ASSURANCE CASE DEVELOPMENT FOR SCIENTIFIC SOFTWARE

A CASE STUDY IN ASSURANCE CASE DEVELOPMENT FOR SCIENTIFIC SOFTWARE

By

Mojdeh Sayari Nejad, B. Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Master of Science in Computer Science

McMaster University

MASTER OF SCIENCE (2017)	McMaster University
(Computer Science)	Hamilton, Ontario, Canada

TITLE:	A Case Study in Assurance Case Development for Scientific Software
AUTHOR:	Mojdeh Sayari Nejad, B.Sc.
SUPERVISOR:	Dr. Spencer Smith
NUMBER OF PAGES:	xiv, 173

Abstract

Assurance Cases have been effectively used for improving the safety of real-time safety systems. However, until now, Assurance Case techniques have not been applied to building confidence in the correctness of Scientific Computing (SC) software.

Our approach is to employ Assurance Case techniques to the case of a specific medical image analysis software, 3dfim+, and then generalize the results/template for other medical and SC software. Using the Goal Structuring Notation (GSN), we develop an Assurance Case to support the top goal that "Program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment." This claim is supported by several sub-claims, including the claims that high quality requirements exist and that the implementation complies with the requirements. The full argument decomposes each sub-claim further until at the bottom level evidence is provided. The evidence provided includes the requirements documentation, test cases and expert review. To simplify the Assurance Case diagram, a new generic module, parameterized over quality, was developed to argue that each quality has been achieved.

Evaluation of the full Assurance Case shows that this approach is feasible for building confidence in SC software, even in the practical situation where confidence is sought, but redesign and reimplementation are not possible. The exercise uncovered issues with the original documentation for 3dfim+, including missing assumptions, and ambiguity with the chosen sign convention. Furthermore, although no errors in output were found, the Assurance Case highlights that confidence in the original 3dfim+ software could be improved through additional checks for input validity.

Acknowledgments

I would like to thank all the people who contributed in some way to the work described in this thesis.

First and foremost, I would like to express my sincere thanks and gratitude to my supervisor Dr. Spencer Smith for his motivation, patience, and the continuous support of my master's studies and research. His guidance helped me in all the time of research and writing of this thesis.

Next, I would like to acknowledge Dr. Michael D. Noseworthy for his assistance with selecting the case study and obtaining the required data for my research.

I also need to thank Dr. Dean Inglis for his valuable contribution to reviewing my SRS document.

My sincere thanks also go to Dr. Alan Wassyng and his research group for sharing their immense knowledge of Assurance Cases with us and for the productive meetings we had every Thursday.

I would also like to thank Dr. Zahra Keshavarz Motamed who showed interest in my work and graciously agreed to serve on my examination committee.

Last but not the least, I would like to thank my parents for supporting me throughout my life. This accomplishment would not have been possible without their constant love and continuous encouragement.

Contents

List of Figures

List of Abbreviations

3dfim+	**3D**imensional **F**unctional **I**ntensity **M**ap+
AFNI	**A**nalysis of **F**unctional **N**euroImages
BOLD	**B**lood **O**xygen **L**evel **D**ependent
CAE	**C**laims **A**rguments **E**vidence
CMS	**C**ompact **M**uon **S**olenoid
EBS	**E**lectrical **B**rain **S**timulation
FDA	**F**ood and **D**rug **A**dministration
fMRI	**f**unctional **M**agnetic **R**esonance **I**maging
GSN	**G**oal **S**tructuring **N**otation
HSE	**H**ealth and **S**afety **E**xecutive
IMDRF	**I**nternational **M**edical **D**evice **R**egulators **F**orum
LHC	**L**arge **H**ardon **C**ollider
MR	**M**agnetic **R**esonance
MRI	**M**agnetic **R**esonance **I**maging
NASA	**N**ational **A**eronautics and **S**pace **A**dministration
NIfTI	**N**euroimaging **Inf**ormatics **T**echnology **I**nitiative
RSC	**R**ailway **S**afety **C**ase
SaMD	**S**oftware **a**s a **M**edical **D**evice
SAP	**S**afety **A**ssesment **P**rinciples
SC	**S**cientific **C**omputing
SEI	**S**oftware **E**ngineering **I**nstitute
SRS	**S**oftware **R**equirements **S**pecification

Chapter 1

Introduction

> *"Many scientists and engineers spend much of their lives writing, debugging, and maintaining software, but only a handful have ever been taught how to do this effectively: after a couple of introductory courses, they are left to rediscover (or reinvent) the rest of programming on their own. The result? Most spend far too much time wrestling with software, instead of doing research, but have no idea how reliable or efficient their programs are." [14]*

- Greg Wilson

Scientists, engineers and society in general are placing an increasing reliance on Scientific Computing (SC) software for decision making and scientific analysis and discovery. A close look at SC software reveals that its size and complexity can cause real challenges for reliability. For instance, the Compact Muon Solenoid (CMS) project [15] consists of software to analyze data from a particle detector designed to discover new fundamental physics particles produced in high-energy collisions in the Large Hadron Collider (LHC). So far, this software project contains around 4.3 million lines of code [15]. If a new fundamental particle is discovered, the physicists need to have confidence that it is real, and not the product of a software error. They also do not want to wonder if a software error has caused them to miss discovering a new particle. Other examples of SC software where correctness is of paramount importance include nuclear safety analysis software and medical image construction and analysis software. The challenge for medical imaging software includes complex algorithms that must operate on disparate and relatively large datasets.

Despite the importance of this software, many SC developers do not seem to pay enough attention to the reliability and correctness of their software. Several techniques exist for building confidence in correctness, such as testing techniques and code reviews, but these techniques seem to be employed in an ad-hoc manner. Each project developer employs the subset of available techniques that satisfies him or her, but we should hold correctness to a higher standard. The goal should be that the evidence needs to be adequate to convince an independent third party. Moreover, we should aim to facilitate a common understanding, so that all specialists and generalist can effectively work together on scientific software projects, while increasing project correctness and acceptance. What we need is an explicit argument, with appropriate evidence, that acceptable confidence in correctness has been achieved. This realization motivates us to investigate Assurance Cases methods and techniques for SC software.

Most of the techniques for building confidence in software correctness are applied after the software implementation is done. Figure 1.1, reported by the Systems Sciences Institute at IBM, shows the relative cost to fix software defects in different stages of software development. As we can see, when a defect gets through during the development process, the earlier it is diagnosed, the easier and cheaper is the rectification of the defect.

Another stage of software development that is not mentioned in Figure 1.1 is the

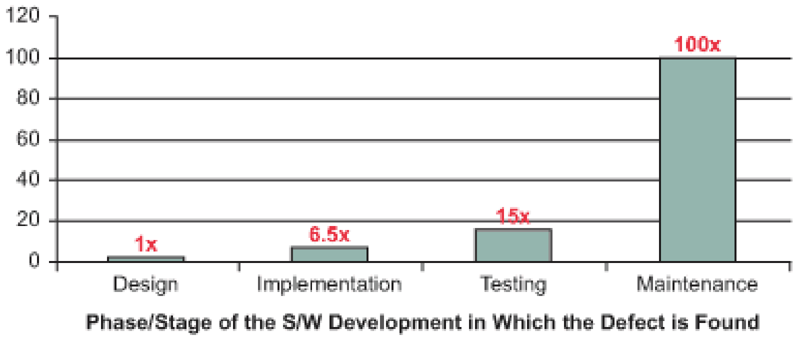


Figure 1.1: Relative costs to fix software defects [1]

documentation and software requirements specification development. In this stage, correcting a defect would be cheaper than the next stage, which is software design. What we need is to check for defects during each stage of software development. This is another prominent idea behind building Assurance Cases.

Assurance Cases, which will be discussed in detail in Chapter 2, have effectively been used for improving the safety of real-time safety systems [4]. However, Assurance Case techniques have not been employed to build a case for SC software correctness. Our goal is to employ the techniques on the specific case of medical image analysis software and then generalize it for other medical and SC software.

In this chapter, first we give an overview of Assurance Cases and our case study. Then we explain our approach to increase reliability in the correctness of our case study.

1.1 Overview of Assurance Cases

According to the definition given by the GSN committee, 2011 [4, p. 5], 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".

Assurance cases were first introduced as safety cases. Safety cases were developed because although safety was being considered, it was often hard to see an overall systematic assurance argument. Moreover, the safety approaches and regulations did not seem to be sufficient to ensure system safety. These regulations and approaches were mostly employed by regulators. But the reality is that developers have more knowledge about what makes their product safe than the regulators. Hence, safety cases were introduced.

Safety cases and in general assurance cases require a clearly articulated argument, supported by evidence. An assurance case mainly consists of a Top Goal, which is the main subject we want our system to be consistent with, Sub-Goals, which are the decompositions of the Top Goal, Strategy, which presents the rationale adopted while making arguments and choosing sub-goals and Solution, which is the evidence supporting the argument. Other elements of assurance cases are mentioned in Chapter 2. Figure 1.2 gives an example of what an assurance case looks like.

In this figure, the Top Goal is " The equipment will fail no more often than once in then thousand operating hours". According to the Strategy S2.3.1, the Top Goal is decomposed into 5 Sub-Goals. Each of the sub-goals then are supported by evidence.

The argument in Figure 1.2 is build using a notation called Goal Structuring Notation (GSN). This is the same notation we have used to build and present our assurance case in this thesis. More details on the GSN is given in Chapter 2. We present our assurance case in Chapter 4.

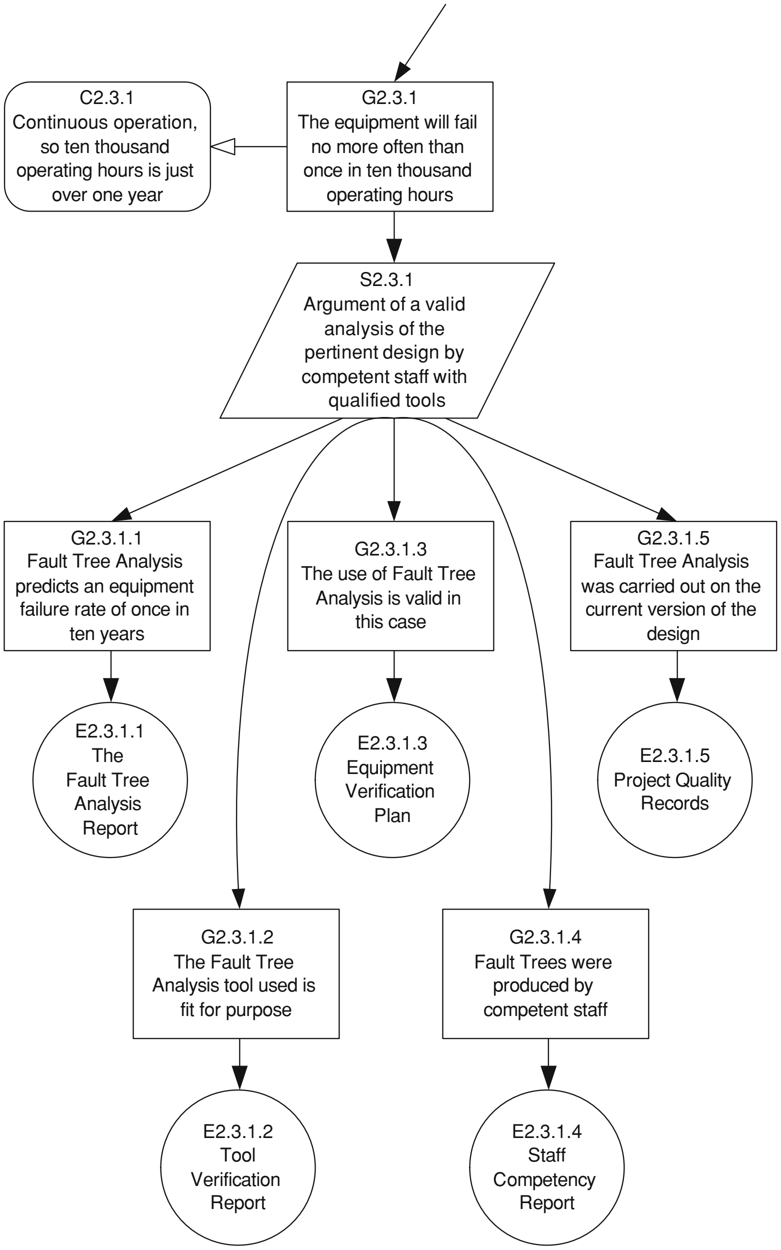

C2.3.1
Continuous operation, so ten thousand operating hours is just over one year
G2.3.1
The equipment will fail no more often than once in ten thousand operating hours
S2.3.1
Argument of a valid analysis of the pertinent design by competent staff with qualified tools
G2.3.1.1
Fault Tree Analysis predicts an equipment failure rate of once in ten years
G2.3.1.3
The use of Fault Tree Analysis is valid in this case
G2.3.1.5
Fault Tree Analysis was carried out on the current version of the design
E2.3.1.1
The Fault Tree Analysis Report
E2.3.1.3
Equipment Verification Plan
E2.3.1.5
Project Quality Records
G2.3.1.2
The Fault Tree Analysis tool used is fit for purpose
G2.3.1.4
Fault Trees were produced by competent staff
E2.3.1.2
Tool Verification Report
E2.3.1.4
Staff Competency Report

Figure 1.2: An example of an assurance case [2]

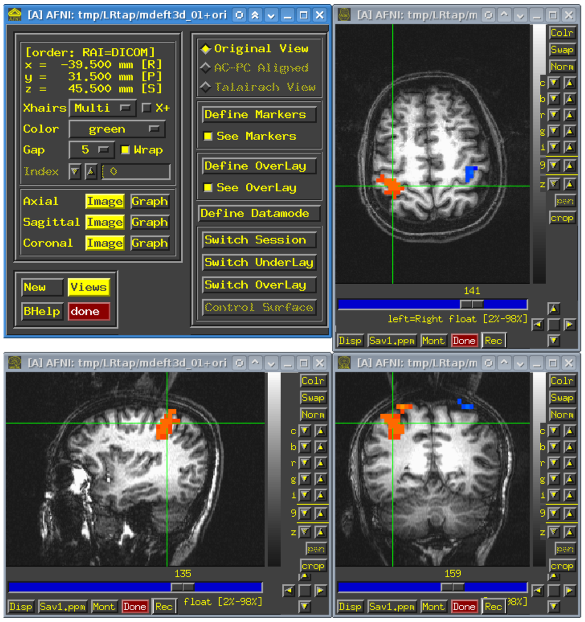
1.2 Overview of Case Study

The purpose of this thesis is to provide a guideline for developing assurance cases to validate and verify SC software. While the eventual goal is developing assurance cases for SC software, first we build an assurance case for a specific case study. Our case study, called 3dfim+, is a medical image analysis software based on the Functional Magnetic Resonance Imaging (fMRI) technique. The motivations for selecting 3dfim+ are mentioned in the next paragraph.

Despite being in use for more than 25 years, some of the common fMRI statistical analyses data have not yet been validated [16]. Furthermore, a recent study [17] has shown that a potentially serious flaw in software commonly used to analyze fMRI data has been discovered, which might invalidate some of the related prior work. According to the US Food and Drug Administration (FDA), in an analysis of 3140 medical device recalls from 1992 to 1998, 242 of them were attributable to software failures [18]. Since the existence of such errors and flaws in this type of software is probable, there is a great interest at this time in raising standards and methods to validate and verify medical software. However, regulatory approval and certifications are expensive and lengthy. One way to reduces the cost and improve the quality is to build assurance cases.

3dfim+ is a tool in the Analysis of Functional NeuroImages (AFNI) package, which is one of the popular packages for processing and displaying fMRI data. It mainly analyzes the activity of the brain by computing the correlation between an ideal signal and the brain signal. The ideal signal is defined arbitrarily by the user. The user need to determine which ideal signal will give them the highest correlation according to the type of the experiment they conduct. The parts of the brain that are correlated to a given ideal signal can be visualized using the tools in the AFNI. Figure 1.3 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-correlated and the least-correlated parts to the ideal signal.



[A] AFNI: tmp/LRtap/mdeft3d_01+ori

```
[order: RAI=DICOM]
x =  -39.500 mm [R]
y =   31.500 mm [P]
z =   45.500 mm [S]
```

Xhairs Multi
Color green
Gap 5 ■ Wrap
Index 0

Figure 1.3: AFNI environment and visualizing the active parts of the brain [3]

More information on 3dfim+ is given in Chapter 3 and Appendix A.

1.3 Overview of Methods

The scope of our work does not include redeveloping or reimplementing 3dfim+. Our goal, as mentioned earlier, is to potentially be able to build an assurance case for the existing software 3dfim+, by treating it as black box.

To verify the correctness of 3dfim+, first 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". Afterwards, we 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 is reviewed by the domain experts 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.

1.4 Thesis Outline

The thesis is organized into four broad parts. In chapter 2, we introduce the assurance case foundation, properties and benefits, and terminology and notation. In chapter 3, we introduce 3dfim+, our case study, and we explain necessary terms and information to understand how the software functions. In chapter 4, we present our assurance case, some sections of our SRS and a test case we developed for verifying and validating the correctness of program 3dfim+. Finally, future work is proposed and conclusions are drawn based on the developed assurance case, SRS and the test case.

Chapter 2

Preliminaries of Assurance Case

In this chapter, we lay out a foundation for understanding assurance cases that includes basic principles, history, terminology, properties and notations. Firstly, the role and importance of assurance cases are discussed. Secondly, we give a survey on the history of assurance cases. Thirdly, the properties of an assurance case are presented. Fourthly, the terminology used for constructing assurance cases is provided. Finally, the notations used for expressing assurance cases are introduced and compared.

2.1 Role of an Assurance Case

To give an overview of assurance cases, we first go through some of the definitions from the literature. In [19, p. 9], an assurance case is defined generally as "A means of increasing well-founded confidence that a system will behave as intended". [4, p. 5] also introduces an assurance case as "An organized argument that a system is acceptable for its intended use with respect to specified concerns (such as safety, security, correctness)". These definitions state the objective of an assurance case;

however, they are vague in terms of how an assurance case increases the confidence and acceptance of a system. A more detailed definition by Rushby [20, p. i] is presented as “Assurance cases are a method for providing assurance for a system by giving an argument to justify a claim about the system, based on evidence about its design, development, and tested behavior”. Rushby’s definition does not point out an organized argument, while one of the benefits of assurance cases is that they provide explicitness through presenting arguments in an organized structure. Furthermore, it does not mention that the reasoning must be true in a given environment. More accurate definitions are given by GSN committee, 2011 [4, p. 5] and Bishop [21], respectively, as “A reasoned and compelling argument, supported by a body of evidence, that a system, service or organization will operate as intended for a defined application in a defined environment” and “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”. While these definitions are good and make explicit the concept of structured argumentation, they do not indicate the motivation of using assurance cases. These definitions do not mention that the development of an assurance case should be integrated with system development from its earlier stage, as opposed to the other standards that are *post facto*, that is they are mostly checked after the development of the system.

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

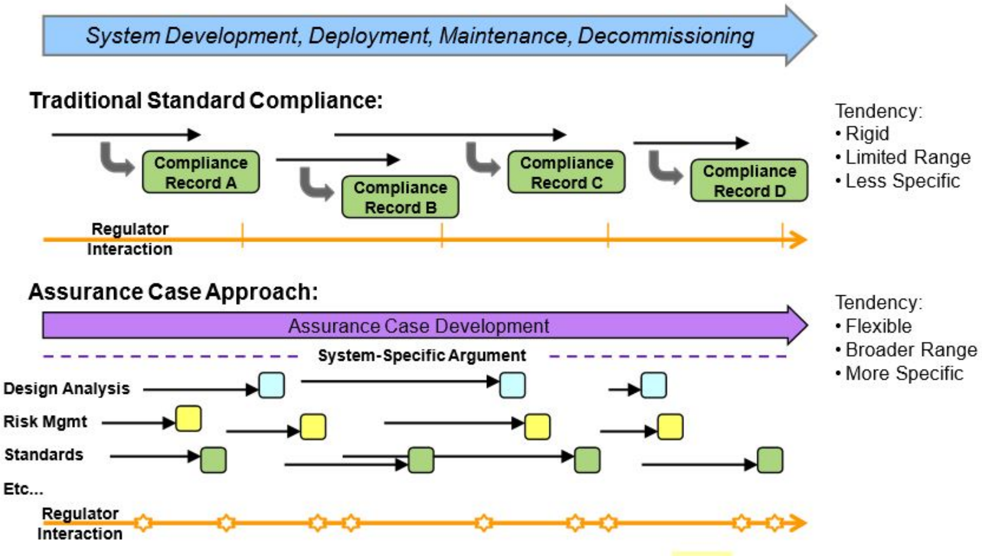


Figure 2.1: Standards-Only vs. Assurance Case Paradigms [4]

expensive. In contrast, 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 reduces the cost, saves time and gives greater freedom in accommodating different standards.

An assurance case does not replace standard compliance. For example, an assurance case itself can indicate what standards are necessary to follow for an argument. In other words, adhering to standards may be a portion of the evidence used in an assurance argument, but it is not adequate by itself. Figure 2.1 compares the assurance case based approach and the standard-based approach. The purpose of this figure is to show that development of an assurance case can be, and should be, done during software development; in contrast to the standard approaches, which are done at the end of software development, or at the end of each phase of software development.

Another significant aspect of assurance case, as noted earlier, is that it presents an organized argument. Without assurance cases, it is hard to have an overall construct of the whole system and its arguments. Assurance case helps us identify where there might be critical quality holes for the system. Moreover, we can identify unnecessary or unrelated arguments or arguments that are missed. With full graphical breakdowns, such as Goal Structuring Notation (GSN), individual claims, arguments, evidence, assumptions, sub-claims, etc. are explicitly stated. We discuss this notation later in this chapter.

Another important feature of the assurance case based approach is an independent assessment. The objective of the independent assessment is to ensure that more than one person or team sees the evidence so as to overcome possible conflicts of interest and blinkered views that may arise from a single assessment.

Another driver for adopting an assurance case based approach is that assurance cases provide an efficient way to connect developer's and stakeholder's contributions to critical system qualities. While other approaches are mostly checked by regulators and not by the developers themselves, assurance cases engage the developers in the system assurance process, which is beneficial as developers have the most knowledge and information over their products and about what improves their quality. Assurance cases also allow all stakeholders to understand their necessary and reasonable roles in liability of the system. Stakeholders get engaged in assurance case development by developing a reasonable level of certainty that the system meets the desired quality at each level. Hence, this approach helps developers and stakeholders both take responsibility for the system.

Given the advantages of assurance cases outlined in the previous paragraphs, developing assurance cases is now a requirement in many fields. For example, currently there is a heavy focus from the US Food and Drug Administration (FDA) on developing assurance cases for medical and safety systems [11]. The motivation for following this approach is to provide an organized structure that demonstrates the desired property is satisfied and to provide a flexible mechanism for an efficient review and involvement of developers, regulators and stakeholders to achieve a knowledge balance.

2.2 History of Assurance Case

As the industrial revolution progressed, the safety of new technologies became a concern and attention began to be paid to prevent the occurrence of failures and learn their causes. In safety-critical industries, manufacturers and operators had to ensure that their systems were adequately safe. The way that safety has been ensured has changed over the past 20 years. Previously, manufacturers and operators claimed safety through the satisfaction of standards and regulatory approvals. However, currently, besides the compliance with standards, operators and manufacturers need to submit a safety case for their systems [5].

Assurance cases were originally developed as safety cases and 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 defense [21]. 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 [4]. Also, there were some attempts to develop assurance cases for security sectors [23]; however, in these cases compliance with the rules is still the preferred approach.

The idea of assurance case began after a number of serious accidents, starting with the Windscale Nuclear Accident in the late 1950s and followed by Piper Alpha

Offshore Oil and Gas Platform Disaster and Clapham Rail Disaster in the 1990s. These accidents drew attention to the safety management in safety-critical sectors. Although there had not been an ignorance of safety concerns in these cases, and safety standards and regulatory approaches had been applied as the norm, these approaches proved to be insufficient for system safety management. Moreover, these approaches lack interaction between regulators and developers which needed to be addressed; hence, a thorough consideration of safety was required.

Currently, in Europe, the responsibility of constructing well-structured arguments has shifted onto the developers to show that their systems achieve acceptable levels of safety. In this section, we outline the history of assurance cases; we describe safety case use in selected safety-critical industries and the reason previous regulatory approaches failed to ensure the system safety.

In the context of this section, the term “safety case” is used for the variety of legislative requirements needed to satisfy the system safety. This term refers to both safety cases and safety reports. The term “installation” is used to describe the site that presents the major hazard. For example, in the nuclear context installation can be a nuclear power station, nuclear fuel fabrication plant, nuclear fuel reprocessing plant, and any facility handling fissionable materials. In the oil and gas platforms, installation can be an oil refinery or an offshore oil and gas facility. Finally the term “operator” describes the owner or employer in charge of the installation.

The nuclear industry has played a major role in development of safety and assurance cases. The most affecting event in this regard was the Windscale accident. This incident was the United Kingdom’s most serious nuclear power accident classified as a Level 5 event [24] and was instrumental in the government setting up new safety regulations. The accident occurred on October 1957 at the Windscale site. The nuclear heating of one of the reactor’s graphite control blocks failed in fully releasing the Winger energy

(Energy stored in the irradiated graphite of a graphite reactor [25]). A second nuclear heating was applied to treat the situation. Attempts to discharge the hot cartridges failed and caused the adjacent uranium cartridges to rupture. The uranium was released and began to oxidize, released radioactivity and caused an explosion. Analysis revealed that the accident was caused by poor staff judgment and faulty instruments; the second nuclear heating was applied too rapidly, which resulted in the burst of one of the cartridges [26]. As a consequence, 2 out of the 3 reactors were shut down. This incident also led to the creation of the NII, which is a part of the Health and Safety Executive (HSE) in 1959 [27]. The HSE published several revised Safety Assessment Principles for Nuclear Facilities (SAPs) in the following years. These principles are primarily intended to be used by the HSE's inspectors to decide whether safety cases for nuclear facilities are adequate [24].

Similar developments have taken place in other industries. In the non-nuclear industrial safety regulations, the fundamentals are the same as the nuclear industry; the primary aim of the safety case is to reduce the probability of an incident occurring. Similarly, for the Britain's offshore oil and gas industry, the first changes to the regulations were prompted by Occidental Petroleum's Piper Alpha Offshore Oil and Gas Platform disaster. This incident occurred on 6 July 1988, in which gas condensate ignited and caused a series of explosions. 165 of the 226 persons on the installation and two of the crew of a rescue craft were killed in only 22 minutes. The death toll was the highest of any accident in the history of offshore operations [28].

Following this accident, the approach being adopted for safety regulations was changed. Nowadays, instead of focusing on prescriptive safety requirements and constructing codes, where the safety is claimed by the regulator, demonstration of safety is done by operators and owners, based on the Offshore Installation (Safety Case) Regulations 2005 [29]. These Regulations were implemented in response to the public inquiry into the Piper Alpha disaster chaired by The Hon Lord Cullen. Lord Cullen's inquiry into the Piper Alpha disaster carefully considered and endorsed the safety case concept which was subsequently applied to the industry. The primary aim of the Regulations is "to reduce the risks from major accident hazards to the health and safety of the workforce employed on offshore installations or in connected activities" [29]. According to these regulations, petrochemical installations need to assess potential hazards and design systems to reliably prevent hazards from happening [28].

According to [30], the UK offshore safety case system has improved the safety management system and had several positive effects. As an evidence, we can consider the modification of the 1970s-built offshore platform. This modification included adding an additional pipeline from another field. During the safety case development, it was realized that there were pressure relief pipes without emergency shutdown valves. Any failure of the pressure relief pipe would cause major consequences. The operator made appropriate modifications according to the safety case. Similar evidence exists that shows operators mitigated the hazardous situations whilst developing a safety case. [30] also states that Australia considered the applicability of the Cullen's report to its offshore hydrocarbon industry and started using a safety case approach.

Currently, safety cases are required for all installations operating in the British waters and in the UK designated areas of the continental shelf. It is not acceptable to operate an installation without having a safety case accepted by HSE [31].

Another example of using safety cases would be the railway industry. Recent changes in the structure of the railways in the United Kingdom brought about by the privatization of British Rail and the introduction of legislation in response to recent railway accidents have introduced a new approach called the Railway Safety case.

On December 1988, an accident at Clapham Junction was caused by a relatively trivial human error [32]. The cause of the disaster was a wire improperly terminated which bypassed a section of safety interlocking circuitry. A signal failed to protect a stationary Basingstoke train, which had been brought to a halt after the signal changed from green to red as it was passed.

The railway industry was shocked by this accident, in which 35 people died and 500 people injured. This accident indicated a need for changing the way in which safety is managed on the railway. As a result, some regulations were introduced in 1994. According to [33], the ultimate objective of the regulations was "To ensure that health and safety standards in the railway industry post-privatization are maintained and, as far as possible, improved". To come up with these regulations, thirteen representative Railway Safety Cases (RSCs) were reviewed [33].

To investigate the safety case approach in the defense industry, we summarize some of the safety case definitions from the standards and handbooks in the following paragraphs. The first definition is taken from the U.K. Ministry of Defense Ship Safety Management System Handbook JSP 430 [34]:

"A safety case is a comprehensive and structured set of safety documentation which is aimed to ensure that the safety of a specific vessel or equipment can be demonstrated by reference to:

- safety arrangements and organization
- safety analyses
- compliance with the standards and best practice
- acceptance tests
- audits
- inspections

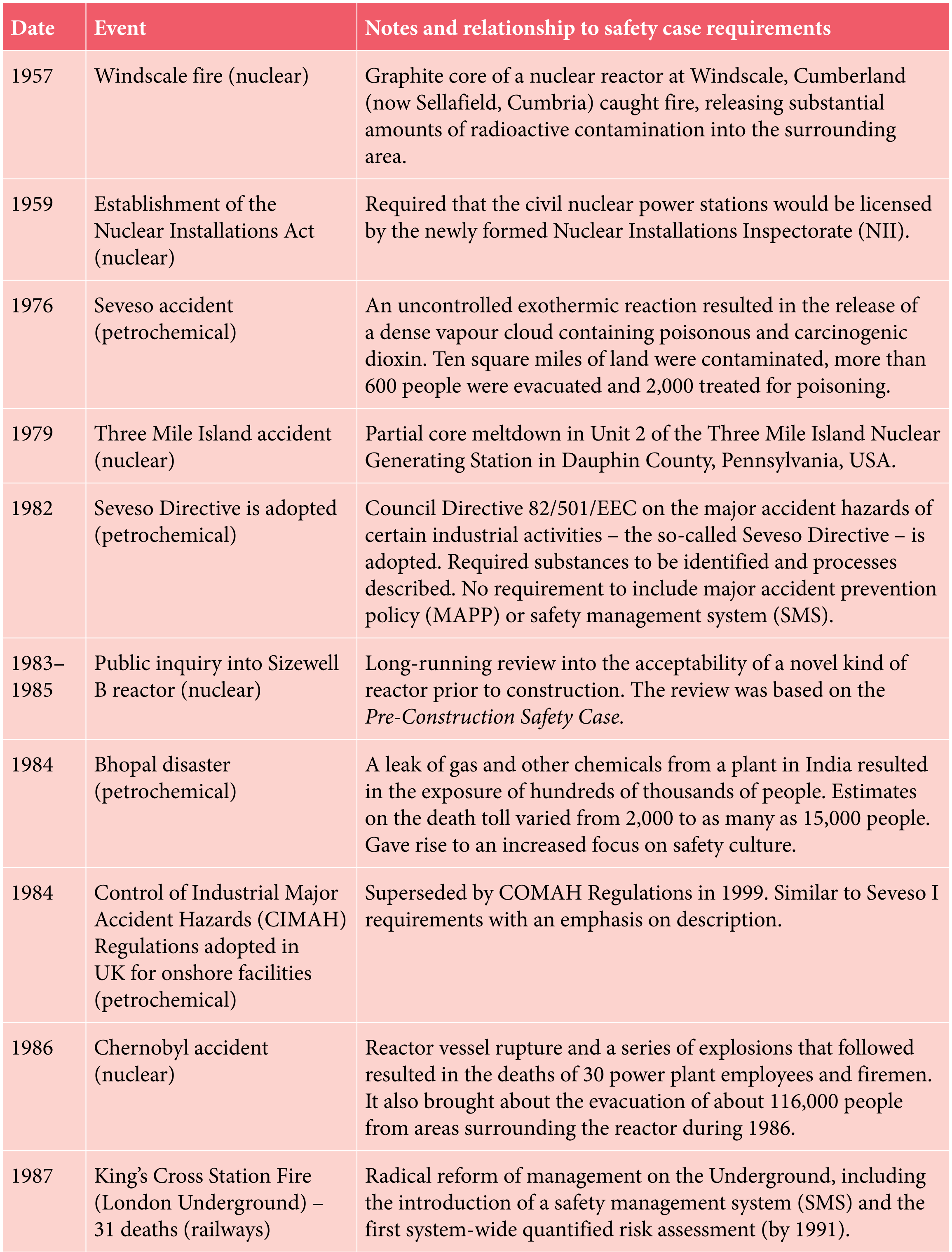
- feedback
- provision made for safe use including emergency arrangement”

Another definition can be found in U.K. Ministry of Defense Standard 00-55 [35] as “The software safety case shall present a well-organised and reasoned justification based on objective evidence, that the software does or will satisfy the safety aspects of the Statement of Technical Requirements and the Software Requirements Specification”.

According to this definition, meeting the requirements ensures some degree of safety. By reviewing the subsequent literature on safety standards, such as the U.K. Defense Standards 00-56 [34], we find that safety case development cannot be left as an activity to be performed towards the end of the system design; it should be an activity that is integrated with the design. “The Safety Case should be initiated at the earliest possible stage in the Safety Programme so that hazards are identified and dealt with while the opportunities for their exclusion exist” [34].

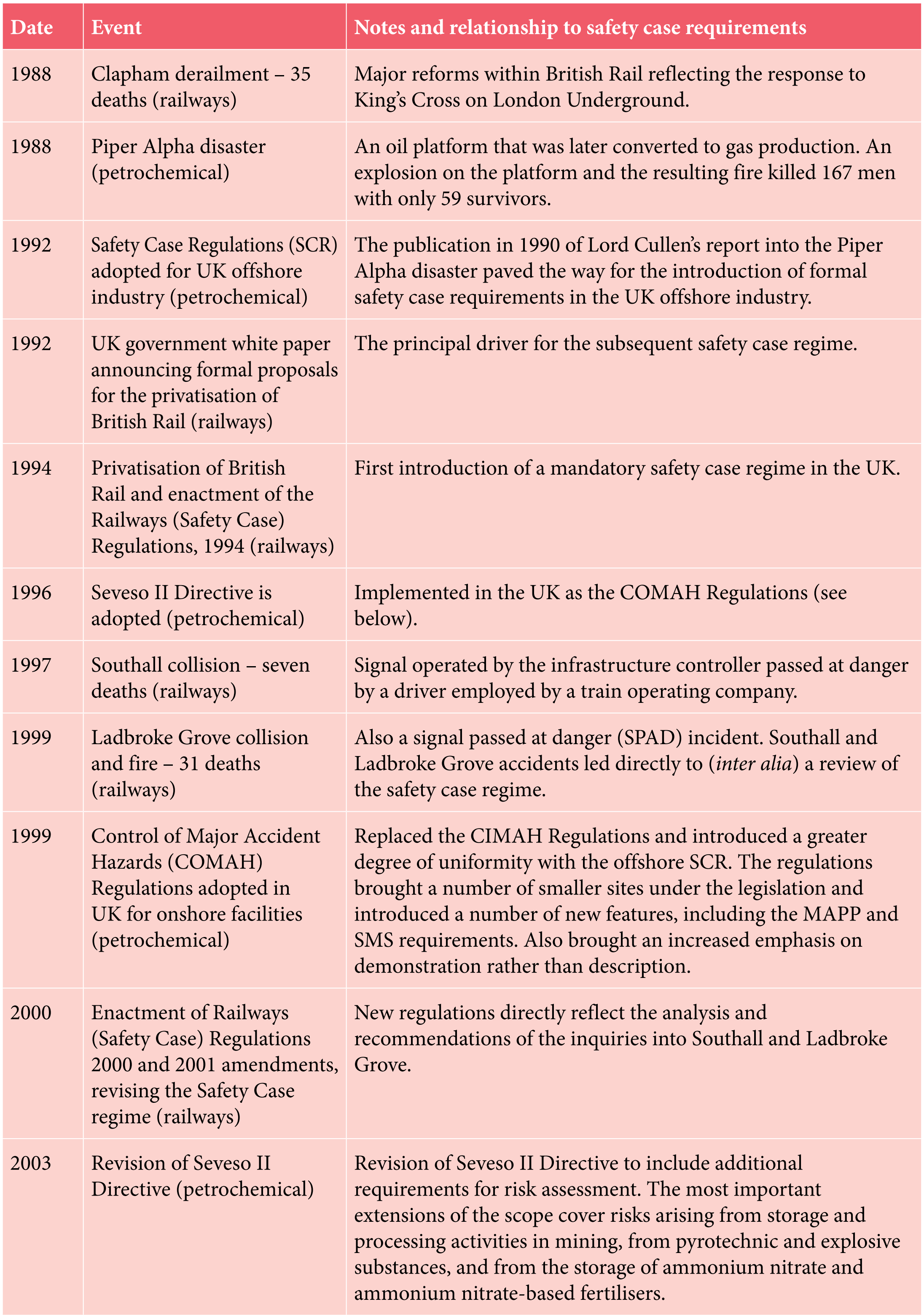
We can see that definitions of acceptable safety have evolved within industry sectors, and the use of safety cases has been emerging. In many UK safety-critical industries, the use of the safety cases is now a regulatory requirement. In Figures 2.2 and 2.3, the significant accidents that resulted in safety regulations changes are mentioned briefly.

In healthcare, there have been first attempts at adopting the safety case concept for medical devices, health informatics and health systems promoted by the FDA [5]. As stated in [36], the very first example of the safety case in this area was the safety case for hospital bed developed by EWICS TC7 medical device group in 2005. The safety case was presented at SAFECOMP 2007 and afterward AdvaMed software group had several meetings with FDA and Software Engineering Institute (SEI) over the safety case. FDA and AdvaMed then held their workshops on assurance cases at 2008. Since then, the FDA has encouraged medical device manufacturers to develop safety



Date	Event	Notes and relationship to safety case requirements
1957	Windscale fire (nuclear)	Graphite core of a nuclear reactor at Windscale, Cumberland (now Sellafield, Cumbria) caught fire, releasing substantial amounts of radioactive contamination into the surrounding area.
1959	Establishment of the Nuclear Installations Act (nuclear)	Required that the civil nuclear power stations would be licensed by the newly formed Nuclear Installations Inspectorate (NII).
1976	Seveso accident (petrochemical)	An uncontrolled exothermic reaction resulted in the release of a dense vapour cloud containing poisonous and carcinogenic dioxin. Ten square miles of land were contaminated, more than 600 people were evacuated and 2,000 treated for poisoning.
1979	Three Mile Island accident (nuclear)	Partial core meltdown in Unit 2 of the Three Mile Island Nuclear Generating Station in Dauphin County, Pennsylvania, USA.
1982	Seveso Directive is adopted (petrochemical)	Council Directive 82/501/EEC on the major accident hazards of certain industrial activities – the so-called Seveso Directive – is adopted. Required substances to be identified and processes described. No requirement to include major accident prevention policy (MAPP) or safety management system (SMS).
1983–1985	Public inquiry into Sizewell B reactor (nuclear)	Long-running review into the acceptability of a novel kind of reactor prior to construction. The review was based on the *Pre-Construction Safety Case.*
1984	Bhopal disaster (petrochemical)	A leak of gas and other chemicals from a plant in India resulted in the exposure of hundreds of thousands of people. Estimates on the death toll varied from 2,000 to as many as 15,000 people. Gave rise to an increased focus on safety culture.
1984	Control of Industrial Major Accident Hazards (CIMAH) Regulations adopted in UK for onshore facilities (petrochemical)	Superseded by COMAH Regulations in 1999. Similar to Seveso I requirements with an emphasis on description.
1986	Chernobyl accident (nuclear)	Reactor vessel rupture and a series of explosions that followed resulted in the deaths of 30 power plant employees and firemen. It also brought about the evacuation of about 116,000 people from areas surrounding the reactor during 1986.
1987	King's Cross Station Fire (London Underground) – 31 deaths (railways)	Radical reform of management on the Underground, including the introduction of a safety management system (SMS) and the first system-wide quantified risk assessment (by 1991).

Figure 2.2: Brief chronological summary of the significant events and changes in the safety regulations for the petrochemical, nuclear and railway industries - part 1 [5]



Date	Event	Notes and relationship to safety case requirements
1988	Clapham derailment – 35 deaths (railways)	Major reforms within British Rail reflecting the response to King's Cross on London Underground.
1988	Piper Alpha disaster (petrochemical)	An oil platform that was later converted to gas production. An explosion on the platform and the resulting fire killed 167 men with only 59 survivors.
1992	Safety Case Regulations (SCR) adopted for UK offshore industry (petrochemical)	The publication in 1990 of Lord Cullen's report into the Piper Alpha disaster paved the way for the introduction of formal safety case requirements in the UK offshore industry.
1992	UK government white paper announcing formal proposals for the privatisation of British Rail (railways)	The principal driver for the subsequent safety case regime.
1994	Privatisation of British Rail and enactment of the Railways (Safety Case) Regulations, 1994 (railways)	First introduction of a mandatory safety case regime in the UK.
1996	Seveso II Directive is adopted (petrochemical)	Implemented in the UK as the COMAH Regulations (see below).
1997	Southall collision – seven deaths (railways)	Signal operated by the infrastructure controller passed at danger by a driver employed by a train operating company.
1999	Ladbroke Grove collision and fire – 31 deaths (railways)	Also a signal passed at danger (SPAD) incident. Southall and Ladbroke Grove accidents led directly to (*inter alia*) a review of the safety case regime.
1999	Control of Major Accident Hazards (COMAH) Regulations adopted in UK for onshore facilities (petrochemical)	Replaced the CIMAH Regulations and introduced a greater degree of uniformity with the offshore SCR. The regulations brought a number of smaller sites under the legislation and introduced a number of new features, including the MAPP and SMS requirements. Also brought an increased emphasis on demonstration rather than description.
2000	Enactment of Railways (Safety Case) Regulations 2000 and 2001 amendments, revising the Safety Case regime (railways)	New regulations directly reflect the analysis and recommendations of the inquiries into Southall and Ladbroke Grove.
2003	Revision of Seveso II Directive (petrochemical)	Revision of Seveso II Directive to include additional requirements for risk assessment. The most important extensions of the scope cover risks arising from storage and processing activities in mining, from pyrotechnic and explosive substances, and from the storage of ammonium nitrate and ammonium nitrate-based fertilisers.

Figure 2.3: Brief chronological summary of the significant events and changes in the safety regulations for the petrochemical, nuclear and railway industries - part 2 [5]

cases for their products. One of the good examples in this regard is the infusion pump safety assurance case guidance issued by the FDA in 2010 [36]. AdvaMed then started working on an example safety assurance case based on the infusion pump guidance with the goal of creating a template. After multiple reviews by the FDA, this assurance case template can be used by manufacturers for their infusion pump submissions to the FDA [36]. However, this template does not claim that it covers all the required aspects of an assurance case, nor that it is correct in all aspects. The FDA has not yet published a final infusion pump guidance document. Figure 2.4 represents the top-level claim and its first level decomposition for the infusion pump safety case.

