Chapter 3

Assurance Cases and Selected Evidence for AortaGeomRecon

In this chapter, we discuss the scope of our work and the assurance case for the correctness of AortaGeomRecon. To provide the necessary evidence for the assurance case the following material is also presented: the Software Requirements Specification, the Design Document, the Module Guide, the test cases, and the algorithm review.

3.1 Assurance Case Development

An Assurance Case is build with claims, subclaims, contexts and the evidences to ensure the goals are fulfilled. By using Astah System Safety presenting in Goal Structuring Notation (GSN) arguments [2][9], we want to show that our software delivers correct outputs when used for its intended use/purpose in its intended environment, and within its assumed operating assumptions. The Figure 3.1 shows the top level of the assurance cases. Having the goal of delivering correct outputs, we are decomposing

the goal into 4 sub goals: GR, GI, GA and GBA, where GR stands for the goals to accomplish for the requirements of the software, GI stands for the goals of the implementation, GA is the goals of inputs satisfy the operational assumptions, and GBA are the goals of defining operational assumptions.

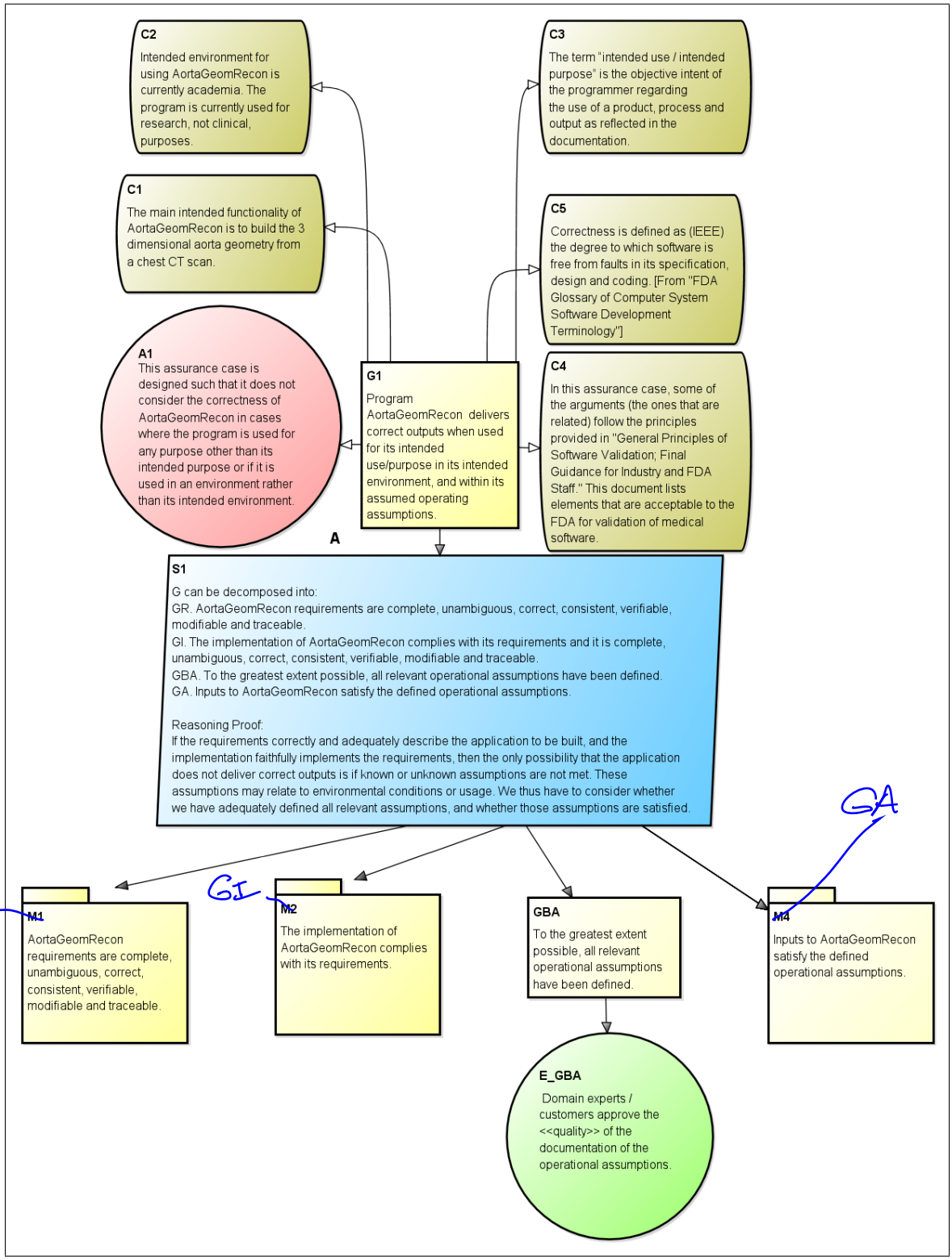

Figure 3.1: AortaGeomRecon Assurance Cases Top Level

3.2 Assurance Case for Software Specification Requirements

The first goal of getting a trusted software is having a complete, unambiguous, correct, consistent, verifiable, modifiable and traceable Software Specification Requirements [13] ~~which~~ that shows the complete breakdown of the requirements with mathematical notation, data models and instance models. ~~It's~~ The SRS is the foundation of the software development, and the design and the implementation will be based on the requirement document.

