CAS 741, CES 741 (Development of Scientific Computing Software)

Fall 2017

22 Assurance Case

Dr. Spencer Smith

Faculty of Engineering, McMaster University

November 24, 2017

Assurance Case

- Administrative details
- Feedback on MG
- Final documentation
- Questions?
- License and copyright
- Assurance cases

Administrative Details

- Course evaluation
  - Nov 23 to Dec 7
  - https://evals.mcmaster.ca
- GitHub issues for colleagues
  - Assigned 1 colleague (see `Repos.xlsx` in repo)
  - Provide at least 5 issues on their MIS
  - Grading as before
  - Due by Tuesday, Dec 5, 11:59 pm
- Today is the last "lecture"
- Next week for presentations
- Following Tuesday for Discussion

Administrative Details: Deadlines

MIS	Week 11	Nov 29
Impl. Present	Week 12	Week of Nov 27
Final Documentation	Week 13	Dec 6

Administrative Details: Presentation Schedule

- Impl. Present
  - **Tuesday: Alexander S., Steven, Alexandre P.**
  - **Friday: Jason, Geneva, Yuzhi**
- Can present anything related to the implementation
  - Code
  - Tools used
  - Testing
  - As always it is fine to show work in progress
  - Good to bring questions to the class

Feedback on MG

- Good work! Good starting point. May have to modify design as a response to MIS
- If multiple anticipated changes map to the same module, you should explain why
- If one anticipated change maps to multiple modules, you should explain why
- You don't have an input parameters module

Final Documentation

- Looking for
  - Revision of documentation
  - Consistency between documents
  - Traceability between documents - should be able to pick a requirement and trace it all the way to testing
  - Effort made to address issues and comments
  - Appropriate challenge level
- Make it easy to see changes from Rev 0
  - Specific explanation in Revision History
  - Comments in tex file

Final Documentation

- Requirements Document revised and improved
- Design Documents revised and improved
- Test Plan revised and improved
- Test Report
- Source Code

Final Documentation: Source Code

- Source code in src folder
- Comments on "what" not "how"
- Identifiers that are consistent, distinctive, and meaningful
- Avoidance of hard-coded constants (other than maybe 0 and 1)
- Appropriate modularization
- Consistent indentation
- Explicit identification of coding standards (see next slide)
- Parameters are in the same order for all functions
- Descriptive names of source code files
- Traceability to modules in module guide

Coding Style

- Having a coding standard is more important than which standard you use
- Examples
  - Google guides
    - Python
    - C++
    - Java
  - Mozilla Developer Network
  - NASA C Style Guide
- Your decisions on style may evolve over the project
- Important to be consistent

Installability and Learnability

- You can test this
- Ask a colleague to install your software
- Run it on a virtual machine, like VirtualBox
- Use a "light weight" VM like docker
- Include installation instructions (INSTALL.txt)
- Include instructions so that someone else can run your tests cases

Final Documentation

- Traceability between documents
- Look for an obvious requirement to see if it is in the requirements document and traceable through the other documents
- Installability - instructions given, makefiles etc to support, means to validate the installation, required libraries are explicitly identified
- Learnability - instructions to get someone started using the software
- Robustness - can the software handle garbage inputs reasonably
- Performance - measured if appropriate
- Usability - measured if appropriate

Final Documentation: Test Report

- Completing what you proposed in your test plan
- You do not need to repeat material from your test plan - the emphasis is not on the rational for test case selection, but on the results.
- If your test plan does not match what you are now testing, edit your test plan to "fake" a rational design process.

Test Report Continued

- Provide specific test cases
- Summarize your test results
  - Test case name
  - Initial state
  - Input
  - Expected results
  - Whether actual output matched expected
- Summarize and explain usability tests - quantify the results
- Performance tests - quantify the results
- Stress tests
- Robustness tests
- After quantification of nonfunctional tests, explain significance of results

Test Report Continued

- In cases where there are many similar tests
  - Summarize the results
  - If the expected result is obvious, you might not need to state it
  - Give an example test case, and explain how similar tests were constructed
  - If the tests were random, describe how they were selected, and how many, but not all of the details
  - Use graphs and tables
  - You need enough information that
    - Someone could reproduce your tests
    - Your test results are convincing
    - Evidence that you have used testing to improve the quality of your project

Test Report Continued

- Summarize changes made in response to test results
- Explain your automated testing set-up (if require more detail than from the test plan)
- Provide traceability to requirements
- Provide traceability to modules
- Make sure you show test results for “bad/abnormal” input

Sample Test Report Documents

- Screenholders
- 2D Physics Based Game (Uses doxygen)
- Follow given template
- Examples are not perfect
- Examples are intended to give you ideas, not to be strictly followed
- You can modify/extend the test report template as appropriate

Questions?