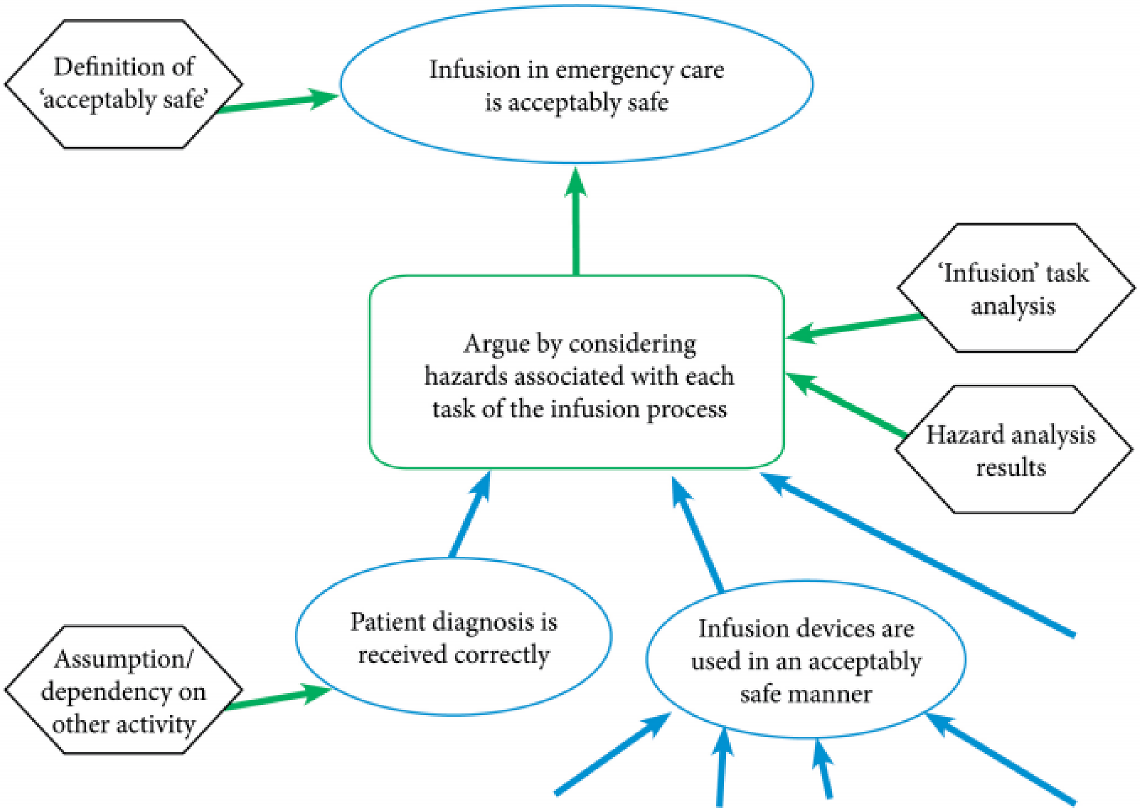

Definition of 'acceptably safe'
Infusion in emergency care is acceptably safe
'Infusion' task analysis
Argue by considering hazards associated with each task of the infusion process
Hazard analysis results
Patient diagnosis is received correctly
Infusion devices are used in an acceptably safe manner
Assumption/ dependency on other activity

Figure 2.4: Proposed clinical safety case top-level claim and decomposition of the high-level claim for pump infusion [5]

In Europe and the United Kingdom, the use of assurance cases is common. They are required in systems such as flight control systems, nuclear reactor shutdown systems, and railroad signaling systems [37]. However, using assurance cases in the United States and Canada is not yet common, but increasing. FDA and the National Aeronautics and Space Administration (NASA) are two of the organization suggesting assurance case use.

The FDA has taken steps towards requiring assurance cases when a device manufacturer submits a medical device for approval. NASA also suggested the use of assurance cases as a part of the development of the Constellation system [37]. Nowadays assurance cases are used not only in the safety-critical domains, but also for other domains such as security [23]. Assurance cases, therefore, are growing in size and complexity and are becoming increasingly used in industry. In this thesis, we have developed an assurance case for the correctness of a medical analysis software which is also categorized as a scientific software. We present this assurance case in Chapter 4.

2.3 Assurance Case Properties, Benefits and Challenges

In this section, we talk about assurance case properties, benefits and existing challenges. As we talked about the role of the assurance cases previously, we only go through the benefits briefly in this section.

The properties an assurance case must have so that we can take advantage of this approach are as follows. According to [5], [4] and [38] an assurance case must be:

Clear and Straightforward An assurance case is a means for communicating arguments. Clarity of the assurance argument helps convince an objective observer that the goal has been achieved.

Easy to navigate An assurance case must be readable and easy to navigate so that a reviewer can understand the pattern used for the reasoning. It must be traceable; tracing from part of an assurance case to the evidence can support the strength of the decision making.

Extensible Assurance cases need to be designed with reusability in mind.

Arguments must be:

Compelling An argument must be convincing. It must persuade the reviewer to agree with the person who made the argument.

Comprehensible An argument must be understandable for the audience and reviewers. Any assumption or context required to understand an argument must be mentioned clearly and completely.

Valid An argument must be a correct representation of the rationale for belief.

An evidence must be:

Relevant Relevance here is considered as the logical relevance; whether an evidence supports its claim.

Complete All claims required to satisfy the property of interest must be supported by some corresponding evidence. However, due to the limitation of knowledge, reaching complete evidence might not be possible. Hence, having some evidence that sufficiently assures the top goal is met is acceptable in many cases. As mentioned in [39] “The Safety Case shall contain a structured argument demonstrating that the evidence contained therein is sufficient to show that the system is safe”. This definition is true about the assurance cases as well.

An assurance case could be a useful tool and could have many benefits if it is developed in a way that has the above-mentioned properties. According to [5], [4] and [38] the benefits of assurance cases include the following:

Improving the development process Development of assurance cases, in parallel with system development encourages a more evolutionary process for development and has several benefits, as mentioned earlier in section 2.1.

Improving understanding and help in producing a reviewable artifact The structure of assurance cases provides a better oversight for the regulators and makes their review easier.

Improving communication An assurance case is a means for communication between stakeholders, such as system designers, manufacturers, operators, managers, regulators and the public.

Help in leveraging assurance resources and integrating the evidence Assurance cases provide a structured means of integrating evidence from diverse sources such as trials, human factors analysis, testing and operational experience. This helps in improving the consistency and completeness of assurance cases.

Improving safety management and plan Developing an assurance case improves understanding of the system hazards as well as the knowledge of the technical and managerial controls required to manage them; hence, an assurance case reduces the risk of the hazards a system may encounter.

However, assurance case development, review, maintenance and reuse is still challenging. The problem is derived from the insufficient and incomplete structuring support for developing assurance cases, especially because the systems requiring an assurance case are usually large and complex. According to [4] and [38], challenges that industries have faced when adopting assurance cases are as follow:

Difficulty in building Using natural language for building assurance cases instead of formal proofs makes the assurance case development challenging. Moreover, tracking among the texts cannot be implemented and none of the current tools provides an acceptable tracking feature; hence work-flow management is difficult. This problem is encountered in maintaining and reusing existing assurance cases as well.

Difficulty in reviewing Free-format text makes review challenging, as it is hard to see the pattern among the arguments and key components might be hidden in the sheer volume. There is no explicit guidance or discipline for judging evidence. There is no explicit way to tell the stakeholders how to evaluate the evidence and how to check if they are inconsistent or conflicting. In many cases, the evidence is compelling, but the reader has to work hard to confirm that it is so. As a result, reasoning based on the evidence can be incomplete, inconclusive and imprecise.

Challenges in maintaining It is not clear how one change in the software impacts the assurance case structure. For example, it might cause breakage of successive dependencies.

Re-usability limits Assurance cases are most of the time hard to reuse. The relationship among claims, arguments and evidence are not often explicitly mentioned.

Confusion between terminology There is no explicit rule to distinguish assurance case terminology. For example, there is no way to determine whether a statement should be mentioned as a context or as an assumption (these terms are defined in chapter 3). In other words, we need guidance on what can be accepted as an assumption and what must be proved.

2.4 Assurance Case Terminology and Notations

In this section, we mention the widely-used notations and terminology for developing structured assurance cases. There are variants of notations that present the assurance case structure graphically. Goal Structuring Notation (GSN) and Claims-Argument-Evidence (CAE) are used in the most of the cases in the literature. These notations can be supported by tools that help to create the graphical structure. In this section, we introduce these two notations and the terminology they use.

In general, two paths can be taken while building a structured assurance case, a top-down approach and a bottom-up approach. In the former, we start with a top-level claim and we try to give arguments to support the claim, and the arguments themselves are supported by some evidence. In the latter approach, first the evidence is provided. Then we try to integrate them to reach our top-level goal using some claims.

The purpose of this section is to give an introduction to the GSN and CAE and their usage in representing arguments. Although other notations exist to present the argumentation and reasoning, we believe that GSN and CAE are the most expressive and complete ones. Other notations often require more text and other resources to support the diagram and the notations that require the reader to read an explanatory text in other accompanying documents are not as effective.

2.4.1 GSN

Goal Structuring Notation (GSN) was developed originally to help structure and visualize safety cases in a readable form. This notation has been used in safety case development for over a decade. According to [2], "The purpose of a Goal Structure is to present an argument that gives the reader a high confidence that the proposition is true". The principal elements forming a GSN diagram are as follows [2]:

Top Goal The primary claim, i.e. the proposition usually proposed by a stakeholder, is represented as the top goal in the GSN. The top goal is advised to be sufficiently general so that it can capture process issues [40]. To express the goal, it should be in the form of subject-verb-object.

Sub-goal Sub-claims are represented as sub-goals in the GSN. A top goal is the consequence of some simpler sub-goals.

Strategy Strategy presents the rationale adopted while making arguments and choosing sub-goals. In other words, a goal is solved by a strategy, and a strategy explains how a goal is split to, and supported by, its sub-goals.

Argument An argument is a connected series of statements intended to establish a proposition [2]. In other words, the top-level goal alongside all the sub-goals and the reasoning that links them form an argument.

Context The context and environment in which arguments should be interpreted is referred to as Context in the GSN. The truth of an argument must be considered in terms of its context. Context is also used to provide additional definitions and other supporting materials required to understand the assurance case. It can also be used to reference out to such materials. The aim of using the context is to make the GSN diagram clearer.

Justification A justification is an extra explanation or rationale that justifies strategy. Sometimes it is not clear whether we should use a context or a justification to define a term. [2] suggests using a context if the definition is used to explain a term and use a justification if we want to explain a decomposition or a strategy.

Assumption Assumptions in the GSN are those that are taken to be true when we propose an argument. It is important to state all of the assumptions that we have made explicitly. The argument may be invalid without considering its assumptions.

Solution The evidence is represented as a solution in the GSN. The solution terminates the threads of argument and supports their truth. It usually points out to the relevant reports, documents, test cases, or any other material that is an evidence.

GSN explicitly represents these elements and the relationships between them. The principal symbols of the GSN are shown in Figure 2.5. A basic GSN structure is shown in Figure 2.6. A more developed example can be seen in Figure 2.7 which is a standard example given by the GSN community [8].

GSN is increasingly being used in safety-critical industries to improve the structure and clarity of safety arguments. The FDA is also proposing that manufacturers take a goal-based approach in their assurance cases, preferably using the CAE paradigm, or the GSN [5]. As our case study, which we will talk about in the next chapter, is a medical software, we use the GSN paradigm to represent our assurance case through this thesis.

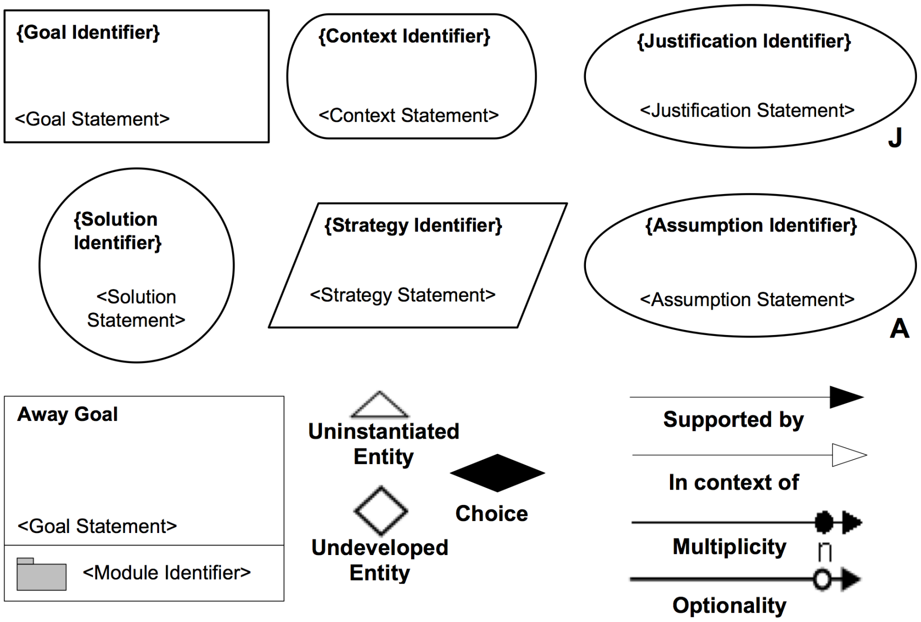


Figure 2.5: Main blocks in the GSN notation [6]

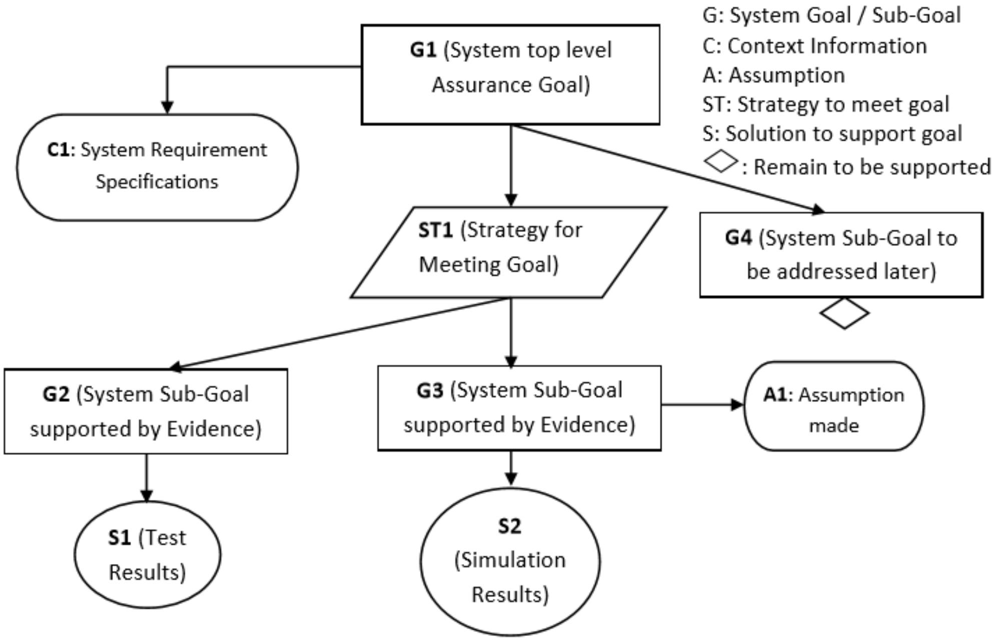

G: System Goal / Sub-Goal
C: Context Information
A: Assumption
ST: Strategy to meet goal
S: Solution to support goal
◇: Remain to be supported
G1 (System top level Assurance Goal)
C1: System Requirement Specifications
ST1 (Strategy for Meeting Goal)
G4 (System Sub-Goal to be addressed later)
G2 (System Sub-Goal supported by Evidence)
G3 (System Sub-Goal supported by Evidence)
A1: Assumption made
S1 (Test Results)
S2 (Simulation Results)

Figure 2.6: A basic GSN structure [7]

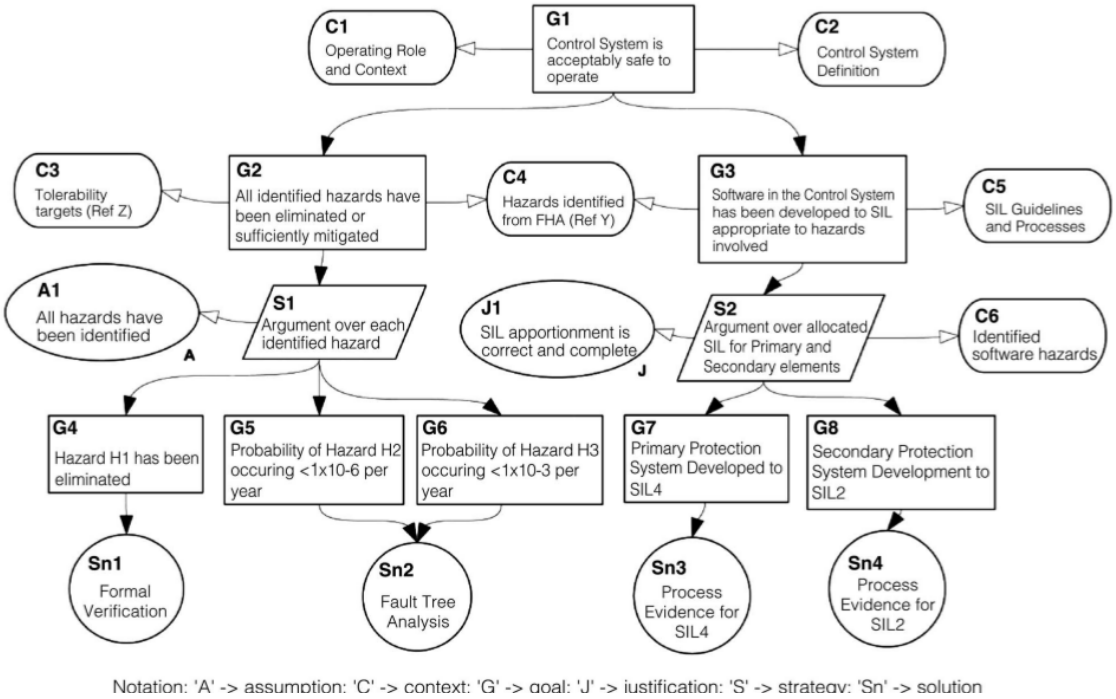

C1
Operating Role and Context
G1
Control System is acceptably safe to operate
C2
Control System Definition
C3
Tolerability targets (Ref Z)
G2
All identified hazards have been eliminated or sufficiently mitigated
C4
Hazards identified from FHA (Ref Y)
G3
Software in the Control System has been developed to SIL appropriate to hazards involved
C5
SIL Guidelines and Processes
A1
All hazards have been identified
A
S1
Argument over each identified hazard
J1
SIL apportionment is correct and complete
J
S2
Argument over allocated SIL for Primary and Secondary elements
C6
Identified software hazards
G4
Hazard H1 has been eliminated
G5
Probability of Hazard H2 occuring <1x10-6 per year
G6
Probability of Hazard H3 occuring <1x10-3 per year
G7
Primary Protection System Developed to SIL4
G8
Secondary Protection System Development to SIL2
Sn1
Formal Verification
Sn2
Fault Tree Analysis
Sn3
Process Evidence for SIL4
Sn4
Process Evidence for SIL2
Notation: 'A' -> assumption: 'C' -> context: 'G' -> goal: 'J' -> justification: 'S' -> strategy: 'Sn' -> solution

Figure 2.7: A standard example of the safety case diagram given by the GSN community [8]

2.4.2 CAE

Claims Arguments Evidence (CAE) is another graphical notation for presenting the structure of an assurance case. CAE structure consists of the following elements [9]:

Claim and Sub-claim Statements about the properties or behaviors of the software are considered as claims and sub-claims in the CAE. These properties can be some functionalities, characteristics or behavior of the system that need to be fulfilled. The higher level claim is usually decomposed to some sub-claims.

Argument Argument is logical links between evidence and claims. It makes the connection between claims and sub-claims and evidence well understood by the audience.

Evidence Evidence is information that supports the claim. Resources of previous experiments, test cases, standards, documents are some of the evidence that is linked to the claims by the arguments and support the claims. A valid evidence should be traceable to the top claim.

A generic example of the assurance case presenting by the CAE notation is shown in Figure 2.8.

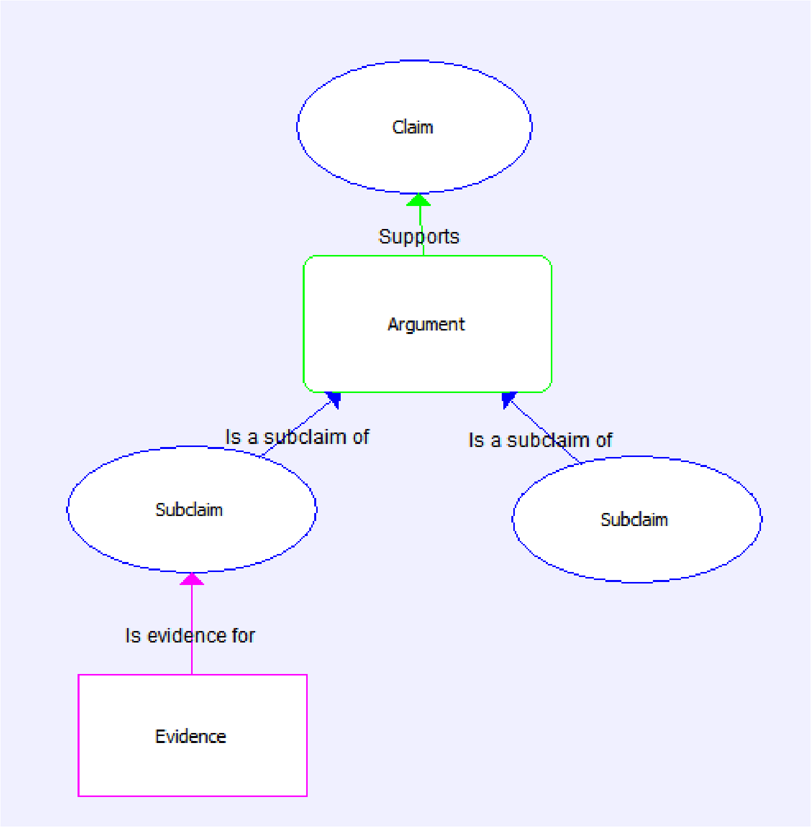


Figure 2.8: Assurance case representation in CAE [9]

In summary, to present an assurance case graphically, the GSN and CAE Graphical notations are widely used. CAE defines nodes for claims, arguments and evidence whereas GSN uses goal oriented presentation style and defines nodes for goals, strategies and solutions. These notations are mostly similar, with some difference in the progression approach. GSN follows a Top–Down approach while creating the assurance case starting with a top-level goal of the system whereas CAE supports a Bottom-Up view starting with the evidence to determine the possible claim. There is no rule to decide which approach or notation should be followed to develop an assurance case and it can be determined by the developers and other people in a team based on their preference.

Chapter 3

Overview of Case Study

As we mentioned in the previous chapter, the FDA has recently encouraged manufacturers to use and submit assurance cases for their products. An assurance case, if developed properly, increases the trustworthiness in the product. Having confidence in the operation of a medical device, or the accuracy of the output of medical software, is of great importance. Such consideration is rarely seen in the medical software domain, and on a larger scale, in the scientific software domain.

Lack of attention to assurance cases in the scientific software domain has motivated us to develop an assurance case for an existing scientific software package. This software is a medical image analysis program in the Analysis of Functional NeuroImages (AFNI) package that is based on the functional Magnetic Resonance Imaging (fMRI) technique. More details on this software and the AFNI package are given later in this chapter.

Another motivation was a recent paper published in 2016 [17], that called some of the fMRI studies into question. As pointed out in [41], around 3500 studies might be invalid because of a bug in one piece of software in the AFNI package [16]. Before this investigation, another bug was reported in AFNI that led to higher error rates than expected and contributed to inflated false positive rates [16]. The community needs to respond to such issues, raise standards and increase trustworthiness. As a result, we selected a medical imaging software called 3dfim+ as our case study and developed an assurance case to increase reliability in the program and to contribute to addressing such problems.

In this chapter, we talk about 3dfim+ and its functionality. We also talk about the Functional Magnetic Resonance Imaging (fMRI), the technique that 3dfim+ is based on, and the principles required to understand this technique. We also explain the way we obtained the raw data from the patient and how the program analyses it.

3.1 Functional Magnetic Resonance Imaging (fMRI)

This section explains how the raw data from an fMRI experiment is analyzed. The aim of such analysis is to determine the regions of the brain in which the brain signal changes upon the presentation of a stimulus.

For centuries, scientists have tried to find the relationship between physical actions and behavior, thought and brain function. The study of brain function started in the 17th century. At that time, it was believed that various areas of the human cerebral cortex have specific functionalities, and these functions are only higher-order, conscious mental functions [42, p. 836]. However, later in the middle of the 19th century, other research showed that a specific part of the cerebral cortex has a causal role in movement [42, p. 836].

This study progressed in the late 19th century by mapping motor function in animals and later in humans. However, this study lacked consistency and accuracy [43].

More reliable work was carried out in the first half of the 20th century. Electrical Brain Stimulation (EBS) allowed the motor map to be defined in greater detail [42]. The problem was that by that time, the studies had come from patients with neurological disorders and there was no way to study healthy individuals [43]. It has only been since the 1990s that scientists have been able to study healthy humans using brain imaging techniques [43].

fMRI is one such technique for mapping human brain activity. 3dfim+ is software that analyses human brain functionality from fMRI data. It is one of the few techniques that enables us to look directly into an individuals' brain while they are thinking or performing an action [16].

3.1.1 Physiological Principles

In this section, the two theoretical foundations required to understand fMRI are briefly explained: Magnetic Resonance Imaging (MRI) and Blood Oxygen Level Dependent (BOLD) Contrast. This chapter serves only as an outline of the basic principles of the fMRI that are related to our case study. More detail can be found in the standard books on the subject, such as [44], [45] and [46].

Conventional Magnetic Resonance Imaging, or MRI, is a technique which gives the anatomical pictures of the brain. Using this technique, we are able to produce images of the human brain with excellent soft tissue contrast. The development of contrast agents suitable for dynamic MRI studies, and improvements in the speed of imaging opened up the possibility of using the technique for functional brain studies [43]. In 1991, the first experiment using MRI to study brain function was performed, imaging the visual cortex whilst the subject was presented with a visual stimulus. A contrast agent was used in this first study, but it was not much later when the first experiment was carried out using blood as a contrast agent. The hemoglobin in the blood has different magnetic properties depending on whether it is oxygenated or not; these differences affect the signal recorded in the MR image.

By imaging a subject at rest and whilst they were carrying out a specific task, it became possible to image brain function in a completely non-invasive way. The fMRI uses such a technique and has brought the study of the human brain into a new era, offering new insights into the relationship between mind and brain.

Blood Oxygen Level Dependent (BOLD) contrast imaging is a method used in fMRI. During the fMRI procedure, a conventional MRI scanner is used, but this technique also takes advantage of two additional phenomena. The first one is that blood contains iron, which is the oxygen-carrying part of hemoglobin inside red blood cells. The existence of iron in blood cause small distortions in the magnetic field around the blood cells. The second key phenomenon underlying fMRI is a physiological principle. According to this principle, whenever any part of the brain becomes active, the small blood vessels in that localized region dilate, causing more blood to rush in. The blood is presumably needed to provide extra oxygen and fuel more glucose for the active brain cells. The result is that a large amount of freshly oxygenated blood pours into any activated brain structure, reducing the amount of oxygen-free (deoxy) hemoglobin. This causes a small change in the magnetic field, and thus the MRI signal, in the active region.

In the early 1990s, it was shown that an MRI scanner can be used to detect this small change in the signal, and thus detect which areas of the brain have been activated. For example, if a patient lying in a scanner is suddenly shown a flash of light, the visual cortex in his brain will become activated, blood flow there will quickly increase, and the MRI signal will change. The result is usually displayed as a patchy area of color, representing the brain area activated. The signal is often called a BOLD signal, standing for Blood Oxygen Level Dependent signal, as shown in Figure 3.1.

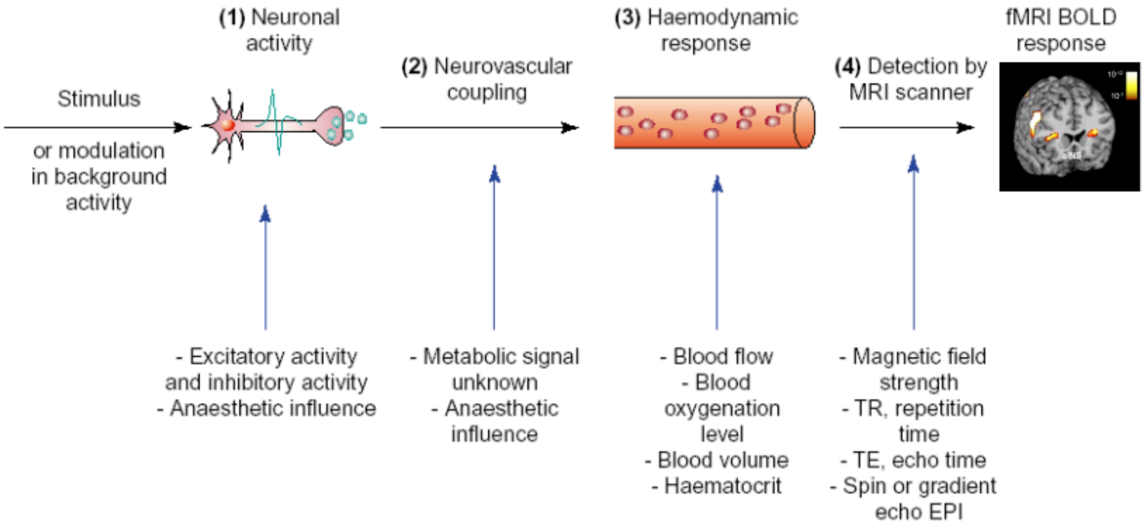

(1) Neuronal activity
(2) Neurovascular coupling
(3) Haemodynamic response
(4) Detection by MRI scanner
fMRI BOLD response
Stimulus
or modulation in background activity
- Excitatory activity and inhibitory activity
- Anaesthetic influence
- Metabolic signal unknown
- Anaesthetic influence
- Blood flow
- Blood oxygenation level
- Blood volume
- Haematocrit
- Magnetic field strength
- TR, repetition time
- TE, echo time
- Spin or gradient echo EPI

Figure 3.1: From Stimulus to Bold [10]

In summary, both conventional and functional MRI use a powerful magnet and radio waves to produce images of the brain. Conventional MRI images show detailed anatomy and are an essential part of modern medicine. In the fMRI, the same scanner is optimized to detect small changes in blood flow in the brain in response to scientifically designed stimuli. In principle, fMRI can be used to observe the activation of brain structures in response to almost any kind of brief stimulation, ranging from sounds to visual images. Currently, fMRI is being used across the world as a powerful neuroscientific research tool to study how the brain works. Although some medical applications are being discovered as well, fMRI may have a long way for being a definite diagnostic tool [47].

3.1.2 Data Acquisition

One of the common methods for obtaining results for a two-state fMRI experiment is to perform a periodic task. In our study, we asked our subject to lie with his head inside an MRI machine and tap his right hand fingers with his thumb in the first 30 seconds. In the next phase, we asked him to do the same activity with his opposite hand. It was necessary to provide some form of clue to inform the subject when he should switch his hands. The clue we used was to switch the lights off and on. As soon as the subject would notice the light change, he would switch to his other hand. Inter-stimulus interval was the same during the experiment. We also asked the subject to try to hold his head as stable as possible, as the disadvantage of this technique is that it is sensitive to head motion and the results might be invalid if motion is captured.

The data obtained from the experiment then was analyzed using the 3dfim+ software. We talk about the analysis of the data in the next section.

3.2 Analysis of fMRI Data Using 3dfim+

To analyze the data from the experiments, the AFNI package was used. AFNI is a set of open-source C programs for processing, analyzing, and displaying fMRI data [48]. AFNI implements existing and novel analysis techniques such as fMRI. It was originally developed at the Medical College of Wisconsin beginning in 1994, largely by Robert W. Cox [48].

3dfim+ is an analysis tool in the AFNI software package. This program mainly calculates the cross-correlation of an ideal reference signal versus the measured fMRI time series for each voxel. In other words, the statistical analysis using 3dfim+ detects the pixels in the image which show a response to the stimulus. These activated areas of the brain then can be displayed using other programs in AFNI, which give the statistical confidence that can be placed in the result.

The aim of such analysis is to produce an image identifying the regions which show a significant signal change in response to the task. Each pixel is assigned a value depending on how much it is correlated to a defined reference signal. The experiment performed was intended to detect activations resulting from a cued motor task. The whole brain of the subject was imaged, in 180 coronal slices of resolution 64 x 64 x 64. As cued by switching the light, the subject was required to tap his fingers as mentioned earlier.

The choice of an appropriate reference waveform is vital for the success of this technique in finding activations. We used a square wave as our ideal signal which was 1 for scans acquired when the subject was tapping the fingers of his right hand and 0 for scans acquired during finger tapping with the subject's opposite hand. Figure 3.2 depicts the activity of the voxel at position $(23, 27, 22)$ versus the ideal signal. For ease of comparison, the values of voxel's activity are scaled to values between 0 and 1. As we can see from the figure, the brain activity and the ideal signal are highly correlated.

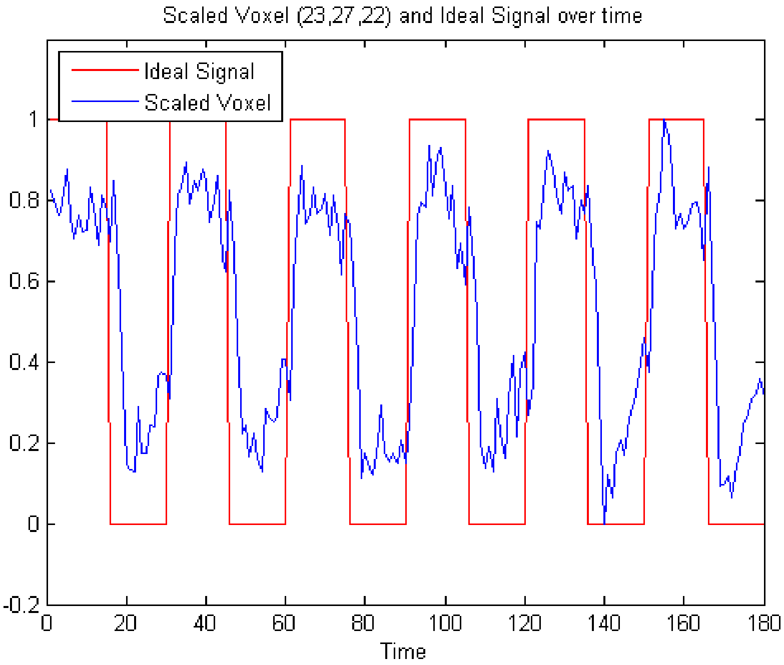


Figure 3.2: Ideal signal versus activity of the voxel at position (23,27,22) over time

The result of 3dfim+ can be visualized in the AFNI environment. This is shown in Figure 3.3. We can see the brain is shown in green and 2 parts of it are shown in red and blue. These parts of the brain are those which are, respectively, the most-correlated and the least-correlated parts to our ideal signal. Voxel at position (23,27,22) is located in the red region of the brain.

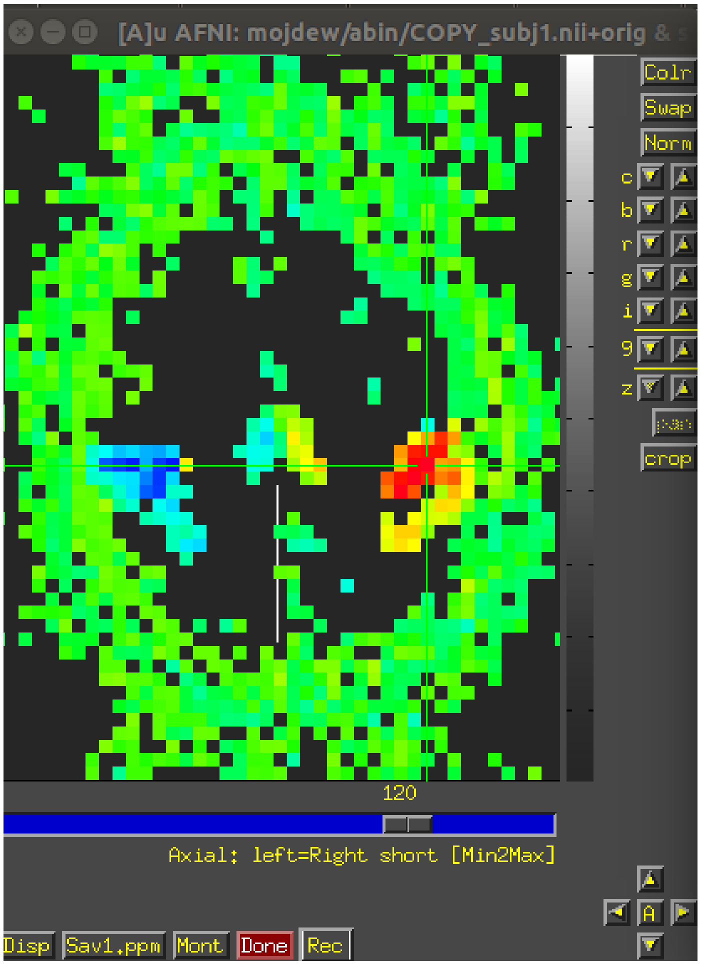

[A]u AFNI: mojdew/abin/COPY_subj1.nii+orig
Colr
Swap
Norm
c
b
r
g
i
9
z
crop
120
Axial: left=Right short [Min2Max]
A
Disp
Sav1.ppm
Mont
Done
Rec

Figure 3.3: Visualizing the brain activity using AFNI tools

Since we know that the BOLD response is mediated by blood flow, it is possible to improve the detection of activations by predicting the shape of the response to the stimulus and calculating correlation coefficients between each pixel time course and this reference waveform. For a reference waveform, the correlation coefficient is calculated as mentioned in the SRS document (Appendix A) and has a value of 1 in case of perfect correlation, a value of zero in case of no correlation, and a value of -1 in case of perfect anti-correlation.

Other necessary statistical background and formulas that 3dfim+ is based on, such as different types of the correlation, are defined in detail in the SRS document given in Appendix A.

Chapter 4

Assurance Case and Selected Evidence

In this chapter, we discuss the scope of our work and the assurance case for the correctness of 3dfim+. To provide the necessary evidence for the assurance case the following material is also presented: the software requirements specification, the test cases, and the expert review.

4.1 Scope Determination

The incorporation of software verification and validation requirements in medical standards has driven interest in assurance cases, as shown by the work on assurance cases for IT systems [49]. In [11], a domain assurance case template (Figure 4.1) is provided as a standard for the development and licensing of medical devices. In the context of medical devices, which can be characterized as embedded real-time systems, safety is a primary concern. Since the domain of the current work is medical image analysis software, rather than a device, the target switches from safety to correctness. However, the top-level goal in the template from [11] is general enough that we can use a similar top-goal for our assurance case.

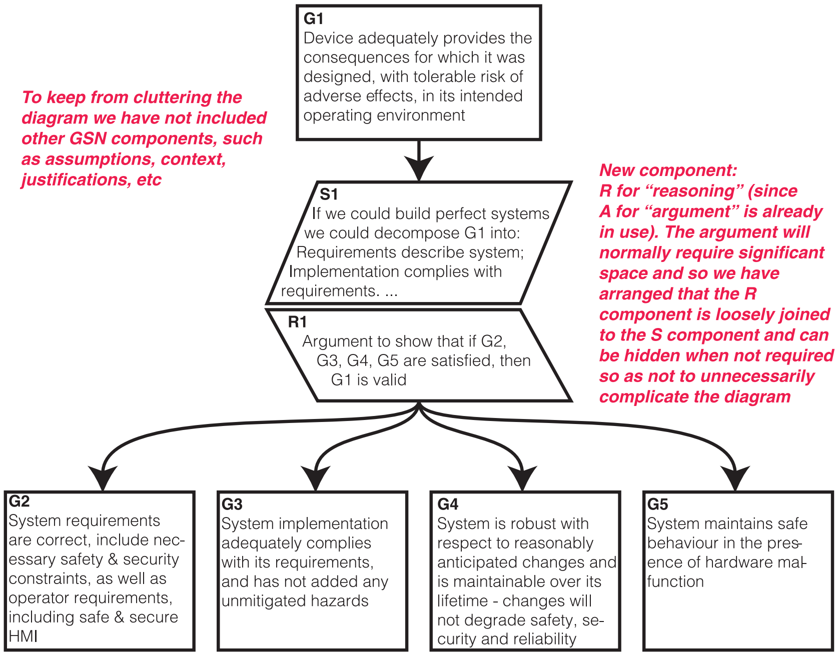

G1
Device adequately provides the consequences for which it was designed, with tolerable risk of adverse effects, in its intended operating environment
To keep from cluttering the diagram we have not included other GSN components, such as assumptions, context, justifications, etc
S1
If we could build perfect systems we could decompose G1 into: Requirements describe system; Implementation complies with requirements. ...
New component: R for "reasoning" (since A for "argument" is already in use). The argument will normally require significant space and so we have arranged that the R component is loosely joined to the S component and can be hidden when not required so as not to unnecessarily complicate the diagram
R1
Argument to show that if G2, G3, G4, G5 are satisfied, then G1 is valid
G2
System requirements are correct, include necessary safety & security constraints, as well as operator requirements, including safe & secure HMI
G3
System implementation adequately complies with its requirements, and has not added any unmitigated hazards
G4
System is robust with respect to reasonably anticipated changes and is maintainable over its lifetime - changes will not degrade safety, security and reliability
G5
System maintains safe behaviour in the presence of hardware malfunction

Figure 4.1: Top-level claim decomposition of the domain assurance case template provided by [11]

The eventual aim is to develop a template for medical software correctness. The first step of our work is to develop an assurance case for the correctness of the program 3dfim+. This program has around 1700 lines of code, so its size and complexity are reasonable for a Masters thesis; it can be tackled and is comprehensible for us as non-experts in the field. Moreover, relatively limited functionality of 3dfim+ makes it easier to test within the available time for the project.

Another reason for selecting 3dfim+ is that its current existing documentation [50] describes the program at a high level; some of the information necessary information is not given in the current manual, such as the mathematical background needed to understand the formulas used, assumptions that input needs to meet, coordinate convention that the software is based on, etc. It is probable that the 3dfim+ developer wants the program to be used only by the people with the relevant expertise, so he published abstract rather than detailed documentation. However, having a complete document, where the necessary background information is provided, the assumptions are mentioned, the conventions are discussed, and the presentation is simple and easy to understand, will be helpful for the people willing to start working with the program. Having such document also improves the software maintainability.

The assurance case developed in this thesis might be used as a guideline or template for developing other assurance cases for other medical analysis software and in general scientific computing software. Using the assurance case would help monitor compliance with the software correctness criteria during the software development process. Additional work remains to verify and validate the approach proposed here, but this thesis provides a first step towards building an assurance case template for medical analysis software.

The development of an assurance case, as mentioned in Chapter 2, is beneficial when it is done concurrently with the software development. We developed the assurance case at the same time as learning the theory behind 3dfim+, which allowed us to do development that was somewhat concurrent. In our assurance case, we discuss the quality of the documentation, design and implementation of the software.

We have used the guidance provided in “General Principles of Software Validation; Final Guidance for Industry and FDA Staff” [18] to develop our assurance case template. This guide outlines generally recognized validation principles that are acceptable to the FDA 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. There are some reasons we picked this guideline.

First, this guideline applies to a different kind of software such as software that is itself a medical device [18]. A definition of the software that is regulated as a medical device is given in [51]. Based on this definition, software regulated as a medical device:

1. "provides the only means and opportunity to capture or acquire data from a medical device for aiding directly in diagnosis or treatment of a patient"; or
2. "replaces a diagnostic or treatment decision made by a physician."