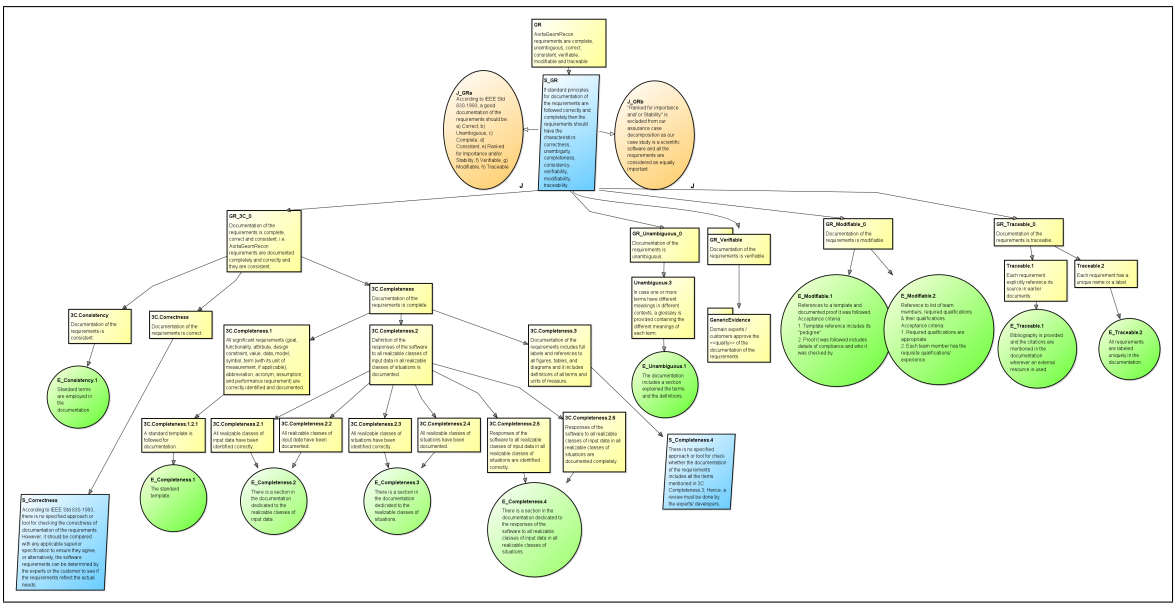

Figure 3.2: AortaGeomRecon Assurance Cases GR

As explains in the assurance case GR, one of most important statement of SRS having these characteristics is using a standard template. ~~This template is provided by the domain expert, and~~ my professor, Dr. Spencer Smith, who has ~~researched on Application of Software Engineering Principles, Methods, Techniques and Tools to~~ We used a template tailored for ~~scienti~~ research software. (Smith 2006, SmithEtAl2007, SmithLaiCai2005) Call bib entries at github.cas.mcmaster.ca/smithr/pub

improve the Sustainability of Research Software.

The chapters in SRS and some of the most important sections in the chapter are explained below:

- Reference Material

  In this section, a Table of Symbols and an Abbreviations and Acronyms table are used to explain every symbol and Abbreviations used in the SRS document. These tables ensure the consistency and the unambiguous characteristics of the document. They are located at the very beginning of the document, so the reader will firstly look at these tables before reading the entire document.

- Introduction

  In the introduction section, we introduced the problems and the scope of the document to the user by explaining the purpose of document, abstracting the scope of requirements and defining the characteristics of intended reader.

- General System Description

  The general system description includes a system context diagram which explains the relationship between the users, the inputs given by the user and the outputs of the AortaGeomRecon program. User responsibility and AortaGeomRecon responsibility are defined such that the reader knows what to do to successfully generate the desired result.

- Specific System Description

  In this section, we are presenting more details about the problem and the specific system to solve the problem. The first subsection Problem Description discussed

on the definition of Organ Segmentation, Coordinate Systems used in medical image problem, Physical System Description which is not available, and Goal Statements which is extract the three-dimensional segmentation of the aorta.

In the next subsection, Solution Characteristics Specification, we started with Assumptions to define clearly the scope of the requirement document. In the subsection Data Definitions, we defined Voxel, Image/Slice, and Volume with mathematical notation so that the developer can easily interpret. Next, in the subsection Instance Model, we showed the mathematical meaning of Region of Interest, and Segmentation, which are the two essential models that the developer must know in order to develop the solution.

4.2 Solution Characteristics Specification

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], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: The 3D image provided by the user must contain a visually distingushable aorta volume [IM1].

A2: User should select a valid region of interest [IM2].

A3: User should input a singular volume (3 dimensional image) even if the data format supports the 4th dimension (time) [IM1].

Figure 3.3: AortaGeomRecon SRS Assumptions

- Requirements

With all the information in the document, we can now present the Functional

Requirements and the Non-Functional Requirements for the program AortaGeomRecon. The Functional Requirements are defined By using the terms we presented in Data Definitions, Instance Model, and based on the other Functional Requirements. The Non-Functional Requirements usually have a measurement such as execution time, the effort of manual works, etc.

5.1 Functional Requirements

R1: Input the following functions, data and parameters:

symbol	description
V	CT Scans volume (DD3)
$Seed\_a$	The seed of ascending aorta centre coordinate (IM2)
$Seed\_d$	The seed of descending aorta centre coordinate (IM2)

R2: Use the volume in R1 to create a second volume, the region of interest (IM1) that contains all voxels of the aorta.

R3: Perform segmentation (IM2) on the volume created in R2.

R4: Visualize a volume (DD3).

Figure 3.4: AortaGeomRecon Functional Requirements

5.2 Nonfunctional Requirements

NFR1: **Usability** AortaGeomRecon allows a user that meets the user characteristics (Section 3.2) to import any DICOM files, input the required parameters, and begin the segmentation effortlessly. The number of steps it takes using AortaGeomRecon should be at least 30% less than the number of steps it takes by using ITK-Snap (bubble method mentioned in Section 2).

NFR2: **Safety** For a valid image, the AortaGeomRecon provides a correct solution, or no answer.

NFR3: **Learnability** The user interface and documentation should allow a user that meets the user characteristics (Section 3.2) to learn how to do an aorta segmentation in at least 30% of the time it takes to learn and use ITK-Snap (bubble method mentioned in Section 2).

NFR4: **Accuracy** For a given image the segmentation found by AortaGeomRecon should match that found by an expert using ITK-Snap. Whether to two segmentations match is something that would be judged by a medical imaging expert.

NFR5: **Consistency** The coordinate system may be modified through the calculations, but any transformations will not alter the meaning of the data.

Other NFRs that might be discussed in the future include verifiability, and reusability.

