Assurance Cases for Scientific Computing Software

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

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

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

ABSTRACT

Assurance cases, which provide an organized and explicit argument for correctness, should be used for certifying Scientific Computing Software (SCS), especially when the software impacts health and safety. Assurance cases have already been effectively used for safety cases for real time systems. Their advantages for SCS include engaging domain experts, producing only necessary documentation, and providing evidence that can potentially be verified/replicated by a third party. This paper illustrates assurance cases for SCS through the correctness case for 3dfim+, an existing medical imaging application. No errors were found in 3dfim+. However, the example still justifies the value of assurance cases, since the existing documentation is shown to have ambiguities and omissions, such as an incompletely defined ranking function and missing details on the coordinate system convention adopted. In addition, a potential concern for the software itself is identified: running the software does not produce any warning about the necessity of using data that matches the assumed parametric statistical model.

CCS CONCEPTS

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

KEYWORDS

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

ACM Reference format:
Spencer Smith, Mojdeh Sayari Nejad, and Alan Wassyng. 2018. Assurance Cases for Scientific Computing Software. In *Proceedings of 40th International Conference on Software Engineering Companion, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE '18 Companion),* 2 pages.
https://doi.org/10.1145/3183440.3195037

1 INTRODUCTION

Are we currently putting too much trust in the quality of Scientific Computing Software (SCS)? Given its role in such fields as nuclear safety analysis and medical imaging, SCS has a significant impact on health and safety related planning and decision making. Although SCS developers do excellent work, do we have enough checks and balances in place for confidence in correctness? Two significant problems exist for SCS in completing a conventional correctness certification exercise through an external body:

- The external body often does not have deep expertise on the problem the software solves, leading to requests for large quantities of documentation, as shown in standards for SCS in the nuclear and medical domains [3, 4].
- SCS developers tend to dislike documentation. As observed by Carver [2], scientists do not view rigid, process-heavy approaches, favourably.

A potential solution to these two problems is to have the SCS developers create an assurance case as they develop their software. Assurance case techniques have been developed and successfully applied for real time safety critical systems [7, 8]. An assurance case presents an organized and explicit argument for correctness (or whatever other software quality is deemed important) through a series of sub-arguments and evidence. Putting the argument in the hands of the experts means that they will work to convince themselves, along with the regulators. They will use the expertise that the regulators do not have; they will be engaged. Significant documentation will still likely be necessary, but now the developers control the documentation.

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

2 EXAMPLE ASSURANCE CASE FOR 3DFIM+

While the eventual goal is developing a template for assurance cases for any SCS, our initial approach is to learn by first building an assurance case for one particular example. Our case study focuses on 3dfim+, a medical image analysis software package that supports Functional Magnetic Resonance Imaging (fMRI). 3dfim+ was selected because it is a reasonably small (approximately 1700 lines of code) and easy to understand example of medical image analysis software. We targeted medical image analysis software

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195037

because a recent study [5] has shown a potentially serious flaw in software commonly used to analyze fMRI data.

To argue for the correctness of 3dfim+, we developed an assurance case and a Software Requirements Specification (SRS) document that contains all the necessary information and mathematical background needed to understand 3dfim+. This document can be used for validation and verification activities, and appears many times as evidence in our assurance case. The SRS was reviewed by a domain expert to provide further evidence. We also developed a test case to illustrate how the results from 3dfim+ can be checked to provide additional evidence of correctness. The full assurance case for 3dfim+ can be found in Nejad 2017 [6].

In our case study assurance case we have defined our top goal as "Program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment." Following one of the standard approaches for assurance cases, and a medical device assurance case [8], we divided the top goal into four sub-goals. The first sub-goal (GR) argues for the quality of the documentation of the requirements, since to make an overall argument for correctness, we need a specification to judge correctness against. The second sub-goal (GD) says that the design complies with the requirements. The third proposes that the implementation also complies with the requirements and the fourth sub-goal (GA) claims that the inputs to 3dfim+ will satisfy the operational assumptions, since we need valid input to make an argument for the correctness of the output.

A convincing argument, along with supporting evidence, needs to be built for each of the four sub-goals. As an example, the decomposition of GR into its sub-goals is shown in Figure 1. This decomposition is based on the IEEE standard 830-1993 [1]. This standard states that good documentation of requirements should be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable and traceable. "Ranked for importance and/or stability" is excluded from the sub-goals in Figure 1 because our domain is SCS. For 3dfim+ to work, all of the requirements are considered to be of equal importance. This is shown as justification J_GRb in Figure 1.

Each of the sub-goals for the argument for GR must be argued in turn, until we reach the bottom of the full argument, where evidence is provided. Evidence consists of concrete facts and documents that can be unambiguously judged. Examples include the following: "a standard template is used," "team member resumes show competence," "reviewers sign off on the requirements," etc.

3 CONCLUDING REMARKS

Besides providing a means to illustrate assurance cases for SCS, the 3dfim+ example provides an opportunity to justify their value for certification. Although no errors were found in the output of the existing software, the rigour of the proposed approach did lead to finding ambiguities and omissions in the existing documentation, such as an incompletely defined ranking function and missing details on the coordinate system convention adopted. In addition, a potential concern for the software itself was identified: running the software does not produce any warning about the obligation of the user to provide data that matches the parametric statistical model employed for the correlation calculations. This is found when trying to build the argument (GA) that the inputs satisfy

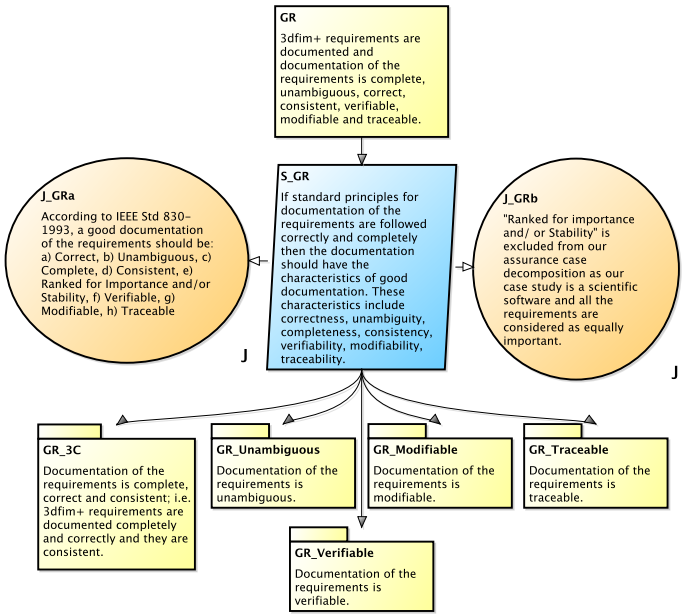

Figure 1: GR decomposition

operational assumptions. This case study demonstrates that an assurance case facilitates SCS certification by providing an explicit argument, artifacts and evidence that can be independently judged.

4 ACKNOWLEDGMENTS

The assistance of Dr. Michael Noseworthy, from St. Joseph's hospital and McMaster University, is gratefully acknowledged. Thanks also to Dr. Dean Inglis for fulfilling the role of expert reviewer.

REFERENCES

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

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

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

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

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

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

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

[8] Alan Wassyng, Neeraj Kumar Singh, Mischa Geven, Nicholas Proscia, Hao Wang, Mark Lawford, and Tom Maibaum. 2015. Can Product-Specific Assurance Case Templates Be Used as Medical Device Standards? *IEEE Design & Test* 32, 5 (2015), 45–55. https://doi.org/10.1109/MDAT.2015.2462720