IMDRF adopted the definition of "Software as a Medical Device (SaMD)" as "software intended to be used for one or more medical purposes that perform these purposes without being part of a hardware medical device " [52]. According to the same resource, the medical purposes are as follows:

- "diagnosis, prevention, monitoring, treatment or alleviation of disease,
- diagnosis, monitoring, treatment, alleviation of or compensation for an injury,
- investigation, replacement, modification, or support of the anatomy or of a physiological process,
- supporting or sustaining life,
- control of conception,
- disinfection of medical devices,
- providing information by means of in vitro examination of specimens derived from the human body;

and does not achieve its primary intended action by pharmacological, immunological or metabolic means, in or on the human body, but which may be assisted in its intended function by such means."

According to [52], SaMD may also:

- "provide means and suggestions for mitigation of a disease;
- provide information for determining compatibility, detecting, diagnosing, monitoring or treating physiological conditions, states of health, illnesses or congenital deformities;
- be an aid to diagnosis, screening, monitoring, determination of predisposition; prognosis, prediction, determination of physiological status."

[52] also states that SaMD is capable of running on general purpose (non-medical purpose) computing platforms and "Computing platforms" include hardware and software resources (e.g. operating system, processing hardware, storage, software libraries, displays, input devices, programming languages etc.).

Medical Device Software can appear in many forms [51]:

- Software that is a component of a medical device
- Software that is an accessory to a medical device
- Standalone software that is intended to run on general purpose computers; also known as "Software only Devices".

There are several classes of medical device software categorized as Class I, Class II, Class III and Class IV medical devices [51]. By definition, "If the software is intended for analyzing device-provided data for the purpose of directly aiding in the treatment or diagnosis of a patient, this would be Class II software” [51].

Medical Device and Diagnostic Industry [53] give several examples of SaMD. Some of the examples include software that allows a generic smartphone to view images for diagnostic purposes obtained from MRI; software that performs image post-processing for aiding the detection of cancer; and treatment planning software that supplies information to be used in a linear accelerator device.

Although it is out of our area of expertise to give a comment on whether 3dfim+ is a SaMD, based on the definitions and examples given for SaMD, 3dfim+ has the potential to be a SaMD of Class II **if it is verified and validated** because:

- It can be used for one or more medical purposes; such as providing information for detecting and diagnosis states of health,
- It is a standalone software that is intended to run on the Unix systems which are general purpose (non-medical purpose) computing platforms,
- It is not a part of a hardware medical device,
- Similar to the example given by [53], it is a software that performs fMRI data post-processing for aiding to understand brain activities.

Another reason for choosing the guideline was it claims that it provides the least burdensome approach to complying with the medical and software scientific and legal requirements after a careful review of the existing related resources. Also, it is a good indicator of what the FDA expects from the developers to ensure compliance with the Quality System regulation with regard to software validation [18].

For these reasons, “General Principles of Software Validation; Final Guidance for Industry and FDA Staff” (GPSV) [18] is an appropriate guideline to consider for the 3dfim+ validation. GPSV recommends that software validation and verification activities must be conducted throughout the entire software lifecycle [18]. This recommendation complies with our approach; the integration of software development and assurance case development.

Although the case study we selected had been implemented previously, and our focus was on the requirements and testing, not redesigning or reimplementing the software, as we learned the about the software while we were developing our assurance case, the process of building the assurance case was concurrent to learning it and it can be considered as a similar approach to the integration of software development and assurance case development. The question we are interested to answer in this thesis is whether confidence can be built in the software based on documentation of the requirements and testing.

4.2 Assurance Case Development

In this section, we discuss the assurance case template we have developed based on the "General Principles of Software Validation" guideline, for validation of medical device software. To develop the template, we considered 3dim+ as a medical device software, based on the reasons mentioned in the previous section; however, we developed the assurance case with the aim of it being used as a template for similar medical device software and scientific software in the future.

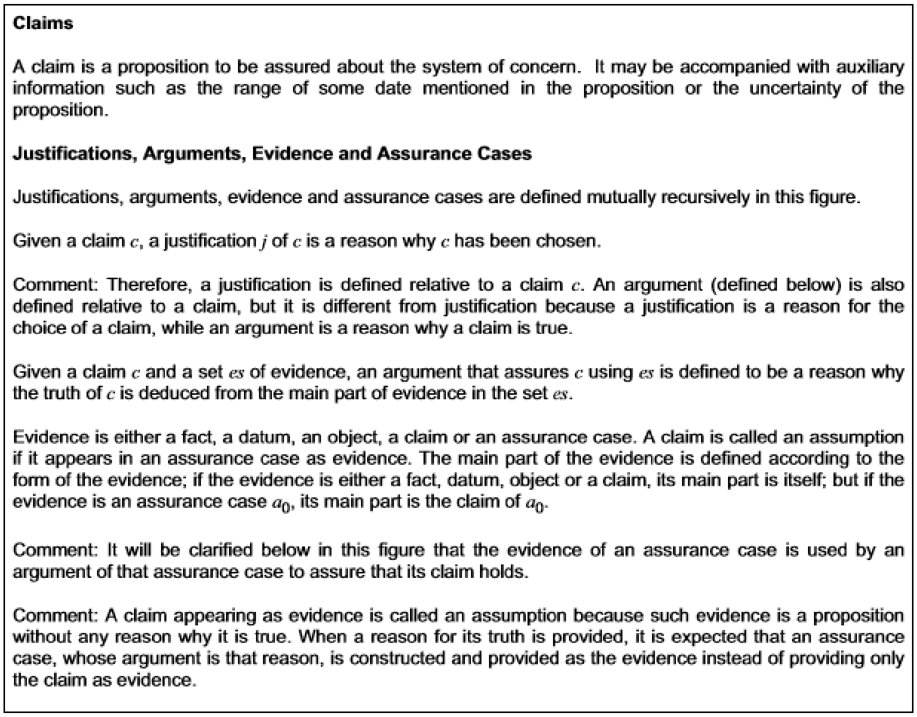
We follow ISO/IEC 15026-2:2011 - part 2 [12] which specifies minimum requirements for the structure and contents of an assurance case to improve the consistency and comparability of assurance cases and to facilitate other uses of assurance cases. Based on this standard "An assurance case includes a top-level claim for a property of a system or product (or set of claims), systematic argumentation regarding this claim, and the evidence and explicit assumptions that underlie this argumentation. Arguing through multiple levels of subordinate claims, this structured argumentation connects the top-level claim to the evidence and assumptions." We have developed our assurance case considering the requirements that need to be applied to the structure of an assurance case according to this standard. The requirements are as follows [12]:

- An assurance case shall have one or more top-level claims that are the ultimate goals of its argumentation.
- An argument shall be supported by one or more claims, evidence, or assumptions.
- A claim shall be supported either by just one argument, or by one or more claims, evidence, or assumptions.
- A claim, argument, evidence, or assumption shall not support itself either directly or indirectly.

Figures 4.2 and 4.3 represent an informative structure of assurance cases.

ISO/IEC 15026-2:2011 does not specify the use of a particular terminology or graphical representation, so we were free to choose among the popular notations, terminologies and tools. As we mentioned in chapter 2, we have chosen the most popular notation called Goal Structuring Notation (GSN), developed by Kelly [54] to make our arguments clear, easy to read and, hence, easy to challenge. To develop the assurance case, we used Astah [55] software, which is a modeling tool designed to be used for assurance case development based on the GSN notation.

Another main resource that we have followed to understand the GSN notation and to develop our assurance case was the book "GSN - The Goal Structuring Notation, A Structured Approach to Presenting Arguments" [2]. It presents a clear explanation of the GSN notation, how to develop arguments, and how to represent them. It also



Claims

A claim is a proposition to be assured about the system of concern. It may be accompanied with auxiliary information such as the range of some date mentioned in the proposition or the uncertainty of the proposition.

Justifications, Arguments, Evidence and Assurance Cases

Justifications, arguments, evidence and assurance cases are defined mutually recursively in this figure.

Given a claim c, a justification j of c is a reason why c has been chosen.

Comment: Therefore, a justification is defined relative to a claim c. An argument (defined below) is also defined relative to a claim, but it is different from justification because a justification is a reason for the choice of a claim, while an argument is a reason why a claim is true.

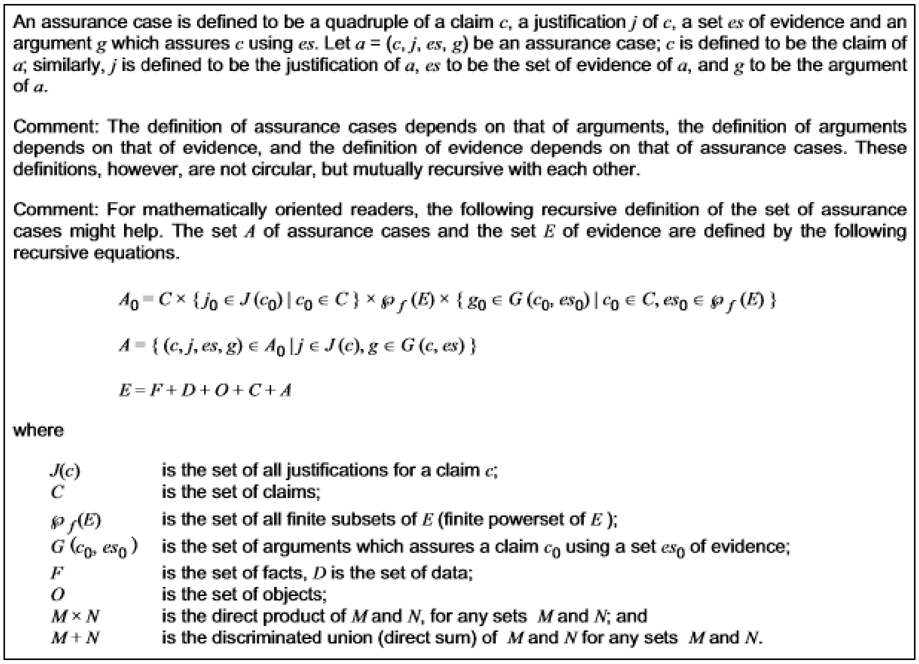
Given a claim c and a set es of evidence, an argument that assures c using es is defined to be a reason why the truth of c is deduced from the main part of evidence in the set es.

Evidence is either a fact, a datum, an object, a claim or an assurance case. A claim is called an assumption if it appears in an assurance case as evidence. The main part of the evidence is defined according to the form of the evidence; if the evidence is either a fact, datum, object or a claim, its main part is itself; but if the evidence is an assurance case a_0, its main part is the claim of a_0.

Comment: It will be clarified below in this figure that the evidence of an assurance case is used by an argument of that assurance case to assure that its claim holds.

Comment: A claim appearing as evidence is called an assumption because such evidence is a proposition without any reason why it is true. When a reason for its truth is provided, it is expected that an assurance case, whose argument is that reason, is constructed and provided as the evidence instead of providing only the claim as evidence.

Figure 4.2: ISO/IEC 15026-2:2011 informative structure of assurance cases - part 1 [12]



An assurance case is defined to be a quadruple of a claim c, a justification j of c, a set es of evidence and an argument g which assures c using es. Let $a = (c, j, es, g)$ be an assurance case; c is defined to be the claim of a; similarly, j is defined to be the justification of a, es to be the set of evidence of a, and g to be the argument of a.

Comment: The definition of assurance cases depends on that of arguments, the definition of arguments depends on that of evidence, and the definition of evidence depends on that of assurance cases. These definitions, however, are not circular, but mutually recursive with each other.

Comment: For mathematically oriented readers, the following recursive definition of the set of assurance cases might help. The set A of assurance cases and the set E of evidence are defined by the following recursive equations.

$$A_0 = C \times \{ j_0 \in J(c_0) \mid c_0 \in C \} \times \wp_f(E) \times \{ g_0 \in G(c_0, es_0) \mid c_0 \in C, es_0 \in \wp_f(E) \}$$

$$A = \{ (c, j, es, g) \in A_0 \mid j \in J(c), g \in G(c, es) \}$$

$$E = F + D + O + C + A$$

where

$J(c)$	is the set of all justifications for a claim c;
C	is the set of claims;
$\wp_f(E)$	is the set of all finite subsets of E (finite powerset of E);
$G(c_0, es_0)$	is the set of arguments which assures a claim c_0 using a set es_0 of evidence;
F	is the set of facts, D is the set of data;
O	is the set of objects;
$M \times N$	is the direct product of M and N, for any sets M and N; and
$M + N$	is the discriminated union (direct sum) of M and N for any sets M and N.

Figure 4.3: ISO/IEC 15026-2:2011 informative structure of assurance cases - part 2 [12]

contains several examples that we have used as a guideline to develop some of our arguments. Figure 4.4 depicts the process of developing an argument.

Using the same process, we have developed our assurance case. Our assurance case consists of many subclaims, which means it cannot legibly by represented on a single page. We have overcome this problem by splitting the argument and presenting sub-structures separately.

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. 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 subgoals 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, 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 the context in which we make it so we must be explicit about what we mean by each term in our goal statement. We could detail the goal statement to include this additional information, but then it would be too long. The solution is to declare the context explicitly. We have defined each term in the top goal in several contexts. We have also made an assumption that must be considered. The assumption

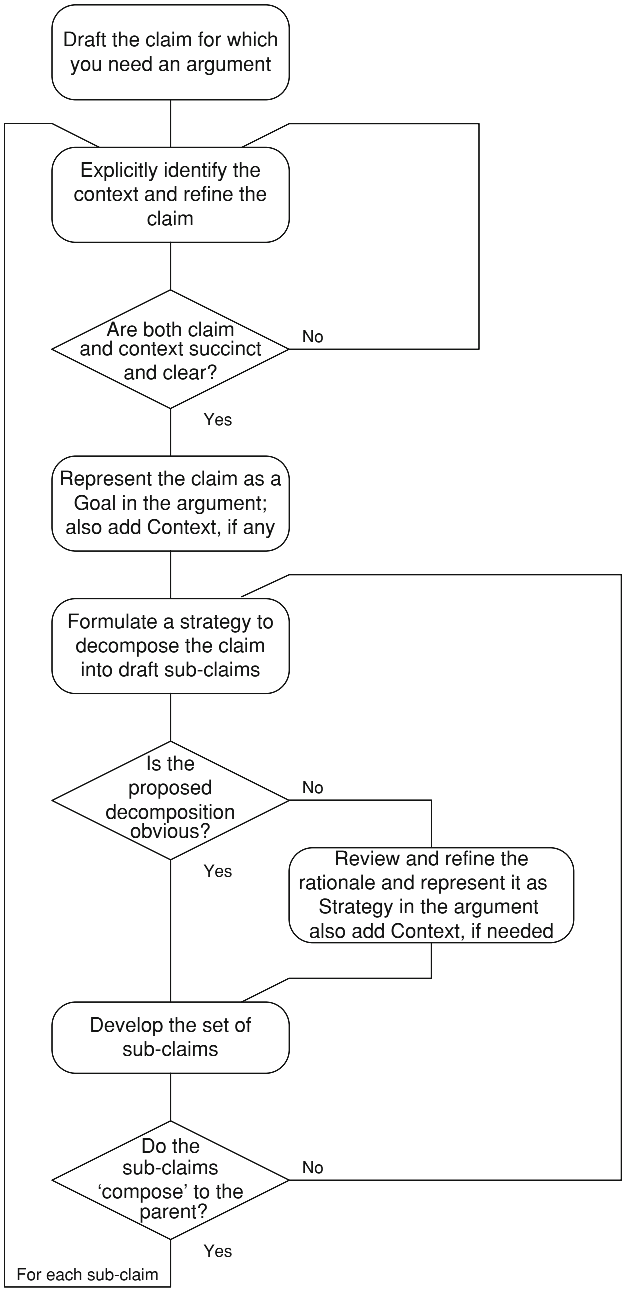

Draft the claim for which you need an argument
Explicitly identify the context and refine the claim
Are both claim and context succinct and clear?
No
Yes
Represent the claim as a Goal in the argument; also add Context, if any
Formulate a strategy to decompose the claim into draft sub-claims
Is the proposed decomposition obvious?
No
Yes
Review and refine the rationale and represent it as Strategy in the argument also add Context, if needed
Develop the set of sub-claims
Do the sub-claims 'compose' to the parent?
No
Yes
For each sub-claim

Figure 4.4: The process of developing an argument [2]

and contexts are given in Figure 4.5.

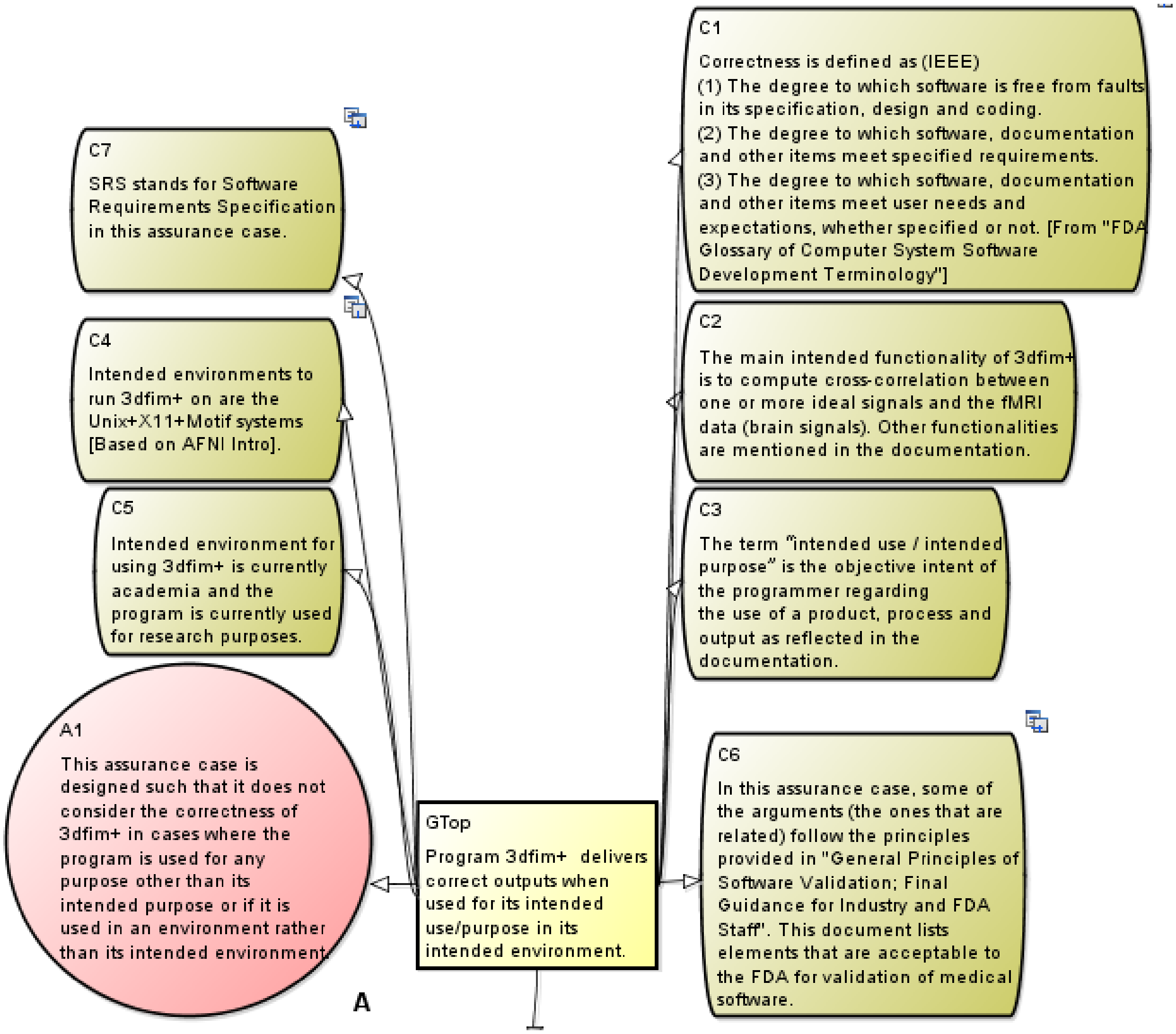
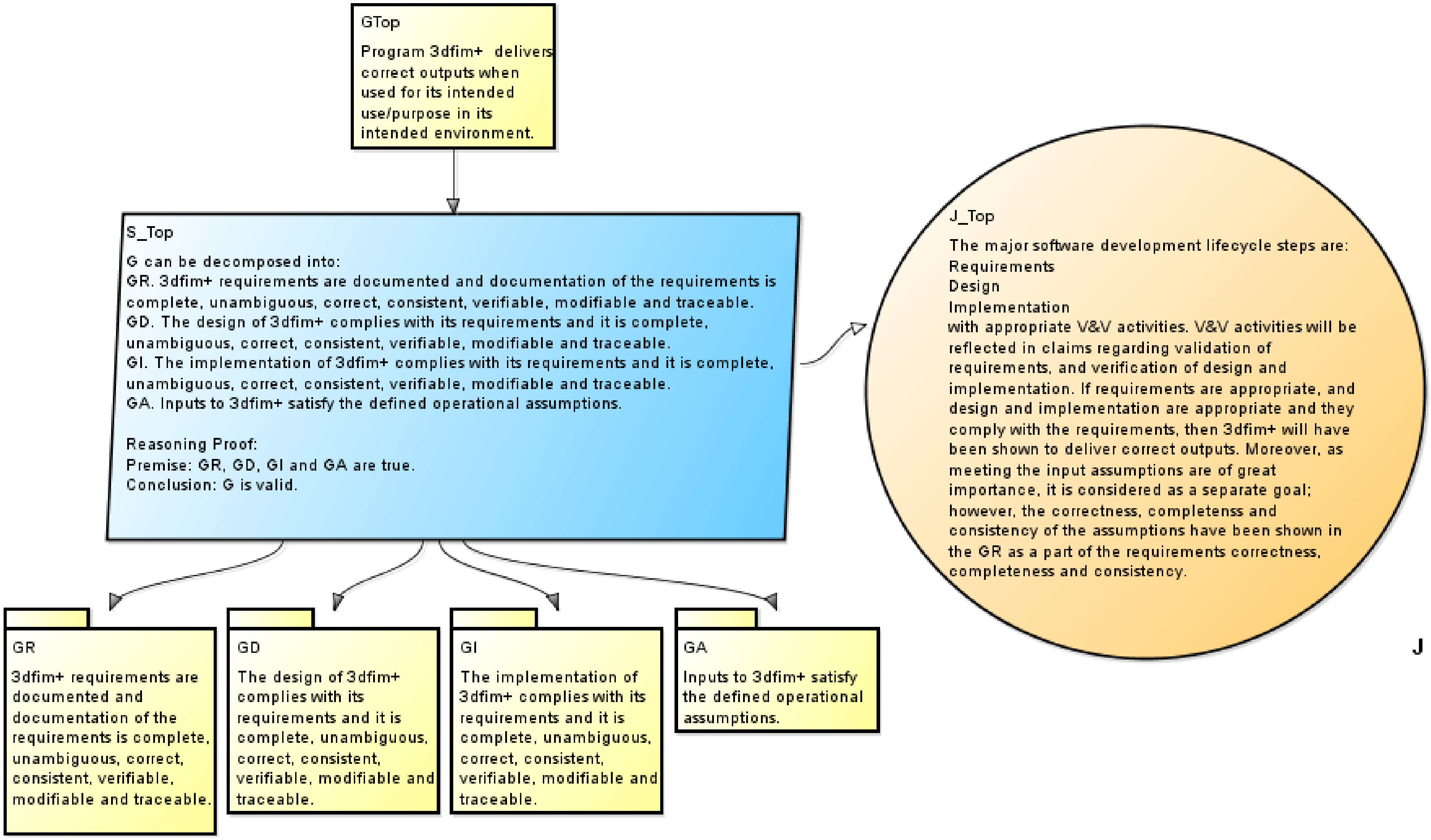

C7
SRS stands for Software Requirements Specification in this assurance case.
C4
Intended environments to run 3dfim+ on are the Unix+X11+Motif systems [Based on AFNI Intro].
C5
Intended environment for using 3dfim+ is currently academia and the program is currently used for research purposes.
A1
This assurance case is designed such that it does not consider the correctness of 3dfim+ in cases where the program is used for any purpose other than its intended purpose or if it is used in an environment rather than its intended environment.
A
C1
Correctness is defined as (IEEE)
(1) The degree to which software is free from faults in its specification, design and coding.
(2) The degree to which software, documentation and other items meet specified requirements.
(3) The degree to which software, documentation and other items meet user needs and expectations, whether specified or not. [From "FDA Glossary of Computer System Software Development Terminology"]
C2
The main intended functionality of 3dfim+ is to compute cross-correlation between one or more ideal signals and the fMRI data (brain signals). Other functionalities are mentioned in the documentation.
C3
The term "intended use / intended purpose" is the objective intent of the programmer regarding the use of a product, process and output as reflected in the documentation.
GTop
Program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment.
C6
In this assurance case, some of the arguments (the ones that are related) follow the principles provided in "General Principles of Software Validation; Final Guidance for Industry and FDA Staff". This document lists elements that are acceptable to the FDA for validation of medical software.

Figure 4.5: Contexts and Assumption in Top Goal

Similar to Figure 4.1, we have divided the top goal to four sub-goals; the first sub-goal argues the quality of the documentation of the requirements (GR), in the second sub-goal we developed an argument for the quality of the design of the program (GD), the third sub-goal is related to the quality of the 3dfim+ implementation (GI), and the last sub-goal is for the input assumptions (GA). More details are provided in Figure 4.6.



GTop

Program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment.

S_Top

G can be decomposed into:
GR. 3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GD. The design of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GI. The implementation of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GA. Inputs to 3dfim+ satisfy the defined operational assumptions.

Reasoning Proof:
Premise: GR, GD, GI and GA are true.
Conclusion: G is valid.

J_Top

The major software development lifecycle steps are:
Requirements
Design
Implementation
with appropriate V&V activities. V&V activities will be reflected in claims regarding validation of requirements, and verification of design and implementation. If requirements are appropriate, and design and implementation are appropriate and they comply with the requirements, then 3dfim+ will have been shown to deliver correct outputs. Moreover, as meeting the input assumptions are of great importance, it is considered as a separate goal; however, the correctness, completenss and consistency of the assumptions have been shown in the GR as a part of the requirements correctness, completeness and consistency.

GR

3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

GD

The design of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

GI

The implementation of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

GA

Inputs to 3dfim+ satisfy the defined operational assumptions.

Figure 4.6: Top Goal of the assurance case and its sub-goals

The main focus in our assurance case was on arguing for GR (quality requirements documentation). The decomposition of GR into its sub-goals is shown in Figure 4.7. This decomposition is based on the IEEE standard 830-1993 [56]. 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 our assurance case as our case study is a scientific software and all requirements are equally important. This is shown as J_GRb in Figure 4.7.

You mention ranked for importance here, and this is shown in your figure,

For each of the sub-goals in Figure 4.7, we have developed arguments and presented them in Figures 4.8, 4.9, 4.11, 4.12, 4.16, 4.15, 4.17 and 4.18. In this case, we need contexts to give definitions for each properties.

We decided to develop an argument for consistency, completeness, and correctness together called 3C. The reason for grouping these qualities is that according to some publications, such as “ The Three Cs of Requirements: consistency, completeness, and correctness” [57], there is an important relationship between completeness, consistency and correctness of software requirements. Improving one of these three qualities may diminish the other one. From another perspective, correctness is a combination of consistency and completeness. So it is important to consider these 3 qualities together. The top level of this argument is shown in Figure 4.8.

We present the sub-goals of the 3Cs in Figures 4.9 for correctness and consistency and 4.11 and 4.12 for completeness.

Spriggs [2] gives an argument for the readiness of a business plan in Figure 4.10.

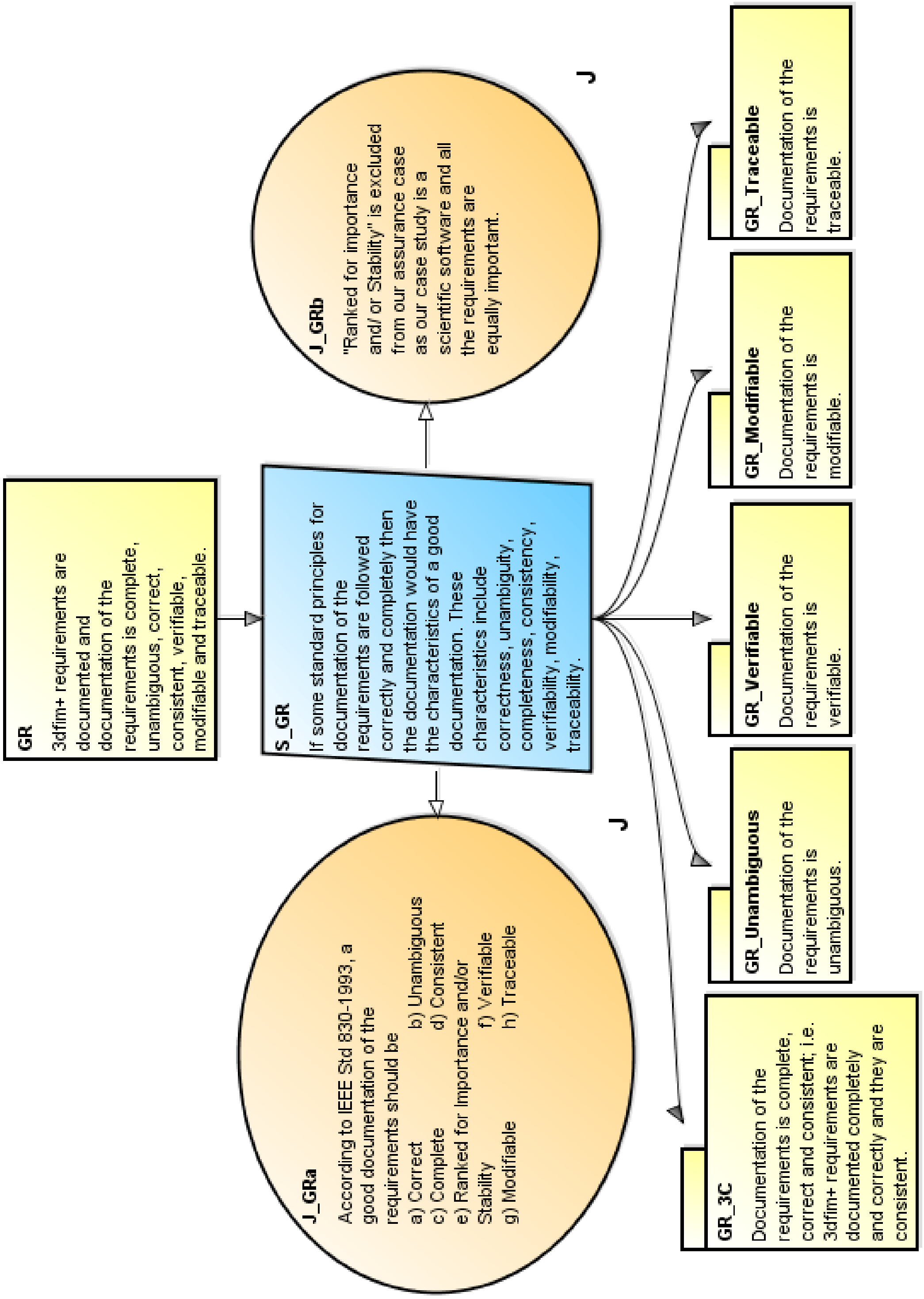

GR
3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
S_GR
If some standard principles for documentation of the requirements are followed correctly and completely then the documentation would have the characteristics of a good documentation. These characteristics include correctness, unambiguity, completeness, consistency, verifiability, modifiability, traceability.
J_GRb
"Ranked for importance and/ or Stability" is excluded from our assurance case as our case study is a scientific software and all the requirements are equally important.
J
J_GRa
According to IEEE Std 830-1993, a good documentation of the requirements should be
a) Correct b) Unambiguous
c) Complete d) Consistent
e) Ranked for Importance and/or Stability f) Verifiable
g) Modifiable h) Traceable
J
GR_Traceable
Documentation of the requirements is traceable.
GR_Modifiable
Documentation of the requirements is modifiable.
GR_Verifiable
Documentation of the requirements is verifiable.
GR_Unambiguous
Documentation of the requirements is unambiguous.
GR_3C
Documentation of the requirements is complete, correct and consistent; i.e. 3dfim+ requirements are documented completely and correctly and they are consistent.

Figure 4.7: GR decomposition

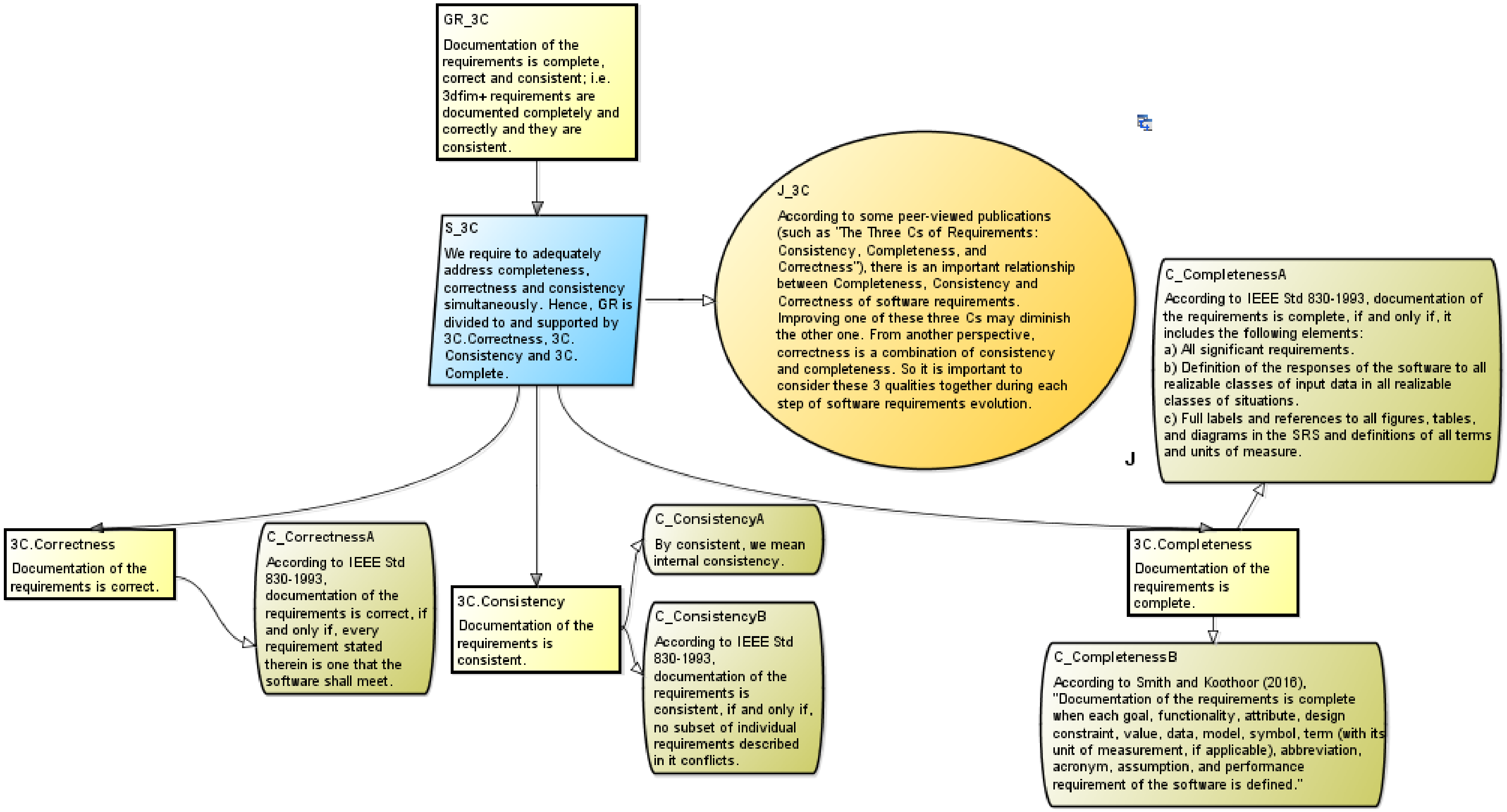

GR_3C
Documentation of the requirements is complete, correct and consistent; i.e. 3dfim+ requirements are documented completely and correctly and they are consistent.
S_3C
We require to adequately address completeness, correctness and consistency simultaneously. Hence, GR is divided to and supported by 3C.Correctness, 3C. Consistency and 3C. Complete.
J_3C
According to some peer-viewed publications (such as "The Three Cs of Requirements: Consistency, Completeness, and Correctness"), there is an important relationship between Completeness, Consistency and Correctness of software requirements. Improving one of these three Cs may diminish the other one. From another perspective, correctness is a combination of consistency and completeness. So it is important to consider these 3 qualities together during each step of software requirements evolution.
J
C_CompletenessA
According to IEEE Std 830-1993, documentation of the requirements is complete, if and only if, it includes the following elements:
a) All significant requirements.
b) Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations.
c) Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.
3C.Correctness
Documentation of the requirements is correct.
C_CorrectnessA
According to IEEE Std 830-1993, documentation of the requirements is correct, if and only if, every requirement stated therein is one that the software shall meet.
3C.Consistency
Documentation of the requirements is consistent.
C_ConsistencyA
By consistent, we mean internal consistency.
C_ConsistencyB
According to IEEE Std 830-1993, documentation of the requirements is consistent, if and only if, no subset of individual requirements described in it conflicts.
3C.Completeness
Documentation of the requirements is complete.
C_CompletenessB
According to Smith and Koothoor (2016), "Documentation of the requirements is complete when each goal, functionality, attribute, design constraint, value, data, model, symbol, term (with its unit of measurement, if applicable), abbreviation, acronym, assumption, and performance requirement of the software is defined."

Figure 4.8: Goal decomposition for Consistency, Completeness, and Correctness

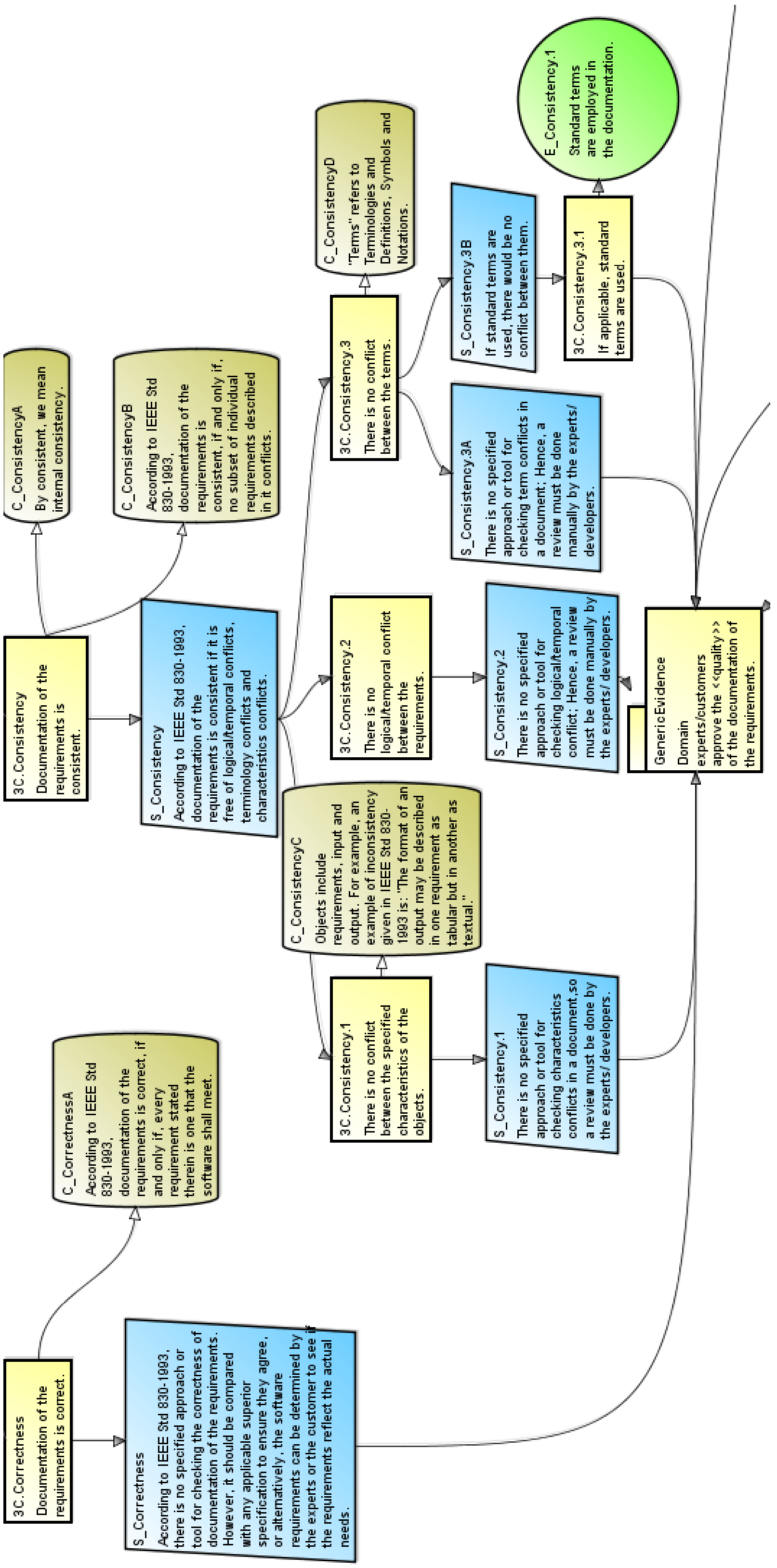

3C.Consistency
Documentation of the requirements is consistent.
C_ConsistencyA
By consistent, we mean internal consistency.
C_ConsistencyB
According to IEEE Std 830-1993, documentation of the requirements is consistent, if and only if, no subset of individual requirements described in it conflicts.
S_Consistency
According to IEEE Std 830-1993, documentation of the requirements is consistent if it is free of logical/temporal conflicts, terminology conflicts and characteristics conflicts.
3C.Consistency.3
There is no conflict between the terms.
C_ConsistencyD
"Terms" refers to Terminologies and Definitions, Symbols and Notations.
S_Consistency.3B
If standard terms are used, there would be no conflict between them.
3C.Consistency.3.1
If applicable, standard terms are used.
E_Consistency.1
Standard terms are employed in the documentation.
S_Consistency.3A
There is no specified approach or tool for checking term conflicts in a document; Hence, a review must be done manually by the experts/ developers.
3C.Consistency.2
There is no logical/temporal conflict between the requirements.
S_Consistency.2
There is no specified approach or tool for checking logical/temporal conflict; Hence, a review must be done manually by the experts/ developers.
GenericEvidence
Domain experts/customers approve the <<quality>> of the documentation of the requirements.
C_ConsistencyC
Objects include requirements, input and output. For example, an example of inconsistency given in IEEE Std 830-1993 is: "The format of an output may be described in one requirement as tabular but in another as textual."
3C.Consistency.1
There is no conflict between the specified characteristics of the objects.
S_Consistency.1
There is no specified approach or tool for checking characteristics conflicts in a document,so a review must be done by the experts/ developers.
C_CorrectnessA
According to IEEE Std 830-1993, documentation of the requirements is correct, if and only if, every requirement stated therein is one that the software shall meet.
3C.Correctness
Documentation of the requirements is correct.
S_Correctness
According to IEEE Std 830-1993, there is no specified approach or tool for checking the correctness of documentation of the requirements. However, it should be compared with any applicable superior specification to ensure they agree, or alternatively, the software requirements can be determined by the experts or the customer to see if the requirements reflect the actual needs.

Figure 4.9: Arguments for Correctness and Consistency

We use a similar argument for completeness of the documentation.