Figure 3.5: AortaGeomRecon Non- Functional Requirements

- Likely Changes and Unlikely Changes

  This section discussed the likely changes that the developer might expect a change in the future works, and the unlikely changes that is most certainly not going to change for a justified reason. The only likely change discussed in the AortaGeomRecon's SRS is regarding the segmentation method. For different

segmentation method, the inputs varies, since the segmentation method is a likely change, the inputs variables are also likely changes. The only unlikely change is the method of retrieve a region of interest. Most methods take a starting point and sizes in different dimensions to get the region of interest.

- Traceability Matrix and Graphs

  The traceability matrices are to provide easy references on what has to be additionally modified if a certain component is changed. Below shows the traceability matrices of different sections:

	DD1	DD2	DD3	IM1	IM2
DD1					
DD2	X				
DD3		X			
IM1			X		
IM2				X	

Table 2: Traceability Matrix Showing the Connections Between Items of Different Sections

Figure 3.6: AortaGeomRecon Traceability Matrix between Data Definitions and Instance Model

	IM1	IM2	R1	R2	R3	R4	NFR1	NFR2	NFR3	NFR4	NFR5
IM1				X							
IM2					X						
R1		X									
R2	X										
R3		X									
R4							X				
NFR1			X	X	X	X			X		
NFR2		X									
NFR3				X	X	X	X				
NFR4		X									
NFR5		X									

Table 3: Traceability Matrix Showing the Connections Between Requirements and Instance Models

Figure 3.7: AortaGeomRecon Traceability Matrix between Requirements and Other sections

	A1	A2	A3
DD1			
DD2			
DD3			X
IM1	X		X
IM2		X	X
LC1	X	X	X
UC1			X

Table 4: Traceability Matrix Showing the Connections Between Assumptions and Other Items

Figure 3.8: AortaGeomRecon Traceability Matrix between Assumptions and Other sections

3.3 Assurance Case for Implementation

The goal of implementation asked the developer to fully comply the design and the implementation with the Software Requirements Specification document. Since we have showed with Assurance Case that our SRS is complete, consistent, and unambiguous, the design and the implementation that fully complying with requirements is correct.

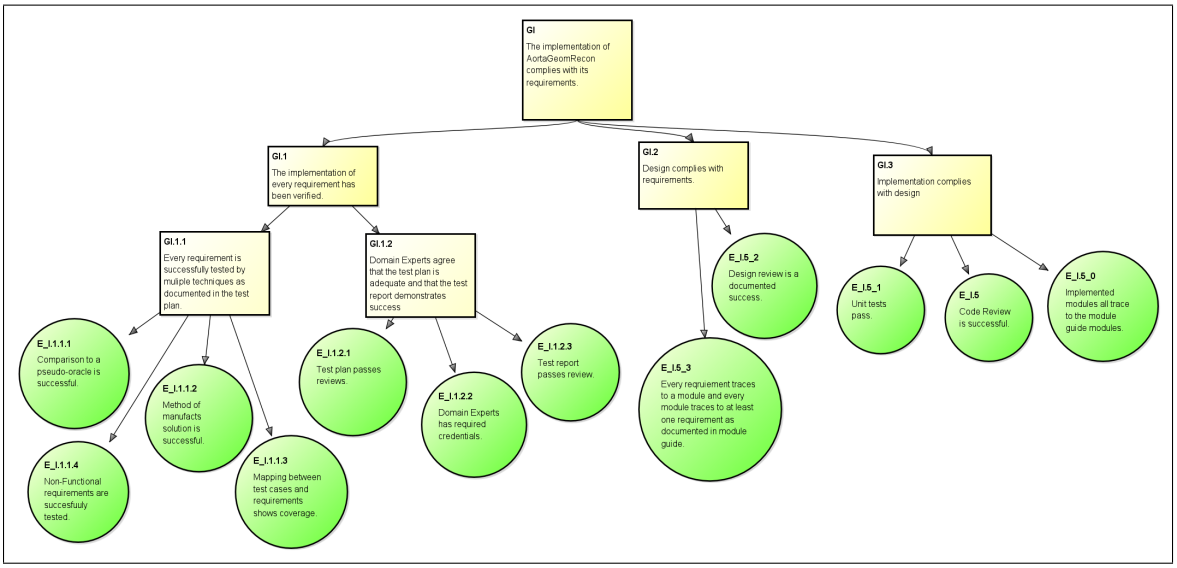

Figure 3.9: AortaGeomRecon Assurance Cases for Implementation

3.3.1 Design Document

The purpose of the Design Document [12] is to explain in details how the algorithm works, and why it worked. Similar to what section 2.2.3 wrote, the design document explains in plan text the workflow of the algorithm. The design document is a piece of evidences that demonstrate unambiguity characteristic.

3.3.1.1 Sphinx - Python Documentation Generator

To implement this Design Document, I used Sphinx, a Python Documentation Generator that can build module's documentation with the comments in the source code. Moreover, using reStructuredText to write the Algorithm Overview, we can build HTML binary which can be published on a web server. Another important section in Design Document is the Glossary. It has rich vocabulary explanation, images, and links to the outside source to let the reader understands as much as possible.