- Questions about MIS documentation?
- Questions about implementation presentations?

No License?

- Can others use your work if you do not include a license?
- See this link for the answer

Copyright

- Your work is automatically afforded protection by copyright law
  - Your cannot infringe on someone else's copyright
  - Must be some creativity
- Additional protection through registration with the copyright office
- Copyright does not apply to the idea, but the expression of the idea
- Trademarks and patents cover concepts and ideas
- In work for hire, copyright belongs to employer
- You can assign your copyright to someone else or a corporation

Rights

- Owner has full and exclusive rights to control who may copy or create a derivative work
- Right to sue for copyright infringement

Licensing

- Permission to others to reproduce or distribute a work
- Licenses are distinguished by the restrictions (conditions)

Proprietary License

- Copyright holder retains all rights
- Cannot copy
- Cannot use
- Cannot modify

GNU General Public License (GPL)

- Can copy the software
- Can distribute the software
- Can charge a fee to distribute the software (which will still include the license information)
- Can make modifications
- Condition – all modifications/uses are also under GPL, source code must be available
- Lesser GPL allows to link to libraries, without automatically falling under GPL conditions

GNU Questions

- Question 1
  - You modify some Linux source files to install Linux on your embedded device
  - You write software to run on this new Linux “box”
  - What software falls under the GPL?
  - Answer
- Question 2
  - You want to distribute object code compiled by gcc, where gcc is under GPL
  - Is your object code under GPL?
  - Answer

BSD and MIT

- Removes "virus" from GPL
- Can copy, distribute, charge a fee, make modifications
- Under the condition that you keep the license intact, credit the author
- Not required to disclose source
- Use at your own risk (cannot sue)

Public Domain

- Do what you want with the code
- No conditions

Copyright and License Related Links

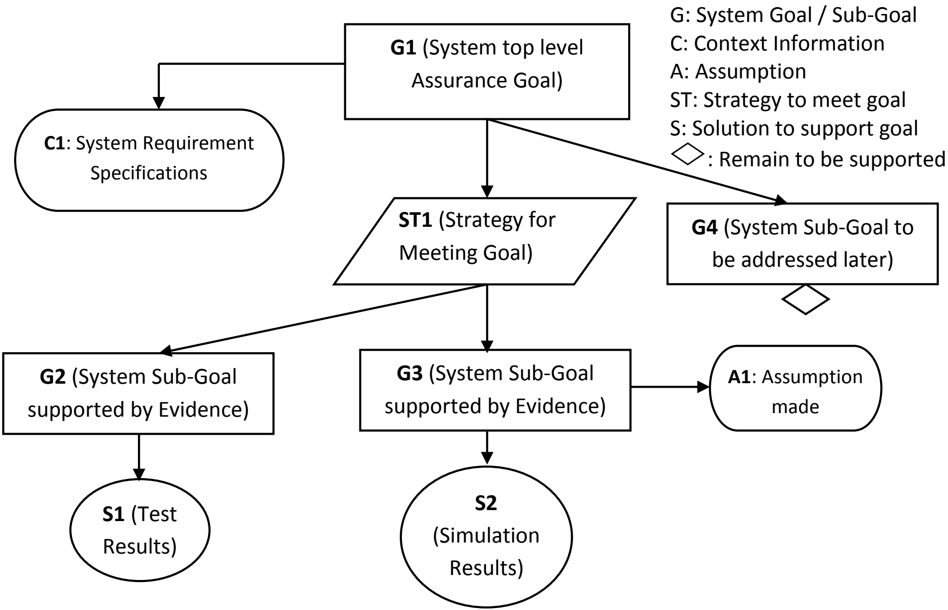
- Developer's guide to copyright law
- Summary of licenses
- Main types of licenses
- Choose a license
- Another summary
- Plain English summaries