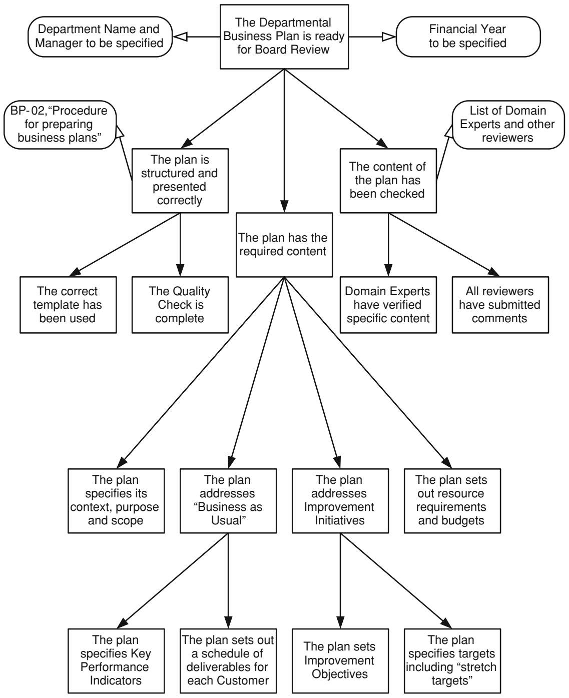

Department Name and Manager to be specified
The Departmental Business Plan is ready for Board Review
Financial Year to be specified
BP-02,"Procedure for preparing business plans"
List of Domain Experts and other reviewers
The plan is structured and presented correctly
The content of the plan has been checked
The plan has the required content
The correct template has been used
The Quality Check is complete
Domain Experts have verified specific content
All reviewers have submitted comments
The plan specifies its context, purpose and scope
The plan addresses "Business as Usual"
The plan addresses Improvement Initiatives
The plan sets out resource requirements and budgets
The plan specifies Key Performance Indicators
The plan sets out a schedule of deliverables for each Customer
The plan sets Improvement Objectives
The plan specifies targets including "stretch targets"

Figure 4.10: A part of the business plan readiness argument Spriggs [2]

We presented the top half of the 3C.Completeness argument as Figure 4.11 with 3 sub-goals. In Figure 4.12 we have repeated this goal again but this time with its sub-goals. As Figure 4.12 shows, most of the sub-goals are directed to a module called GenericEvidence, instead of ending up to an evidence. We discuss this module in the next paragraphs.

The content of the documentation of the requirements must be reviewed and verified by domain experts. Spriggs [2] gives a decomposition for this argument shown in Figure 4.13. We have developed a similar decomposition in one of our modules, called GenericEvidence. 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

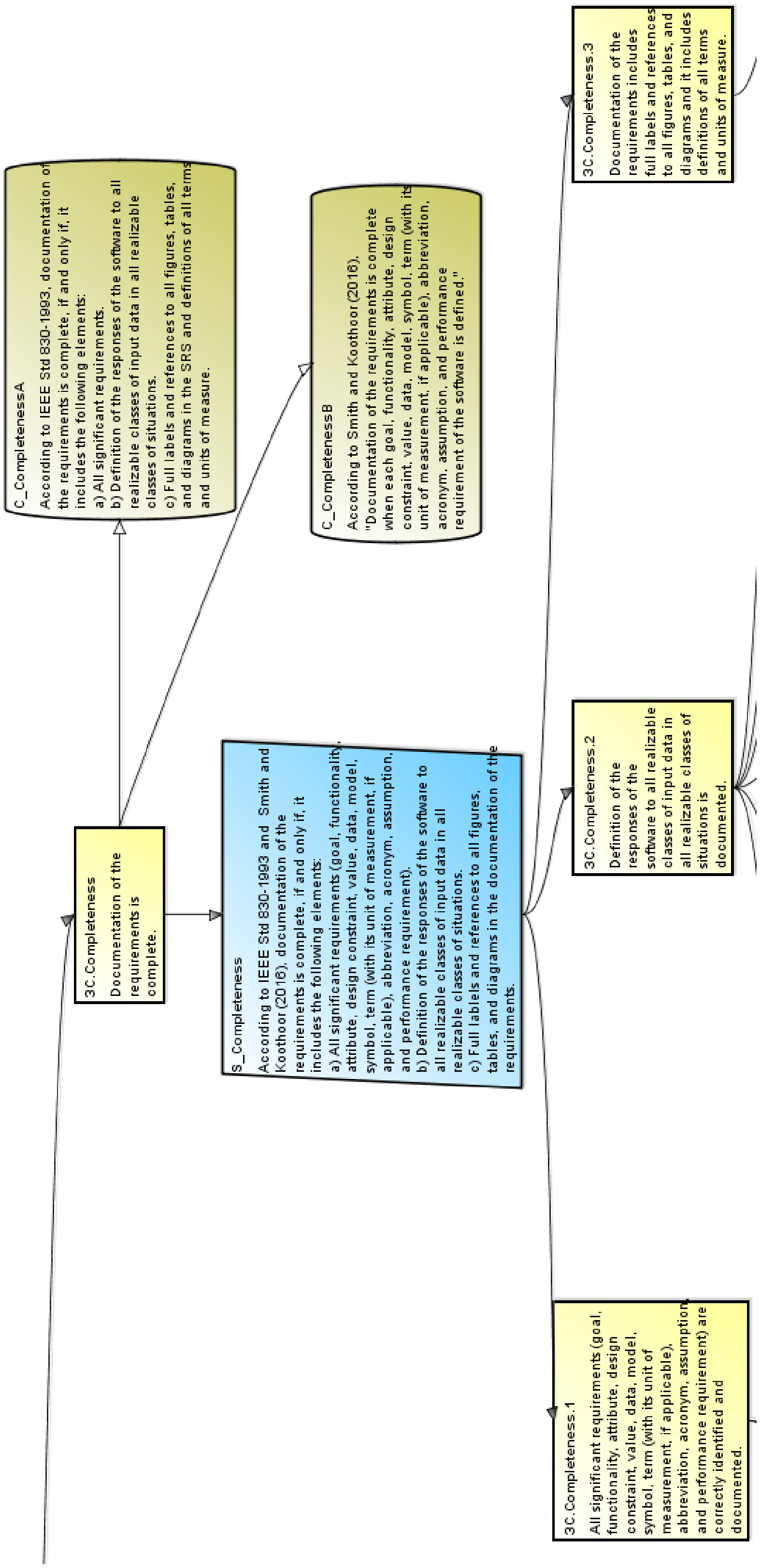

C_CompletenessA
According to IEEE Std 830-1993, documentation of the requirements is complete, if and only if, it includes the following elements:
a) All significant requirements.
b) Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations.
c) Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.
C_CompletenessB
According to Smith and Koothoor (2016), "Documentation of the requirements is complete when each goal, functionality, attribute, design constraint, value, data, model, symbol, term (with its unit of measurement, if applicable), abbreviation, acronym, assumption, and performance requirement of the software is defined."
3C.Completeness
Documentation of the requirements is complete.
S_Completeness
According to IEEE Std 830-1993 and Smith and Koothoor (2016), documentation of the requirements is complete, if and only if, it includes the following elements:
a) All significant requirements (goal, functionality, attribute, design constraint, value, data, model, symbol, term (with its unit of measurement, if applicable), abbreviation, acronym, assumption, and performance requirement).
b) Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations.
c) Full lablels and references to all figures, tables, and diagrams in the documentation of the requirements.
3C.Completeness.3
Documentation of the requirements includes full labels and references to all figures, tables, and diagrams and it includes definitions of all terms and units of measure.
3C.Completeness.2
Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations is documented.
3C.Completeness.1
All significant requirements (goal, functionality, attribute, design constraint, value, data, model, symbol, term (with its unit of measurement, if applicable), abbreviation, acronym, assumption, and performance requirement) are correctly identified and documented.

Figure 4.11: Top level argument for Completeness

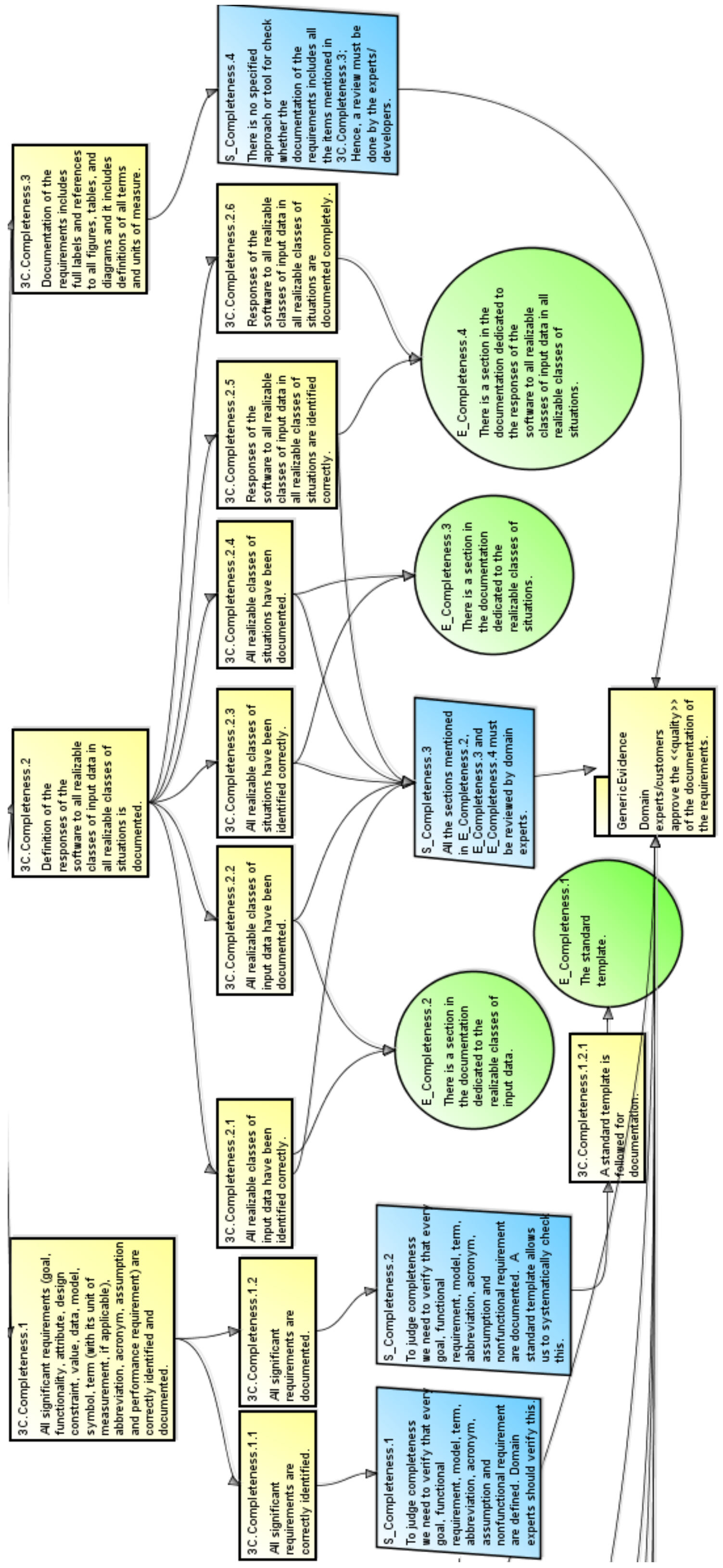

3C.Completeness.1
All significant requirements (goal, functionality, attribute, design constraint, value, data, model, symbol, term (with its unit of measurement, if applicable), abbreviation, acronym, assumption and performance requirement) are correctly identified and documented.
3C.Completeness.2
Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations is documented.
3C.Completeness.3
Documentation of the requirements includes full labels and references to all figures, tables, and diagrams and it includes definitions of all terms and units of measure.
3C.Completeness.1.1
All significant requirements are correctly identified.
3C.Completeness.1.2
All significant requirements are documented.
3C.Completeness.2.1
All realizable classes of input data have been identified correctly.
3C.Completeness.2.2
All realizable classes of input data have been documented.
3C.Completeness.2.3
All realizable classes of situations have been identified correctly.
3C.Completeness.2.4
All realizable classes of situations have been documented.
3C.Completeness.2.5
Responses of the software to all realizable classes of input data in all realizable classes of situations are identified correctly.
3C.Completeness.2.6
Responses of the software to all realizable classes of input data in all realizable classes of situations are documented completely.
S_Completeness.4
There is no specified approach or tool for check whether the documentation of the requirements includes all the items mentioned in 3C.Completeness.3; Hence, a review must be done by the experts/ developers.
S_Completeness.1
To judge completeness we need to verify that every goal, functional requirement, model, term, abbreviation, acronym, assumption and nonfunctional requirement are defined. Domain experts should verify this.
S_Completeness.2
To judge completeness we need to verify that every goal, functional requirement, model, term, abbreviation, acronym, assumption and nonfunctional requirement are documented. A standard template allows us to systematically check this.
E_Completeness.2
There is a section in the documentation dedicated to the realizable classes of input data.
S_Completeness.3
All the sections mentioned in E_Completeness.2, E_Completeness.3 and E_Completeness.4 must be reviewed by domain experts.
E_Completeness.3
There is a section in the documentation dedicated to the realizable classes of situations.
E_Completeness.4
There is a section in the documentation dedicated to the responses of the software to all realizable classes of input data in all realizable classes of situations.
3C.Completeness.1.2.1
A standard template is followed for documentation.
E_Completeness.1
The standard template.
GenericEvidence
Domain experts/customers approve the <<quality>> of the documentation of the requirements.

Figure 4.12: Sub-goals for Completeness

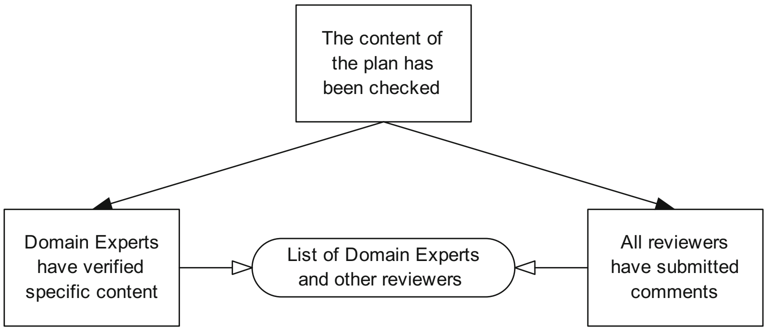

The content of the plan has been checked
Domain Experts have verified specific content
List of Domain Experts and other reviewers
All reviewers have submitted comments

Figure 4.13: An example of an argument for review from [2]

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 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 4.14. .

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” [2]. This module is shown in Figure 4.14.

According to IEEE Std 830-1993 [56], documentation of the requirements is verifiable, if and only if, every requirement stated therein is verifiable. Further information and definitions of verifiability of documentation are mentioned as contexts in Figure 4.15.

In Figure 4.16, we presented the argument for having unambiguous documentation of the requirements and its associated sub-goals. Having unambiguous requirements is important; misinterpretation of requirements is the source of 40% of all bugs in

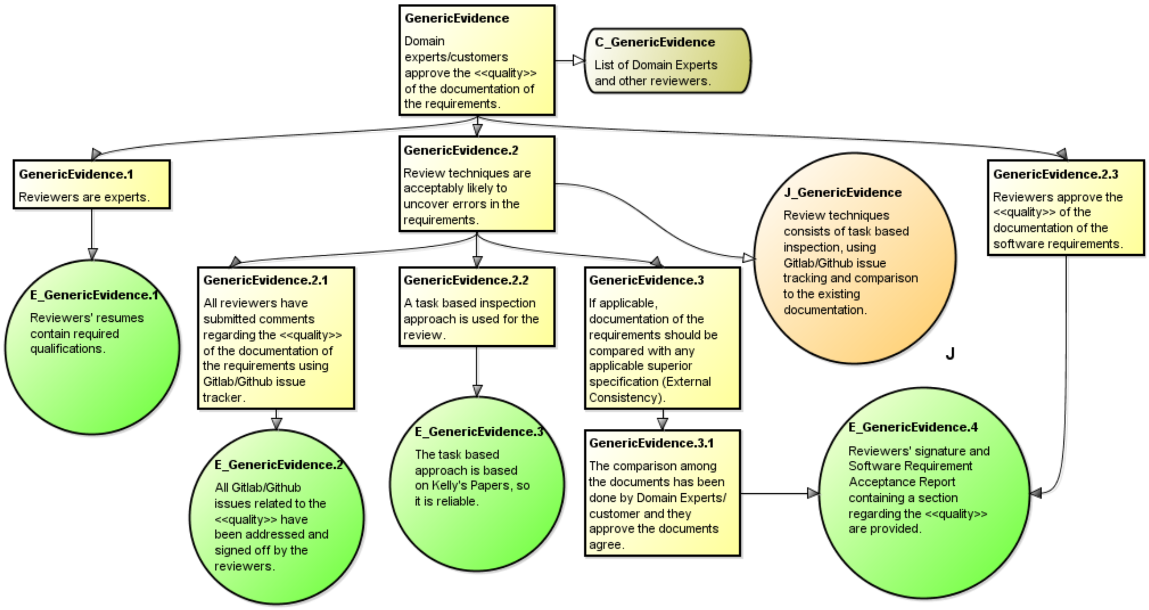

GenericEvidence
Domain experts/customers approve the <<quality>> of the documentation of the requirements.
C_GenericEvidence
List of Domain Experts and other reviewers.
GenericEvidence.1
Reviewers are experts.
GenericEvidence.2
Review techniques are acceptably likely to uncover errors in the requirements.
GenericEvidence.2.3
Reviewers approve the <<quality>> of the documentation of the software requirements.
J_GenericEvidence
Review techniques consists of task based inspection, using Gitlab/Github issue tracking and comparison to the existing documentation.
J
E_GenericEvidence.1
Reviewers' resumes contain required qualifications.
GenericEvidence.2.1
All reviewers have submitted comments regarding the <<quality>> of the documentation of the requirements using Gitlab/Github issue tracker.
GenericEvidence.2.2
A task based inspection approach is used for the review.
GenericEvidence.3
If applicable, documentation of the requirements should be compared with any applicable superior specification (External Consistency).
E_GenericEvidence.2
All Gitlab/Github issues related to the <<quality>> have been addressed and signed off by the reviewers.
E_GenericEvidence.3
The task based approach is based on Kelly's Papers, so it is reliable.
GenericEvidence.3.1
The comparison among the documents has been done by Domain Experts/ customer and they approve the documents agree.
E_GenericEvidence.4
Reviewers' signature and Software Requirement Acceptance Report containing a section regarding the <<quality>> are provided.

Figure 4.14: GenericEvidence module used as a pattern in our assurance case

delivered software [58]. For a documentation of the requirements to be unambiguous, each requirement must be described using a single unique term. Each term also must have one meaning. In case a term has several meanings, the term must be mentioned in a glossary and its particular meanings should be given specifically.

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 changes” [59]. 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. The argument for a modifiable documentation of the requirements and its associated sub-goals is shown as Figure 4.17.

According to IEEE Std 830-1993 [56], software requirements are traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation. Two types of

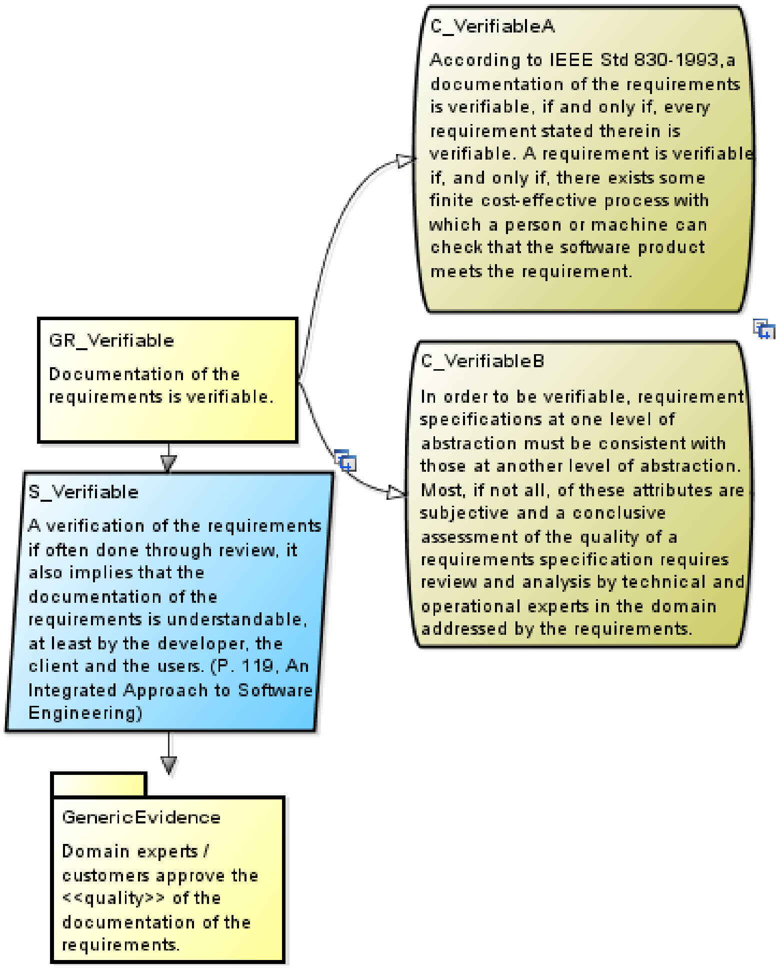

C_VerifiableA
According to IEEE Std 830-1993,a documentation of the requirements is verifiable, if and only if, every requirement stated therein is verifiable. A requirement is verifiable if, and only if, there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement.
GR_Verifiable
Documentation of the requirements is verifiable.
C_VerifiableB
In order to be verifiable, requirement specifications at one level of abstraction must be consistent with those at another level of abstraction. Most, if not all, of these attributes are subjective and a conclusive assessment of the quality of a requirements specification requires review and analysis by technical and operational experts in the domain addressed by the requirements.
S_Verifiable
A verification of the requirements if often done through review, it also implies that the documentation of the requirements is understandable, at least by the developer, the client and the users. (P. 119, An Integrated Approach to Software Engineering)
GenericEvidence
Domain experts / customers approve the <<quality>> of the documentation of the requirements.

Figure 4.15: Argument for verifiability of documentation of requirements

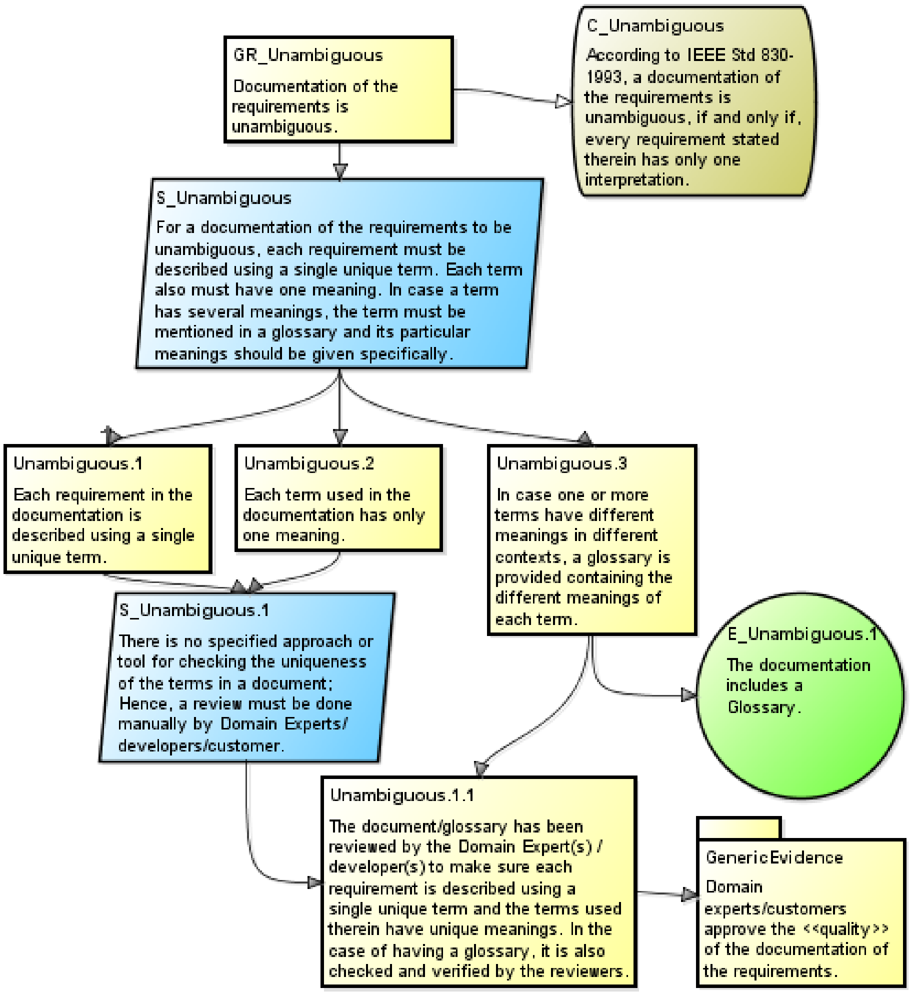

GR_Unambiguous
Documentation of the requirements is unambiguous.
C_Unambiguous
According to IEEE Std 830-1993, a documentation of the requirements is unambiguous, if and only if, every requirement stated therein has only one interpretation.
S_Unambiguous
For a documentation of the requirements to be unambiguous, each requirement must be described using a single unique term. Each term also must have one meaning. In case a term has several meanings, the term must be mentioned in a glossary and its particular meanings should be given specifically.
Unambiguous.1
Each requirement in the documentation is described using a single unique term.
Unambiguous.2
Each term used in the documentation has only one meaning.
Unambiguous.3
In case one or more terms have different meanings in different contexts, a glossary is provided containing the different meanings of each term.
S_Unambiguous.1
There is no specified approach or tool for checking the uniqueness of the terms in a document; Hence, a review must be done manually by Domain Experts/ developers/customer.
E_Unambiguous.1
The documentation includes a Glossary.
Unambiguous.1.1
The document/glossary has been reviewed by the Domain Expert(s) / developer(s) to make sure each requirement is described using a single unique term and the terms used therein have unique meanings. In the case of having a glossary, it is also checked and verified by the reviewers.
GenericEvidence
Domain experts/customers approve the <<quality>> of the documentation of the requirements.

Figure 4.16: Goal decomposition for an unambiguous documentation of requirements

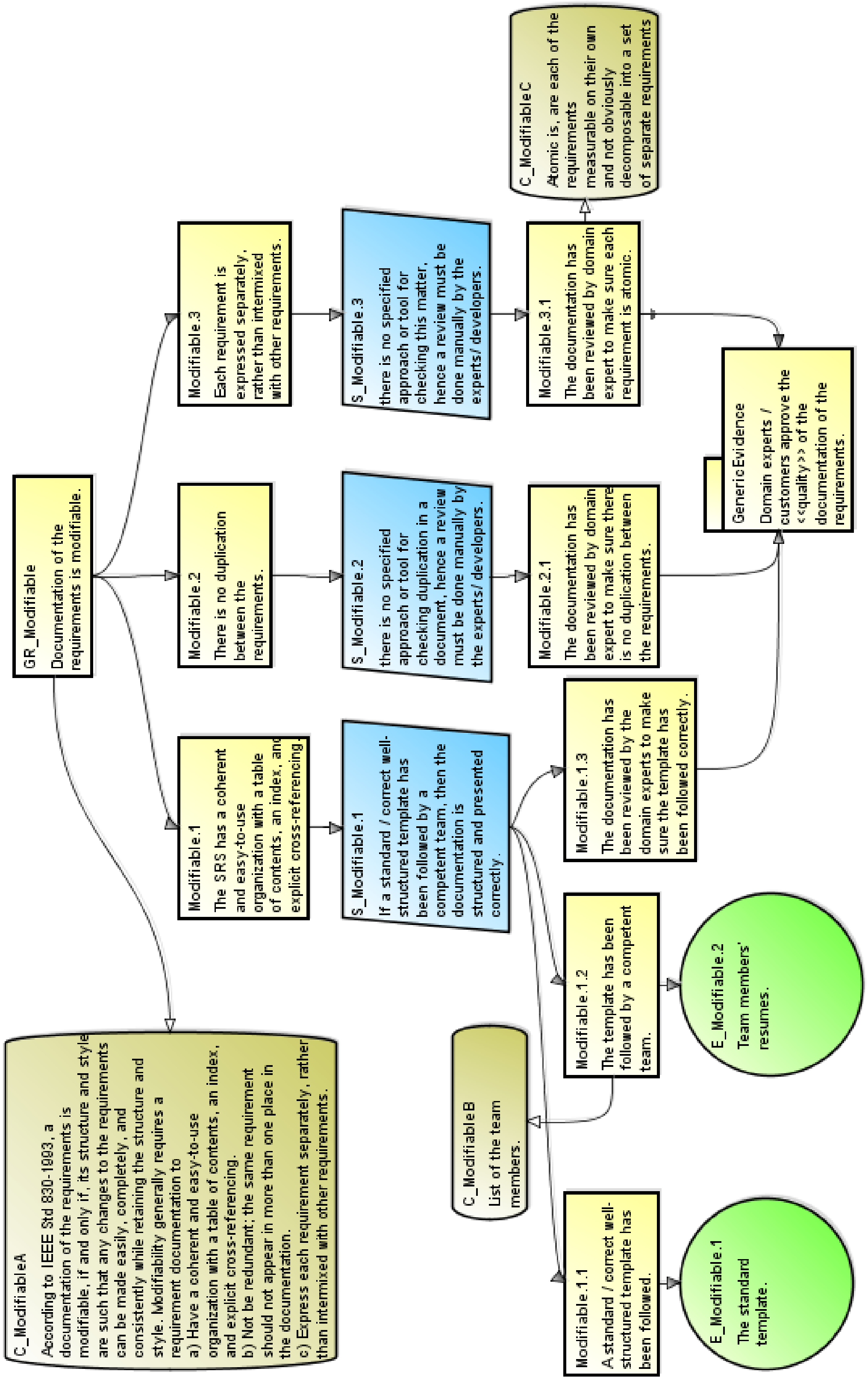

GR_Modifiable
Documentation of the requirements is modifiable.
C_ModifiableA
According to IEEE Std 830-1993, a documentation of the requirements is modifiable, if and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style. Modifiability generally requires a requirement documentation to
a) Have a coherent and easy-to-use organization with a table of contents, an index, and explicit cross-referencing.
b) Not be redundant; the same requirement should not appear in more than one place in the documentation.
c) Express each requirement separately, rather than intermixed with other requirements.
Modifiable.1
The SRS has a coherent and easy-to-use organization with a table of contents, an index, and explicit cross-referencing.
Modifiable.2
There is no duplication between the requirements.
Modifiable.3
Each requirement is expressed separately, rather than intermixed with other requirements.
S_Modifiable.1
If a standard / correct well-structured template has been followed by a competent team, then the documentation is structured and presented correctly.
S_Modifiable.2
there is no specified approach or tool for checking duplication in a document, hence a review must be done manually by the experts/ developers.
S_Modifiable.3
there is no specified approach or tool for checking this matter, hence a review must be done manually by the experts/ developers.
C_ModifiableB
List of the team members.
Modifiable.2.1
The documentation has been reviewed by domain expert to make sure there is no duplication between the requirements.
Modifiable.3.1
The documentation has been reviewed by domain expert to make sure each requirement is atomic.
C_ModifiableC
Atomic is, are each of the requirements measurable on their own and not obviously decomposable into a set of separate requirements
Modifiable.1.1
A standard / correct well-structured template has been followed.
Modifiable.1.2
The template has been followed by a competent team.
Modifiable.1.3
The documentation has been reviewed by the domain experts to make sure the template has been followed correctly.
GenericEvidence
Domain experts / customers approve the <<quality>> of the documentation of the requirements.
E_Modifiable.1
The standard template.
E_Modifiable.2
Team members' resumes.

Figure 4.17: Argument for modifiability of documentation of requirements

traceability exist:

- Backward Traceability: Each requirement explicitly referencing its source in earlier documents.
- Forward Traceability: Each requirement in the requirements documentation has a unique name or reference number.

The argument we have developed for traceability of the documentation is shown in Figure 4.18.

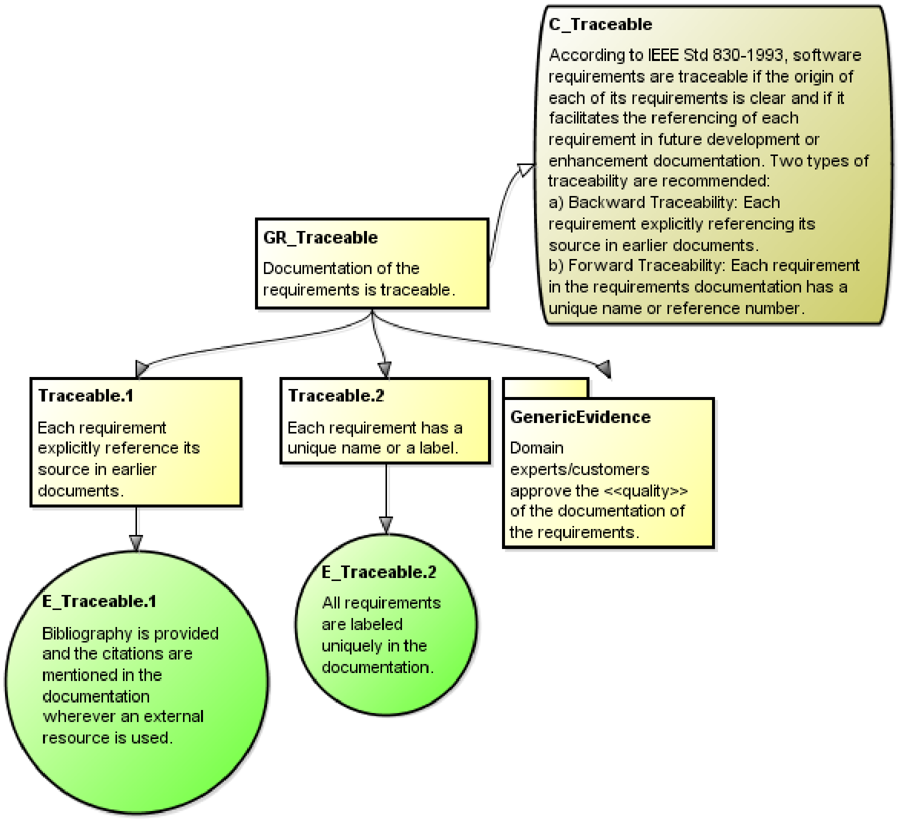

C_Traceable
According to IEEE Std 830-1993, software requirements are traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation. Two types of traceability are recommended:
a) Backward Traceability: Each requirement explicitly referencing its source in earlier documents.
b) Forward Traceability: Each requirement in the requirements documentation has a unique name or reference number.
GR_Traceable
Documentation of the requirements is traceable.
Traceable.1
Each requirement explicitly reference its source in earlier documents.
Traceable.2
Each requirement has a unique name or a label.
GenericEvidence
Domain experts/customers approve the <<quality>> of the documentation of the requirements.
E_Traceable.1
Bibliography is provided and the citations are mentioned in the documentation wherever an external resource is used.
E_Traceable.2
All requirements are labeled uniquely in the documentation.

Figure 4.18: Goal decomposition for document traceability

The next step after the requirement specification is typically the software design and then the software implementation. Having a documented requirement specification can facilitate the stage. The arguments for the software design and implementation are shown in Figures 4.19 and 4.20. As the focus of our work was on the quality of the documentation, these goals are undeveloped and need further development. This is indicated in the diagrams by the little diamonds at the bottom of the goals.

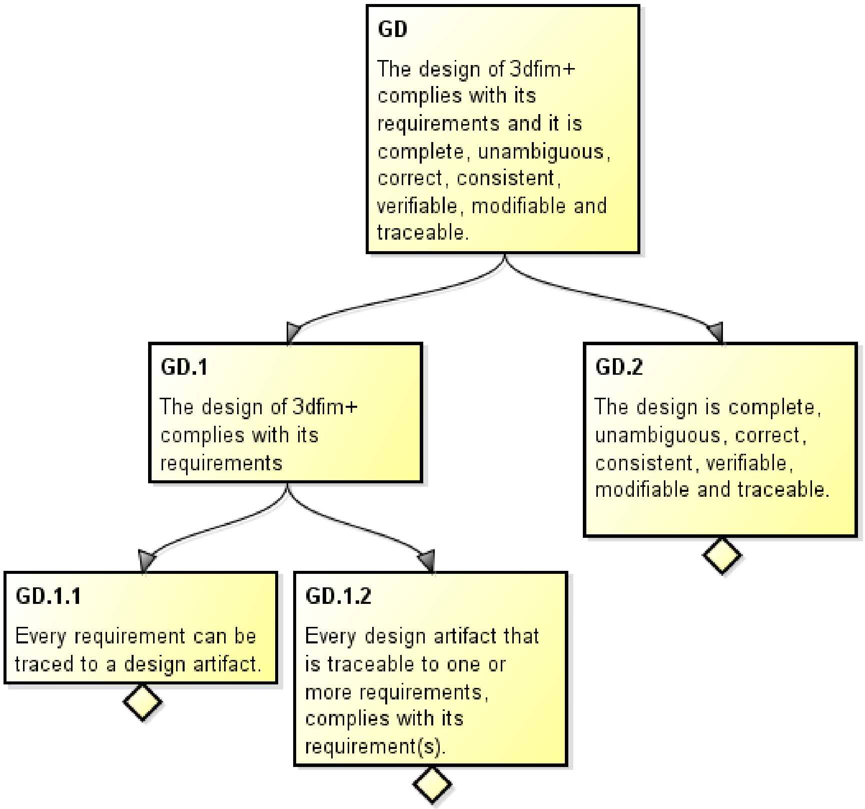

GD
The design of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GD.1
The design of 3dfim+ complies with its requirements
GD.2
The design is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GD.1.1
Every requirement can be traced to a design artifact.
GD.1.2
Every design artifact that is traceable to one or more requirements, complies with its requirement(s).

Figure 4.19: Goal decomposition for software design

In addition, 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. This argument is shown in Figure 4.21.

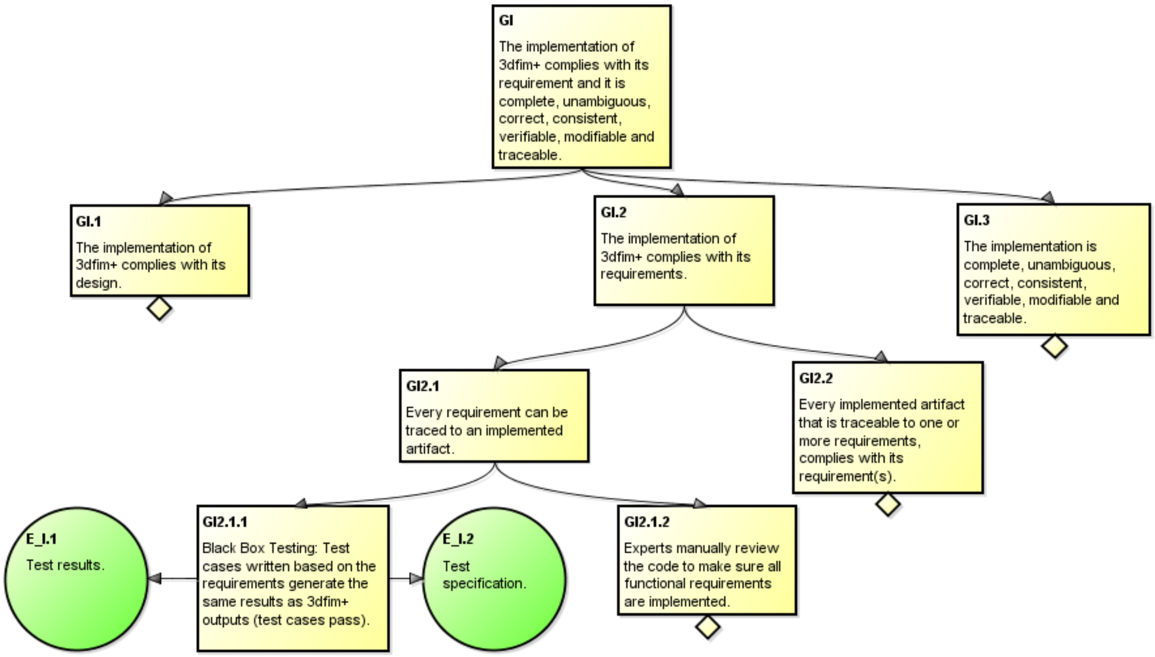

GI
The implementation of 3dfim+ complies with its requirement and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GI.1
The implementation of 3dfim+ complies with its design.
GI.2
The implementation of 3dfim+ complies with its requirements.
GI.3
The implementation is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GI2.1
Every requirement can be traced to an implemented artifact.
GI2.2
Every implemented artifact that is traceable to one or more requirements, complies with its requirement(s).
E_I.1
Test results.
GI2.1.1
Black Box Testing: Test cases written based on the requirements generate the same results as 3dfim+ outputs (test cases pass).
E_I.2
Test specification.
GI2.1.2
Experts manually review the code to make sure all functional requirements are implemented.

Figure 4.20: Goal decomposition for software implementation

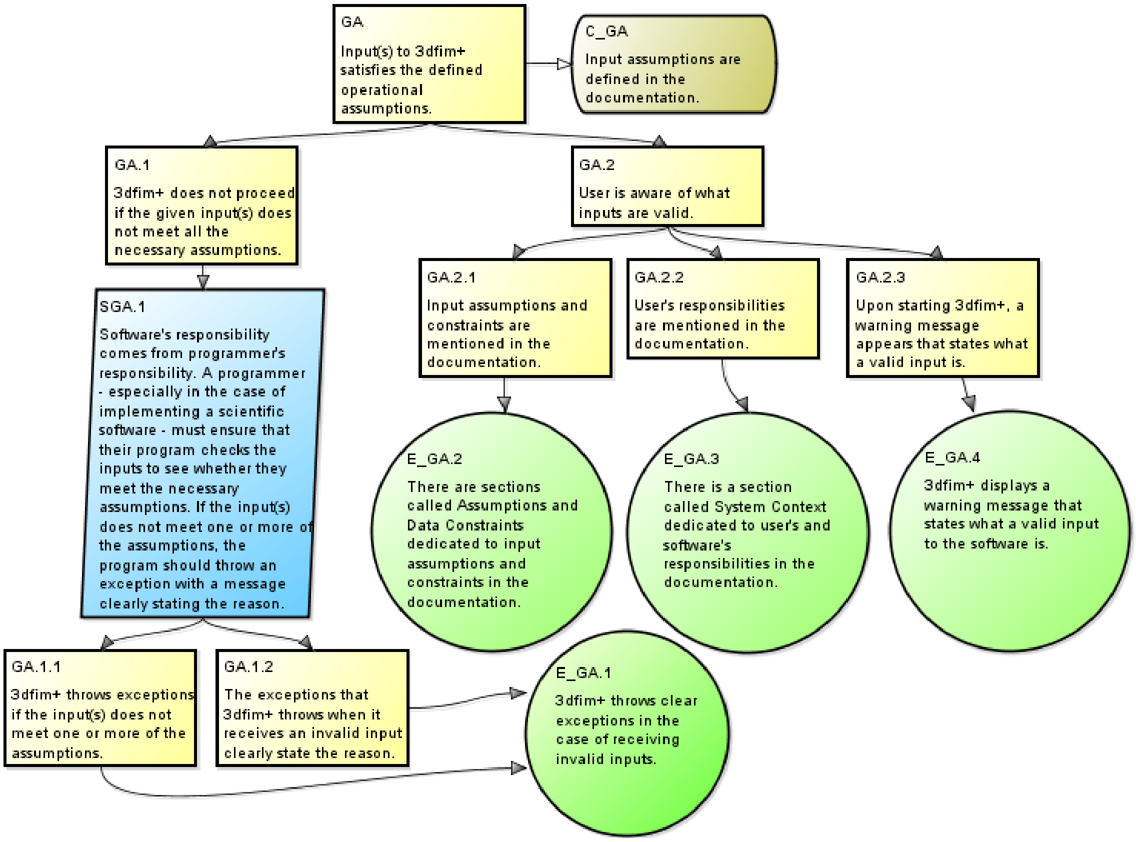

GA
Input(s) to 3dfim+ satisfies the defined operational assumptions.
C_GA
Input assumptions are defined in the documentation.
GA.1
3dfim+ does not proceed if the given input(s) does not meet all the necessary assumptions.
GA.2
User is aware of what inputs are valid.
SGA.1
Software's responsibility comes from programmer's responsibility. A programmer - especially in the case of implementing a scientific software - must ensure that their program checks the inputs to see whether they meet the necessary assumptions. If the input(s) does not meet one or more of the assumptions, the program should throw an exception with a message clearly stating the reason.
GA.2.1
Input assumptions and constraints are mentioned in the documentation.
GA.2.2
User's responsibilities are mentioned in the documentation.
GA.2.3
Upon starting 3dfim+, a warning message appears that states what a valid input is.
E_GA.2
There are sections called Assumptions and Data Constraints dedicated to input assumptions and constraints in the documentation.
E_GA.3
There is a section called System Context dedicated to user's and software's responsibilities in the documentation.
E_GA.4
3dfim+ displays a warning message that states what a valid input to the software is.
GA.1.1
3dfim+ throws exceptions if the input(s) does not meet one or more of the assumptions.
GA.1.2
The exceptions that 3dfim+ throws when it receives an invalid input clearly state the reason.
E_GA.1
3dfim+ throws clear exceptions in the case of receiving invalid inputs.