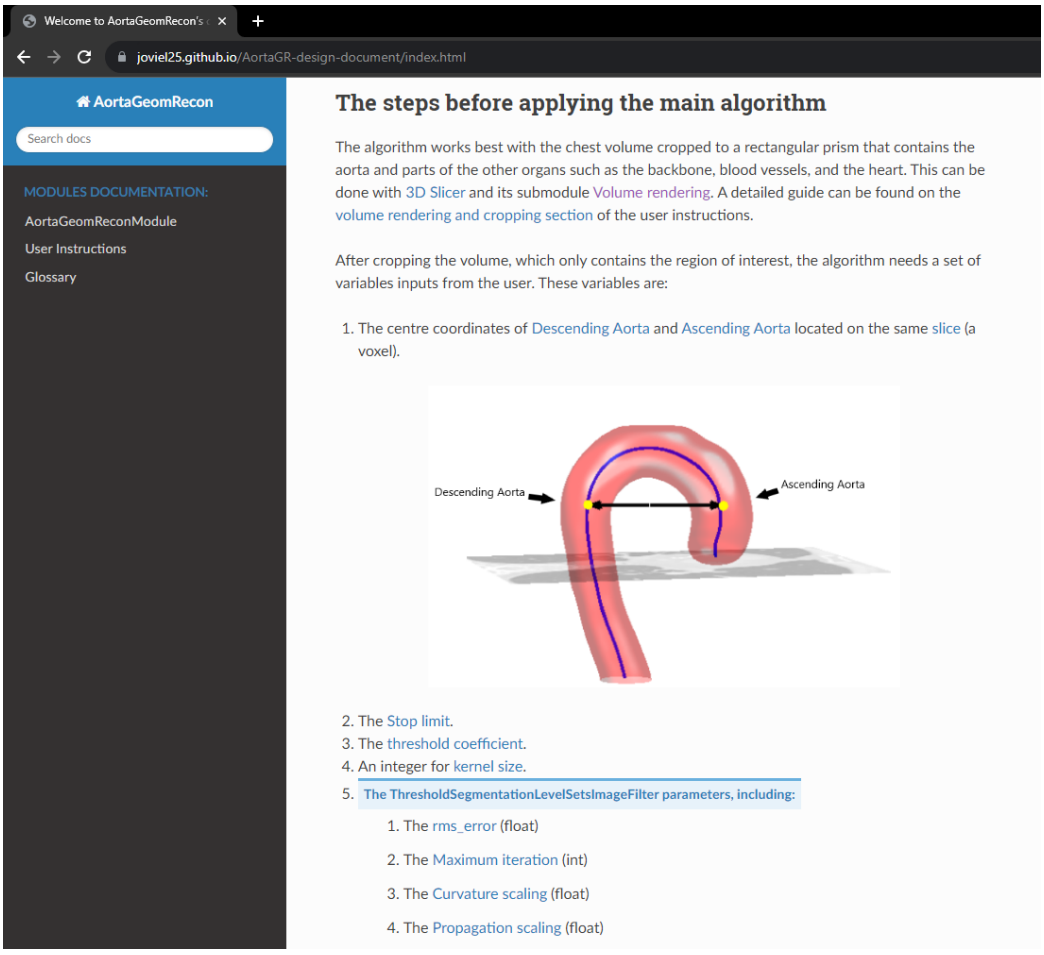

Figure 3.10: AortaGeomRecon Design Document website

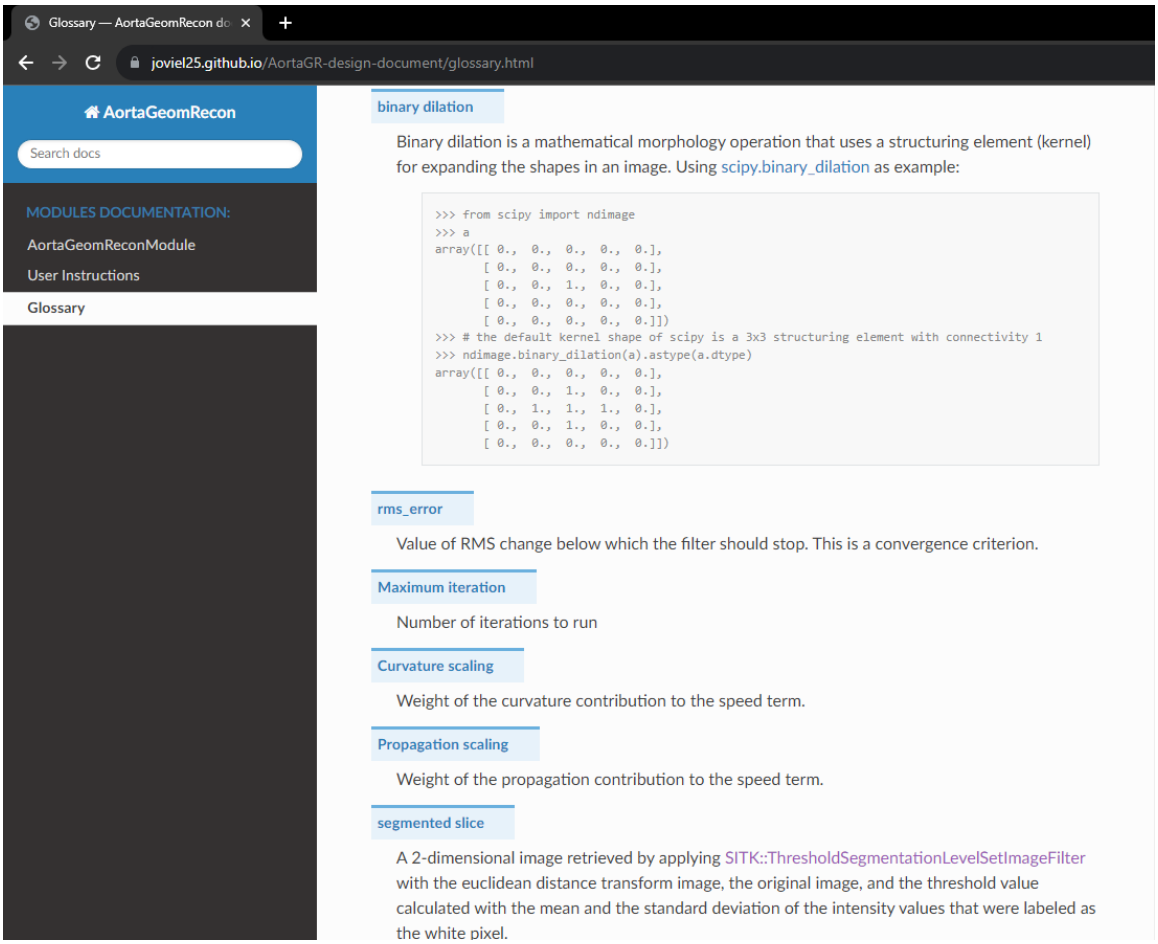

Figure 3.11: AortaGeomRecon Design Document Glossary

3.3.2 Module Guide

Another important documentation to show that the design is complete, correct, and consistent design is Module Guide. In the design of the software, we first list the anticipated changes such that we are expecting a change to this piece of requirement of information, we will keep it a single module so when the changes did happen, we only need to concern about the module and its dependency modules. The anticipated changes are listed in the Figure 3.12 below.