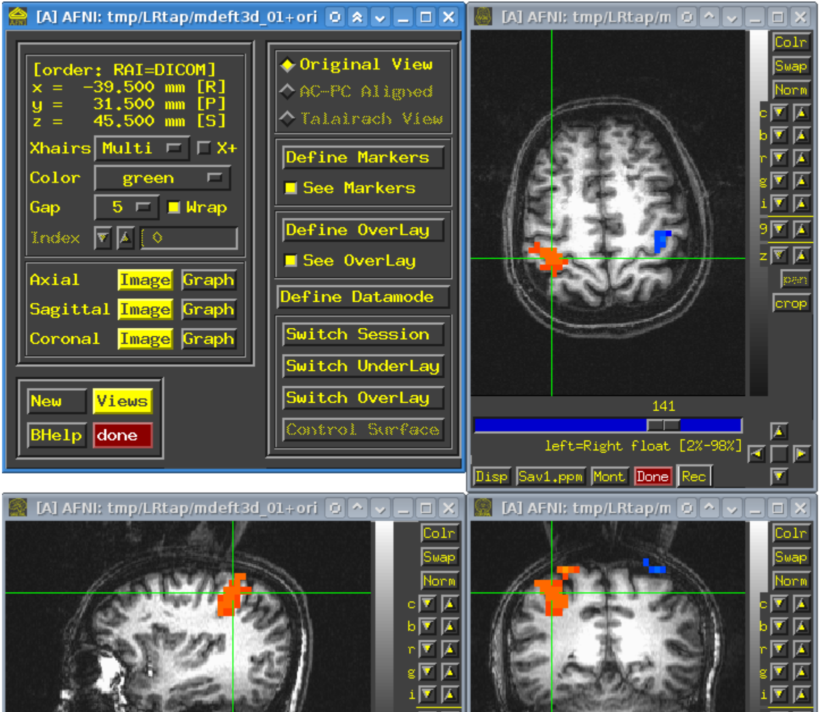
Assurance Cases in Scientific Computing [1]

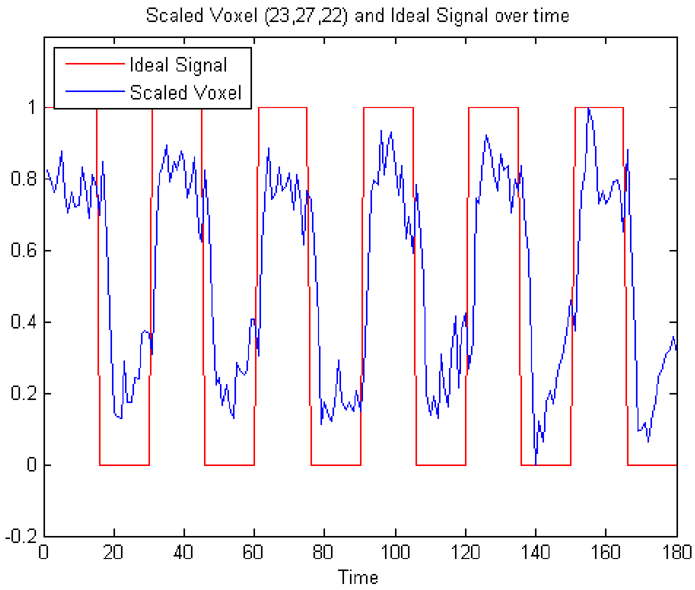
- Assurance cases
  - Organized and explicit argument for correctness
  - Successfully used for safety critical systems
- Advantages for SC
  - Engaging domain experts
  - Producing necessary and relevant documentation
  - Evidence that can be verified/replicated by a third party
- Example of 3dfim+
  - No errors found
  - However
    - Documentation ambiguities
    - No warning about parametric statistical model