Figure 4.21: Argument for inputs satisfying the defined operational assumptions

4.3 Software Requirements Specification Development

Having a documented software requirements specification (SRS) is one of the most important principles of software validation [18]. Such document provides a baseline for software validation and verification. Having an SRS is necessary for conducting a complete software validation process [18]. Another reason for the importance of the SRS is that the only way to judge the correctness of scientific software is by comparing it to a specification of the requirements [13].

Another advantage of having a requirement specification is that the input assumptions, theoretical and mathematical background information, the scope of the software and in general all supporting information can be explicitly and clearly mentioned therein. As mentioned in the previous section, requirement documentation can improve software qualities such as correctness, completeness, understandability, readability, verifiability, reusability and maintainability.

Moreover, the SRS is part of the evidence for the assurance case. For example, C2 in Figure 4.5, E_Traceable.1 and E_Traceable.2 in Figure 4.18 and E_GA.2 and E_GA.3 in Figure 4.30 are some of the elements in our assurance case where the evidence comes from the SRS. The SRS that we have developed for 3dfim+ is included in Appendix 1.

An SRS must be of high quality and have all the quality features that are previously represented in Figure 4.7. To come up with a high-quality SRS, we have followed the IEEE 830-1993 [56] standard. It describes the content of a good SRS, alongside presenting several sample SRS outlines. According to this IEEE Standard, a good SRS provides several benefits such as:

- Facilitating the understanding of the software. It helps the user decide whether the software is a good choice for their needs by giving full description of the functionality the software performs.
- Enabling early corrections. Through carefully reviewing and documenting the requirements, it is more probable that inconsistencies and misunderstandings can be revealed earlier when they are easier to fix.
- Reducing the development effort. The design is done based on the requirements, and once the requirements are fully and carefully documented in the SRS, designing the corresponding components and modules is easier.
- Reducing the validation and verification effort. The SRS provides information based on what validation and verification activities are conducted. It provides a baseline against which compliance can be evaluated.
- Facilitating the comparison of the different tools and software.
- Improving the communication between experts.
- Improving understandability for the users.

We followed a requirements template for scientific software with some modifications. This template is given by Smith [13] and is also based on the IEEE Standard 830. The template is represented in Figure 4.22.

Our template is represented in Figure 4.22. Most of the sections are borrowed from the Smith template (Figure 4.22). The major modifications from the Smith template are discussed below:

- In section 4 we do not mention the System Behavior. Instead we mention it in section 5 (Requirements). We have also modified section 4 and added more sub-sections to it where necessary.

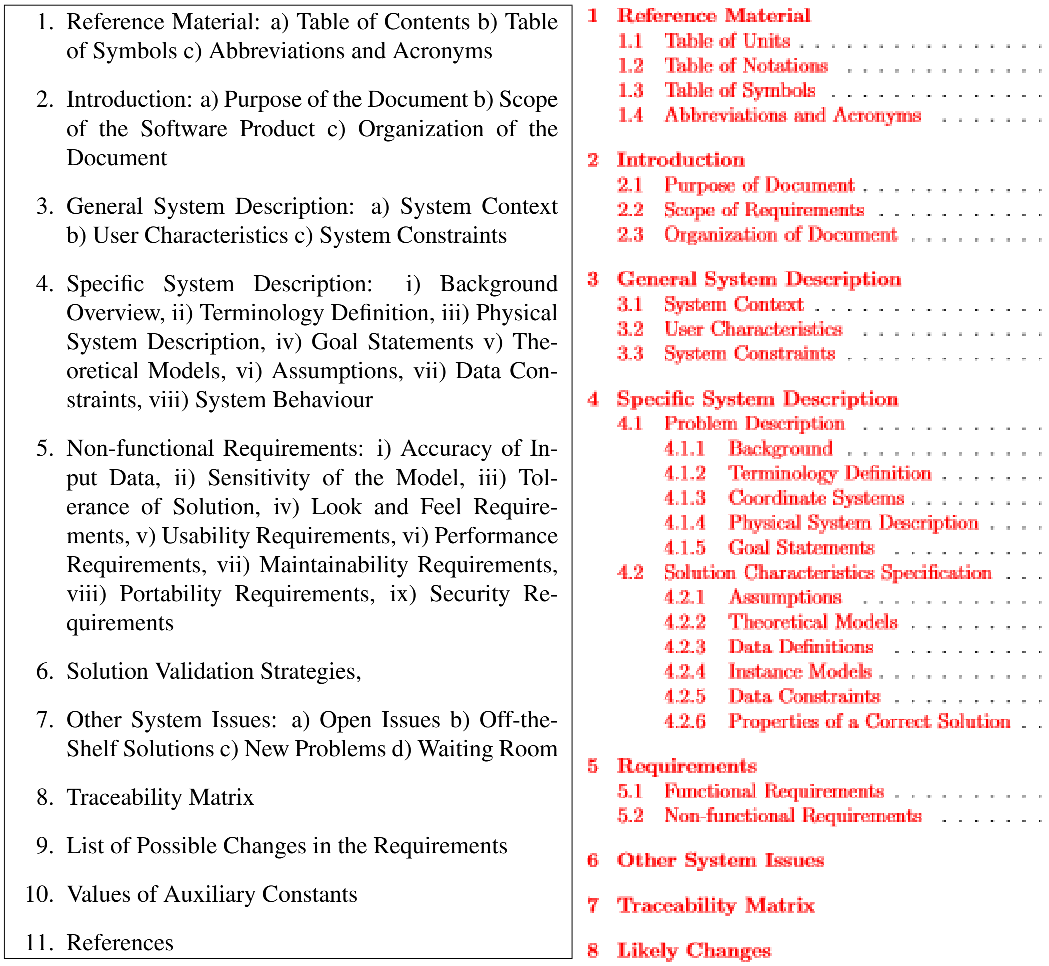
- We have changed section 5 to contain both functional and non-functional requirements and our emphasis is on the functional requirements.

- We do not provide a section for Solution Validation Strategies, we discuss them in a separate document (Verification and Validation Plan). Values of Auxiliary Constant are mentioned in the theoretical models and we do not provide a separate section for them. We also have sections for Traceability Matrix and System Issues.

Before discussing the different sections in our SRS in detail, we need to clarify some of the terms.

Requirement: According to [18], a requirement "can be any need or expectation for a system or for its software. Requirements reflect the stated or implied needs of the customer, and may be market-based, contractual, or statutory, as well as an organization's internal requirements". Software requirements are typically derived from the functionality that has been allocated to software and is typically stated in functional terms.

In this thesis, we derived some of the requirements such as functional requirements from the existing documentation [50]. This documentation focuses on the functionality of the software and the terms are explained at a high level. Other requirements such as assumptions are derived based on the mathematical models and formulas that are used in the program. These formulas are given in the



1. Reference Material: a) Table of Contents b) Table of Symbols c) Abbreviations and Acronyms
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: i) Background Overview, ii) Terminology Definition, iii) Physical System Description, iv) Goal Statements v) Theoretical Models, vi) Assumptions, vii) Data Constraints, viii) System Behaviour
5. Non-functional Requirements: i) Accuracy of Input Data, ii) Sensitivity of the Model, iii) Tolerance of Solution, iv) Look and Feel Requirements, v) Usability Requirements, vi) Performance Requirements, vii) Maintainability Requirements, viii) Portability Requirements, ix) Security Requirements
6. Solution Validation Strategies,
7. Other System Issues: a) Open Issues b) Off-the-Shelf Solutions c) New Problems d) Waiting Room
8. Traceability Matrix
9. List of Possible Changes in the Requirements
10. Values of Auxiliary Constants
11. References

1 Reference Material
1.1 Table of Units . . . .
1.2 Table of Notations . . . .
1.3 Table of Symbols . . . .
1.4 Abbreviations and Acronyms . . . .

2 Introduction
2.1 Purpose of Document . . . .
2.2 Scope of Requirements . . . .
2.3 Organization of Document . . . .

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

4 Specific System Description
4.1 Problem Description . . . .
4.1.1 Background . . . .
4.1.2 Terminology Definition . . . .
4.1.3 Coordinate Systems . . . .
4.1.4 Physical System Description . . . .
4.1.5 Goal Statements . . . .
4.2 Solution Characteristics Specification . . .
4.2.1 Assumptions . . . .
4.2.2 Theoretical Models . . . .
4.2.3 Data Definitions . . . .
4.2.4 Instance Models . . . .
4.2.5 Data Constraints . . . .
4.2.6 Properties of a Correct Solution . .

5 Requirements
5.1 Functional Requirements . . . .
5.2 Non-functional Requirements . . . .

6 Other System Issues

7 Traceability Matrix

8 Likely Changes

Figure 4.22: (a) Requirement template provided by [13] and (b) Table of Content of Our SRS

existing document [50], but the assumptions are not mentioned. We investigated other resources to come up with a complete documentation of the requirements.

Specification: A specification is “a document that states requirements” [18]. Another definition is given by [56] as “A specification that documents the requirements of a system or system component. It typically includes functional requirements, performance requirements, interface requirements, design requirements [attributes and constraints], development [coding] standards, etc.”.

In this thesis, whenever we use the term specification, we are referring to the specification that documents the requirements of the software.

Requirement Analysis: Requirement Analysis is defined as “(1) The process of studying user needs to arrive at a definition of a system, hardware, or software requirements. or (2) The process of studying and refining system, hardware, or software requirements” [56].

In this thesis, we conducted the requirement analysis as explained in the second definition. We studied and refined the software requirements based on the existing documentation and implementation.

SRS: According to the IEEE definition [56], “The SRS is a specification for a particular software product, program, or set of programs that performs certain functions in a specific environment.”

In this thesis, the SRS we developed is a specification for the 3dfim+, which is a particular program that performs certain functions in its intended environment.

Moreover, our template complies with GPSV principles for SRS. GPSV specifies the information an SRS should contain. This information alongside the parts of our SRS that contains such information is given as follows.

- "All software system inputs"; The inputs are mentioned in Section 4.1.5 (Goal Statements) and are explained in detail in Section 4.2.3 (Data Definitions) (Figures 4.23, 4.24 and 4.25).

4.1.5 Goal Statements

Given an fMRI time series (DD6), one or more ideal time series (DD7) and zero or more orthogonal time series (DD13):

GS1: Estimate the Pearson correlation coefficients between the (best) ideal time series and the fMRI time series at each voxel over time.

GS2: Estimate the Spearman correlation coefficient between the (best) ideal time series and the fMRI time series at each voxel over time.

GS3: Estimate the quadrant correlation between the (best) ideal time series and the fMRI time series at each voxel over time.

GS4: In case of having multiple ideal signals, report the index number for the best ideal time series.

GS5: Calculate the percentage change in the fMRI time series due to the (best) ideal time series relative to the average for each voxel.

GS6: Calculate the percentage change in the fMRI time series due to the (best) ideal time series relative to the baseline for each voxel.

GS7: Calculate the fMRI time series baseline for each voxel.

GS8: Calculate the fMRI time series average for each voxel.

GS9: Calculate the percentage change in the fMRI time series due to the (best) ideal time series relative to the topline for each voxel.

GS10: Calculate the fMRI time series topline quantity for each voxel.

GS11: Calculate the standard deviation of the residuals at each voxel between the fMRI dataset and corresponding data estimation.

Figure 4.23: Goal Statements for 3dfim+

- "All software system outputs". We mention all the outputs in Section 5.1 as Functional Requirements (Figures 4.26).

Number	DD6
Name	3d+time
Label	**Mathematical Representation of 3d+time Dataset**
Symbol	$X : \mathbb{R}^{m \times n \times p \times q}$
Equation	-
Description	3d+time datasets are 4D datasets that have a temporal component, a time dimension that is the time intervals during scanning, collecting and concatenating datasets together. 3d+time datasets are the basic units of the fMRI.
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	GS1, DD8, DD14, DD16, DD17, DD19, IM1, IM2, IM3, IM4, IM9, IM10, R1

Number	DD7
Name	Ideal Signal
Label	**Mathematical Representation of Ideal (Reference) Signal (Time Series)**
Symbol	$r : \mathbb{R}^n$
Equation	-
Description	Ideal signal is a waveform of choice.
Source	https://en.wikipedia.org/wiki/Square_wave
Ref. By	GS1, DD16, IM1, IM2, IM3, IM4, IM6, IM9, IM10, IM12, R1, R7, R8, R11

Figure 4.24: Some of the inputs for 3dfim+

Number	DD13
Name	Orthogonal
Label	**Orthogonal Time Series**
Symbol	$\phi : \mathbb{R}^n$
Equation	-
Description	Time series that is perpendicular to the baseline (DD11). Two polynomials are orthogonal if their inner product is zero. We define an inner product for two functions by integrating their product. $\int_a^b \phi(x)\text{base}(x)dx = 0$
Source	https://www.johndcook.com/OrthogonalPolynomials.pdf
Ref. By	GS1, IM2, IM6, IM7, IM8, IM12, R1

Figure 4.25: Another input for 3dfim+

- "All functions that the software system will perform". The functions are mentioned briefly in Section 4.1.5 as Goal Statements and are discussed more in details in Section 4.2.4 as Instance Models (Figures 4.23, 4.27 and 4.28). The Goal Statement does not contain non-functional requirements.

- "All performance requirements that the software will meet, (e.g. data throughput, reliability, and timing)". When developing the SRS, although our focus was on the functional requirements, we dedicated a section (Section 5.2) for non-functional requirements.

- "The definition of all external and user interfaces, as well as any internal software-to-system interfaces". Refining the interfaces was not in the scope of our work.

- "How users will interact with the system". We explained the user interaction with the software system, and mentioned user's responsibilities in Section 3.1 as

5.1 Functional Requirements

R1: Input the following functions, data and parameters:

symbol	**description**
X	fMRI data as a 3d+time dataset in NIfTI format (DD6)
$pnum$	degree of the polynomial in the baseline model (DD12)
ϕ	orthogonal time series function(s) (DD13)
r	reference time series function(s) (DD7)
p	threshold for voxels' intensity (DD14)
$cval$	comparing value for correlation coefficient screen display (DD15)

R2: Use the inputs in R1 to estimate the vector of unknown parameters β (IM2) at each voxel (from IM2).

R3: Calculate the Pearson correlation coefficient at each voxel between X and (best) r (from IM1).

R4: Calculate the Spearman correlation coefficient at each voxel between X and (best) r (from IM3).

R5: Calculate the quadrant correlation coefficient at each voxel between X and (best) r (from IM4).

R6: In case of having multiple ideal signals r, report the index number k (DD17) for the best ideal time series r_k (DD16) (from IM5).

R7: Calculate the percentage change in X due to the (best) ideal time series (DD7, DD16) relative to the *Baseline* (IM6) for each voxel (from IM9).

Figure 4.26: 3dfim+ outputs

Number	IM1
Name	Pearson Model
Label	**Calculating Pearson Correlation Coefficient Between the Reference Signal and the Input Dataset**
Input	$X : \mathbb{R}^{m \times n \times p \times q}$, $r : \mathbb{R}^q$
Output	$\rho_{ijk}(X, r) = \frac{\sum_{l=1}^{q}(x_{ijkl} - \bar{x}_{ijk})(r_l - \bar{r})}{[\sum_{l=1}^{q}(x_{ijkl} - \bar{x}_{ijk})^2 (r_l - \bar{r})^2]^{\frac{1}{2}}}$
Description	The formula calculates the Pearson correlation coefficient (T1) between the ideal time series r (DD7) and the 3d+time dataset X (DD6). $\bar{x}_{ijk}$ and $\bar{r}$ are sample means (DD1) defining as follows: $\bar{x}_{ijk} = \frac{\sum_{l=1}^{q} x_{ijkl}}{q}$ $\bar{r} = \frac{\sum_{i=1}^{q} r_i}{q}$ Note that assumptions A1, A2, A3, A4 and A5 must hold while calculating this correlation. We also assumed that $r = r_k$ (DD16) in case of having more than one ideal signal.
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	R2, R3, LC2

Number	IM12
Name	Standard Deviation of the Residuals
Label	**Calculating The Standard Deviation of the Residuals at Each Voxel Between the fMRI Dataset and Corresponding Data Estimation**
Input	$X : \mathbb{R}^{m \times n \times p \times q}$, $\phi_i : \mathbb{R}^q$, $r_i : \mathbb{R}^q$
Output	$\hat{\sigma}_{ijk} = \sqrt{\frac{\sum_{l=1}^{q}(X_{ijkl} - \hat{X}_{ijkl})^2}{q - n_b - n_o - n_i}}$
Description	Extending the theoretical model T8 to the fMRI dataset, we have: $\hat{X}_{ijkl} = (M^T M)^{-1} M^T X_{ijkl}$ Using theoretical models T5, T6 and T7 we can calculate thestandard deviation of the residuals: $\hat{\sigma}_{ijk} = \sqrt{\frac{\sum_{l=1}^{q}(X_{ijkl} - \hat{X}_{ijkl})^2}{q - pnum - n_o - n_i}}$ Where: $pnum$ is the polynomial degree (DD12), n_o is the number of orthogonal time series (DD13), and n_i depends on the number of ideal time series (DD7) such that: $n_i = \begin{cases} 1 \text{ if we have 1 ideal time series} \\ 2 \text{ if we have more than one ideal time series} \end{cases}$
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	R13

Figure 4.27: Some of the Instance Models for 3dfim+

Number	IM2
Name	fMRI Dataset Model
Label	**Mathematical Model of Measured fMRI Dataset To Find Fit Coefficients**
Input	X, $\phi_i \in \mathbb{R}^q$, $r_i \in \mathbb{R}^q$
Output	$\beta_{ijk}^T = [\beta_0, \beta_1, \cdots, \gamma_1, \gamma_2, \cdots, \alpha_1, \alpha_2, \cdots]^T$
Description	Correlation analysis of each voxel's time series in X (DD6) with reference signal(s) r_i (DD7) where: $$M = \begin{bmatrix} 1 & 1 & \cdots & \phi_{1_1} & \cdots & r_{1_1} & \cdots \\ 1 & 2 & \cdots & \phi_{1_2} & \cdots & r_{1_2} & \cdots \\ 1 & 3 & \cdots & \phi_{1_3} & \cdots & r_{1_3} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ 1 & f & \cdots & \phi_{1_f} & \cdots & r_{1_f} & \cdots \end{bmatrix}$$ $$X_{ijk} = y_{ijk} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_f \end{bmatrix} \beta_{ijk}^* = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \gamma_1 \\ \gamma_2 \\ \vdots \\ \alpha_1 \\ \alpha_2 \\ \vdots \end{bmatrix} \epsilon_{ijk} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_f \end{bmatrix}$$ The equation can be also written as: $X_{ijk} = M\beta_{ijk}^* + \epsilon_{ijk}$ where: M is the data model consisting of baseline (DD11), orthogonal time series ϕ_i's (DD13) and ideal time series r_i's (DD7). β_{ijk}^{*T} is the vector of unknown fit coefficients for each voxel v_{ijk}. ϵ_{ijk} is the noise at a specific voxel v_{ijk} over time. α 's are the fit coefficient for ideal signals. β's are the fit coefficient for baseline. γ's are the fit coefficient for orthogonal time series.
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf

Figure 4.28: Another Instance Model for 3dfim+

System Context (Figure 4.29).

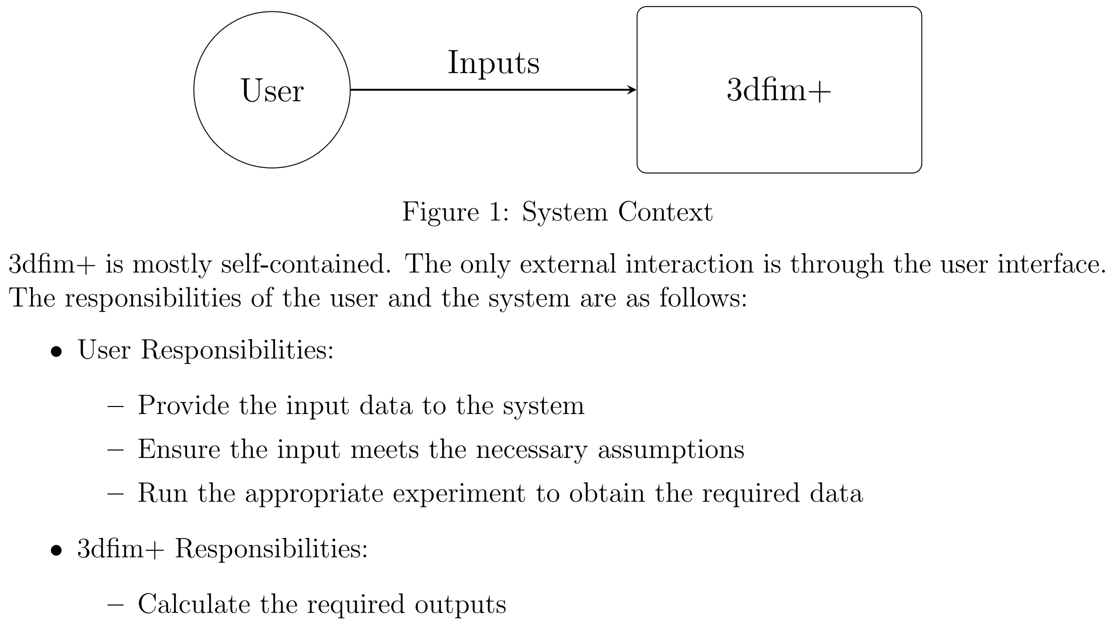

User
Inputs
3dfim+

Figure 1: System Context

3dfim+ is mostly self-contained. The only external interaction is through the user interface. The responsibilities of the user and the system are as follows:

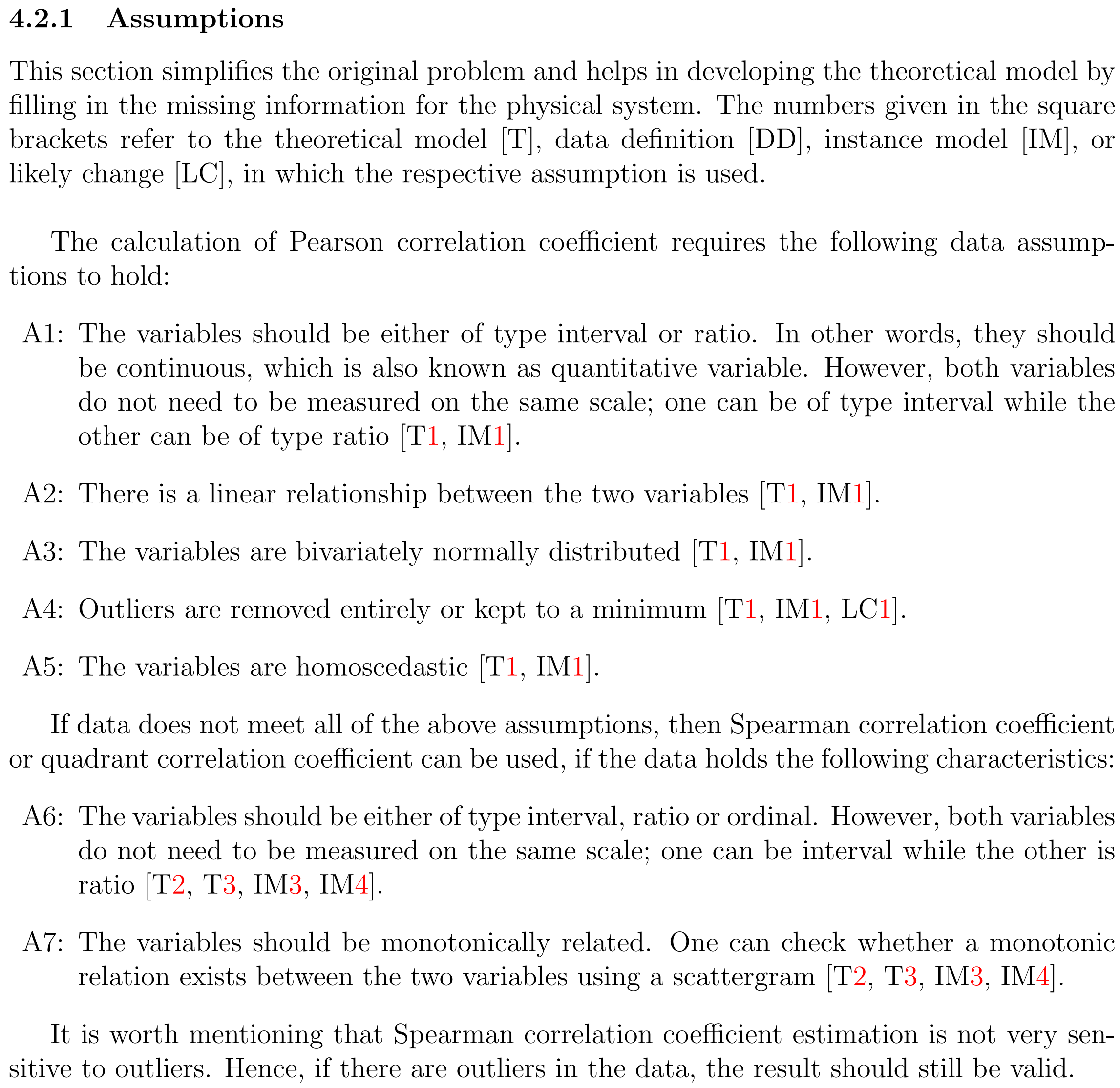
- User Responsibilities:
  - Provide the input data to the system
  - Ensure the input meets the necessary assumptions
  - Run the appropriate experiment to obtain the required data
- 3dfim+ Responsibilities:
  - Calculate the required outputs

Figure 4.29: System Context for 3dfim+

- "Required response times"; This information is not applicable to our work.

- "The intended operating environment for the software, if this is a design constraint (e.g., hardware platform, operating system)". The intended environment to run the program 3dfim+ is mentioned in Section 3.3 as System Constraints. As 3dfim+ had been already designed and implemented by the time of writing our version of SRS, the information provided in this section is for user's information and not for design purposes.

- "All ranges, limits, defaults, and specific values that the software will accept". We have mentioned the inputs that the software accepts and Section 4.2.1 as Assumptions (Figure 4.30). The Assumptions section emphasis on the importance of the input assumptions. Ideally, an accurate software should not proceed if its input does not meet the necessary assumption. Although 3dfim+ does not provide input assumptions, we have investigated and documented the input assumptions based on the mathematical formulas that are used in the program.



4.2.1 Assumptions

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [T], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

The calculation of Pearson correlation coefficient requires the following data assumptions to hold:

A1: The variables should be either of type interval or ratio. In other words, they should be continuous, which is also known as quantitative variable. However, both variables do not need to be measured on the same scale; one can be of type interval while the other can be of type ratio [T1, IM1].

A2: There is a linear relationship between the two variables [T1, IM1].

A3: The variables are bivariately normally distributed [T1, IM1].

A4: Outliers are removed entirely or kept to a minimum [T1, IM1, LC1].

A5: The variables are homoscedastic [T1, IM1].

If data does not meet all of the above assumptions, then Spearman correlation coefficient or quadrant correlation coefficient can be used, if the data holds the following characteristics:

A6: The variables should be either of type interval, ratio or ordinal. However, both variables do not need to be measured on the same scale; one can be interval while the other is ratio [T2, T3, IM3, IM4].

A7: The variables should be monotonically related. One can check whether a monotonic relation exists between the two variables using a scattergram [T2, T3, IM3, IM4].

It is worth mentioning that Spearman correlation coefficient estimation is not very sensitive to outliers. Hence, if there are outliers in the data, the result should still be valid.

Figure 4.30: Input Assumptions for 3dfim+

- "All safety related requirements, specifications, features, or functions that will be implemented in software". We indirectly focused on safety by building confidence in the quality of correctness. For medical analysis software, like 3dfim+, safety is tied to correctness, since using incorrect software would be unsafe.

Another integral part of a requirement documentation is Terminology. This section consists of the mathematical concepts and their meaning. As our case study is scientific software, it is necessary to include this section in our documentation. The reason is that terminology often has different meanings, and to avoid potential confusion, such information should be provided in this section. Terminology along with the Background section provide enough information to allow understanding of the later sections: Goal Statements, Assumptions, Theoretical Models, and Instance Models. In the Theoretical Models section, the models are presented as they would be specified in any general mathematical text book. The models are general mathematical models and not specific for the case study.

4.4 Test Case Development

To validate the implementation of 3dfim+, we have developed test cases based on the Functional Requirements we have derived and documented in our SRS. Since our case study is a scientific software, validation through testing is challenging. As mentioned earlier, such software is based on mathematical and physical formulas and success in one test case does not necessarily ensure success for another test case [13].

We developed one test case per each functional requirement, to compare their results with the results of 3dfim+. As an evidence, we explain one of our test cases. This test case checks the correctness of the Pearson Correlation Coefficient calculation, which is one of the main functionalities of 3dfim+.

We used MATLAB software to develop our test case and AFNI to visualize and get the indices of voxels. Our input consists of 180 frames of 64×64×28 images. In this test case, we decided to find the minimum and the maximum Pearson correlation coefficients. These values appear in a voxel whose activity has the highest correlation with the ideal signal (for the maximum correlation coefficient) and a voxel whose activity has the lowest correlation to the ideal signal (for the minimum correlation coefficient).

We got the same results for the maximum and the minimum Pearson correlation coefficients in both software. The maximum Pearson correlation coefficient was calculated as 0.7802 and the minimum Pearson correlation coefficient was calculated as -0.775843. Despite of achieving the same results, the locations of these results were not consistent. Using 3dfim+, the maximum correlation coefficient was found at location $(50, 27, 21)$ whereas the voxel with the highest correlation coefficient was located at $(43, 28, 22)$ in MATLAB. A difference of $(7, -1, -1)$ between these two locations is noticeable.

Moreover, this difference was also seen in the case of the minimum correlation coefficient. The voxel with the minimum correlation coefficient was found at location $(28, 27, 23)$ using 3dfim+, whereas in MATLAB, the minimum correlation coefficient belonged to the voxel at position $(21, 28, 24)$. As we can see, in this case the difference is also $(7, -1, -1)$. This experiment shows that the coordinate system and indices conventions are different in MATLAB and 3dfim+. Figure 4.31 depicts this comparison.

To avoid confusion, the coordinate system that is followed by 3dfim+ should be mentioned in the software requirements specification. We added a section for the coordinate systems used in these programs in our SRS and explained the differences between these conventions clearly.

Although most scientific software face Oracle problems [60], in our case we knew what the right answer would be. We had the outputs from 3dfim+ and we could check our results against them. In all test cases, our results match the results from 3dfim+. Hence, we can claim that we increased the confidence in the correctness of 3dfim+.

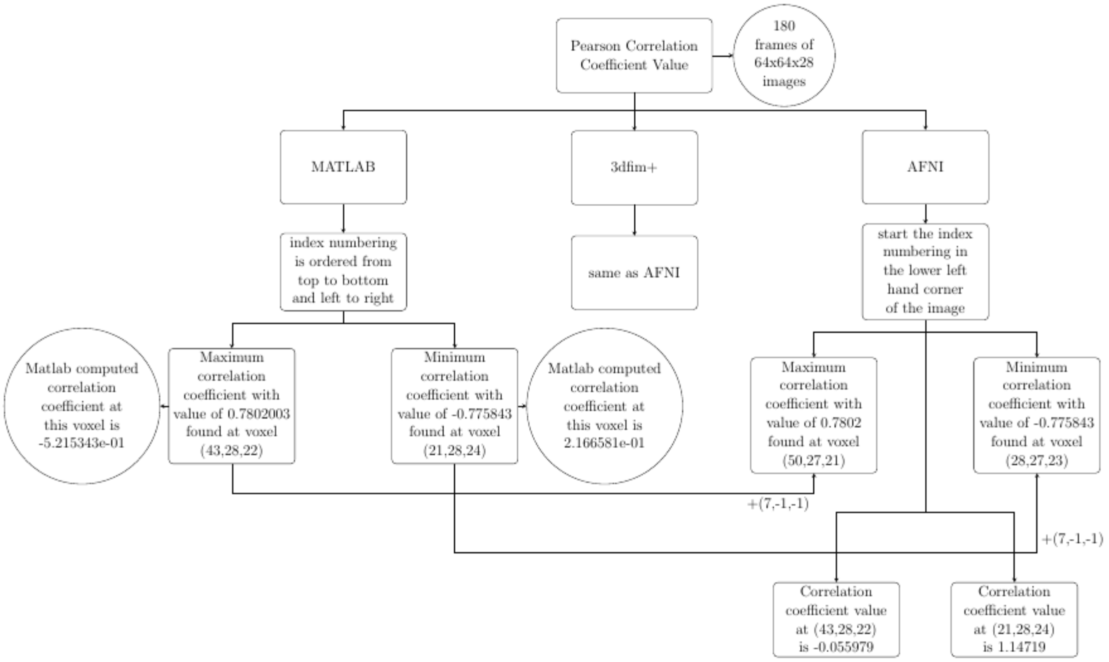

Pearson Correlation Coefficient Value
180 frames of 64x64x28 images
MATLAB
3dfim+
AFNI
index numbering is ordered from top to bottom and left to right
same as AFNI
start the index numbering in the lower left hand corner of the image
Matlab computed correlation coefficient at this voxel is -5.215343e-01
Maximum correlation coefficient with value of 0.7802003 found at voxel (43,28,22)
Minimum correlation coefficient with value of -0.775843 found at voxel (21,28,24)
Matlab computed correlation coefficient at this voxel is 2.166581e-01
Maximum correlation coefficient with value of 0.7802 found at voxel (50,27,21)
Minimum correlation coefficient with value of -0.775843 found at voxel (28,27,23)
+(7,-1,-1)
+(7,-1,-1)
Correlation coefficient value at (43,28,22) is -0.055979
Correlation coefficient value at (21,28,24) is 1.14719

Figure 4.31: Comparison of the results we got from different software for the minimum and maximum value of Pearson Correlation Coefficient between the brain signals and the ideal signal

4.5 Expert Review

Another evidence in our assurance case is the domain experts review. Review of SRS is important to reach a common platform between software engineers and scientists. Properly reviewed SRS acts as an agreement between the scientists and the software engineers regarding the deliverables of the project.

Domain experts review appears in our assurance case as “Domain experts/customers approve the «quality» of the documentation of the requirements”. To ensure our SRS is of high quality, a task-based inspection approach has been done for the review. Review techniques consist of task-based inspection using Github issue tracking. The task-based approach is based on Kelly’s work [54].

To initiate the review process, we have assigned a set of tasks which need to be completed. Every task is framed as a question in a specific section of the SRS which needs to be answered after reading the corresponding section in the SRS document. We used Github issue tracking for our discussion. We selected some of the tasks and posted them as issues on Github. A domain expert went through all the tasks and gave us suggestions. According to his suggestions, we made the necessary modifications. This review increased the confidence in our SRS. The set of tasks is given in Appendix C.

Chapter 5

Conclusion and Future Works

In this chapter, we provide a summary of the thesis as well as the future work.

5.1 Thesis Summary

This thesis has provided insight into constructing assurance cases for scientific software. The principal objective of our assurance case was to present a clear argument that the program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment. The Goal Structuring Notation (GSN), presented within this paper, has been developed to provide a clear and structured approach for developing and presenting our argument. The assurance case we developed is the first contribution to employing this approach for scientific software. This assurance case can be used as a guideline for the future assurance case developments in this domain. For example, our approach to using a generic argument for the quality-related claims, which improves reusability, can be used in other assurance cases.

An assurance case presents an overall picture of the system. It puts the evidence such as testing and documentation together in a coherent way. Hence, it facilitates communicating among the people engaged in a project. It also enables external parties to judge about the system.

Moreover, we developed an SRS for 3dfim+ which has been significantly improved in comparison to the existing documentation. For instance, the documentation provided by the developer of 3dfim+ does not mention the input assumptions Nor does it mention the coordinate system 3dfim+ uses. This particular omission can cause considerable confusion, especially when somebody wants to compare the results of the 3dfim+ with the results of another software for the same experiment. Mathematical formulas and concepts in the original documentation are too abstract and require further explanation. In our SRS, we addressed these problems. However, we can not quantify the risk and cost associated with mistakes/changes in the software.

Our SRS delivers the information required to understand how 3dfim+ works, what assumptions should an input meet, and what mathematical formulas are used to implement it. The SRS as evidence appears many times in our assurance case and its correctness has been verified by a domain expert.

Further evidence to our assurance case is the test case development. By developing test cases and the matching results between 3dfim+ and our test cases, we improved confidence in the correctness of 3dfim+.

The main problems we encountered throughout our work can be summarized as follows:

- Using natural language for building assurance cases instead of formal proofs made the assurance case development challenging.
- Distinguishing between some terminologies used in assurance cases, such as assumption and context, was challenging.

- Tool support for building assurance cases was limited and made it difficult for us to develop our assurance case. Moreover, with the current tool support, assurance cases are hard to maintain.
- There was no way to mathematically prove the relationship between the elements in the assurance case.
- There was no way to ensure that the evidence we would provide is sufficient. In many cases, the evidence was compelling, but the reader had to work hard to confirm that it is so.

Some of the open questions related to the use of the assurance cases are:

- To what extent does this approach improve the software reliability?
- How does one measure the effectiveness of developing assurance cases for software?
- How is the assurance case itself evaluated?

5.2 Future Works

Based on our work and our review of past and current work on assurance cases, we can identify a number of directions for the future development of assurance cases. We believe the outlined problems and open questions can be tackled effectively if more research is done on the following fronts:

- Assurance Case Methodology Enhancement: As mentioned earlier in this chapter, not all the definitions given for the assurance case terminology are distinguishable. Conducting more research on this area and having better and more explicit definitions, along with more examples, can help with developing assurance cases.

- 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; e.g. to hide the details and only show the title to improve readability, and release the details upon a click on the goals. No tool supports implementation for links between the elements to replicate changes; if we make a change in one element, other elements attached directly or indirectly to it remain unchanged. Improving tools in this regard will make assurance cases easier to navigate and readable. It also improves maintainability.

- 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 from about assurance cases.

- Developing Argument Templates and Linkage to Standards such as IEEE 830-1993: Assurance cases are costly to develop, so we should seek more efficient means of construction. Being able to develop templates for specific categories of systems or software will be extremely valuable. Especially if these patterns are approved by their stakeholders and corresponding responsible organizations, or they are linked to a standard, this can assist future development of assurance cases and reduce the risk and time of the development.

- Extension to other Areas: Assurance cases can be used in other areas that require assurance. Nowadays, assurance cases have been used to a limited degree, especially in the US and Canada, but may well be used more widely in future.

- Adding Formality to Assurance Cases: The means of expressing confidence in assurance cases and the top-level claims need more formality and rigor. Adding formality justifies the claim decomposition and the credibility of the evidence.

- Quantifying the Risk Associated with Mistakes/Changes: Quantifying the cost and risk of making a change or finding a mistake in 3dfim+ could be explored in future work.

Bibliography

[1] D. Edwards, “Devops: Shift left with continuous testing by using automation and virtualization.” [Online]. Available: https://www.ibm.com/devops/method/experience/deliver/dibbe_edwards_devops_shift_left/

[2] J. Spriggs, *GSN - The Goal Structuring Notation, A Structured Approach to Presenting Arguments*, ser. NASA contractor report. Hayling Island, UK: Springer, 2012.

[3] “File:AFNI screenshot.” [Online]. Available: https://commons.wikimedia.org/wiki/File:AFNI_screenshot.png

[4] D. Rinehart, J. Knight, and J. Rowanhill, *Current practices in constructing and evaluating assurance cases with applications to aviation*, ser. NASA contractor report, 2015. [Online]. Available: https://books.google.ca/books?id=AVl7nQAACAAJ

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

[6] R. Calinescu, S. Gerasimou, and I. Habli, “Engineering trustworthy self-adaptive software.” [Online]. Available: https://www-users.cs.york.ac.uk/simos/ENTRUST/

[7] D. Gade and S. Deshpande, “Assurance driven software design using assurance case based approach,” *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 3, no. 2320-9801, October 2015.

[8] GSN Community, “GSN community standard version 1,” November 2011. [Online]. Available: http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf

[9] Adelard, “Claims, arguments and evidence (CAE).” [Online]. Available: https://www.adelard.com/asce/choosing-asce/cae.html

[10] K. Preuschoff, “Physiological basis of the BOLD signal.” [Online]. Available: http://www.fil.ion.ucl.ac.uk/spm/course/slides10-zurich/Kerstin_BOLD.pdf

[11] A. Wassyng, N. K. Singh, M. Geven, N. Proscia, H. Wang, M. Lawford, and T. Maibaum, “Can product-specific assurance case templates be used as medical device standards?” *IEEE Design & Test*, vol. 32, no. 5, pp. 45–55, 2015. [Online]. Available: https://doi.org/10.1109/MDAT.2015.2462720

[12] ISO/IEC, “Systems and software engineering – systems and software assurance – part 2: Assurance case,” Feb 2011. [Online]. Available: https://www.iso.org/standard/52926.html

[13] W. S. Smith, “Systematic development of requirements documentation for general purpose scientific computing software,” in *Proceedings of the 14th IEEE International Requirements Engineering Conference, RE 2006*,

Minneapolis / St. Paul, Minnesota, 2006, pp. 209–218. [Online]. Available: http://www.ifi.unizh.ch/req/events/RE06/

[14] "Software carpentry," 2006. [Online]. Available: http://www.scipy.org

[15] "CMS offline software." [Online]. Available: https://github.com/cms-sw/cmssw

[16] C. Pernet and T. Nichols, "Has a software bug really called decades of brain imaging research into question?" September 2016. [Online]. Available: https://www.theguardian.com/science/head-quarters/2016/sep/30/has-a-software-bug-really-called-decades-of-brain-imaging-research-into-question

[17] A. Eklunda, T. Nichols, and H. Knutssona, "A methodology for safety case development," *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, vol. 113, no. 28, pp. 7900–7905, 2016.

[18] "General principles of software validation; final guidance for industry and FDA staff," 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, Tech. Rep., January 2002.

[19] C. B. Weinstock, "Assurance cases," 2008. [Online]. Available: http://www.seas.upenn.edu/~lee/10cis541/lecs/Assurance.Cases.Tutorial-6.pdf

[20] J. Rushby, "The interpretation and evaluation of assurance cases," Computer Science Laboratory, SRI International, Menlo Park, CA, Tech. Rep. SRI-CSL-15-01, July 2015, available at http://www.csl.sri.com/users/rushby/papers/sri-csl-15-1-assurance-cases.pdf.

[21] P. Bishop and R. Bloofield, “A methodology for safety case development,” *Industrial Perspectives of Safety-critical Systems: Proceedings of the Sixth Safety-critical Systems Symposium, Birmingham 1998*, 1989.

[22] T. Kelly, “Arguing safety – a systematic approach to safety case management,” Ph.D. dissertation, York University, Department of Computer Science Report YCST, May 1999.