AC1: The specific hardware on which the software is running.

AC2: The format of the initial input data.

AC3: The algorithm to segment the aorta.

AC4: The data structures to store the input parameters required to execute the algorithm.

AC5: The methods to create a user interface.

AC6: The methods to retrieve a region of interest.

AC7: The methods to visualize a volume.

AC8: How the overall control of the calculations is orchestrated.

AC9: The format of the final output data.

Figure 3.12: AortaGeomRecon Anticipated Changes

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). The Secrets' field in a module decomposition is a brief statement of the design decision hidden by the module. The Services' field specifies what the module will do without documenting how to do it. For each module, a suggestion for the implementing software is given under the Implemented By title. If the entry is OS, this means that the module is provided by the operating system or by standard programming language libraries. AortaGeomRecon means the module will be implemented by the AortaGeomRecon software.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Input Format Module

M3: Input Parameter Module

M4: Control Module

M5: GUI Module

M6: Volume Visualization Module

M7: Crop Volume Module

M8: Aorta Segmentation Module

M9: Image Processing Module

M10: Multidimensional Array Processing Module

M11: Digital Enhancement Module

Figure 3.13: AortaGeomRecon Modules

7.2.2 Input Parameter Module (M3)

Secrets: The data structure for input parameters, how the values are input and how the values are verified. The load and verify secrets are isolated to their own access programs.

Services: Gets input from user, stores input and verifies that the input parameters comply with physical and software constraints. Throws an error if a parameter violates a physical constraint. Throws a warning if a parameter violates a software constraint. Stored parameters can be read individually, but write access is only to redefine the entire set of inputs.

Implemented By: AortaGeomRecon

Source: AortaSegmenter class' atributes

7.2.3 Control Module (M4)

Secrets: The algorithm for coordinating the running of the program.

Services: Provides the main program's entry point, the ability to jump from a program state to another.

Implemented By: AortaGeomRecon

Source: AortaGeomReconDisplayModuleWidget module

7.2.4 Volume Visualization Module (M6)

Secrets: The methods which allow users to visualize a 3D Volume.

Services: Display the aorta images and vtk 3D geometry.

Implemented By: 3D Slicer

Figure 3.14: AortaGeomRecon Module Decomposition Example

Now that we have listed the anticipated changes and the modules, we are using traceability matrices to show the relationships between the modules and the anticipated changes, and the modules between the requirements. This indicates that the design is fully complying with the requirements, as we stated in GI.2 in the Figure 3.9.

Req.	Modules
R1	M1, M2, M3, M4
R2	M3, M7
R3	M8, M9, M10
R4	M6
NFR1	M3, M4, M5
NFR2	M4, M8, M9, M10
NFR3	M3, M4, M5 M6, M7, M8
NFR4	M7, M8, M9, M10
NFR5	M3

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M2
AC3	M8
AC4	M3
AC5	M5
AC6	M7
AC7	M6
AC8	M4
AC9	M9, M10

Table 3: Trace Between Anticipated Changes and Modules

Figure 3.15: AortaGeomRecon Modules Traceability Matrices

On top of relating the modules to the requirements, we are relating the actual source code to the modules, which is a strong evidence of our implementation has

fully complying with the requirements. Since our requirements has proven to be correct, complete, and consistent, the implementation must also be trustworthy.

M6 Volume Visualization Module	3D Slicer's Volume Rendering Module 3D Slicer's Volume Rendering Source Code
M7 Crop Volume Module	3D Slicer's Crop Volume Module 3D Slicer's Crop Volume Module Source Code
M8 Aorta Segmentation Module	AortaSegmenter class
M9 Image Processing Module	SimpleITK
M10 Multi-Dimensional Array Processing Module	NumPy
M11 Digital Enhancement Module	

```
# AortaGeomReconDisplayModule.py
# https://github.com/smiths/aorta/blob/main/src/SlicerExtension/
    AortaGeometryReconstructor/AortaGeomReconDisplayModule/
    AortaGeomReconDisplayModule.py#L739-L769
    def transform_image(self, cropped_volume):
        """
        Histogram Equalization for Digital Image Enhancement.
        https://levelup.gitconnected.com/introduction-to-histogram-
            equalization-for-digital-image-enhancement-420696db9e43
        """
        cropped_image = sitkUtils.PullVolumeFromSlicer(cropped_volume)
        img_array = sitk.GetArrayFromImage(
            (sitk.Cast(sitk.RescaleIntensity(cropped_image), sitk.
                sitkUInt8)))
        histogram_array = np.bincount(img_array.flatten(), minlength=256)
        num_pixels = np.sum(histogram_array)
        histogram_array = histogram_array/num_pixels
        chistogram_array = np.cumsum(histogram_array)
        transform_map = np.floor(255 * chistogram_array).astype(np.uint8)
        img_list = list(img_array.flatten())
        eq_img_list = [transform_map[p] for p in img_list]
        eq_img_array = np.reshape(np.asarray(eq_img_list), img_array.
            shape)
        eq_img = sitk.GetImageFromArray(eq_img_array)
        eq_img.CopyInformation(cropped_image)
        median = sitk.MedianImageFilter()
        median_img = sitk.Cast(median.Execute(eq_img), sitk.sitkUInt8)
        self._cropped_image = median_img
```

Table 4: Trace Between Modules and Code

Figure 3.16: AortaGeomRecon Part of the Traceability Matrix on Modules and Code

3.3.3 Test Plan

GI.1.2 claim states that a test plan must be adequate and test report demonstrates success. Unlike the other algorithm that can easily be tested with a ground truth test case, our ground truth case is build by using another more accurate method such as ITK-Snap's bubble method, then manually erase the unwanted pixels.

3.3.3.1 Build "Ground Truth" data

Instead of using ITK-Snap's bubble method, we build ground truth test case with our existing algorithm and updated algorithm. If there exists a difference between two results, our domain experts can make a visual comparison of both test cases to see which algorithm to use for the future.

