition and the credibility of the evidence.

7 CONCLUDING REMARKS

This work has motivated assurance cases for SCS. Assurance cases have already been effectively used for safety cases for real time systems. For SCS their advantages include engaging domain experts, producing only necessary documentation, and providing evidence that can potentially be verified/replicated by a third party. The engagement of the domain experts is noteworthy because scientist end user developers have historically shown a distrust of software engineering techniques and principles. In particular, SCS developers tend not to favour full documentation of requirements. However, their motivation should improve because an assurance case shows the necessity and value of an SRS. As more examples and tools become available, adoption of assurance cases in SCS should become more prevalent.

How to document an assurance case for SCS was illustrated via the example of the medical image analysis software, 3dfim+. The 3dfim+ software analyzes activity in the brain by computing the correlation between the measured and an ideal brain signal. This example was partly chosen because of recent concerns about the validity of fMRI (Functional Magnetic Resonance Imaging) studies. The concerns centre around whether a parametric model is appropriate for fMRI data. Although the software itself cannot determine whether it is the appropriate model, the user should ask themselves this question. The assurance case highlighted how the user can be informed of the mathematical assumptions of the model through the SRS, and the details of their responsibility through in-program warnings.

The value of assurance cases for SCS was justified. Although no errors were found in the software outputs from 3dfim+, the exercise did highlight problems with the original documentation

and software. The existing documentation was shown to have ambiguities and omissions, such as an incompletely defined ranking function and missing details on the coordinate system convention adopted. In addition, a potential concern for the software itself was identified. As mentioned above, running the software does not produce any warning about the obligation of the user to provide data that matches the parametric statistical model employed for the correlation calculations.

8 ACKNOWLEDGMENTS

The assistance of Dr. Michael Noseworthy, from St. Joseph's hospital and McMaster University, is gratefully acknowledged. The 3dfim+ case study came directly from his advice and fMRI demonstration. Thanks also to Dr. Dean Inglis for fulfilling the role of expert reviewer.

REFERENCES

[1] Peter G. Bishop and Robin E. Bloofield. 1989. A methodology for safety case development. In *Industrial Perspectives of Safety-critical Systems: Proceedings of the Sixth Safety-critical Systems Symposium, Birmingham 1998*. Springer-Verlag, London, UK, 1–10.

[2] Fletcher J. Buckley, A.M. Davis, and J.W. Horch. 1993. *IEEE Recommended Practice for Software Requirements Specifications*. Technical Report. The institute of Electrical and Electronics Engineers, Inc., New York, USA.

[3] Charles B.Weinstock, Howard F.Lipson, and John Goodenough. 2007. *Arguing Security – Creating Security Assurance Cases*. Technical Report. Software Engineering Institute Carnegie Mellon University, 4500 Fifth Avenue, Pittsburgh, PA. http://resources.sei.cmu.edu/asset_files/WhitePaper/2013_019_001_293637.pdf

[4] Canadian Nuclear Safety Commission (CNSC). 2000. *Computer Programs Used in Design and Safety Analyses of Nuclear Power Plants and Research Reactors*. Technical Report G-149. Minister of Public Works and Government Services Canada.

[5] Jeffrey C. Carver, Richard P. Kendall, Susan E. Squires, and Douglass E. Post. 2007. Software Development Environments for Scientific and Engineering Software: A Series of Case Studies. In *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*. IEEE Computer Society, Washington, DC, USA, 550–559. https://doi.org/10.1109/ICSE.2007.77

[6] CDRH. 2002. *General Principles of Software Validation; Final Guidance for Industry and FDA Staff*. Technical Report. US Department Of Health and Human Services Food and Drug Administration FDA, Center for Devices and Radiological CDRH, Health Center for Biologics Evaluation and Research, Rockville, MD.

[7] Center for Devices and Radiological Health, CDRH. 2002. *General Principles of Software Validation; Final Guidance for Industry and FDA Staff*. Technical Report. US Department Of Health and Human Services Food and Drug Administration Center for Devices and Radiological Health Center for Biologics Evaluation and Research, York, England.