[23] C. B.Weinstock, H. F.Lipson, and J. Goodenough, “Arguing security – creating security assurance cases,” Software Engineering Institute Carnegie Mellon University, 4500 Fifth Avenue, Pittsburgh, PA, Tech. Rep., January 2007. [Online]. Available: http://resources.sei.cmu.edu/asset_files/WhitePaper/2013_019_001_293637.pdf

[24] Office for Nuclear Regulation, “A guide to nuclear regulation in the UK,” 2016. [Online]. Available: http://www.onr.org.uk/documents/a-guide-to-nuclear-regulation-in-the-uk.pdf

[25] European Nuclear Society, “Wigner energy.” [Online]. Available: https://www.euronuclear.org/info/encyclopedia/w/wigner-energy.htm

[26] M. Ragheb, “Windscale accident,” May 2015. [Online]. Available: http://mragheb.com/NPRE%20457%20CSE%20462%20Safety%20Analysis%20of%20Nuclear%20Reactor%20Systems/Windscale%20%20Accident.pdf

[27] I. Davis, “The British nuclear industry: Status and prospects,” January 2009. [Online]. Available: https://www.cigionline.org/sites/default/files/british_nuclear_industry.pdf

[28] Australia Department of Primary Industries and Energy, *Report of the consultative committee on safety in the offshore petroleum industry.* Department of Primary Industries and Energy, 1991.

[29] L. Cullen, *The public inquiry into the Piper Alpha disaster*, 1st ed. Health and Safety Executive, 1996.

[30] P. Wilkinson, "Safety cases: Success or failure?" Tech. Rep., May 2002. [Online]. Available: http://rickduley.webs.com/doc/SafetyCases-SuccessOrFailure.pdf

[31] Health and Safety Executive, *A guide to the Offshore Installations (Safety Case) Regulations 2005*, 3rd ed. Health and Safety Executive, 2005.

[32] C. Edwards, *Railway Safety Cases.* London: Springer London, 1997, pp. 317–322. [Online]. Available: http://dx.doi.org/10.1007/978-1-4471-0921-1_18

[33] Health and Safety Executive, "Evaluation of the railways (safety case) regulations," Tech. Rep., 2004. [Online]. Available: http://www.hse.gov.uk/research/rrpdf/rr192.pdf

[34] UK Ministry of Defence, Ed., *JSP 430 - Ship Safety Management System Handbook.* Ministry of Defence, January 1996.

[35] ——, *00-55 Requirements of Safety Related Software in Defence Equipment.* Ministry of Defence, August 1997.

[36] S. Eagles, "Safety assurance cases: What the medical device industry is doing." [Online]. Available: https://www.umsec.umn.edu/sites/www.umsec.umn.edu/files/SSS%20what%20the%20medical%20device%20industry%20is%20doing.pdf

[37] C. Weinstock, "Assurance cases and confidence," August 2013. [Online]. Available: https://insights.sei.cmu.edu/sei_blog/2013/08/assurance-cases-and-confidence.html

[38] T. Kelly, "Evidence based certification: The safety case approach," 2008. [Online]. Available: https://www.umsec.umn.edu/sites/www.umsec.umn.edu/files/TimKelly.pdf

[39] UK Ministry of Defence, Ed., *00-56 Safety Management Requirements for Defence Systems.* Ministry of Defence, December 1996.

[40] S. Wilson, T. Kelly, and J. McDermid, *Safety Case Development: Current Practice, Future Prospects.* London: Springer London, 1997, pp. 135–156. [Online]. Available: http://dx.doi.org/10.1007/978-1-4471-0921-1_6

[41] T. Nichols, "Bibliometrics of cluster inference," July 2016. [Online]. Available: http://blogs.warwick.ac.uk/nichols/entry/bibliometrics_of_cluster/

[42] E. Kandel and J. Schwartz, *Principles of Neural Science*, 5th ed. McGraw-Hill Education / Medical, 2012.

[43] S. Clare, "Functional MRI : Methods and applications," Ph.D. dissertation, University of Nottingham, October 1997.

[44] S. H. Faro and F. B.Mohamed, *BOLD fMRI: A Guide to Functional Imaging for Neuroscientists.* Springer, July 2013.

[45] K. Uludag, K. Ugurbil, and L. Berliner, *fMRI: From Nuclear Spins to Brain Functions.* Springer, September 2016.

[46] C. Mulert and L. Lemieux, *EEG - fMRI: Physiological Basis, Technique, and Applications.* Springer Science & Business Media, October 2009.

[47] "fMRI brain scans may diagnose mental illness," July 2013. [Online]. Available: https://www.promises.com/articles/mental-health/fmri-brain-scans-may-diagnose-mental-illness

[48] AFNI, "Intro." [Online]. Available: https://afni.nimh.nih.gov/

[49] R. Bloomfield and P. Bishop, "Safety and assurance cases: Past, present and possible future," Tech. Rep., 2010. [Online]. Available: https://www.researchgate.net/publication/225907154_Safety_and_Assurance_Cases_Past_Present_and_Possible_Future_-_an_Adelard_Perspective

[50] B. Douglas Ward, ""AFNI program: 3dfim+"." [Online]. Available: https://afni.nimh.nih.gov/pub/dist/doc/program_help/3dfim+.html

[51] Health Canada, "Software regulated as a medical device." [Online]. Available: http://www.hc-sc.gc.ca/dhp-mps/md-im/activit/announce-annonce/md_qa_software_im_qr_logicels-eng.php#q1

[52] D. Spanou, "Software as a medical device (SaMD): Key definitions," International Medical Device Regulators Forum, Tech. Rep., December 2013.

[53] MDDI Staff, "Protecting software as a medical device," November 2016. [Online]. Available: http://www.mddionline.com/blog/devicetalk/protecting-software-medical-device-11-18-16

[54] T. Kelly, "A systematic approach to safety case management," *SAE International*, no. 04AE-149, 2003.

[55] "Astah." [Online]. Available: http://astah.net/

[56] F. J. Buckley, A. Davis, and J. Horch, “IEEE recommended practice for software requirements specifications,” The institute of Electrical and Electronics Engineers, Inc., New Yors, USA, Tech. Rep., December 1993.

[57] D. Zowghi and V. Gervasi, “The three Cs of requirements: Consistency, completeness, and correctness,” Tech. Rep.

[58] “Writing unambiguous requirements,” June 2006. [Online]. Available: http://tynerblain.com/blog/2006/06/12/writing-unambiguous-requirements/

[59] L. Northrop, “Achieving product qualities through software architecture practices,” 2004. [Online]. Available: http://www.sei.cmu.edu/architecture/cseet04.pdf

[60] J. Carver, N. Hong, and G. Thiruvathukal, *Software Engineering for Science*, ser. Chapman & Hall/CRC Computational Science. CRC Press, 2016. [Online]. Available: https://books.google.ca/books?id=xSgNDgAAQBAJ

[61] T. Kelly and R. Weaver, “The goal structuring notation – a safety argument notation,” in *Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases*, 2004.

[62] R. Chuse, *Pressure vessels: the ASME code simplified*, 7th ed. McGraw-Hill, 1993.

[63] Nuffield Department of Clinical Neurosciences, University of Oxford Medical Science Division, “Introduction to fMRI.” [Online]. Available: https://www.ndcn.ox.ac.uk/divisions/fmrib/what-is-fmri/introduction-to-fmria

[64] FDA, “Glossary of computer system software development terminology (8/95),” November 2014. [Online]. Available: http://www.mddionline.com/blog/devicetalk/protecting-software-medical-device-11-18-16

Appendix A

Software Requirements Specification for 3dfim+

Software Requirements Specification for 3dfim+

Mojdeh Sayari Nejad and Spencer Smith

June 23, 2017

Contents

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

For basic units in SI (Système International d'Unités) the symbol is given in the table below followed by a description of the unit with the SI name.

symbol	**unit**	**SI**
s	time	second

Table 1: Table of Units

3dfim+ calculates cross-correlation of two sequences of data. Correlation coefficients are not influenced by the units and the two sequences of data can be measured in different units. Indeed, the calculations for correlation coefficients were designed such that the units of measurement do not affect the calculation. As a result, we do not provide units for them.

1.2 Table of Notations

Through this document, some notations are used to define mathematical expressions. These notations are given below in table 2 followed by a description. Some of the notations are chosen from [1].

1.3 Table of Symbols

Table 3 summarizes the symbols used in this document. The symbols are listed in alphabetical order.

symbol	**type**	**description**

a	$\mathbb{R}$	variable
A	$\mathbb{R}^n$	sample dataset of size n
$Average$	$\mathbb{R}$	average quantity for fMRI dataset
b	$\mathbb{R}$	variable
B	$\mathbb{R}^n$	sample dataset of size n
base	$\mathbb{R}^n$	baseline signal
$Baseline$	$\mathbb{R}$	baseline quantity for fMRI dataset
$cval$	$\mathbb{R}$	a threshold variable
d	$\mathbb{N}^+$	sample size
f	$\mathbb{N}$	number of frames
k	$\mathbb{N}$	index of best ideal signal
M	$\mathbb{R}^{m\times n}$	data model consisting of baseline, orthogonal and ideal time series
MSE	$\mathbb{R}$	mean square error
p	$\mathbb{R}$	threshold for voxel's intensity
$pnum$	$\mathbb{W}$	degree of the polynomial in the baseline model
r	$\mathbb{R}^n$	ideal signal
r_k	$\mathbb{R}^n$	best ideal signal
s	$\mathbb{R}$	sample variance
sb	$\mathbb{R}^{m\times n\times p}$	sub-brick
slc	$\mathbb{R}^{m\times n}$	slice
SSE	$\mathbb{R}$	sum of squared errors
t	$\mathbb{R}$	time
$Topline$	$\mathbb{R}$	topline quantity for fMRI dataset
v	$\mathbb{R}$	voxel
X	$\mathbb{R}^{m\times n\times p\times q}$	3d+time dataset
α	$\mathbb{R}$	fit coefficient for ideal signal
β	$\mathbb{R}$	fit coefficient for baseline
β^*	$\mathbb{R}^n$	vector of fit coefficients
ϵ	$\mathbb{R}^n$	noise vector
γ	$\mathbb{R}$	fit coefficient for orthogonal time series
σ	$\mathbb{R}$	sample standard deviation
σ_r	$\mathbb{R}$	standard deviation of the residuals
ρ	$\mathbb{R}$	Pearson correlation coefficient
ρ_s	$\mathbb{R}$	Spearman correlation coefficient
ρ_q	$\mathbb{R}$	quadrant correlation coefficient

ϕ	$\mathbb{R}^n$	orthogonal time series

Table 3: Table of Symbols

1.4 Abbreviations and Acronyms

Table 4 contains the abbreviations and acronyms used in this document.

symbol	**description**
2D	2-Dimensional
3D	3-Dimensional
3dfim+	3-Dimensional Functional Intensity Map+
4D	4-Dimensional
A	Assumption
AFNI	Analysis of Functional NeuroImages
DD	Data Definition
DICOM	Digital Imaging and Communications in Medicine
fMRI	functional Magnetic Resonance Imaging
GS	Goal Statement
IM	Instance Model
LC	Likely Change
LPI	Left-Posterior-Inferior
LPS	Left-Posterior-Superior
MRI	Functional magnetic resonance imaging
NIfTI	Neuroimaging Informatics Technology Initiative
R	Requirement
RAI	Right-Anterior-Inferior
RAS	Right-Anterior-Superior
SRS	Software Requirements Specification
T	Theoretical Model
WCS	World Coordinate System

Table 4: Abbreviations and Acronyms

symbol	Description
$\bar{\ }$	over bar indicating arithmetic mean
$\mathbb{N}$	set of natural numbers
$\mathbb{N}^n$	set of natural vectors of size n
$\mathbb{N}^{m\times n}$	set of natural 2D matrices of size $m \times n$
$\mathbb{R}$	set of real numbers
$\mathbb{R}^n$	sequence of real numbers (set of real vectors) of size n
$\mathbb{R}^{m\times n\times p}$	set of 3D real matrices of size $m \times n \times p$
$\mathbb{R}^{m\times n\times p\times q}$	sequence of length of q of 3D real matrices of size $m \times n \times p$
$\mathbb{W}$	set of whole numbers
a_i	i^{th} entry of a matrix
a_{ij}	entry (i, j) of a 2D matrix
a_{ijk}	entry (i, j, k) of a 3D matrix
a_{ijkl}	entry (i, j, k) of a 3D matrix in a sequence of 3D matrices at time l
A^T	transpose of a matrix: $A^T_{ij} = A_{ji}$
rank(a_{ij},A)	rank of element (i, j) in a 2D matrix A

Table 2: Table of Notations

2 Introduction

This document provides an overview of the Software Requirements Specification (SRS) for the program 3dfim+ [2]. 3dfim+ mainly calculates the cross-correlation of an ideal reference signal versus the measured fMRI time series for each voxel. The current section explains the purpose of this document, the scope of the software, the organization of the document and the characteristics of the intended readers.

2.1 Purpose of Document

The main purpose of this document is to provide sufficient information to understand what 3dfim+ does. The goals and theoretical models used in the 3dfim+ implementation are provided, with an emphasis on explicitly identifying assumptions and unambiguous definitions.

2.2 Scope of Requirements

The responsibilities of the user and the 3dfim+ are as follows:

- User Responsibilities: Users are responsible to provide appropriate inputs to the program and ensure that the inputs meet the assumptions mentioned in 4.2.1.

- 3dfim+ Responsibilities: Upon receiving appropriate inputs, the program is intended to compute the cross-correlation of each voxel's activity over time with a user specified reference time series. Other outputs are mentioned in R6 to R13.

2.3 Organization of Document

The organization of this document follows the template for an SRS for scientific computing software proposed by [3] and [4]. The presentation follows the standard pattern of presenting goals, theories, definitions and assumptions. The goal statements are refined to the theoretical models, and theoretical models to the instance models. For readers that would like a more bottom-up approach, they can start reading the instance models in Section 4.2.4 and trace back to find any additional information they require.

3 General System Description

This section provides general information about the system, identifies the interfaces between the system and its environment, and describes the user characteristics and the system constraints.

3.1 System Context

Figure 1 shows the system context. A circle represents an external entity outside the software, the user in this case. A rectangle represents the software system itself. Arrows are used to show the data flow between the system and its environment.

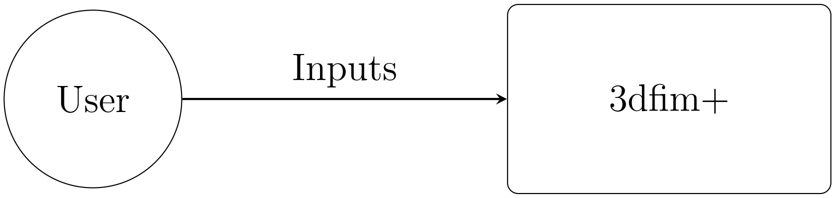


Figure 1: System Context

3dfim+ is mostly self-contained. The only external interaction is through the user interface. The responsibilities of the user and the system are as follows:

- User Responsibilities:
  - Provide the input data to the system
  - Ensure the input meets the necessary assumptions
  - Run the appropriate experiment to obtain the required data

- 3dfim+ Responsibilities:
  - Calculate the required outputs

3.2 User Characteristics

The end user of 3dfim+ should have an understanding of undergraduate Level 1 Linear Algebra.

3.3 System Constraints

Intended environment to run the program on are the Unix+X11+Motif systems [5].

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

4.1 Problem Description

The main purpose of 3dfim+ is to calculate the cross-correlation between voxels and a reference signal over time. Other outputs of the program are mentioned in R6 to R13.

4.1.1 Background

This section provides information necessary to understand the correlation.

4.1.1.1 Basics of Correlation

Correlation is used to measure strength of association between two variables. Correlation coefficients are standardized; they vary between +1 and -1 and describe strength and direction of the association.

If a variable is correlated to itself, the resulting value is called autocorrelation or serial correlation. In this case the variable is being compared to itself with a time shift. Otherwise, if we have two different variables, the output is called cross-correlation.

If the value of the correlation coefficient is near to +1 or -1, there is a strong degree of association between the two variables. A value near to zero represents a weak correlation between the variables.

4.1.1.2 Visual Representation of Correlation

To study the possible correlation between two variables, we can produce a graph called scatter diagram or scattergram. Axes represent values of two variables, and corresponding values are shown by a dot. Figure 2 shows a sample of a scattergram of two sample variables a and b.

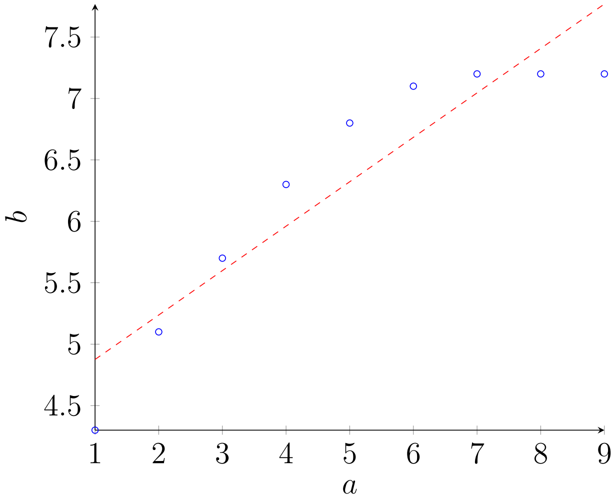


Figure 2: A sample of a scattergram and regression line of two variables a and b

The red dashed line in the graph shows linear regression, which represents the best-fit straight line through the points. The nearer the points are to this line, the stronger the association between the two variables is.

4.1.1.3 Different Types of Correlation

We can categorize correlation based on the nature of inputs and the relationship between them as follows:

- Positive and Negative Correlation: Positive correlation occurs when two variables change in the same direction. In other words, both variables either increase or decrease. A sample scattergram of a positive correlation is shown in Figure 3 .

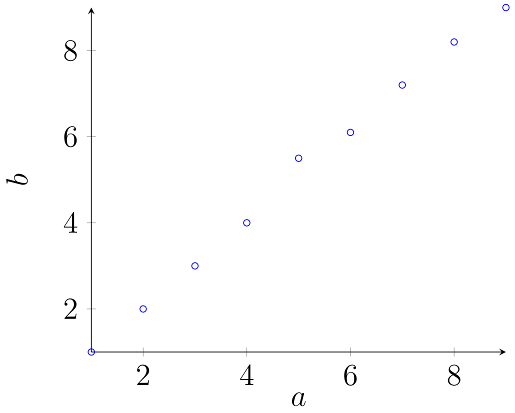


Figure 3: A sample positive correlation

There is a negative correlation between variables if one variable increases while the other decreases. In other words, two variables change in the opposite directions. A sample scattergram showing a negative correlation is shown in Figure 4.

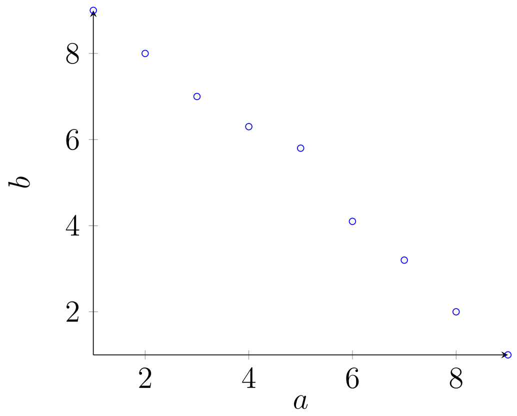


Figure 4: A sample negative correlation

- Linear versus Non-Linear Correlation: If the ratio between two variables remains the same, there exists a linear correlation between them. In this case, there is a straight line relationship between those variables. If the ratio does not remain constant over time, the correlation is called non-linear. When a relation is non-existent or random, correlation coefficients are near zero.

- Parametric versus Non-parametric Correlation: Parametric correlation uses data information such as mean and standard deviation while non-parametric correlation does not need such information. So if the data type is interval or ratio, we use a parametric estimation such as Pearson correlation coefficient and if the level of measurement is either ordinal or nominal, we use a non-parametric estimation, such as Spearman correlation coefficient. Moreover, to use a parametric correlation data distribution should be approximately normal. It is important to choose an appropriate correlation to get valid results.

- Pearson Correlation Coefficient: Pearson correlation is the most commonly used type of correlations. This correlation, signified by ρ, is a linear correlation used in statistics to measure the degree of linear relationship between paired data.

- Spearman Correlation Coefficient: Spearman correlation coefficient, denoted by ρ_S, is a statistical measure of the strength of a monotonic relationship between the observation ranks. We can consider this correlation as a non-parametric version of the Pearson correlation that measures the strength of association between two ranked variables. This rank-based estimator is highly efficient and is robust to outliers [6].

- Quadrant Correlation Coefficient: As we mentioned previously, an estimation procedure can be endowed with robustness properties by using a rank statistics [7]. Quadrant correlation coefficient is a non-parametric estimator that computes the correlation coefficient between the sign of deviations from medians using ranked data.

4.1.1.4 Effect Size

The correlation coefficient representing the strength of relationship between two variables is referred to as the effect size. We can use either Cohen's (1998) [8] or Evans (1996) [9] standard shown in Tables 5 and 6 respectively, to interpret the effect size.

Table 5: Cohen's effect size

Strength of Association	Positive Coefficient	Negative Coefficient
Small	0.1 to 0.29	-0.1 to -0.29
Medium	0.3 to 0.49	-0.3 to -0.49
Large	0.5 to 1	-0.5 to -1

Table 6: Evans' effect size

Strength of Association	Positive Coefficient	Negative Coefficient
Very Weak	0.00 to 0.19	0.00 to -0.19
Weak	0.20 to 0.39	-0.20 to -0.39
Moderate	0.40 to 0.59	-0.40 to -0.59
Strong	0.60 to 0.79	-0.60 to -0.79
Very Strong	0.8 to 1	-0.8 to -1

Note that correlation coefficient of 0 does not imply that there is no relationship between the variables. For example, a value of 0 for a Pearson correlation coefficient only indicates that there is no linear association between the variables. However, other relationships, such as quadratic relationship, can exist between them.

Also note that a coefficient of +1 means that there is no variation between the data points and the line of best fit.

4.1.2 Terminology Definition

This subsection provides definitions for the terms that are used in the subsequent sections with the purpose of reducing ambiguity and making it easier to understand the requirements.

- Arithmetic Mean: The arithmetic mean of a set of data, also referred to as mean or sample mean, is computed as the sum of all the values in the dataset divided by the count of all data points in the dataset.

- Variance: Variance is a measure of how far the numbers in a set are spread out. It measures the distance between each number in the set from the mean of the numbers in the set. It is calculated as the average of the squared differences between each number in the set and the mean.

- Standard Deviation: Standard deviation is a measure that is used to quantify the amount of variation of a set of data values. It is computed as the square root of the variance. Standard deviation is used when a sample of data from an entire population is available.

- Nominal Data: Nominal data also known as categorical data is a type of data that is categorized but there is no order between the categories.

- Ranked Data: Ranked data is a set of variables that for any two of them, one is ranked either equal to or lower than or higher than the other one. The relationship between these variables is called ranking. More information is provided in DD4.

- Ordinal Data: Ordinal type is when there is a clear ordering of variables, but the difference between values is inconsistent. Rating between 0 and 10 is an example of this kind of variables. The difference between rate 2 and 4 is not necessarily the same as the difference between rate 6 and 8.

- Interval Data: For an interval variable, order is important as for an ordinal variable. In addition, the interval between the values are equally spaced. For example, temperature is considered as an interval variable. The difference between 50 degrees and 60 degrees is the same as the difference between 70 degrees and 80 degrees.

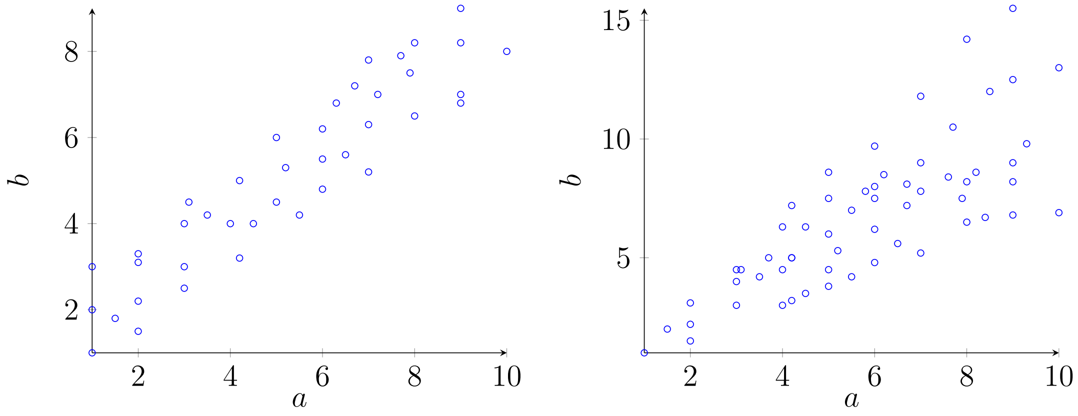
- Ratio Data: A ratio variable has all the properties of an interval variable. Moreover, when the value of the variable is equal to 0, it means that there is none of that variable. For example, a value of 0 for a variable such as height means we have no height. Note that ratio data can also be considered as an interval data and an ordinal data. In other words, ratio data $\subset$ interval data $\subset$ ordinal data.

  The definition of nominal, ordinal, interval, and ratio variables, known as level of measurement, was first developed by Stevens (1946) [10]. The level of measurement determines which statistical measures are appropriate for the specific need. Note that for calculating Pearson correlation coefficients, variables need to have a level of measurement at least equal to interval. The reason is that we need to compute mean of variables for Pearson correlation coefficients and computing an average is meaningful only when the intervals between values are equally spaced.

If data is ordinal, Spearman correlation coefficients or quadrant correlation coefficients are used instead.

- Homoscedasiticity: Homoscedacsiticity happens when both variables are normally distributed around the regression line. It means that the variances along the regression line remain similar while moving along the line.

  When using Pearson correlation coefficient as a measure, violation in homoscedasiticity may result in over-estimating the goodness of the fit. Figure 5 shows this characteristic.



(a) Homoscedasiticity

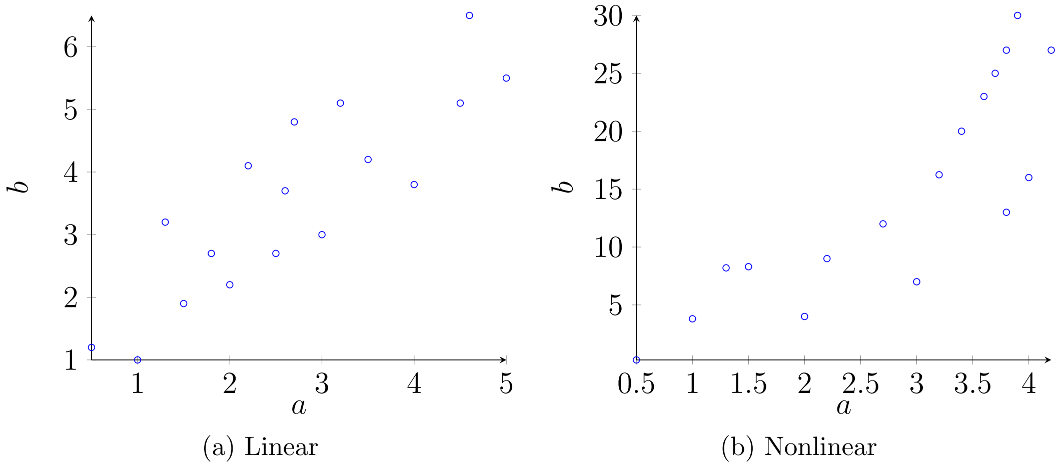
(b) Heteroscedasiticity

Figure 5

- Bivariate normal distribution: When each variable is normally distributed itself and is also normally distributed at all levels of the other variable, the distribution is bivariate normal. If this assumption is met, the only type of statistical relationship that can exist between the two variables is a linear relationship. However, if the assumption is violated, a non-linear relationship may exist. It is important to determine if a non-linear relationship exists between two variables before describing the results using Pearson correlation coefficient.

- Outlier: An outlier is a data point that does not follow the general pattern of the data and its value is extremely different from the rest of the data, such that it has a large effect on some parameters such as mean of the data and consequently on Pearson correlation coefficient and the regression line. Pearson correlation coefficient is sensitive to outliers, so if data point removal is not allowed, we should use a non-parametric estimation such as Spearman correlation coefficient.

- Linearity: Linearity is a mathematical relationship between two variables that can be represented as a straight line. If the relationship between the variables is non-linear, Pearson correlation coefficient is not an appropriate statistic for measuring the association. Figure 6 visualizes this relationship.

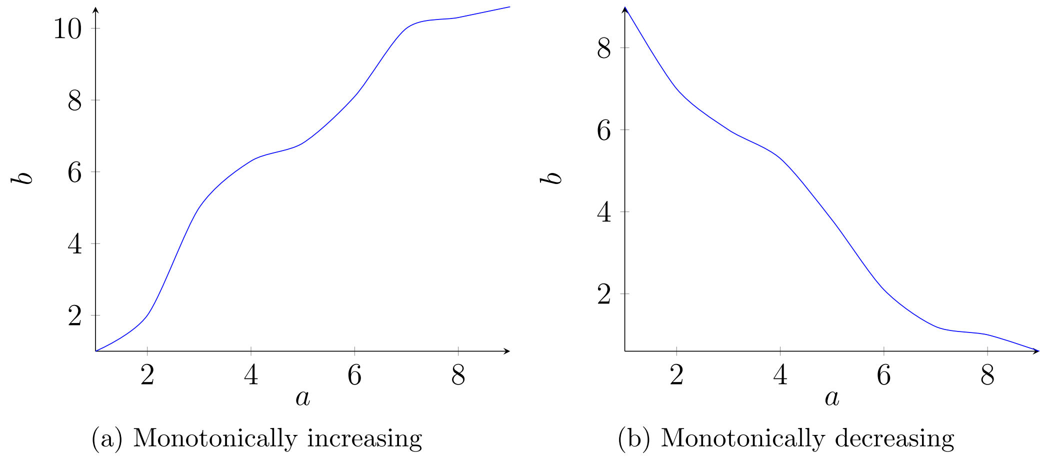


(a) Linear

(b) Nonlinear

Figure 6

- Monotonic function: Monotonic function $b(a)$ is a function where increasing in the value of a results in either always increasing or always decreasing in the value of b. Figure 7 visualizes this function.



(a) Monotonically increasing

(b) Monotonically decreasing

Figure 7

4.1.3 Coordinate Systems

While working with medical images, it is necessary to be familiar with the different coordinate systems of the medical literarure and how data (voxels' orientation) is interpreted in different medical and non-medical software. Each coordinate system uses one or more numbers (coordinates) to uniquely determine the position of a point (in the medical context, we refer to each point as a voxel). The purpose of this section is to introduce some of the coordinate systems related to the medical imaging. There are different coordinate systems to represent data. A knowledge of the following coordinate systems is needed to be able to work with the medical images.

4.1.3.1 Cartesian Coordinate System

A Cartesian coordinate system is a coordinate system that specifies each point uniquely in a 2D plane by a pair of numerical coordinates or in a 3D space by three numerical coordinates. We assume a right-hand Cartesian coordinate system throughout this document.

4.1.3.2 World Coordinate System

World Coordinate System (WCS) is a Cartesian coordinate system that describes the physical coordinates associated with a model such as a MRI scanner or a patient. While each model has its own coordinate system, without a universal coordinate system such as WCS, they cannot interact with each other. For model interaction to be possible, their coordinate systems must be transformed into the WCS. Figure 10 shows the WCS corresponding space and axes.

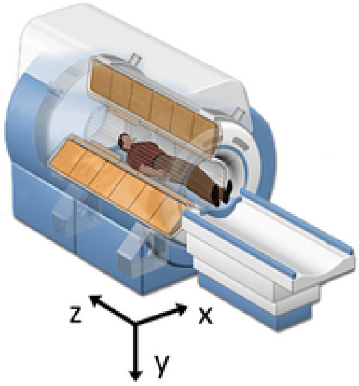


Figure 8: World Coordinate System Space and Axes [11]

4.1.3.3 Anatomical Coordinate System

Anatomical coordinate system, also known as patient coordinate system, is a right-handed 3D coordinate system which describes the standard anatomical position of a human using the following 3 orthogonal planes:

- Axial / Transverse plane: is a plane parallel to the ground that separates the body into head (superior) and tail (inferior) positions.
- Coronal / Frontal plane: is a plane perpendicular to the ground that divides the body into front (anterior) and back (posterior) positions.
- Sagittal / Median plane: is a plane that divides the body into right and left positions.

Figure 9 shows this coordinate system.

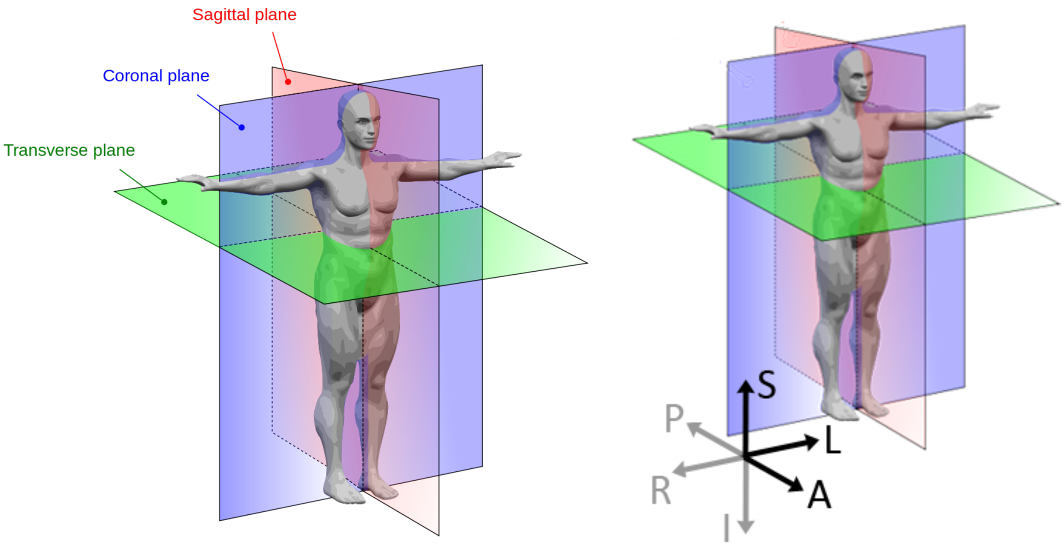


Figure 9: Anatomical Coordinate System Space and Axes [11]

Medical applications follow an anatomical coordinate system to store voxels in sequences. Depending on how the data is stored, this coordinate system can be divided into different bases. The most common ones are:

- LPS Coordinate System:

  The LPS coordinate system, also known as DICOM (patient) coordinate system, is a left-hand coordinate system used in DICOM images. In this system, voxels are ordered from left to right in a row, rows are ordered from posterior to anterior, and slices are stored from inferior to superior. In other words, it is an LPI system.
  LPS stands for Left-Posterior-Superior which indicates the directions that spatial axes are increasing.

- RAS Coordinate System:

  LPI is a right-hand coordinate system for voxel orientation. It stores 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. This system is the preferred basis for Neurological applications such as 3dfim+ and is used in NIfTI files. The increasing position order is RAS.

4.1.3.4 Image Coordinate System

To specify locations in an image we need to know to which coordinate system it is referenced. Different software may use different orders as their index convention.

- Image Coordinate System for Matlab:

  In Matlab, index numbering starts at the upper left corner. To express the position of point (x, y, z), we should consider that the x axis increases from left to right, the y axis increases to the bottom and the z axis increases backward.

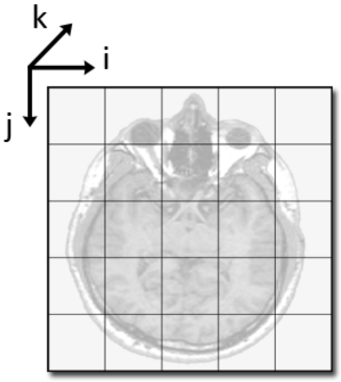


Figure 10: Image Coordinate System Space and Axes in Matlab [11]

- Image Coordinate System for AFNI:

  In AFNI, the lower left hand corner of the image is considered as the origin, which represents the position of the first voxel (0,0,0).

  If we are using different file formats and software, we need to transform their coordinate systems into WCS.

4.1.4 Physical System Description

We do not study the physical system for MRI or how the data is actually generated.

4.1.5 Goal Statements

Given an fMRI time series (DD6), one or more ideal time series (DD7) and zero or more orthogonal time series (DD13):

GS1: Estimate the Pearson correlation coefficients between the (best) ideal time series and the fMRI time series at each voxel over time.

GS2: Estimate the Spearman correlation coefficient between the (best) ideal time series and the fMRI time series at each voxel over time.

GS3: Estimate the quadrant correlation between the (best) ideal time series and the fMRI time series at each voxel over time.

GS4: In case of having multiple ideal signals, report the index number for the best ideal time series.

GS5: Calculate the percentage change in the fMRI time series due to the (best) ideal time series relative to the average for each voxel.

GS6: Calculate the percentage change in the fMRI time series due to the (best) ideal time series relative to the baseline for each voxel.

GS7: Calculate the fMRI time series baseline for each voxel.

GS8: Calculate the fMRI time series average for each voxel.

GS9: Calculate the percentage change in the fMRI time series due to the (best) ideal time series relative to the topline for each voxel.

GS10: Calculate the fMRI time series topline quantity for each voxel.

GS11: Calculate the standard deviation of the residuals at each voxel between the fMRI dataset and corresponding data estimation.

4.2 Solution Characteristics Specification

In this section, necessary information to understand the meaning of instance models, presented in subsection 4.2.4, is provided.

4.2.1 Assumptions

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [T], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

The calculation of Pearson correlation coefficient requires the following data assumptions to hold:

A1: The variables should be either of type interval or ratio. In other words, they should be continuous, which is also known as quantitative variable. However, both variables do not need to be measured on the same scale; one can be of type interval while the other can be of type ratio [T1, IM1].

A2: There is a linear relationship between the two variables [T1, IM1].

A3: The variables are bivariately normally distributed [T1, IM1].

A4: Outliers are removed entirely or kept to a minimum [T1, IM1, LC1].

A5: The variables are homoscedastic [T1, IM1].

If data does not meet all of the above assumptions, then Spearman correlation coefficient or quadrant correlation coefficient can be used, if the data holds the following characteristics:

A6: The variables should be either of type interval, ratio or ordinal. However, both variables do not need to be measured on the same scale; one can be interval while the other is ratio [T2, T3, IM3, IM4].

A7: The variables should be monotonically related. One can check whether a monotonic relation exists between the two variables using a scattergram [T2, T3, IM3, IM4].

It is worth mentioning that Spearman correlation coefficient estimation is not very sensitive to outliers. Hence, if there are outliers in the data, the result should still be valid.

4.2.2 Theoretical Models

This section focuses on the general equations and laws that 3dfim+ is based on. In this document, we considered indexing starts from 1.

Number	T1
Name	Pearson
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. The equation can be also written as: $\rho(A, B) = \frac{\sum_{i=1}^{n} a_i b_i - n\bar{a}\bar{b}}{(n-1)\sigma_{a_i}\sigma_{b_i}}$ Where σ_a and σ_b are standard deviations (DD3). Assumptions A1, A2, A3, A4 and A5 must hold when calculating this correlation.
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf http://www.statstutor.ac.uk/resources/uploaded/pearsons.pdf
Ref. By	IM1

Number	T2
Name	Spearman
Label	**Calculating Spearman Correlation Coefficient**
Equation	$\rho_s(A,B) = \frac{\sum\limits_{i=1}^{n} (\text{rank}(a_i,A) - \frac{n+1}{2})(\text{rank}(b_i,B) - \frac{n+1}{2})}{\sqrt{\sum\limits_{i=1}^{n} (\text{rank}(a_i,A) - \frac{n+1}{2})^2 (\text{rank}(b_i,B) - \frac{n+1}{2})^2}}$
Description	This formula calculates Spearman correlation coefficient ρ_s applied to two sample datasets $A : \mathbb{R}^n$ and $B : \mathbb{R}^n$ both of size n. ρ_s is the Spearman correlation coefficient between A and B. $\text{rank}(a_i, A)$ and $\text{rank}(b_i, B)$ are rank functions (DD4). This formula can also be written as: $\rho_s(A,B) = 1 - \frac{6 \sum\limits_{i=1}^{n} h_i^2}{n(n^2-1)}$ h_i is the difference between paired ranked variables: $h_i = \text{rank}(a_i, A) - \text{rank}(b_i, B)$ Note that assumptions A6 and A7 must hold while calculating this correlation.
Source	http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf
Ref. By	IM3

Number	T3
Name	Quadrant
Label	**Calculating Quadrant Correlation Coefficient**
Equation	$\rho_q(A, B) = \frac{\sum\limits_{i=1}^{n}(\text{sign}(\text{rank}(a_i,A) - \frac{n+1}{2}))(\text{sign}(\text{rank}(b_i,B) - \frac{n+1}{2}))}{\sqrt{\sum\limits_{i=1}^{n}((\text{rank}(a_i,A) - \frac{n+1}{2}))^2((\text{rank}(b_i,B) - \frac{n+1}{2}))^2}}$
Description	This formula calculates the quadrant (sign) correlation coefficient ρ_q using the rank function (DD4) and sign function (DD5) applied to two sample datasets $A : \mathbb{R}^n$ and $B : \mathbb{R}^n$ both of size n. ρ_q is the quadrant correlation coefficient between A and B. Note that assumptions A6 and A7 must hold while calculating this correlation.
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf https://books.google.ca/books?id=-058B6kg32sC&pg=PA19&lpg=PA19&dq=quadrant+correlation&source=bl&ots=diTd_d0tou&sig=vfZXlpyTf2BzVWYUAQYpZQSjiv4&hl=en&sa=X&ved=0ahUKEwi4g7DP45LSAhXpy4MKHfFPCU04ChDoAQg-MAY#v=onepage&q=quadrant%20correlation&f=false
Ref. By	IM4

Number	T4
Name	Linear Regression
Label	**Linear Regression Model**
Equation	$f(t, x) = x_1\omega_1(t) + x_2\omega_2(t) + \cdots + x_n\omega_n(t)$
Description	Regression is the task of finding the best fit for a model through a set of data points. Given data points (t_i, y_i) where $i = 1, \cdots, m$, we want to find the vector x of size n $(m > n)$ of parameters that gives the best fit to the data by the model function $f(t, x)$. The terms in the linear model $f(t, x)$ are either constant, i.e. $\omega_i(t) = 1$ or the product of a parameter x_i and a function $\omega_i(t)$. The above equation is called a linear regression equation and the fitting line that it generates is called line of best fit. If the data is linear, then the line of best fit is straight; otherwise, it is a curve. One of the common methods for estimating the linear regression is least squares method. (T8).
Source	[12]
Ref. By	T5, T8

Number	T5
Name	SSE
Label	**Sum of Squared Errors**
Equation	$SSE = \sum_{i=1}^{n}(y_i - f(t_i, x))^2$
Description	SSE is the sum of squared residuals. Here, the residual refers to the difference between the data y_i and the estimated value $f(t_i, x)$ (T4).
Source	https://en.wikipedia.org/wiki/Residual_sum_of_squares
Ref. By	T6, IM12

Number	T6
Name	MSE
Label	**Mean Squared Error**
Equation	$MSE = \frac{1}{n}(\sum_{i=1}^{n}(y_i - f(t_i, x))^2)$
Description	MSE is the mean of the SSE (T5).
Source	https://en.wikipedia.org/wiki/Mean_squared_error
Ref. By	T7, IM12

Number	T7
Name	Residuals Deviation
Label	**Standard Deviation of the Residuals**
Equation	$\sigma_r = \sqrt{MSE}$
Description	MSE is the mean squared error (T6).
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf https://brownmath.com/stat/infregr.htm
Ref. By	IM12

Number	T8
Name	Least Squares
Label	**Linear Least Squares**
Equation	$Ax \cong b$
Description	Given the best fit model $f(t,x)$ (T4) and data points (t_i, y_i), $i = 1, \cdots, m$, we want to find an estimation for x. Least squares tries to minimize the residual as follows: $$min\sum_{i=1}^{m}(y_i - f(t_i, x))^2$$ The matrix representation is $$b = Ax + \epsilon$$ Where: A is a $m \times n$ matrix with entries $a_{ij} = \omega_j(t_i)$, b is a $m \times 1$ vector where $b_i = y_i$, x is a $n \times 1$ vector of parameters, and ϵ is a $m \times 1$ vector of errors. If $m > n$, the system is overdetermined and there is no exact solution for x. Instead, our goal is to minimize some norm of the residual vector $r = b - Ax$ as a function of x : $$min\|Ax - b\|_2^2$$ If we use 2-norm as the approximation, the method is called least squares and takes the form of $Ax \cong b$. We can show that: $$\hat{x} = (A^TA)^{-1}A^Tb$$ The estimated fit is then given by: $$\hat{b} = A\hat{x} = A(A^TA)^{-1}A^Tb$$ The residual vector $\hat{\epsilon}$ is : $b - A\hat{x} = b - A(A^TA)^{-1}A^Tb$.
Source	[12]
Ref. By	T4, IM12

4.2.3 Data Definitions

This section provides the mathematical formulas of the arithmetic concepts used in this document.