3.3.3.2 GitHub Actions workflows

This leads to our Continuous Integration infrastructure, implemented with GitHub Actions workflow. A workflow is a configurable and automated process that will run one or more jobs on the desired system. GitHub Actions workflow used a YAML file to define the events and the commands to be executed on the temporary system, which has the build of the repository. [7]

We have set up two automated process which happens on each "push" event and "pull" event. A "push" event implies that something is changed in one or multiple commits, therefore there is a need to verify whether the commits have bugs that need extra fixes. A "pull" event happens when a feature branch is going to merge with the main branch. Since our main branch is protected, any update to the main branch must be merged by using a pull-request. Before a pull-request can be approved, the

continuous integration tests are examined and until there are no errors, a pull-request cannot be merged with the main branch.

The first automated process is a linter. A linter is a tool for static code analysis to flag programming errors, bugs, stylistic errors and suspicious constructs. We used Python Flake8 as our linter to find bugs and errors, and ensures that program's readability by striking the source code with Google's published Python Style Guide. [17]

The second automated process is our continuous integration tests. By setting up Git Large File System (LFS) and upload to pre-build ground truth test data in the repository, we can now pass the cropped volume as the inputs' data, the same Aorta seeds and the hyperparameters that we have used to generate the ground truth test data to the algorithm and verify the results. To compare the volumes, we used Dice similarity coefficient.

3.3.4 Algorithm Review

The algorithm review was an idea started with Code Walkthrough. A code walkthrough is one of the methods that can increase the participants' confidence, or finding program's bugs or error. For an Algorithm Review, we have not discussed what the program should do; we are presenting the algorithm to the domain expert and asking them if the design and implementation fulfill the implementation objectives.

3.3.4.1 Tool used in Algorithm Review

Spyder is a free and open source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. The Variable Explorer

allows the user to interactively browse the variables and the objects in debugging mode. [**?**]

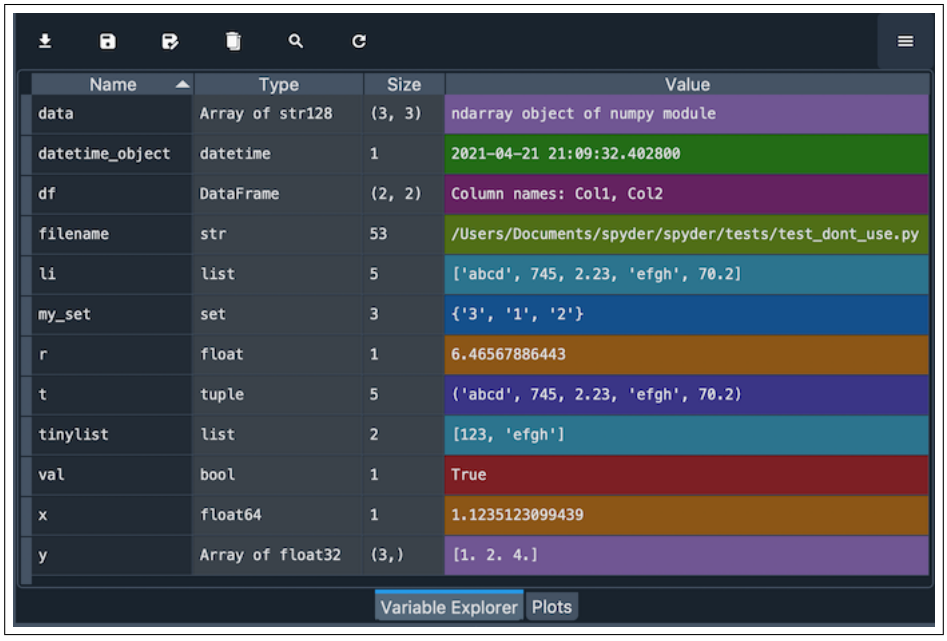

Figure 3.17: Spyder Variable Explorer [16]

This feature allows us to execute the program step by step, and see what happens to the variable (segmentation result) when executing the segmentation algorithm.

3.3.4.2 Algorithm Review with Kailin Chu

The first algorithm review was done with Kailin Chu, who is a biomedical engineering student and started working the semi-automacial aorta segmentation algorithm as a summer researcher. Along with Smith Spencer, we were aiming to increase our confidence in this code walkthrough. This code walkthrough did not increase our confidence in the software, because the code was developed by Kailin from two years ago, so some details and design decisions are missing, and some variables were decided

by trial and error. Despite that the code walkthrough has turned into an algorithm review, and it did not achieve what we wanted in the first place, this meeting was still very helpful. Knowing that the algorithm was partially based on trial and error, I was able to improve the algorithm and reaching a better result in a more efficient way.

3.3.4.3 Algorithm Review with Dr. Dean Inglis

The second algorithm review was conducted with Dr. Dean Inglis, an experienced professor, Medical Image Analyst, and Software Developer. We presented our segmentation algorithm to him and requested validation of our approach or suggestions for a potentially superior algorithm. Dr. Dean Inglis provided his insights on the algorithm, which we meticulously recorded on the GitHub issue tracker. These insights will guide the developer responsible for enhancing the program. This meeting significantly reinforced our confidence in both the software and our endeavors, whether the idea of developing an extension module for 3D Slicer or the algorithm itself.

3.4 Assurance Case for Operational Assumptions

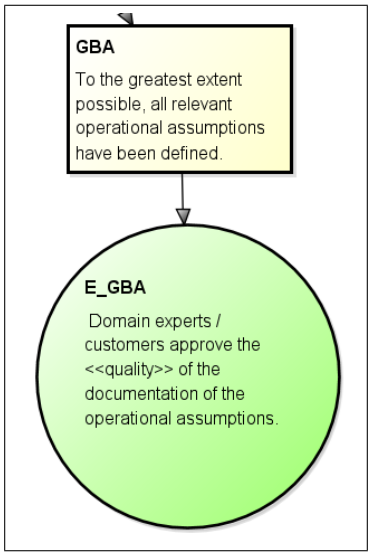

Figure 3.18: AortaGeomRecon Assurance Case Operational Assumptions

The evidence for the statement "To the greatest extent possible, all relevant operational assumptions have been defined" is quite simple as only Domain experts/customers approve the quality of the documentation. However, finalizing this evidence takes many efforts from the beginning of the project to the end of the project, because we want to continuously improve the quality of the content matching the most recent updates of the software.