[8] G.M. Cleland, M.A. Sujan, I. Habli, and J. Medhurst. 2012. *Evidence: Using Safety Cases in Industry and Healthcare*. Health Foundation, London. https://books.google.ca/books?id=o8z-Ms9o3DMC

[9] CSA. 1999. *Quality assurance of analytical, scientific, and design computer programs for nuclear power plants*. Technical Report N286.7-99. Canadian Standards Association, 178 Rexdale Blvd. Etobicoke, Ontario, Canada M9W 1R3.

[10] CSA. 2009. *Guideline for the application of N286.7-99, Quality assurance of analytical, scientific, and design computer programs for nuclear power plants*. Technical Report N286.7.1-09. Canadian Standards Association, 5060 Spectrum Way, Suite 100, Mississauga, Ontario, Canada L4W 5N6, 1-800-463-6727.

[11] Anders Eklunda, Thomas Nichols, and Hans Knutssona. 2016. A methodology for safety case development. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)* 113, 28 (2016), 7900–7905.

[12] ESA. February 1991. *ESA Software Engineering Standards, PSS-05-0 Issue 2*. Technical Report. European Space Agency.

[13] Dipak Gade and Santosh Deshpande. 2015. Assurance Driven Software Design using Assurance Case Based Approach. *International Journal of Innovative Research in Computer and Communication Engineering* 3, 10 (October 2015), 9121–9127.

[14] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. 2003. *Fundamentals of Software Engineering* (2nd ed.). Prentice Hall, Upper Saddle River, NJ, USA.

[15] David Gries and Fred B. Schneider. 1993. *A logical approach to discrete math*. Springer-Verlag Inc., New York.

[16] John Hatcliff, Mats Heimdahl, Mark Lawford, Tom Maibaum, Alan Wassyng, and Fred Wurden. 2009. A Software Certification Consortium and its Top 9 Hurdles. *Electronic Notes in Theoretical Computer Science* 238, 4 (2009), 11–17. https://doi.org/10.1016/j.entcs.2009.09.002

[17] Timothy Hickey, Qun Ju, and Maarten H. Van Emden. 2001. Interval Arithmetic: From Principles to Implementation. *J. ACM* 48, 5 (Sept. 2001), 1038–1068. https://doi.org/10.1145/502102.502106

[18] IEEE. 1998. *Recommended Practice for Software Requirements Specifications*. Technical Report IEEE Std 830-1998. The institute of Electrical and Electronics Engineers, Inc. 1–40 pages. https://doi.org/10.1109/IEEESTD.1998.88286

[19] Upulee Kanewala and Anders Lundgren. 2016. Automated Metamorphic Testing of Scientific Software. In *Software Engineering for Science*, Jeffrey C. Carver, Neil Chue Hong, and George Thiruvathukal (Eds.). Taylor & Francis, Boca Raton, FL, Chapter Examples of the Application of Traditional Software Engineering Practices to Science, 151–174.

[20] Diane F. Kelly. 2007. A Software Chasm: Software Engineering and Scientific Computing. *IEEE Software* 24, 6 (2007), 120–119. https://doi.org/10.1109/MS.2007.155

[21] Diane F. Kelly, W. Spencer Smith, and Nicholas Meng. 2011. Software Engineering for Scientists. *Computing in Science & Engineering* 13, 5 (October 2011), 7–11.

[22] T.P. Kelly. 1999. *Arguing Safety – A Systematic Approach to Safety Case Management*. Ph.D. Dissertation. York University, Department of Computer Science Report YCST.

[23] Tim Kelly. 2003. *A Systematic Approach to Safety Case Management*. Technical Report 04AE-149. SAE International.

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

[25] Mojdeh Sayari Nejad. 2017. *A Case Study in Assurance Case Development for Scientific Software*. Master's thesis. McMaster University, Hamilton, ON, Canada.

[26] L. Northrop. 2004. Achieving Product Qualities Through Software Architecture Practices. (2004). http://www.sei.cmu.edu/architecture/cseet04.pdf

[27] Office for Nuclear Regulation. 2016. A guide to Nuclear Regulation in the UK. (2016). http://www.onr.org.uk/documents/a-guide-to-nuclear-regulation-in-the-uk.pdf

[28] David L. Parnas and P.C. Clements. 1986. A Rational Design Process: How and Why to Fake it. *IEEE Transactions on Software Engineering* 12, 2 (February 1986), 251–257.