Number	DD1
Name	Mean
Label	**Calculating Arithmetic Mean**
Symbol	¯
Equation	$\bar{a} = \frac{1}{d} \sum_{i=1}^{d} a_i$
Description	This formula calculates arithmetic mean, also referred as sample mean or mean for a dataset containing d values.
Source	http://mathworld.wolfram.com/ArithmeticMean.html
Ref. By	T1, IM1

Number	DD2
Name	Variance
Label	**Calculating Sample Variance**
Symbol	s^2
Equation	$s_a^2 = \frac{1}{d} \sum_{i=1}^{d} (a_i - \bar{a})^2$
Description	This formula calculates sample variance of a dataset containing d values.
Source	http://mathworld.wolfram.com/SampleVariance.html
Ref. By	DD3

Number	DD3
Name	Standard Deviation
Label	**Calculating Sample Standard Deviation**
Symbol	σ
Equation	$\sigma_a = \sqrt{s_a^2} = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(a_i - \bar{a})^2}$
Description	This formula calculates sample standard deviation, that is the square root of the sample variance (DD2) when applied to a dataset containing d values.
Source	http://mathworld.wolfram.com/StandardDeviation.html
Ref. By	T1

Number	DD4
Name	Rank
Label	**Rank Function**
Symbol	rank()
Equation	$\text{rank} : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{N}$
Description	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: $\text{rank}(a, A) : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{N}$ $\text{rank}(a, A) \equiv \text{avg}(\text{indexSet}(a, \text{sort}(A)))$ $\text{indexSet}(a, B) : \mathbb{R} \times \mathbb{R}^n \rightarrow \text{ set of } \mathbb{N}$ $\text{indexSet}(a, B) \equiv \{j : \mathbb{N} \mid j \in [1..\lvert B \rvert] \wedge B_j = a : j\}$ $\text{sort}(A) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ $\text{sort}(A) \equiv B : \mathbb{R}^n, \text{ such that}$ $\forall(a : \mathbb{R} \mid a \in A : \exists(b : \mathbb{R} \mid b \in B : b = a) \wedge \text{count}(a, A) = \text{count}(b, B)) \wedge \forall(i : \mathbb{N} \mid i \in [1..\lvert A \rvert - 1] : B_i \leq B_{i+1})$ $\text{count}(a, A) : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{N}$ $\text{count}(a, A) : +(x : \mathbb{N} \mid x \in A \wedge x = a : 1)$ $\text{avg}(C) : \text{ set of } \mathbb{N} \rightarrow \mathbb{R}$ $\text{avg}(C) \equiv +(x : \mathbb{N} \mid x \in C : x)/\lvert C \rvert$ The above equations use the Gries and Schneider notation [13, p. 143] for set building and evaluation of an operator applied over a set of values. Specifically, the expression $(*x : X \mid R : P)$ 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.
Source	https://en.wikipedia.org/wiki/Ranking
Ref. By	T2, T3

Number	DD5
Name	Sign
Label	**Sign Function**
Symbol	sign()
Equation	$\text{sign(a)} = \begin{cases} 1 & a > 0 \\ 0 & a = 0 \\ -1 & a < 0 \end{cases}$
Description	Given a variable a, the sign function returns 1 if a is positive, 0 if a is equal to zero, and -1 if a is negative.
Source	`https://en.wikipedia.org/wiki/Sign_function`
Ref. By	T3

Number	DD6
Name	3d+time
Label	**Mathematical Representation of 3d+time Dataset**
Symbol	$X : \mathbb{R}^{m \times n \times p \times q}$
Equation	-
Description	3d+time datasets are 4D datasets that have a temporal component, a time dimension that is the time intervals during scanning, collecting and concatenating datasets together. 3d+time datasets are the basic units of the fMRI.
Source	`http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf`
Ref. By	GS1, DD8, DD14, DD16, DD17, DD19, IM1, IM2, IM3, IM4, IM9, IM10, R1

Number	DD7
Name	Ideal Signal
Label	**Mathematical Representation of Ideal (Reference) Signal (Time Series)**
Symbol	$r : \mathbb{R}^n$
Equation	-
Description	Ideal signal is a waveform of choice.
Source	https://en.wikipedia.org/wiki/Square_wave
Ref. By	GS1, DD16, IM1, IM2, IM3, IM4, IM6, IM9, IM10, IM12, R1, R7, R8, R11

Number	DD8
Name	Sub-brick
Label	**Sub-brick**
Symbol	sb : $\mathbb{R}^{m \times n \times p}$
Equation	-
Description	A dataset (DD6) is comprised of one or more sub-bricks. Each sub-brick is a 3D array of numbers.
Source	https://msu.edu/~zhuda/fmri_class/labs/lab2/afni01_intro.pdf
Ref. By	DD9

Number	DD9
Name	Slice
Label	**Slice**
Symbol	slc : $\mathbb{R}^{m \times n}$
Equation	-
Description	A sub-brick (DD8) consists of slices. Each move in the Z plane is considered as one slice.
Source	https://msu.edu/~zhuda/fmri_class/labs/lab2/afni01_intro.pdf
Ref. By	DD10

Number	DD10
Name	Voxel
Label	**Voxel**
Symbol	$v : \mathbb{R}$
Equation	–
Description	A slice (DD9) consists of $n \times n$ voxels. A real number is assigned to each voxel which reports its activation significance. Figure 11 is provided for a better understanding.
Source	https://msu.edu/~zhuda/fmri_class/labs/lab2/afni01_intro.pdf
Ref. By	GS1

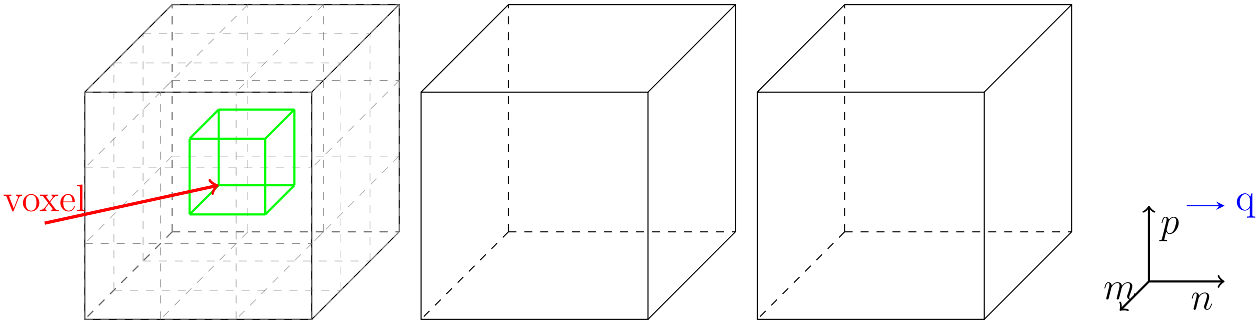


Figure 11: 3x3x3 dataset consisting of 3 sub-bricks

Number	DD11
Name	Baseline
Label	**Baseline Model**
Symbol	base: $\mathbb{R}^n \rightarrow \mathbb{R}^n$
Equation	$\text{base}(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x + a_1$
Description	The average signal level from which a signal departs and to which it returns. Baseline is modeled as a function of time. *pnum* (DD12) is used to set the degree of the polynomial in baseline model.
Source	http://dalspace.library.dal.ca/bitstream/handle/10222/37440/Rukhshinda-Jabeen-MSc-CHEM-August-2013.pdf?sequence=6
Ref. By	DD13, IM2, IM6, IM7, IM8

Number	DD12
Name	Polynomial Degree
Label	**Polynomial Degree of Baseline Model**
Symbol	$pnum$: $\mathbb{W}$
Equation	-
Description	*pnum* indicates the degree of the polynomial in the baseline model. For example, $pnum = 0$ indicates a constant baseline, $pnum = 1$ is used to model a linear baseline and $pnum = 2$ removes any quadratic trend in data and so on. The default of the 3dfim+ is $pnum = 1$.
Source	https://afni.nimh.nih.gov/pub/dist/doc/program_help/3dfim+.html
Ref. By	DD11, IM2, IM6, IM7, IM8, R1, IM12

Number	DD13
Name	Orthogonal
Label	**Orthogonal Time Series**
Symbol	$\phi : \mathbb{R}^n$
Equation	-
Description	Time series that is perpendicular to the baseline (DD11). Two polynomials are orthogonal if their inner product is zero. We define an inner product for two functions by integrating their product. $\int_a^b \phi(x)\text{base}(x)dx = 0$
Source	https://www.johndcook.com/OrthogonalPolynomials.pdf
Ref. By	GS1, IM2, IM6, IM7, IM8, IM12, R1

Number	DD14
Name	Threshold
Label	**Threshold For Voxels' Intensity**
Symbol	$p : \mathbb{R}; 0 \leq p \leq 1.0$
Equation	-
Description	p is a variable between 0 and 1. By default $p = 0.0999$. 3dfim+ calculates the average image intensity for the first sub-brick of the X (DD6) in the time series and then excludes any voxel whose intensity is less than $p *$ average. This process decreases the run time of the program.
Source	https://afni.nimh.nih.gov/pub/dist/doc/program_help/3dfim+.html
Ref. By	R1

Number	DD15
Name	Correlation Coefficient Comparing Value
Label	**Comparing Value For Correlation Coefficient Screen Display**
Symbol	$cval : \mathbb{R}$; $0 \leq cval \leq 1$
Equation	-
Description	*cval* is used to control the correlation coefficient values displayed on the user's screen as the output of the program 3dfim+. The correlation coefficient value for each voxel is printed on the screen only if the absolute value of the computed correlation coefficient is greater than or equal to *cval*.
Source	https://afni.nimh.nih.gov/pub/dist/doc/program_help/3dfim+.html
Ref. By	R1

Number	DD16
Name	Best Ideal
Label	**Best Ideal Signal**
Symbol	$r_k : \mathbb{R}^n$
Equation	-
Description	When multiple ideal signals (DD7) are defined, each of them is separately correlated with the dataset A (DD6). For each voxel, one of the signals is the most highly correlated one to that voxel's activity. We call this signal the best ideal signal for that voxel. Consider the g ideal signals $r_1, r_2, \cdots, r_g$. For each voxel: $$r_k = \underset{i=1\cdots g}{\underset{r_i}{\operatorname{argmax}}} \mid \rho(A, r_i) \mid \text{ (DD18)}$$ In this case, r_k is the best ideal signal.
Source	https://afni.nimh.nih.gov/pub/dist/doc/program_help/3dfim+.html
Ref. By	GS1, DD17, IM5, IM6, IM7, IM8, IM9, IM10, R6, R7, R8, R11

Number	DD17
Name	Best Index
Label	**Index of Best Ideal Signal**
Symbol	$k : \mathbb{N}$
Equation	-
Description	The index of the best ideal signal (DD16) is called the best index. Consider the g ideal signals $r_1, r_2, \cdots, r_g$ and a dataset A (DD6). For each voxel: $r_k = \underset{\substack{r_i \\ i=1\cdots g}}{\text{argmax}} \mid \rho(A, r_i) \mid$ (DD18) In this case, the kth ideal signal is the best ideal signal and k is the best index.
Source	`https://afni.nimh.nih.gov/pub/dist/doc/program_help/3dfim+.html`
Ref. By	R6, IM5

Number	DD18
Name	argmax
Label	**Argmax Function**
Symbol	argmax: $(\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$
Equation	-
Description	Given a function f defined on a set D, argmax function is defined as follows: $\underset{x \in D}{\text{argmax}} f(x) := \{x \mid \forall y \in D : f(x) \geq f(y)\}$
Source	`https://www.cs.ubc.ca/~schmidtm/Documents/2016_540_Argmax.pdf`
Ref. By	DD16, DD17, IM5

Number	DD19
Name	Peak
Label	**Peak to Peak**
Symbol	$pp() : \mathbb{R}$
Equation	$pp(A) = \max\limits_{i=1 \cdots n} (a_i) - \min\limits_{i=1 \cdots n} (a_i)$ where $a_i \in A$
Description	Peak to peak function calculates the variation among the elements in a dataset (DD6).
Source	-
Ref. By	IM9, IM10, IM11

4.2.4 Instance Models

In this section, we express the 3dfim+ functionality mathematically.

The goal GS1 to GS11 is solved by IM1 to IM12.

Number	IM1
Name	Pearson Model
Label	**Calculating Pearson Correlation Coefficient Between the Reference Signal and the Input Dataset**
Input	$X : \mathbb{R}^{m \times n \times p \times q}$, $r : \mathbb{R}^q$
Output	$\rho_{ijk}(X, r) = \frac{\sum\limits_{l=1}^{q}(x_{ijkl} - \bar{x}_{ijk})(r_l - \bar{r})}{[\sum\limits_{l=1}^{q}(x_{ijkl} - \bar{x}_{ijk})^2 (r_l - \bar{r})^2]^{\frac{1}{2}}}$
Description	The formula calculates the Pearson correlation coefficient (T1) between the ideal time series r (DD7) and the 3d+time dataset X (DD6). $\bar{x}_{ijk}$ and $\bar{r}$ are sample means (DD1) defining as follows: $\bar{x}_{ijk} = \frac{\sum\limits_{l=1}^{q} x_{ijkl}}{q}$ $\bar{r} = \frac{\sum\limits_{i=1}^{q} r_i}{q}$ Note that assumptions A1, A2, A3, A4 and A5 must hold while calculating this correlation. We also assumed that $r = r_k$ (DD16) in case of having more than one ideal signal.
Source	`http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf`
Ref. By	R2, R3, LC2

Number	IM2
Name	fMRI Dataset Model
Label	**Mathematical Model of Measured fMRI Dataset To Find Fit Coefficients**
Input	X, $\phi_i \in \mathbb{R}^q$, $r_i \in \mathbb{R}^q$
Output	$\beta_{ijk}^T = [\beta_0, \beta_1, \cdots, \gamma_1, \gamma_2, \cdots, \alpha_1, \alpha_2, \cdots]^T$
Description	Correlation analysis of each voxel's time series in X (DD6) with reference signal(s) r_i (DD7) where: $$M = \begin{bmatrix} 1 & 1 & \cdots & \phi_{1_1} & \cdots & r_{1_1} & \cdots \\ 1 & 2 & \cdots & \phi_{1_2} & \cdots & r_{1_2} & \cdots \\ 1 & 3 & \cdots & \phi_{1_3} & \cdots & r_{1_3} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ 1 & f & \cdots & \phi_{1_f} & \cdots & r_{1_f} & \cdots \end{bmatrix}$$ $$X_{ijk} = y_{ijk} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_f \end{bmatrix} \beta_{ijk}^* = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \gamma_1 \\ \gamma_2 \\ \vdots \\ \alpha_1 \\ \alpha_2 \\ \vdots \end{bmatrix} \epsilon_{ijk} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_f \end{bmatrix}$$ The equation can be also written as: $X_{ijk} = M\beta_{ijk}^* + \epsilon_{ijk}$ where: M is the data model consisting of baseline (DD11), orthogonal time series ϕ_i's (DD13) and ideal time series r_i's (DD7). β_{ijk}^{*T} is the vector of unknown fit coefficients for each voxel v_{ijk}. ϵ_{ijk} is the noise at a specific voxel v_{ijk} over time. α 's are the fit coefficient for ideal signals. β's are the fit coefficient for baseline. γ's are the fit coefficient for orthogonal time series.
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	IM6, IM7, IM8, R2, R12

The fMRI data we get from brain activity can be modeled as a factor of the model M. The data is composed of baseline, orthogonal time series, reference signal time series and noise. In the X matrix, the first columns indicate baseline (DD11) of the signals we get from each voxel's activity. The first column is for the constant baseline, the second column indicates the linear baseline, etc. *pnum* (DD12) indicates the degree of the baseline polynomial. After baseline columns, we have orthogonal time series (DD13) columns shown by ϕ's. We can have zero or more orthogonal time series. The next columns are for the reference time series. We can define one or more ideal time series.

Number	IM3
Name	Spearman Model
Label	**Calculating Spearman Correlation Coefficient Between the Reference Signal and the Input Dataset**
Input	$X : \mathbb{R}^{m \times n \times p \times q}$, $r : \mathbb{R}^q$
Output	$\rho_{s_{ijk}}(X, r) = \frac{\sum_{l=1}^{q}(\text{rank}(x_{ijkl}, X_{kl}) - \frac{q+1}{2})(\text{rank}(r_l, r) - \frac{q+1}{2})}{\sqrt{\sum_{l=1}^{q}(\text{rank}(x_{ijkl}, X_{kl}) - \frac{d+1}{2})^2(\text{rank}(r_l, r) - \frac{q+1}{2})^2}}$
Description	The above formula calculates Spearman correlation coefficient (T2) between the ideal time series r (DD7) and the 3d+time dataset X (DD6). Assumptions A6 and A7 must hold while calculating this correlation.
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	R4

Number	IM4
Name	Quadrant Model
Label	**Calculating Quadrant Correlation Coefficient Between the Reference Signal and the Input Dataset**
Input	$X : \mathbb{R}^{m \times n \times p \times q}$, $r : \mathbb{R}^q$
Output	$\rho_{q_{ijk}}(X, r) = \frac{\sum_{l=1}^{q} (\text{sign}(\text{rank}(x_{ijkl}, X_{kl}) - \frac{q+1}{2}))(\text{sign}(\text{rank}(r_l, r) - \frac{q+1}{2}))}{\sqrt{\sum_{l=1}^{q} ((\text{rank}(x_{ijkl}, X_{kl}) - \frac{q+1}{2}))^2 ((\text{rank}(r_l, r) - \frac{q+1}{2}))^2}}$
Description	The above formula calculates quadrant correlation coefficient between the ideal time series r (DD7) and the 3d+time dataset X (DD6). Note that assumptions A6 and A7 must hold while calculating this correlation.
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	R5

Number	IM5
Name	Best Index Model
Label	**Finding the Index of the Most Highly Correlated Ideal Time Series with the Dataset**
Input	$X : \mathbb{R}^{m \times n \times p \times q}$, $r_i : \mathbb{R}^q$
Output	$k : \mathbb{N}$ such that $r_k = \underset{i=1 \cdots g}{\underset{r_i}{\text{argmax}}} \mid \rho(X, r_i) \mid$
Description	The program gives an integer upon requesting the best index. argmax (DD18) returns the best ideal signal r_k (DD16) and index k is the best index (DD17).
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	R6

Number	IM6
Name	Baseline Quantity
Label	**Calculating Baseline Quantity for fMRI Dataset**
Input	$X : \mathbb{R}^{m\times n\times p\times q}$, $r_i : \mathbb{R}^q$
Output	$Baseline = \sum_{i=1}^{c} \beta_i . avg(\mathrm{base}_i) + \sum_{j=1}^{h} \gamma_j . avg(\phi_i) + \hat{\alpha}.\min(r_k)$
Description	The program returns a real number for the *Baseline* computed as mentioned in the output above. We assume that the polynomial baseline model is of order c ($pnum = c$ (DD12)) and we have h orthogonal time series (DD13). base_i indicates the baseline model (DD11) of degree i. $avg()$ function calculates the average value of its input over time. $\hat{\alpha}$ is the fit coefficient for the (best) ideal time series (DD16). min() function outputs the minimum value of the (best) ideal time series (DD7) over time. β and γ are defined in IM2.
Source	`http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf`
Ref. By	IM9, R7, R9

Number	IM7
Name	Average Quantity
Label	**Calculating Average Quantity for fMRI Dataset**
Input	$X : \mathbb{R}^{m\times n\times p\times q}$, $r_i : \mathbb{R}^q$
Output	$Average = \sum_{i=1}^{c} \beta_i.avg(\text{base}_i) + \sum_{j=1}^{h} \gamma_j.avg(\phi_i) + \hat{\alpha}.avg(r_k)$
Description	The program returns a real number for the *Average* computed based on the formula mentioned in the output. We assume that the polynomial baseline model is of order c ($pnum = c$ (DD12)) and we have h orthogonal time series (DD13). base_i indicates the baseline model (DD11) of degree i. $avg()$ function calculates the average value of its input over time. $\hat{\alpha}$ is the fit coefficient for the (best) ideal time series (DD16). β and γ are defined in IM2.
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	IM10, R8, R10

Number	IM8
Name	Topline Quantity
Label	**Calculating Topline Quantity for fMRI Dataset**
Input	$X : \mathbb{R}^{m \times n \times p \times q}$, $r_i : \mathbb{R}^q$
Output	$\mathit{Topline} = \sum_{i=1}^{c} \beta_i.\mathit{avg}(\text{base}_i) + \sum_{j=1}^{h} \gamma_j.\mathit{avg}(\phi_i) + \hat{\alpha}.\max(r_k)$
Description	The program returns a real number for the *Topline* computed based on the above formula. We assume that the polynomial baseline model is of order c ($\mathit{pnum} = c$ (DD12)) and we have h orthogonal time series (DD13). base_i indicates the baseline model (DD11) of degree i. $\mathit{avg}()$ function calculates the average value of its input over time. $\hat{\alpha}$ is the fit coefficient for the (best) ideal time series (DD16). max() function outputs the maximum value of the (best) ideal time series (DD16) over time. β and γ are defined in IM2.
Source	`http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf`
Ref. By	IM11, R11, R12

Number	IM9
Name	Baseline Percentage Change
Label	**Calculating Percentage Change in the fMRI Dataset Relative to Baseline**
Input	$Baseline$ (IM6), $r_i : \mathbb{R}^q$
Output	$\%base = 100.\frac{\hat{\alpha}.pp(r_k)}{Baseline}$
Description	The formula calculates the percentage change in the fMRI dataset (DD6) due to the (best) ideal time series (DD7, DD16) relative to the *Baseline*(IM6) for each voxel. $\hat{\alpha}$ is the fit coefficient for the (best) ideal time series. $pp()$ is the peak to peak function (DD19) which calculates the variation of the (best) ideal time series as follows: $$pp(r_k) = \max_{j=1\cdots d}(r_{k_j}) - \min_{j=1\cdots d}(r_{k_j})$$
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	R7

Number	IM10
Name	Average Percentage Change
Label	**Calculating Percentage Change in the fMRI Dataset Relative to Average**
Input	*Average* (IM7), $r_i : \mathbb{R}^q$
Output	$\% avg = 100.\frac{\hat{\alpha}.pp(r_k)}{Average}$
Description	The formula calculates the percentage change in the fMRI dataset (DD6) due to the (best) ideal time series (DD7, DD16) relative to the *Average* (IM7) for each voxel. $\hat{\alpha}$ is the fit coefficient for the (best) ideal time series. $pp()$ is the peak to peak function (DD19) which calculates the variation of the (best) ideal time series as follows: $$pp(r_k) = \max_{j=1 \cdots d}(r_{k_j}) - \min_{j=1 \cdots d}(r_{k_j})$$
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	R8

Number	IM11
Name	Topline Percentage Change
Label	**Calculating Percentage Change in the fMRI Dataset Relative to Topline**
Input	$\mathit{Topline}$(IM8), $r_i : \mathbb{R}^q$
Output	$\%top = 100.\frac{\hat{\alpha}.pp(r_k)}{\mathit{Topline}}$
Description	The formula calculates the percentage change in the fMRI dataset (DD6) due to the (best) ideal time series (DD7, DD16) relative to the *Topline* (IM8) for each voxel. $\hat{\alpha}$ is the fit coefficient for the (best) ideal time series. $pp()$ is the peak to peak function (DD19) which calculates the variation of the (best) ideal time series as follows: $$pp(r_k) = \max_{j=1\cdots d}(r_{k_j}) - \min_{j=1\cdots d}(r_{k_j})$$
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	R11

Number	IM12
Name	Standard Deviation of the Residuals
Label	**Calculating The Standard Deviation of the Residuals at Each Voxel Between the fMRI Dataset and Corresponding Data Estimation**
Input	$X : \mathbb{R}^{m \times n \times p \times q}$, $\phi_i : \mathbb{R}^q$, $r_i : \mathbb{R}^q$
Output	$\hat{\sigma}_{ijk} = \sqrt{\frac{\sum_{l=1}^{q}(X_{ijkl}-\hat{X}_{ijkl})^2}{q-n_b-n_o-n_i}}$
Description	Extending the theoretical model T8 to the fMRI dataset, we have: $\hat{X}_{ijkl} = (M^T M)^{-1} M^T X_{ijkl}$ Using theoretical models T5, T6 and T7 we can calculate thestandard deviation of the residuals: $\hat{\sigma}_{ijk} = \sqrt{\frac{\sum_{l=1}^{q}(X_{ijkl}-\hat{X}_{ijkl})^2}{q - pnum - n_o - n_i}}$ Where: $pnum$ is the polynomial degree (DD12), n_o is the number of orthogonal time series (DD13), and n_i depends on the number of ideal time series (DD7) such that: $n_i = \begin{cases} 1 \text{ if we have 1 ideal time series} \\ 2 \text{ if we have more than one ideal time series} \end{cases}$
Source	http://homepage.usask.ca/~ges125/fMRI/AFNIdoc/3dfim+.pdf
Ref. By	R13

4.2.5 Data Constraints

Data constraints on the input are as follows:

- Dimensions of reference signals (DD7) and orthogonal time series (DD13) should match.

Data constraints on the output are as follows:

- Correlation coefficients ρ (IM1),ρ_s (IM3),ρ_q (IM4) must lie between -1 and 1.

4.2.6 Properties of a Correct Solution

Whether we use Pearson, Spearman or quadratic correlation coefficient estimation, the value of the computed correlation coefficients should be between -1 and 1.

5 Requirements

This section provides functional and non-functional requirements for 3dfim+.

5.1 Functional Requirements

R1: Input the following functions, data and parameters:

symbol	**description**
X	fMRI data as a 3d+time dataset in NIfTI format (DD6)
$pnum$	degree of the polynomial in the baseline model (DD12)
ϕ	orthogonal time series function(s) (DD13)
r	reference time series function(s) (DD7)
p	threshold for voxels' intensity (DD14)
$cval$	comparing value for correlation coefficient screen display (DD15)

R2: Use the inputs in R1 to estimate the vector of unknown parameters β (IM2) at each voxel (from IM2).

R3: Calculate the Pearson correlation coefficient at each voxel between X and (best) r (from IM1).

R4: Calculate the Spearman correlation coefficient at each voxel between X and (best) r (from IM3).

R5: Calculate the quadrant correlation coefficient at each voxel between X and (best) r (from IM4).

R6: In case of having multiple ideal signals r, report the index number k (DD17) for the best ideal time series r_k (DD16) (from IM5).

R7: Calculate the percentage change in X due to the (best) ideal time series (DD7, DD16) relative to the *Baseline* (IM6) for each voxel (from IM9).

R8: Calculate the percentage change in X due to the (best) ideal time series (DD7, DD16) relative to the *Average* (IM7) for each voxel (from IM10).

R9: Calculate the fMRI dataset X *Baseline* quantity for each voxel (from IM6).

R10: Calculate the fMRI dataset X *Average* quantity for each voxel (from IM7).

R11: Calculate the percentage change in X due to the (best) ideal time series (DD7, DD16) relative to the *Topline* (IM8) for each voxel (from IM11).

R12: Calculate the fMRI dataset X *Topline* quantity for each voxel (IM8).

R13: Calculate the standard deviation of the residuals at each voxel between the fMRI dataset and corresponding data estimation (from IM12).

5.2 Non-functional Requirements

Considering the use of this program in the research, as well as keeping an eye on its future use in the clinical practice, the priority non-functional requirements are correctness, reliability, verifiability, understandability, reusability and maintainability.

6 Other System Issues

N/A

7 Traceability Matrix

A traceability matrix is given for instance models and assumptions.

Input Assumptions	IMA	IM3	IM4	
A1	✓			
A2	✓			
A3	✓			
A4	✓			
A5	✓			
A6		✓		✓
A7		✓		✓

Table 7: Traceability matrix between instance models and input assumptions

8 Likely Changes

LC1: A4 - Although outliers can have deleterious effects on statistical analyses, some people prefer not to exclude them reasoning the outliers are parts of the dataset.

LC2: IM1 - There are other methods of calculating correlation coefficients such as Kendall rank correlation which is likely to be used instead of Pearson correlation. Input data assumptions might be different from method to method.

References

[1] G. H. Golub and C. F. V. Loan, *Matrix Computation*, 3rd ed. The Johns Hopkins University Press, 1996.

[2] B. Douglas Ward, “”afni program: 3dfim+”.” [Online]. Available: https://afni.nimh.nih.gov/pub/dist/doc/program_help/3dfim+.html

[3] N. Koothoor, “A document drive approach to certifying scientific computing software,” Master’s thesis, McMaster University, Hamilton, Ontario, Canada, 2013. [Online]. Available: http://hdl.handle.net/11375/13266

[4] W. S. Smith and L. Lai, “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. Ȧgerfalk, and N. Kraiem, Eds. Paris, France: In conjunction with 13th IEEE International Requirements Engineering Conference, 2005, pp. 107–121.

[5] AFNI, “Intro.” [Online]. Available: https://afni.nimh.nih.gov/

[6] J. D. Kloke and J. W. McKean, “Rfit: Rank-based estimation for linear models,” *The R Journal*, vol. 4, no. 2, pp. 57–64, 2012.

[7] G. Shevlyakov and P. Smirnov, “Robust estimation of the correlation coefficient: An attempt of survey,” *AUSTRIAN JOURNAL OF STATISTICS*, vol. 40, no. 1 and 2, pp. 147–156, 2011.

[8] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. Lawrence Erlbaum Associates, 1998.

[9] J. D. Evans, *Straightforward statistics for the behavioral sciences.* Pacific Grove, CA: Brooks/Cole Publishing Company, 1996.

[10] S. S. Stevens, “Afni program: 3dfim+,” *Science*, vol. 103, pp. 677–680, 1946.

[11] “Coordinate systems,” June 2014. [Online]. Available: https://www.slicer.org/wiki/Coordinate_systems

[12] M. T. Heath, *Scientific Computing : An Introductory Survey*, 2nd ed. McGraw-Hill, 2002.

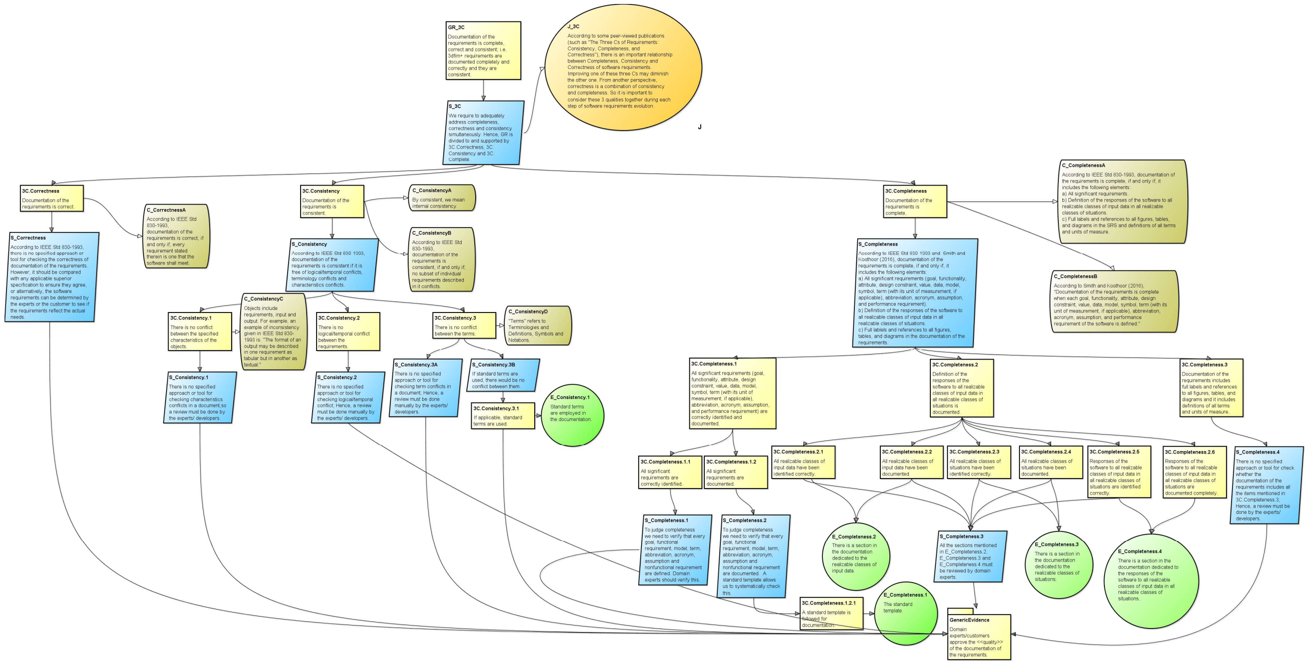
[13] D. Gries and F. B. Schneider, *A logical approach to discrete math.* Springer-Verlag Inc., 1993.

[14] D. L. Parnas and P. Clements, “A rational design process: How and why to fake it,” *IEEE Transactions on Software Engineering*, vol. 2, no. 2, pp. 251–257, 1986.

[15] E. A. Lee, *Structure and Interpretation of Signals and Systems*, 2011.

Appendix B

Assurance Case for 3dfim+



GR_3C
Documentation of the requirements is complete, correct and consistent; i.e. 3dfim+ requirements are documented completely and correctly and they are consistent.

J_3C
According to some peer-viewed publications (such as "The Three Cs of Requirements: Consistency, Completeness, and Correctness"), there is an important relationship between Completeness, Consistency and Correctness of software requirements. Improving one of these three Cs may diminish the other one. From another perspective, correctness is a combination of consistency and completeness. So it is important to consider these 3 qualities together during each step of software requirements evolution.

J

S_3C
We require to adequately address completeness, correctness and consistency simultaneously. Hence, GR is divided to and supported by 3C.Correctness, 3C. Consistency and 3C. Complete.

3C.Correctness
Documentation of the requirements is correct.

C_CorrectnessA
According to IEEE Std 830-1993, documentation of the requirements is correct, if and only if, every requirement stated therein is one that the software shall meet.

S_Correctness
According to IEEE Std 830-1993, there is no specified approach or tool for checking the correctness of documentation of the requirements. However, it should be compared with any applicable superior specification to ensure they agree, or alternatively, the software requirements can be determined by the experts or the customer to see if the requirements reflect the actual needs.

3C.Consistency
Documentation of the requirements is consistent.

C_ConsistencyA
By consistent, we mean internal consistency.

C_ConsistencyB
According to IEEE Std 830-1993, documentation of the requirements is consistent, if and only if, no subset of individual requirements described in it conflicts.

S_Consistency
According to IEEE Std 830 1993, documentation of the requirements is consistent if it is free of logical/temporal conflicts, terminology conflicts and characteristics conflicts.

C_ConsistencyC
Objects include requirements, input and output. For example, an example of inconsistency given in IEEE Std 830-1993 is: "The format of an output may be described in one requirement as tabular but in another as textual."

3C.Consistency.1
There is no conflict between the specified characteristics of the objects.

S_Consistency.1
There is no specified approach or tool for checking characteristics conflicts in a document,so a review must be done by the experts/ developers.

3C.Consistency.2
There is no logical/temporal conflict between the requirements.

S_Consistency.2
There is no specified approach or tool for checking logical/temporal conflict; Hence, a review must be done manually by the experts/ developers.

3C.Consistency.3
There is no conflict between the terms.

C_ConsistencyD
"Terms" refers to Terminologies and Definitions, Symbols and Notations.

S_Consistency.3A
There is no specified approach or tool for checking term conflicts in a document; Hence, a review must be done manually by the experts/ developers.

S_Consistency.3B
If standard terms are used, there would be no conflict between them.

3C.Consistency.3.1
If applicable, standard terms are used.

E_Consistency.1
Standard terms are employed in the documentation.

3C.Completeness
Documentation of the requirements is complete.

C_CompletenessA
According to IEEE Std 830-1993, documentation of the requirements is complete, if and only if, it includes the following elements:
a) All significant requirements.
b) Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations.
c) Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.

C_CompletenessB
According to Smith and Koothoor (2016), "Documentation of the requirements is complete when each goal, functionality, attribute, design constraint, value, data, model, symbol, term (with its unit of measurement, if applicable), abbreviation, acronym, assumption, and performance requirement of the software is defined."

S_Completeness
According to IEEE Std 830 1993 and Smith and Koothoor (2016), documentation of the requirements is complete, if and only if, it includes the following elements:
a) All significant requirements (goal, functionality, attribute, design constraint, value, data, model, symbol, term (with its unit of measurement, if applicable), abbreviation, acronym, assumption, and performance requirement).
b) Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations.
c) Full lablels and references to all figures, tables, and diagrams in the documentation of the requirements.

3C.Completeness.1
All significant requirements (goal, functionality, attribute, design constraint, value, data, model, symbol, term (with its unit of measurement, if applicable), abbreviation, acronym, assumption, and performance requirement) are correctly identified and documented.

3C.Completeness.2
Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations is documented.

3C.Completeness.3
Documentation of the requirements includes full labels and references to all figures, tables, and diagrams and it includes definitions of all terms and units of measure.

3C.Completeness.1.1
All significant requirements are correctly identified.

3C.Completeness.1.2
All significant requirements are documented.

3C.Completeness.2.1
All realizable classes of input data have been identified correctly.

3C.Completeness.2.2
All realizable classes of input data have been documented.

3C.Completeness.2.3
All realizable classes of situations have been identified correctly.

3C.Completeness.2.4
All realizable classes of situations have been documented.

3C.Completeness.2.5
Responses of the software to all realizable classes of input data in all realizable classes of situations are identified correctly.

3C.Completeness.2.6
Responses of the software to all realizable classes of input data in all realizable classes of situations are documented completely.

S_Completeness.4
There is no specified approach or tool for check whether the documentation of the requirements includes all the items mentioned in 3C.Completeness.3; Hence, a review must be done by the experts/ developers.

S_Completeness.1
To judge completeness we need to verify that every goal, functional requirement, model, term, abbreviation, acronym, assumption and nonfunctional requirement are defined. Domain experts should verify this.

S_Completeness.2
To judge completeness we need to verify that every goal, functional requirement, model, term, abbreviation, acronym, assumption and nonfunctional requirement are documented. A standard template allows us to systematically check this.

E_Completeness.2
There is a section in the documentation dedicated to the realizable classes of input data.

S_Completeness.3
All the sections mentioned in E_Completeness.2, E_Completeness.3 and E_Completeness.4 must be reviewed by domain experts.

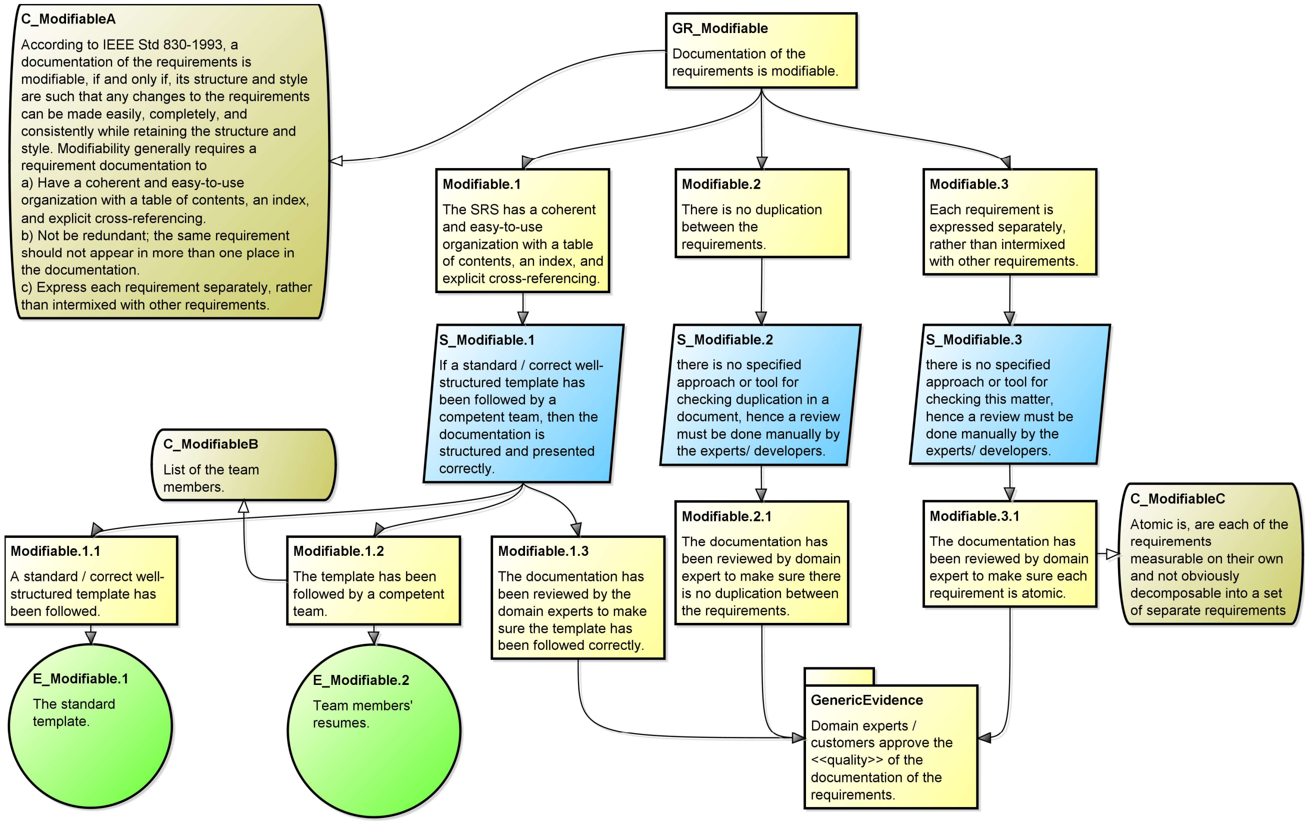
E_Completeness.3
There is a section in the documentation dedicated to the realizable classes of situations.

E_Completeness.4
There is a section in the documentation dedicated to the responses of the software to all realizable classes of input data in all realizable classes of situations.

3C.Completeness.1.2.1
A standard template is followed for documentation.

E_Completeness.1
The standard template.

GenericEvidence
Domain experts/customers approve the <<quality>> of the documentation of the requirements.



C_ModifiableA

According to IEEE Std 830-1993, a documentation of the requirements is modifiable, if and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style. Modifiability generally requires a requirement documentation to
a) Have a coherent and easy-to-use organization with a table of contents, an index, and explicit cross-referencing.
b) Not be redundant; the same requirement should not appear in more than one place in the documentation.
c) Express each requirement separately, rather than intermixed with other requirements.

GR_Modifiable

Documentation of the requirements is modifiable.

Modifiable.1

The SRS has a coherent and easy-to-use organization with a table of contents, an index, and explicit cross-referencing.