3.4.1 User Manual

A user manual serves the purpose of documenting the all operational assumptions. When the user getting unexpected results by using this software, they can always refer to the user manual to see what pieces are different. Our user manual is initially located in GitHub repo's README, which is only available to the developers invited

as the contributors. The content includes the installation of the software, importing the extension modules, import inputs data, and perform segmentation. The user manual is also available publicly on the design document website, assuming that the users might not be the repository contributors.

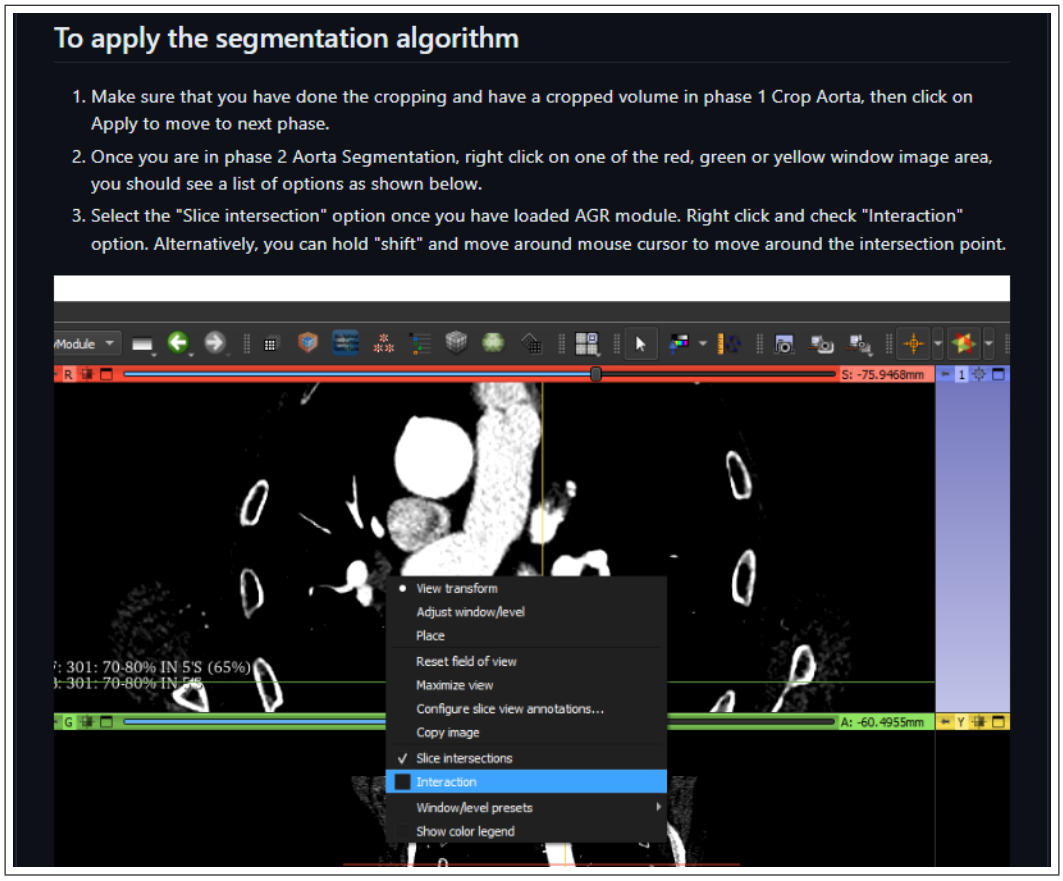

Figure 3.19: AortaGeomRecon User Manual on GitHub README

3.4.2 User Instruction Video

Videos are an effective way to engage your audience and deliver information in a way that's easy to follow along and understand. A better instructional content is a YouTube Video where I make step-by-step instruction with voice over to instruct

user. The video is not listed publicly on YouTube, but the users who have access to the GitHub repository or Design Document website can access this video by the URL link.