[29] Cyril Pernet and Tom Nichols. 2016. Has a software bug really called decades of brain imaging research into question? (September 2016). https://www.theguardian.com/science/head-quarters/2016/sep/30/has-a-software-bug-really-called-decades-of-brain-imaging-research-into-question

[30] David J. Rinehart, John C. Knight, and Jonathan Rowanhill. 2015. *Current Practices in Constructing and Evaluating Assurance Cases with Applications to Aviation*. Technical Report CR-2014-218678. National Aeronautics and Space Administration (NASA), Langley Research Centre, Hampton, Virginia.

[31] Patrick J. Roache. 1998. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, New Mexico.

[32] John Rushby. 2015. *The Interpretation and Evaluation of Assurance Cases*. Technical Report SRI-CSL-15-01. Computer Science Laboratory, SRI International, Menlo Park, CA. Available at http://www.csl.sri.com/users/rushby/papers/sri-csl-15-1-assurance-cases.pdf.

[33] Judith Segal. 2008. Models of Scientific Software Development. In *Proceedings of the First International Workshop on Software Engineering for Computational Science and Engineering (SECSE 2008)*. In conjunction with the 30th International Conference on Software Engineering (ICSE), ACM, Leipzig, Germany, 1–6. http://www.cse.msstate.edu/~SECSE08/schedule.htm

[34] Judith Segal and Chris Morris. 2008. Developing Scientific Software. *IEEE Software* 25, 4 (July/August 2008), 18–20.

[35] W. Spencer Smith and Nirmitha Koothoor. 2016. A Document-Driven Method for Certifying Scientific Computing Software for Use in Nuclear Safety Analysis. *Nuclear Engineering and Technology* 48, 2 (April 2016), 404–418. https://doi.org/10.1016/j.net.2015.11.008

[36] W. Spencer Smith and Lei Lai. 2005. A New Requirements Template for Scientific Computing. In *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, J. Ralyté, P. Ågerfalk, and N. Kraiem (Eds.). In conjunction with 13th IEEE International Requirements Engineering Conference, IEEE, Paris, France, 107–121.

[37] W. Spencer Smith, Lei Lai, and Ridha Khedri. 2007. Requirements Analysis for Engineering Computation: A Systematic Approach for Improving Software Reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation* 13, 1 (February 2007), 83–107.

[38] John Spriggs. 2012. *GSN - The Goal Structuring Notation, A Structured Approach to Presenting Arguments*. Springer, Hayling Island, UK. https://doi.org/10.1007/978-1-4471-2312-5

[39] United States Department of Energy. 2003. *Assessment Criteria and Guidelines for Determining the Adequacy of Software Used in the Safety Analysis and Design of Defense Nuclear Facilities*. Technical Report CRAD - 4.2.4.1. Office of Environment, Health, Safety & Security, Department of Energy, United States of America.

[40] U.S. Food and Drug Administration. 2014. Infusion Pumps Total Product Life Cycle: Guidance for Industry and FDA Staff. on-line. (December 2014).
[41] Hans van Vliet. 2000. *Software Engineering (2nd ed.): Principles and Practice.* John Wiley & Sons, Inc., New York, NY, USA.
[42] B. Douglas Ward. 2000. *Program 3dfim+.* Biophysics Research Institute, Medical College of Wisconsin.
[43] Alan Wassyng, Neeraj Kumar Singh, Mischa Geven, Nicholas Proscia, Hao Wang, Mark Lawford, and Tom Maibaum. 2015. Can Product-Specific Assurance Case Templates Be Used as Medical Device Standards? *IEEE Design & Test* 32, 5 (2015), 45–55. https://doi.org/10.1109/MDAT.2015.2462720
[44] Didar Zowghi and Vincenzo Gervasi. 2013. *The Three Cs of Requirements: Consistency, Completeness, and Correctness.* Technical Report. Faculty of Information Technology University of Technology, Sydney , Australia.