Modifiable.2

There is no duplication between the requirements.

Modifiable.3

Each requirement is expressed separately, rather than intermixed with other requirements.

S_Modifiable.1

If a standard / correct well-structured template has been followed by a competent team, then the documentation is structured and presented correctly.

S_Modifiable.2

there is no specified approach or tool for checking duplication in a document, hence a review must be done manually by the experts/ developers.

S_Modifiable.3

there is no specified approach or tool for checking this matter, hence a review must be done manually by the experts/ developers.

C_ModifiableB

List of the team members.

Modifiable.1.1

A standard / correct well-structured template has been followed.

Modifiable.1.2

The template has been followed by a competent team.

Modifiable.1.3

The documentation has been reviewed by the domain experts to make sure the template has been followed correctly.

Modifiable.2.1

The documentation has been reviewed by domain expert to make sure there is no duplication between the requirements.

Modifiable.3.1

The documentation has been reviewed by domain expert to make sure each requirement is atomic.

C_ModifiableC

Atomic is, are each of the requirements measurable on their own and not obviously decomposable into a set of separate requirements

E_Modifiable.1

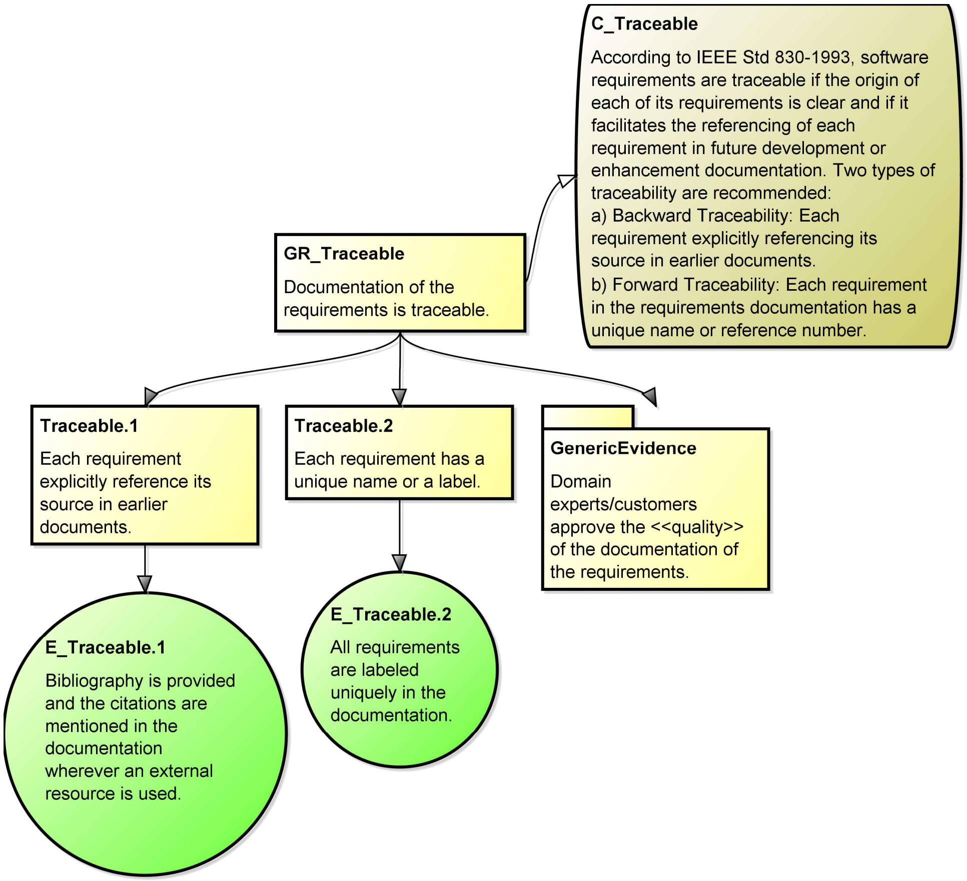
The standard template.

E_Modifiable.2

Team members' resumes.

GenericEvidence

Domain experts / customers approve the <<quality>> of the documentation of the requirements.



C_Traceable

According to IEEE Std 830-1993, software requirements are traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation. Two types of traceability are recommended:
a) Backward Traceability: Each requirement explicitly referencing its source in earlier documents.
b) Forward Traceability: Each requirement in the requirements documentation has a unique name or reference number.

GR_Traceable

Documentation of the requirements is traceable.

Traceable.1

Each requirement explicitly reference its source in earlier documents.

Traceable.2

Each requirement has a unique name or a label.

GenericEvidence

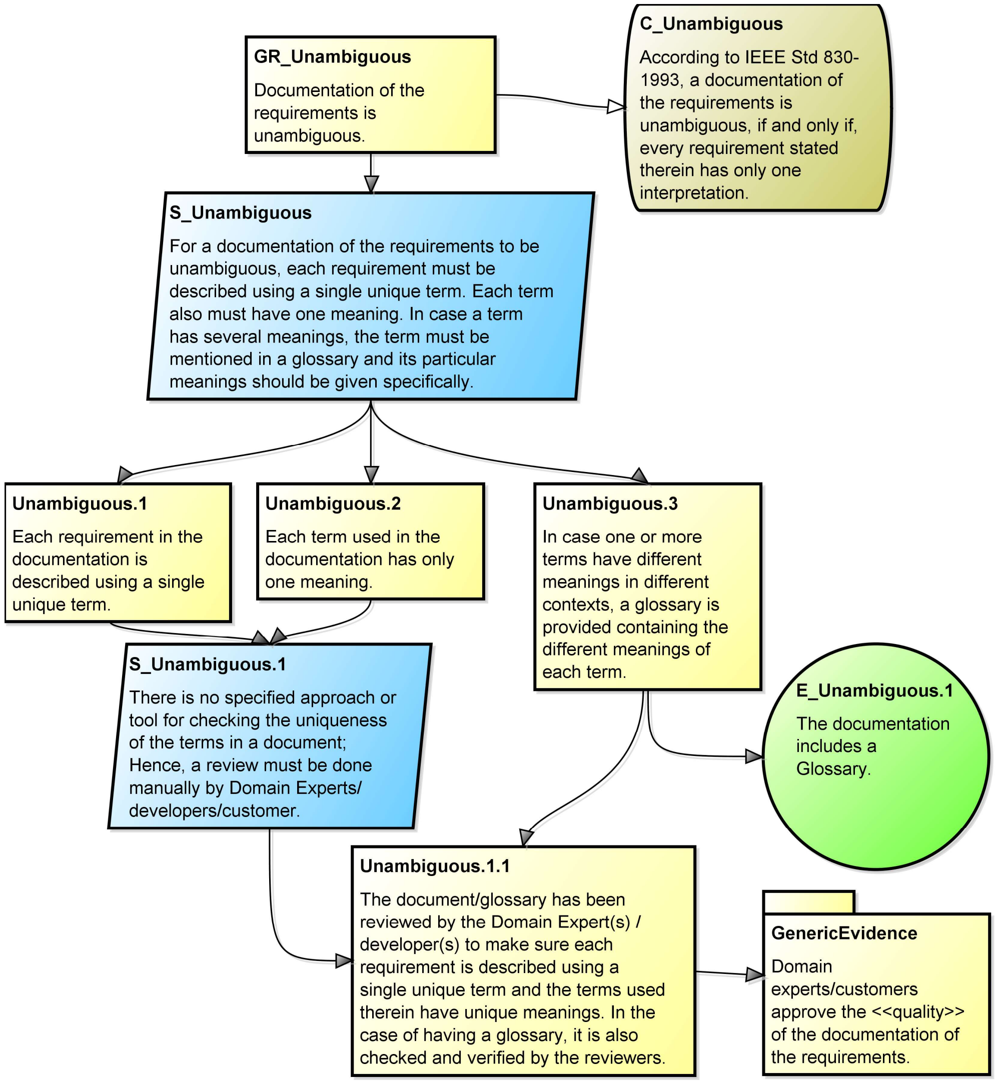
Domain experts/customers approve the <<quality>> of the documentation of the requirements.

E_Traceable.1

Bibliography is provided and the citations are mentioned in the documentation wherever an external resource is used.

E_Traceable.2

All requirements are labeled uniquely in the documentation.



GR_Unambiguous

Documentation of the requirements is unambiguous.

C_Unambiguous

According to IEEE Std 830-1993, a documentation of the requirements is unambiguous, if and only if, every requirement stated therein has only one interpretation.

S_Unambiguous

For a documentation of the requirements to be unambiguous, each requirement must be described using a single unique term. Each term also must have one meaning. In case a term has several meanings, the term must be mentioned in a glossary and its particular meanings should be given specifically.

Unambiguous.1

Each requirement in the documentation is described using a single unique term.

Unambiguous.2

Each term used in the documentation has only one meaning.

Unambiguous.3

In case one or more terms have different meanings in different contexts, a glossary is provided containing the different meanings of each term.

S_Unambiguous.1

There is no specified approach or tool for checking the uniqueness of the terms in a document; Hence, a review must be done manually by Domain Experts/ developers/customer.

E_Unambiguous.1

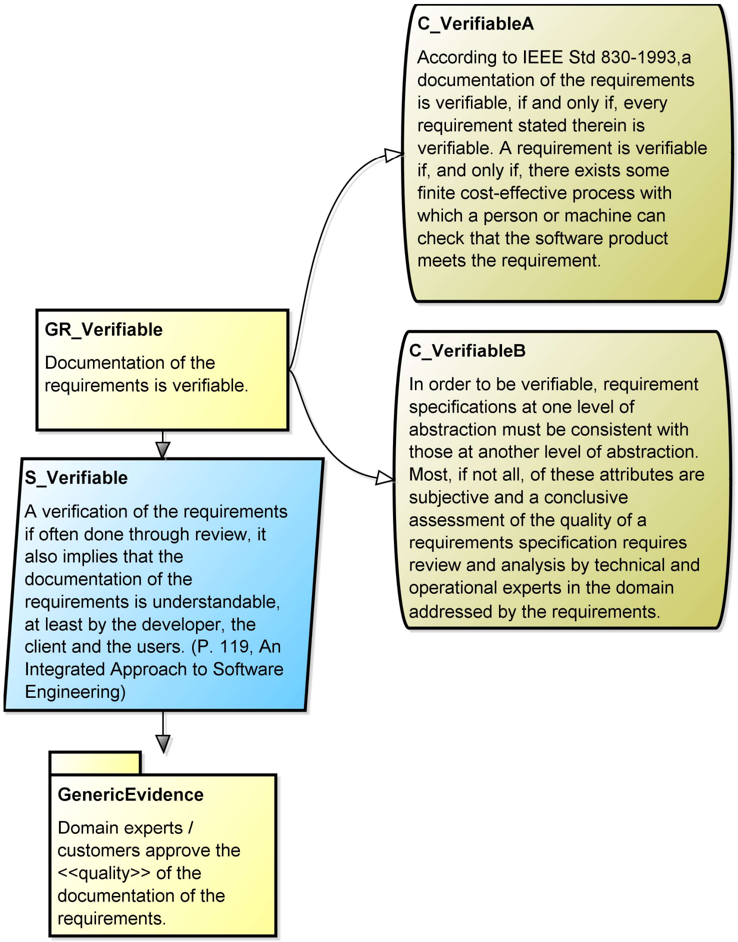
The documentation includes a Glossary.

Unambiguous.1.1

The document/glossary has been reviewed by the Domain Expert(s) / developer(s) to make sure each requirement is described using a single unique term and the terms used therein have unique meanings. In the case of having a glossary, it is also checked and verified by the reviewers.

GenericEvidence

Domain experts/customers approve the <<quality>> of the documentation of the requirements.



GR_Verifiable

Documentation of the requirements is verifiable.

C_VerifiableA

According to IEEE Std 830-1993,a documentation of the requirements is verifiable, if and only if, every requirement stated therein is verifiable. A requirement is verifiable if, and only if, there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement.

C_VerifiableB

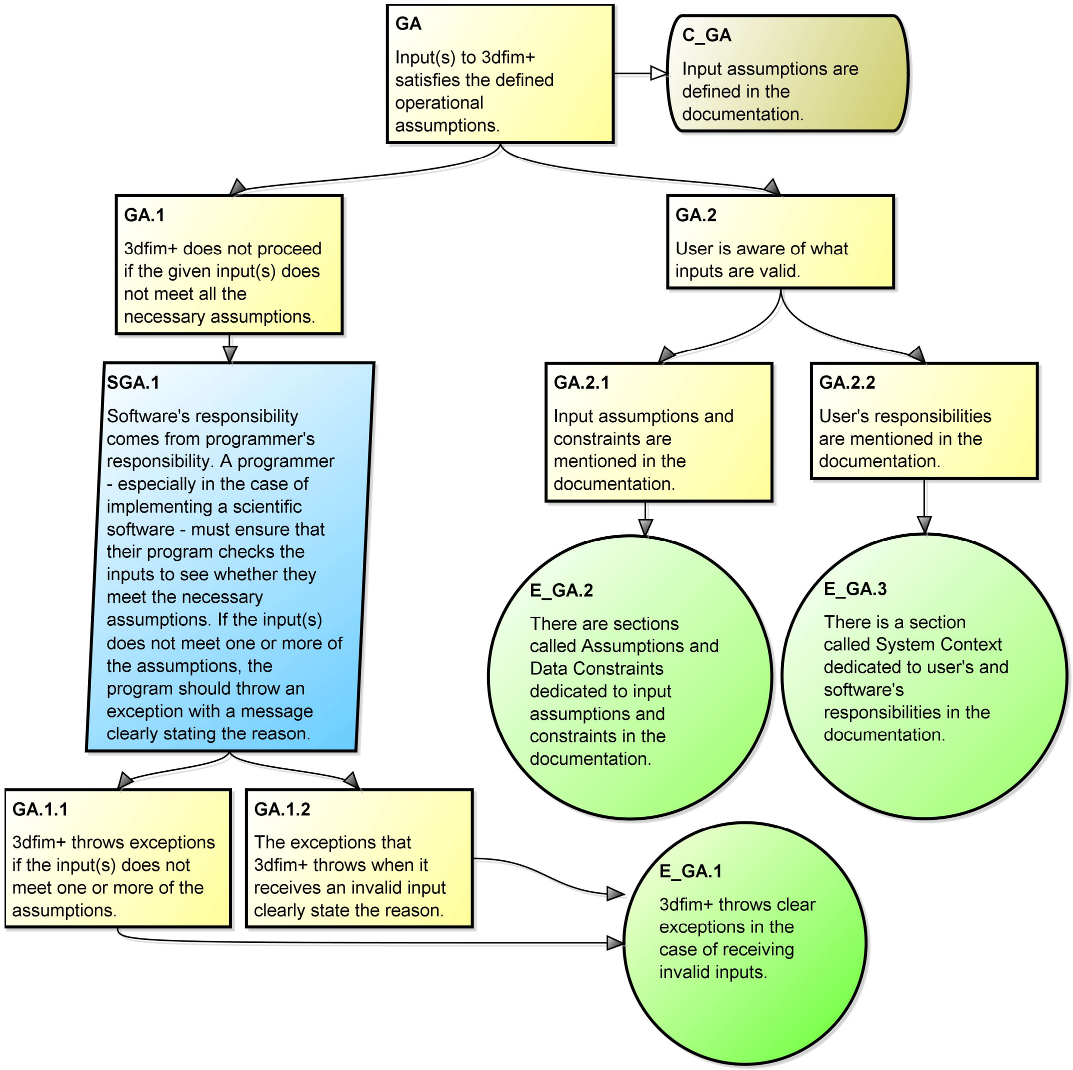
In order to be verifiable, requirement specifications at one level of abstraction must be consistent with those at another level of abstraction. Most, if not all, of these attributes are subjective and a conclusive assessment of the quality of a requirements specification requires review and analysis by technical and operational experts in the domain addressed by the requirements.

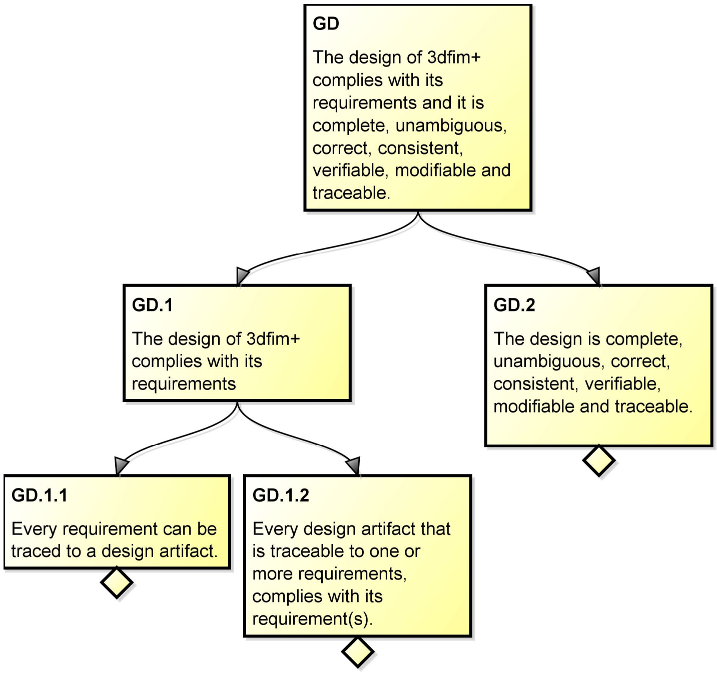
S_Verifiable

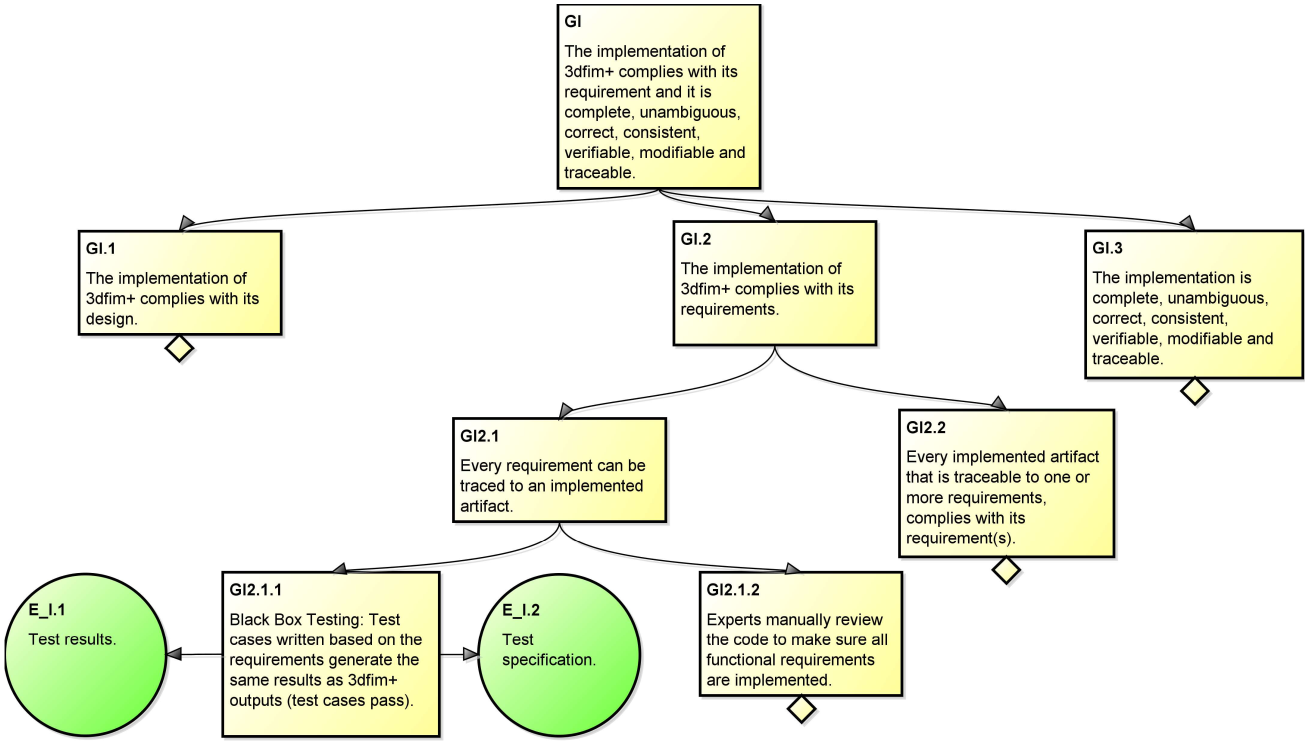
A verification of the requirements if often done through review, it also implies that the documentation of the requirements is understandable, at least by the developer, the client and the users. (P. 119, An Integrated Approach to Software Engineering)

GenericEvidence

Domain experts / customers approve the <<quality>> of the documentation of the requirements.


GA
Input(s) to 3dfim+ satisfies the defined operational assumptions.
C_GA
Input assumptions are defined in the documentation.
GA.1
3dfim+ does not proceed if the given input(s) does not meet all the necessary assumptions.
GA.2
User is aware of what inputs are valid.
SGA.1
Software's responsibility comes from programmer's responsibility. A programmer - especially in the case of implementing a scientific software - must ensure that their program checks the inputs to see whether they meet the necessary assumptions. If the input(s) does not meet one or more of the assumptions, the program should throw an exception with a message clearly stating the reason.
GA.2.1
Input assumptions and constraints are mentioned in the documentation.
GA.2.2
User's responsibilities are mentioned in the documentation.
E_GA.2
There are sections called Assumptions and Data Constraints dedicated to input assumptions and constraints in the documentation.
E_GA.3
There is a section called System Context dedicated to user's and software's responsibilities in the documentation.
GA.1.1
3dfim+ throws exceptions if the input(s) does not meet one or more of the assumptions.
GA.1.2
The exceptions that 3dfim+ throws when it receives an invalid input clearly state the reason.
E_GA.1
3dfim+ throws clear exceptions in the case of receiving invalid inputs.


GD
The design of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GD.1
The design of 3dfim+ complies with its requirements
GD.2
The design is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GD.1.1
Every requirement can be traced to a design artifact.
GD.1.2
Every design artifact that is traceable to one or more requirements, complies with its requirement(s).



GI

The implementation of 3dfim+ complies with its requirement and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

GI.1

The implementation of 3dfim+ complies with its design.

GI.2

The implementation of 3dfim+ complies with its requirements.

GI.3

The implementation is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

GI2.1

Every requirement can be traced to an implemented artifact.

GI2.2

Every implemented artifact that is traceable to one or more requirements, complies with its requirement(s).

E_I.1

Test results.

GI2.1.1

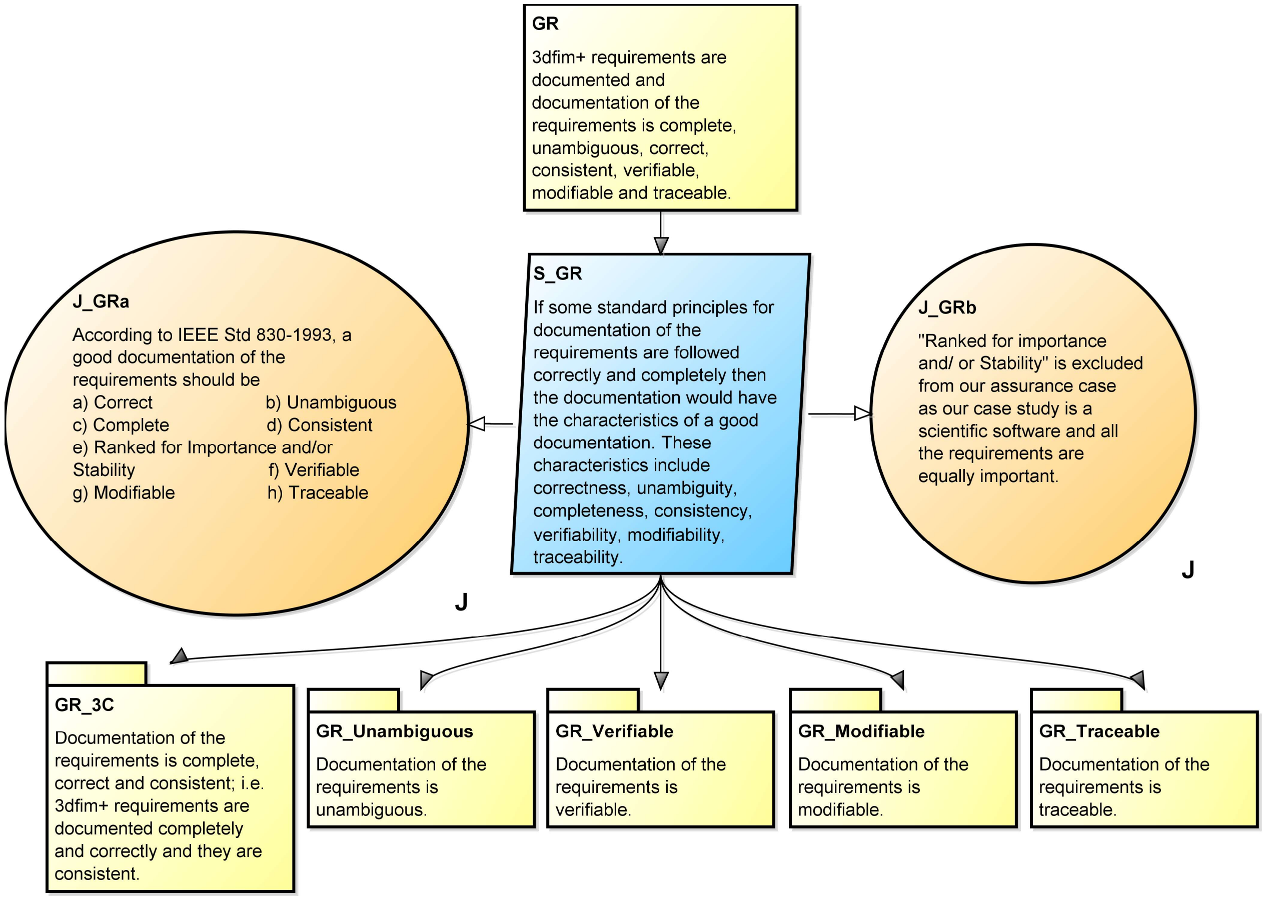
Black Box Testing: Test cases written based on the requirements generate the same results as 3dfim+ outputs (test cases pass).

E_I.2

Test specification.

GI2.1.2

Experts manually review the code to make sure all functional requirements are implemented.



GR

3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

S_GR

If some standard principles for documentation of the requirements are followed correctly and completely then the documentation would have the characteristics of a good documentation. These characteristics include correctness, unambiguity, completeness, consistency, verifiability, modifiability, traceability.

J_GRa

According to IEEE Std 830-1993, a good documentation of the requirements should be

a) Correct b) Unambiguous
c) Complete d) Consistent
e) Ranked for Importance and/or Stability f) Verifiable
g) Modifiable h) Traceable

J_GRb

"Ranked for importance and/ or Stability" is excluded from our assurance case as our case study is a scientific software and all the requirements are equally important.

GR_3C

Documentation of the requirements is complete, correct and consistent; i.e. 3dfim+ requirements are documented completely and correctly and they are consistent.

GR_Unambiguous

Documentation of the requirements is unambiguous.

GR_Verifiable

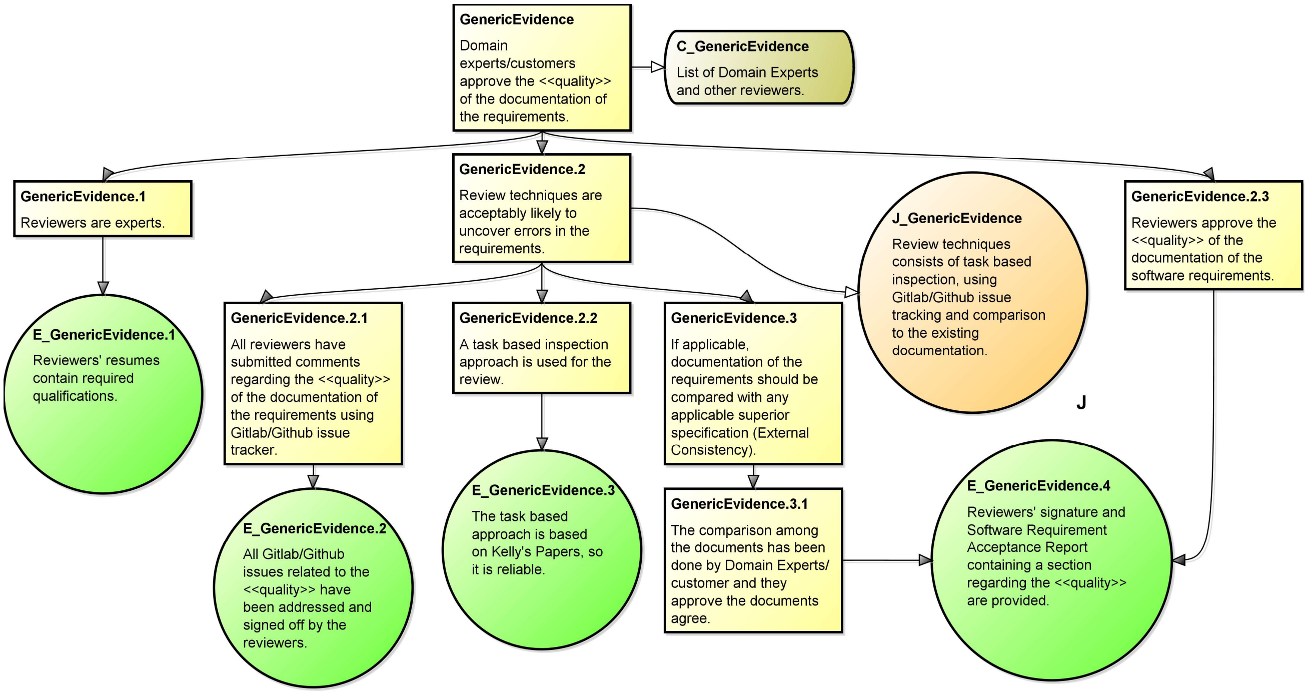
Documentation of the requirements is verifiable.

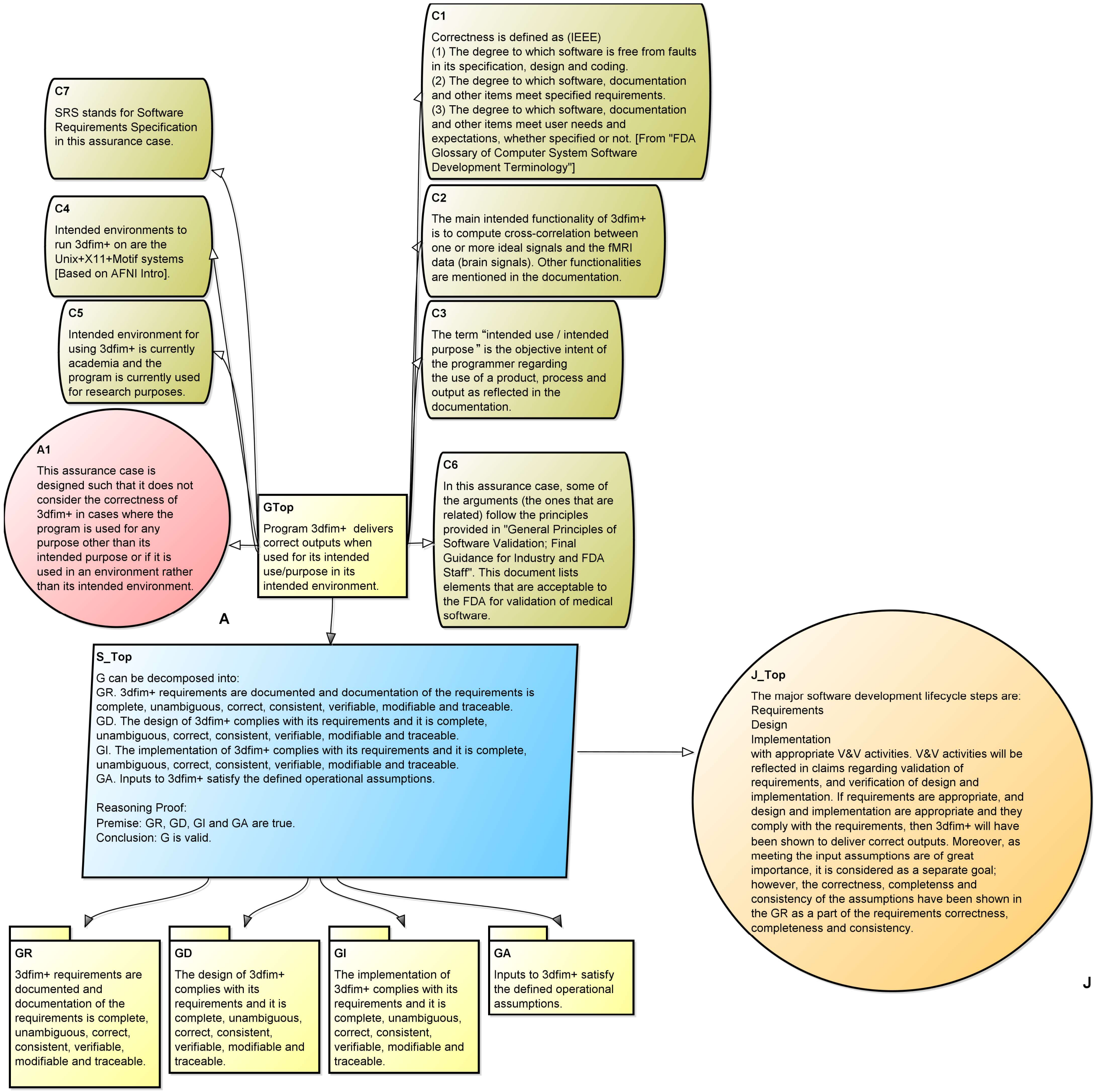
GR_Modifiable

Documentation of the requirements is modifiable.

GR_Traceable

Documentation of the requirements is traceable.


GenericEvidence
Domain experts/customers approve the <<quality>> of the documentation of the requirements.
C_GenericEvidence
List of Domain Experts and other reviewers.
GenericEvidence.1
Reviewers are experts.
GenericEvidence.2
Review techniques are acceptably likely to uncover errors in the requirements.
J_GenericEvidence
Review techniques consists of task based inspection, using Gitlab/Github issue tracking and comparison to the existing documentation.
J
GenericEvidence.2.3
Reviewers approve the <<quality>> of the documentation of the software requirements.
E_GenericEvidence.1
Reviewers' resumes contain required qualifications.
GenericEvidence.2.1
All reviewers have submitted comments regarding the <<quality>> of the documentation of the requirements using Gitlab/Github issue tracker.
GenericEvidence.2.2
A task based inspection approach is used for the review.
GenericEvidence.3
If applicable, documentation of the requirements should be compared with any applicable superior specification (External Consistency).
E_GenericEvidence.2
All Gitlab/Github issues related to the <<quality>> have been addressed and signed off by the reviewers.
E_GenericEvidence.3
The task based approach is based on Kelly's Papers, so it is reliable.
GenericEvidence.3.1
The comparison among the documents has been done by Domain Experts/ customer and they approve the documents agree.
E_GenericEvidence.4
Reviewers' signature and Software Requirement Acceptance Report containing a section regarding the <<quality>> are provided.



C7

SRS stands for Software Requirements Specification in this assurance case.

C4

Intended environments to run 3dfim+ on are the Unix+X11+Motif systems [Based on AFNI Intro].

C5

Intended environment for using 3dfim+ is currently academia and the program is currently used for research purposes.

A1

This assurance case is designed such that it does not consider the correctness of 3dfim+ in cases where the program is used for any purpose other than its intended purpose or if it is used in an environment rather than its intended environment.

C1

Correctness is defined as (IEEE)
(1) The degree to which software is free from faults in its specification, design and coding.
(2) The degree to which software, documentation and other items meet specified requirements.
(3) The degree to which software, documentation and other items meet user needs and expectations, whether specified or not. [From "FDA Glossary of Computer System Software Development Terminology"]

C2

The main intended functionality of 3dfim+ is to compute cross-correlation between one or more ideal signals and the fMRI data (brain signals). Other functionalities are mentioned in the documentation.

C3

The term “intended use / intended purpose” is the objective intent of the programmer regarding the use of a product, process and output as reflected in the documentation.

GTop

Program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment.

C6

In this assurance case, some of the arguments (the ones that are related) follow the principles provided in "General Principles of Software Validation; Final Guidance for Industry and FDA Staff". This document lists elements that are acceptable to the FDA for validation of medical software.

S_Top

G can be decomposed into:
GR. 3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GD. The design of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GI. The implementation of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GA. Inputs to 3dfim+ satisfy the defined operational assumptions.

Reasoning Proof:
Premise: GR, GD, GI and GA are true.
Conclusion: G is valid.

J_Top

The major software development lifecycle steps are:
Requirements
Design
Implementation
with appropriate V&V activities. V&V activities will be reflected in claims regarding validation of requirements, and verification of design and implementation. If requirements are appropriate, and design and implementation are appropriate and they comply with the requirements, then 3dfim+ will have been shown to deliver correct outputs. Moreover, as meeting the input assumptions are of great importance, it is considered as a separate goal; however, the correctness, completenss and consistency of the assumptions have been shown in the GR as a part of the requirements correctness, completeness and consistency.

GR

3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

GD

The design of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

GI

The implementation of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

GA

Inputs to 3dfim+ satisfy the defined operational assumptions.

Appendix C

Task list for Scientists - 3dfim+ SRS Review

Task list for Scientists - 3dfim+ SRS Review

Mojdeh Sayari Nejad and Spencer Smith

June 23, 2017

Contents

1 Purpose of Document

This document is intended to act as a guide to review the SRS document. The scope of this document is to involve the scientists in reading and reviewing the SRS document. To initiate the review process, we have assigned a set of tasks which need to be completed. Every task is framed as a question in a specific section of the SRS which needs to be answered after reading the corresponding section in the SRS document. We will use Github issue tracking for our discussion.

The SRS is an abstract document which says *what* problem is being solved, but does not say *how* to solve it. SRS will be used as a starting point for subsequent development phases, including writing the test plan and the software verification and validation plan. Review of SRS document is important to reach a common platform between software engineers and scientists. Any changes required in the software are finalized after the review of SRS. Properly reviewed SRS acts as an agreement between the scientists and the software engineers regarding the deliverables of the project.

2 Questions for reviewers

We would like all the scientists involved in this project to go through the SRS document fully, review the document and give us suggestions. However, we do understand if you cannot go through the whole document and review it.

Tasks that are marked with ** have been reviewed by Dr. Dean Inglis, and we have addressed the issues he brought to our attention.

1: ** Please let us know if the notations used in the Table of Notations are consistent with the ones usually employed in the literature. Specifically, is the notation used for the "sequence of length of q of 3D real matrices" intuitive and easy to understand? - **Section 1.2 in SRS**.

(Positive answer to this question is a part of the Generic Evidence for Consistency in our assurance for 3dfim+.)

2: Please let us know if any of the symbols used in the Table of Symbols is inconsistent with symbols usually employed in the literature.

3: ** Please read Scope of Requirements and System Context. Is the given division of responsibilities between the user and the system correct? -**Sections 2.2 and 3.1 in SRS**

(Positive answer to this question is a part of the evidence for GA in our assurance case for 3dfim+.)

4: Please go through Problem Description section and let us know if the information given in this section is adequate for teaching an undergraduate student in Science or Engineering the basics of correlation. Please let us know if any needed background information is missed. - **Section 4.1 in SRS**

5: Please go through the Assumptions and let us know if any other data assumptions should be considered while calculating the correlation coefficients. - **Section 4.2.1 in SRS**

6: ** Please read assumptions A1 to A7 and let us know if the first 5 assumptions are reasonable with respect to 3dfim+ and Pearson correlation coefficient estimation and whether A6 and A7 are reasonable with respect to 3dfim+ and Spearman and quadrant correlation coefficient estimations.

Also please let us know if any other data assumptions should be considered while calculating these correlation coefficients.

- **Section 4.2.1 Assumptions A1 to A7 in SRS**

(Positive answer to the first part of this question and/or addressing the second part of the question is a part of the evidence for Completeness (3C.Completeness.1) in our assurance case for 3dfim+.)

7: ** Please let us know if all symbols in Theoretical Model T1 are defined. Is enough information provided that you could calculate the Pearson correlation coefficient if you are given datasets A and B. - **T1 in section 4.2.2**

(Positive answer to this question is an evidence for Completeness and Correctness in our assurance case for 3dfim+.)

8: Please let us know if Theoretical Model T2 is explained clearly or needs any additional information. - **T2 in section 4.2.2**

9: Please let us know if Theoretical Model T3 is explained clearly or needs any additional information. Also can you please clarify the necessity of using quadrant correlation coefficient? If you are aware of a good reference that explains it, please let us know. - **T3 in section 4.2.2**

10: Please let us know if Data Definition DD4 (Rank Function) 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. - **DD4 in section 4.2.3**

11: Please read DD7 and let us know when we use multiple ideal signals. - **DD7 in section 4.2.3**

12: Please let us know if Data Definitions DD8, DD9 and DD10 make sense. If the definitions are not correct or are ambiguous, please provide us

with a good resource.
Moreover, in DD10, we made an assumption that the dimensions of a slice are of same size. We would like to know if this assumption is always true. - **DD8, DD9 and DD10 in section 4.2.3**

13: Please verify if Figure 11 is correct and is consistent with our definitions. - **Figure 11 in SRS**.

14: Please read Data Definitions DD11 and DD12 and let us know if these terms are explained correctly with respect to 3dfim+. Also can you please tell us how you determine the right value for *pnum* while using 3dfim+. - **DD11 and DD12 in Section 4.2.3**

15: ** Please read Data Definition DD13. Can you please explain what is the purpose of using orthogonal time series and when we use multiple orthogonal time series. If you know a good reference that explains this time series or define an equation for it, please let us know. - **DD13 in Section 4.2.3**

16: Please read Data Definitions DD14 and DD15. They are two of the values we can define with 3dfim+ commands. Do the descriptions for these Data Definitions correctly explain their usage?- **DD14 and DD15 in Section 4.2.3**

17: Please go through the Data Definitions DD16 to DD19 and let us know if they are defined correctly according to 3dfim+. Also please tell us if the symbols we use are unambiguous. - **DD16 to DD19 in Section 4.2.3 in SRS**

18: Please read Instance Models IM1, IM3 and IM4. We tried to extend Theoretical Models T1, T2 and T3, respectively, to 3dfim+. Are the inputs and outputs defined correctly?- **IM1, IM3 and IM4 in Section 4.2.4**

19: Please read Instance Model IM2 and the paragraph below it and let us know if the symbols we are using match the ones from the literature, if inputs and outputs are defined correctly and whether the description is complete and unambiguous. - **IM2 in Section 4.2.4**

20: ** Please read Instance Models IM6 to IM8 and let us know if they are defined correctly with respect to 3dfim+. Also please let us know whether you found the equations given for the outputs easy to understand. - **IM6, IM7 and IM8 in Section 4.2.4**

(Positive answers to these questions are part of the evidence for GR_Verifiable in our assurance case for 3dfim+.)

21: Please read Instance Models IM9 to IM11 and let us know if they are defined and calculated correctly with respect to 3dfim+. - **IM9, IM10 and IM11 in Section 4.2.4**

22: In Instance Model IM12, we extended the Theoretical Model T8 to fMRI dataset. Please read the description, input and output and let us know if they are correct and unambiguous. - **IM12 in Section 4.2.4**

23: Please read the table given in Functional Requirement R1. If we missed some of the 3dfim+ input parameters, please let us know. - **R1 in Section 5.1**

24: Please read the functional requirements R2 to R13 in Section 5.1. Do you feel that these requirements completely cover the functionality of 3dfim+? Do you think the requirements are atomic? That is, are each of the requirements measurable on their own and not obviously decomposable into a set of separate requirements. **- R2 to R13 in Section 5.1**

(Positive answers to these questions are part of the evidence for 3C.Completeness.1, Modifiable.3 in our assurance case for 3dfim+.)