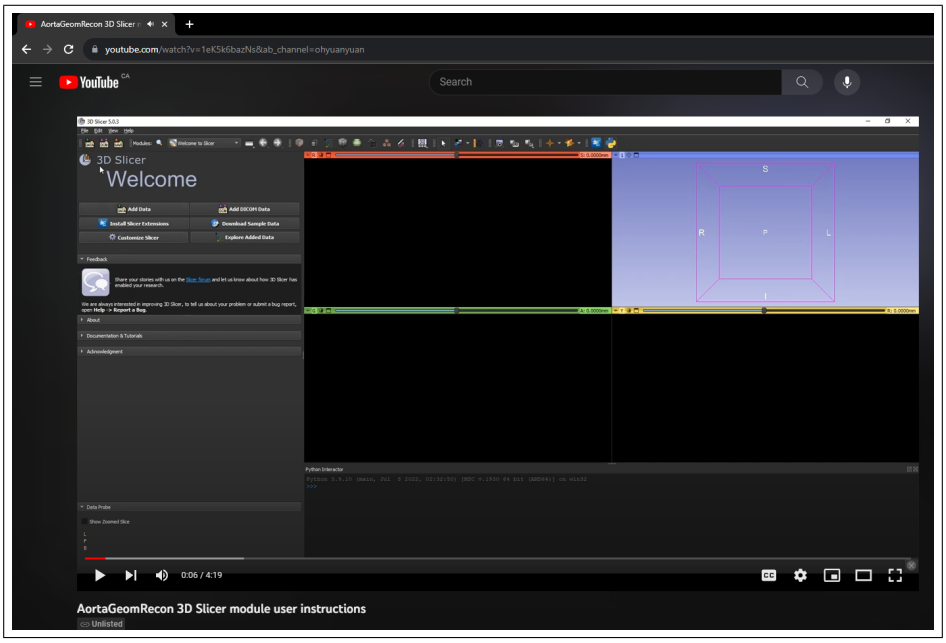

Figure 3.20: AortaGeomRecon User Instructions on YouTube

3.5 Assurance Case for Inputs Assumptions

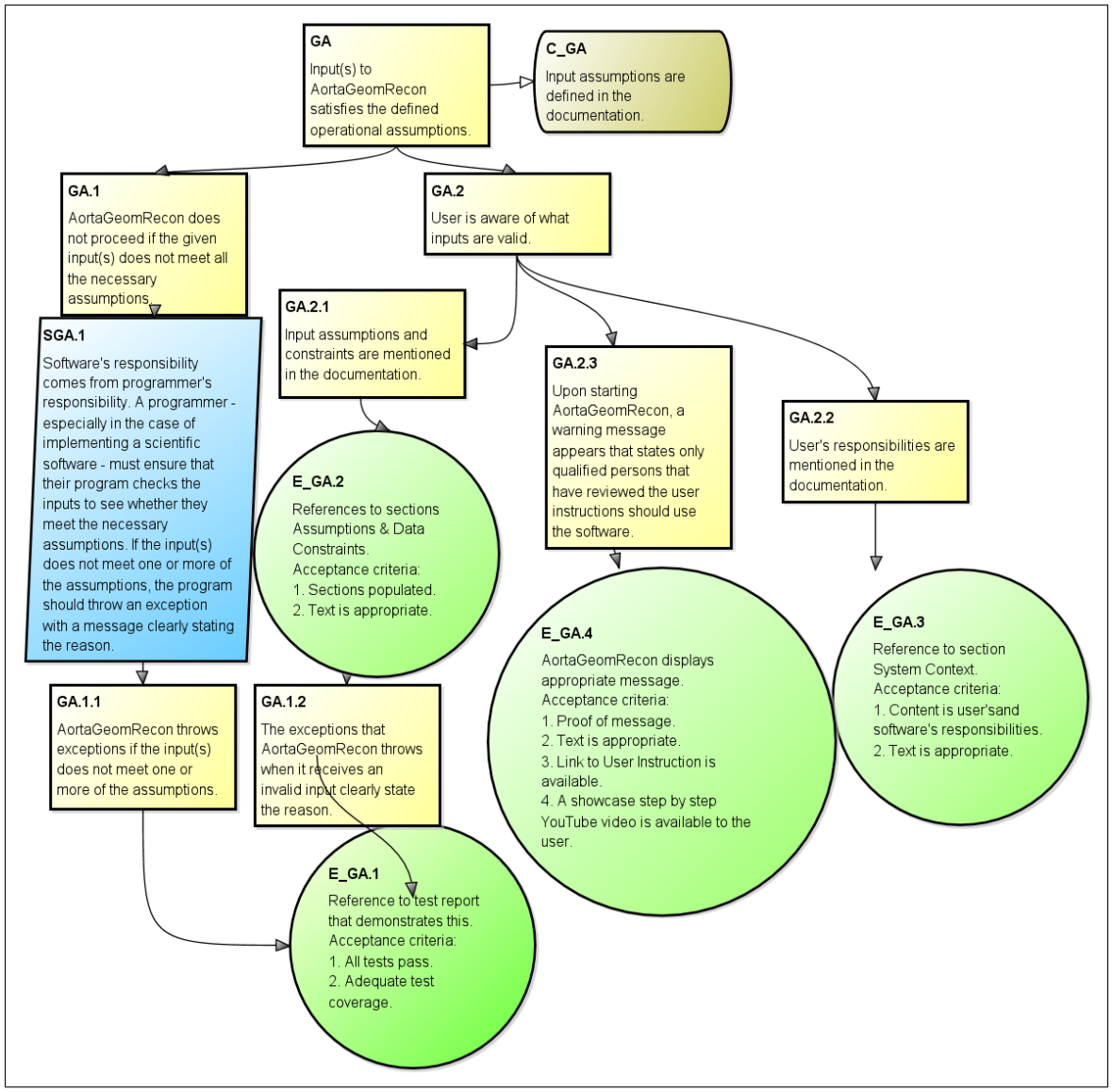

Figure 3.21: AortaGeomRecon Assurance Case Inputs Assumptions

This statement requires the user know what inputs are valid, and only used the valid inputs in the software.

3.5.1 Warning Message

As we initially planned, this piece of information is available in the User Manual and User Instruction Video, where we showed the user how to import DICOM patient's data, and operate on the inputs' data till we get a segmentation result. A user who has read the User Manual and watched the instruction video should know what inputs are valid. Therefore, in the AortaGeomRecon module, we need to effectively guide the user to the User Manual, whether the user has used this software before or it is a first time user.

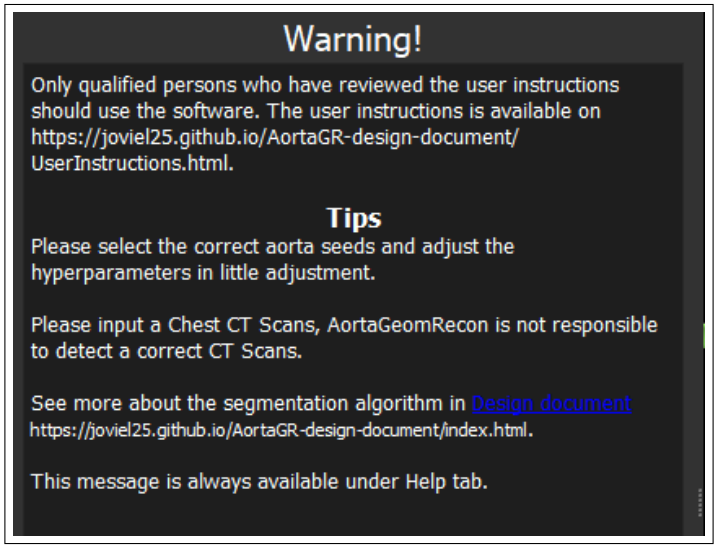

Figure 3.22: AortaGeomRecon Warning Message

As mentioned in the section 2.3.3.2, when the user first starting 3D Slicer and click on the AGR module, this warning message appears. The user must clicks on the Confirm button to continue to the next steps. With the warning message shown to the user, it is now the user's responsibility to use the valid inputs for AGR, which the program will deliver the correct outputs if the other operations are performed correctly.