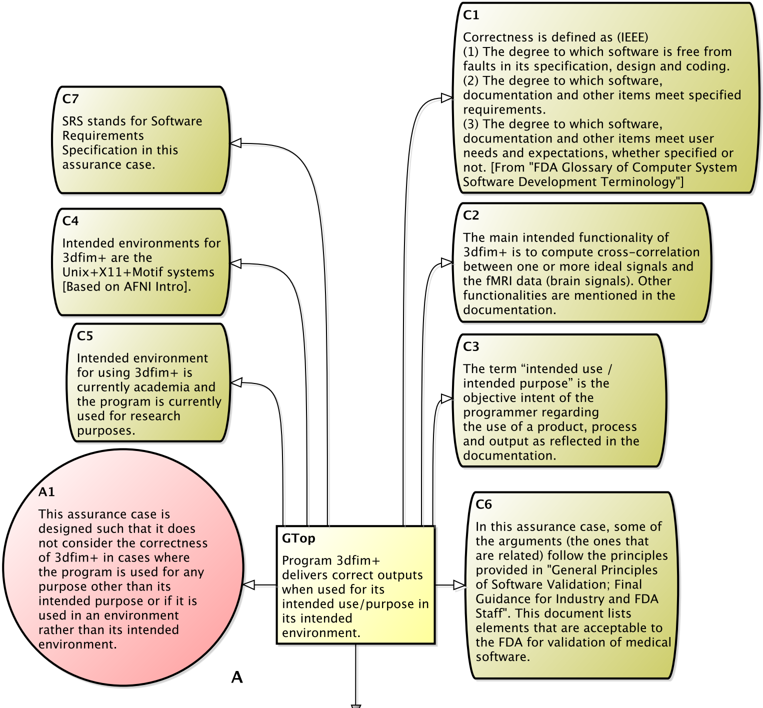
Assurance Cases in SC Motivation

- Do we put too much trust in the quality of SCS?
- Are enough checks and balances in place, especially for safety related software?
- Problems with imposing external requirements for certification
  - External body does not have expertise
  - SCS developers dislike documentation
- Solution – Assurance Cases by experts
  - Experts engaged
  - Relevant documentation
- Current techniques of development and testing still used, but arguments will no longer be ad hoc and incompletely documented

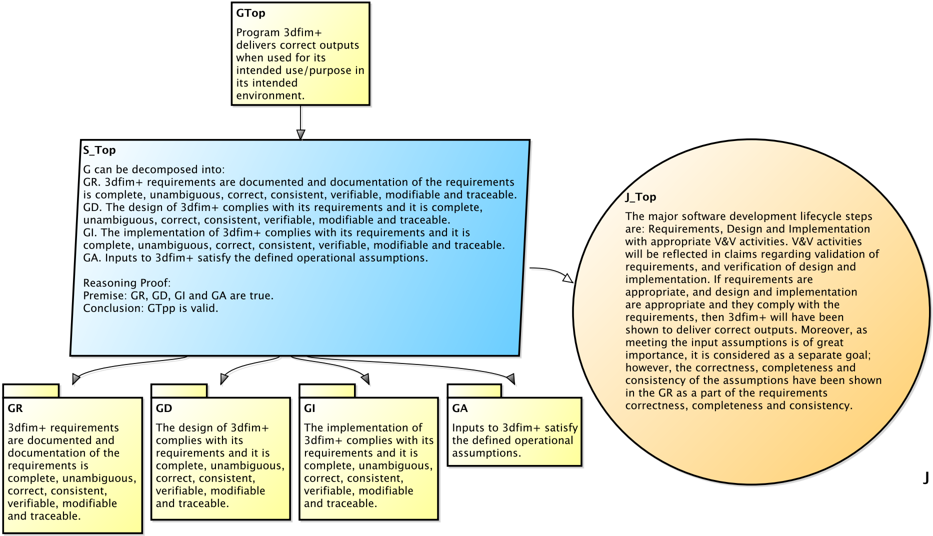

G: System Goal / Sub-Goal
C: Context Information
A: Assumption
ST: Strategy to meet goal
S: Solution to support goal
: Remain to be supported
G1 (System top level Assurance Goal)
C1: System Requirement Specifications
ST1 (Strategy for Meeting Goal)
G4 (System Sub-Goal to be addressed later)
G2 (System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

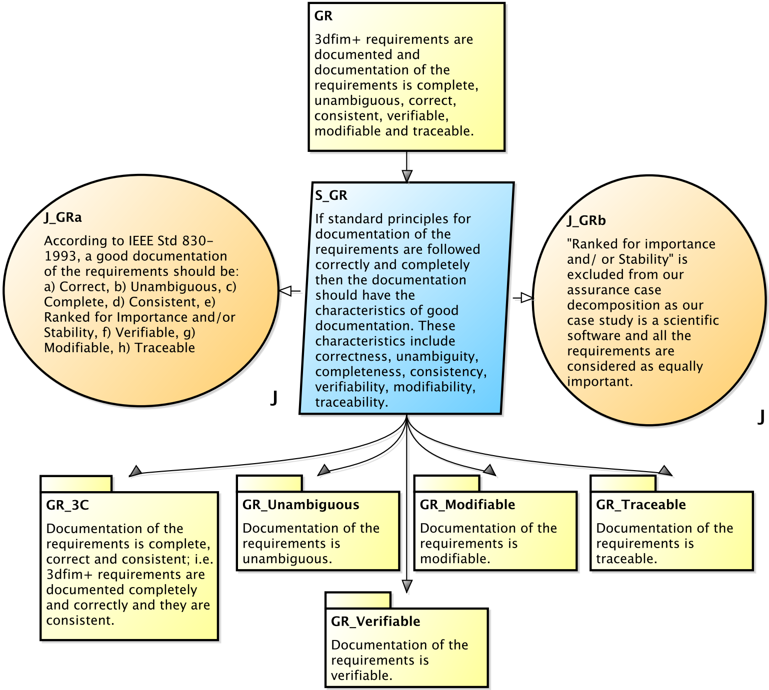

[A] AFNI: tmp/LRtap/mdeft3d_01+ori
[order: RAI=DICOM]
x = -39.500 mm [R]
y = 31.500 mm [P]
z = 45.500 mm [S]
Xhairs Multi
X+
Color green
Gap 5
Wrap
Index
Axial Image Graph
Sagittal Image Graph
Coronal Image Graph
New Views
BHelp done
Original View
AC-PC Aligned
Talairach View
Define Markers
See Markers
Define OverLay
See OverLay
Define Datamode
Switch Session
Switch UnderLay
Switch OverLay
Control Surface
[A] AFNI: tmp/LRtap/m
Colr
Swap
Norm
crop
141
left=Right float [2%-98%]
Disp
Sav1.ppm
Mont
Done
Rec

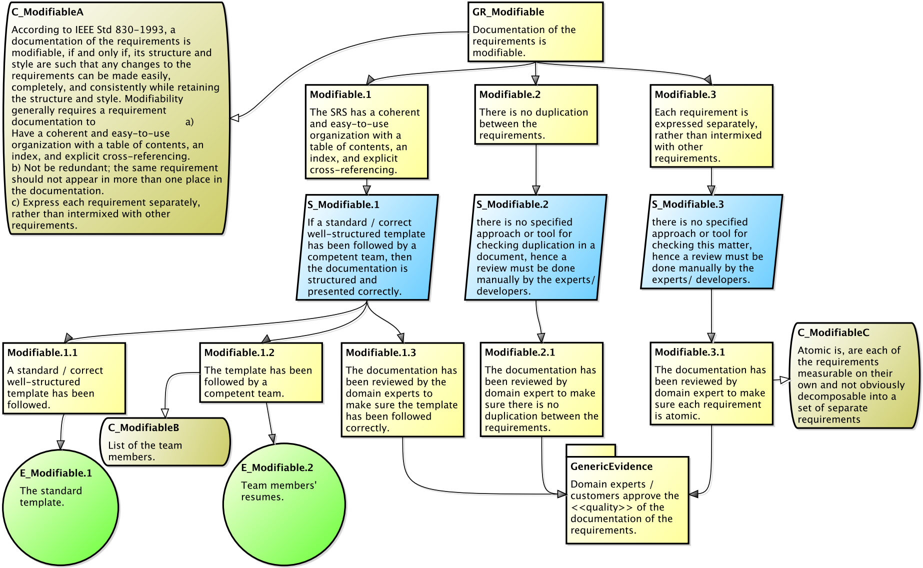

Scaled Voxel (23,27,22) and Ideal Signal over time
Ideal Signal
Scaled Voxel
1
0.8
0.6
0.4
0.2
0
-0.2
0
20
40
60
80
100
120
140
160
180
Time

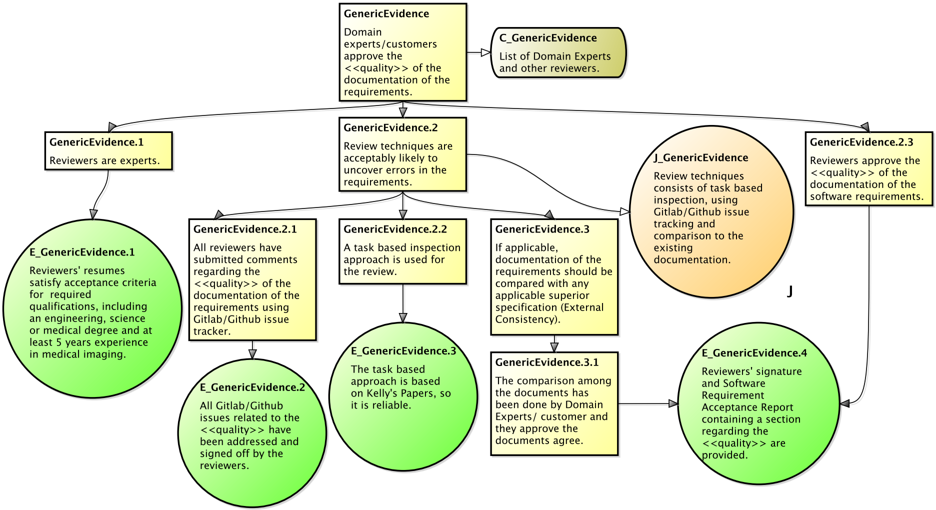

C7
SRS stands for Software Requirements Specification in this assurance case.
C4
Intended environments for 3dfim+ are the Unix+X11+Motif systems [Based on AFNI Intro].
C5
Intended environment for using 3dfim+ is currently academia and the program is currently used for research purposes.
A1
This assurance 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.
GTop
Program 3dfim+ delivers correct outputs when used for its intended use/purpose in 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.
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.

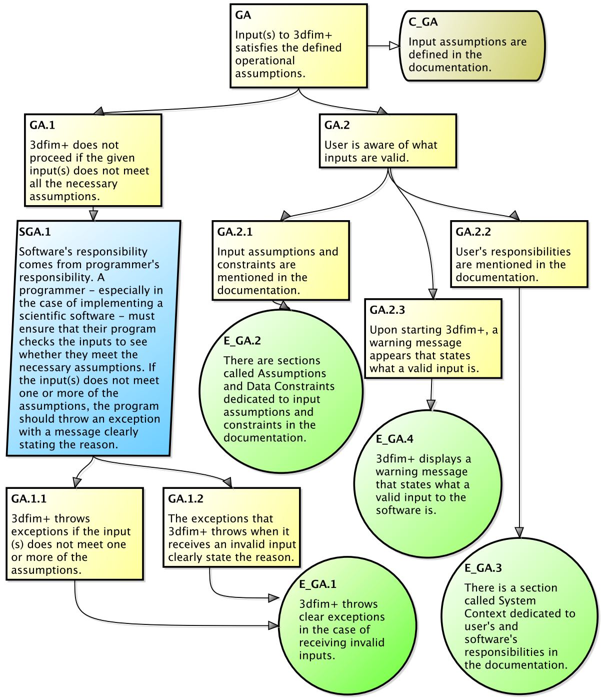
A


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: GTpp is valid.
J_Top
The major software development lifecycle steps are: Requirements, Design and 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 is of great importance, it is considered as a separate goal; however, the correctness, completeness and consistency of the assumptions have been shown in the GR as a part of the requirements correctness, completeness and consistency.
J
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.


GR
3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
S_GR
If standard principles for documentation of the requirements are followed correctly and completely then the documentation should have the characteristics of 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
J_GRb
"Ranked for importance and/ or Stability" is excluded from our assurance case decomposition as our case study is a scientific software and all the requirements are considered as equally important.
J
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_Modifiable
Documentation of the requirements is modifiable.
GR_Traceable
Documentation of the requirements is traceable.
GR_Verifiable
Documentation of the requirements is verifiable.


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.
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
C_ModifiableB
List of the team members.
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.


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 satisfy acceptance criteria for required qualifications, including an engineering, science or medical degree and at least 5 years experience in medical imaging.
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.


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.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.
E_GA.3
There is a section called System Context dedicated to user's and software's responsibilities in the documentation.

Proposed Changes to 3dfim+

- No mistakes found in calculations
- Goal of original software was not certification
- Problems found
  - GR goal not satisfied
    - Not complete, verifiable, modifiable or traceable
    - Coordinate system information missing
    - Ambiguous rank function
  - Inputs not checked in code
  - User not informed of their responsibility to use tool with correct statistical model

Concluding Remarks

- Hopefully motivated assurance cases for SC
- Quality is improved by looking at a problem from different perspectives, assurance cases provide a systematic and rigorous way to introduce a new perspective
- An assurance cases will likely use the same documentation and ideas used in CAS 741
- However, an assurance case can focus and direct efforts right from the start of the project

References I

W. Spencer Smith, Mojdeh Sayari Nejad, and Alan Wassyng.
Assurance cases for scientific computing software.
Prepared for ICSE Submission, November 2017.