Software Design Document
for
Image Analysis and Geometry Reconstruction

Version 1.1

Prepared by Nicol Barrios, Ralph Belleca, Ted Kim, Silvano Medina, Robin Mok, Alejandra Olvera, Demetrius Parker, Sabino Ramirez, Mary Semerdjian, Jason Tejada

Dr. Mathias Brieu (Biomedical)

December 09, 2021

Revision History

Name	Date	Reason For Changes	Version
Silvano	10/08/2021	Update Document	1.0
Silvano	12/09/2021	Update Document	1.1
Mary	12/09/2021	Update Document	1.1

1. Introduction

1.1 Purpose

The purpose of the research project is to create an application that will provide an efficient and convenient option to generate 3D anatomical models from various sets of DICOM (MRI) images. This is the first version of the design. The scope of the project is to create a web application that takes DICOM (MRI) images of the pelvic system as input.

1.2 Document Conventions

The project is following a simple methodology: segmentation and refinement. After taking an MRI image as input, the application will use segmentation to the region of interest and generate a 3D model. Lastly, the model will undergo refinement which includes smoothing and repair procedures. The priority of the project is to simplify the traditional process of manual thresholding.

1.3 Intended Audience and Reading Suggestions

The document is mainly written for developers as well as doctors and other healthcare professionals. It will include technical terminologies as well as diagrams and tables that will be easy to comprehend to the general public.

1.4 System Overview

The goal of the project is to create a web application that does two things: create a high quality 3D anatomical model from an MRI image through the use of segmentation and smoothing tools, provide a user-friendly interface that is easy to navigate for first-time users.

2. Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

2.1 Assumptions and Dependencies

Describe any assumptions or dependencies regarding the software and its use. These may concern such issues as:

- Related software or hardware
- Operating systems
    - Windows
    - MacOS
    - Linux
- End-user characteristics
    - Understanding what the user wants when using the site
        - Make it as easy as possible for the user to use and understand in terms of human anatomy
- Possible and/or probable changes in functionality

2.2 General Constraints

Describe any global limitations or constraints that have a significant impact on the design of the system's software (and describe the associated impact). Such constraints may be imposed by any of the following (the list is not exhaustive):

- Hardware or software environment
    - Application software that has access to the internet or Wi-Fi
        - Runs on cloud web server
- End-user environment
    - Up-to-date web browser
        - Security updates
- Availability or volatility of resources
    - MRI scans for testing and AI training
- Interface/protocol requirements

- Data repository and distribution requirements
    - Cost of using:
        - Google Collab
        - NVIDIA
- Security requirements (or other such regulations)
- Memory and other capacity limitations
- Performance requirements
- Network communications
    - The user has to have a stable internet connection
- Verification and validation requirements (testing)
- Other means of addressing quality goals
- Other requirements described in the requirements specification

You will not need to include all of these. Only the ones that will influence the design of your software

2.3 Goals and Guidelines

Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system's software. For each such goal or guideline, unless it is implicitly obvious, describe the reason for its desirability. Feel free to state and describe each goal in its own subsubsection if you wish. Such goals might be:

Goals

- The KISS principle ("Keep it simple stupid!")
    - Please keep it simple, where the user can understand it easily
- Emphasis on speed versus memory use
    - Two critical uses, the user should understand
- The product should work, look, or "feel" like an existing product

Guidelines

- Follow an uncomplicated and straightforward guideline
- Make it easy to use for the user

2.4 Development Methods

Briefly describe the method or approach used for this software design. If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of these methods. If several methods were seriously considered, then each such method should be mentioned, along with a brief explanation of why all or part of it was used or not used.

The liaison required us to use 3D Slicer and Blender. Now, using 3D Slicer was our main goal because we could analyze the shape of the female pelvis and its key organs. We were able to identify different female pelvis parts by using DICOM MRI files. After analysis and implementation, we then found our key organ. Last, we would export the STL file from 3D Slicer and import the STL file to Blender software.

The team followed the female-pelvis-anatomical approach for this project. They hosted client bi-weekly meetings on Tuesdays and weekly meetings every Friday.

3. Architectural Strategies

Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

- Use of a particular type of product (programming language, database, library, etc. ...)
- Reuse of existing software components to implement various parts/features of the system
- Future plans for extending or enhancing the software
- User interface paradigms (or system input and output models)
- Hardware and/or software interface paradigms
- Error detection and recovery
- Memory management policies
- External databases and/or data storage management and persistence
- Distributed data or control over a network
- Generalized approaches to control
- Concurrency and synchronization
- Communication mechanisms
- Management of other resources
    - 3D Slicer is used to make 3D models of three organs: the vagina, bladder, rectum, and pelvic bone.
    - We decided to use the Nvidia plugin, open-source medical image visualization, and analysis application. This plugin has three methods: fully automatic segmentation, boundary point-based segmentation, and deep grow segmentation.
        - We concluded that boundary point-based segmentation was the best method to use for this project due to the time it takes to create a model and its smoothness.
    - We plan to make models of three organs: the bladder, vagina, rectum, and pelvic bone.
    - The language used will be python.

Each significant strategy employed should probably be discussed in its own subsection. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose).

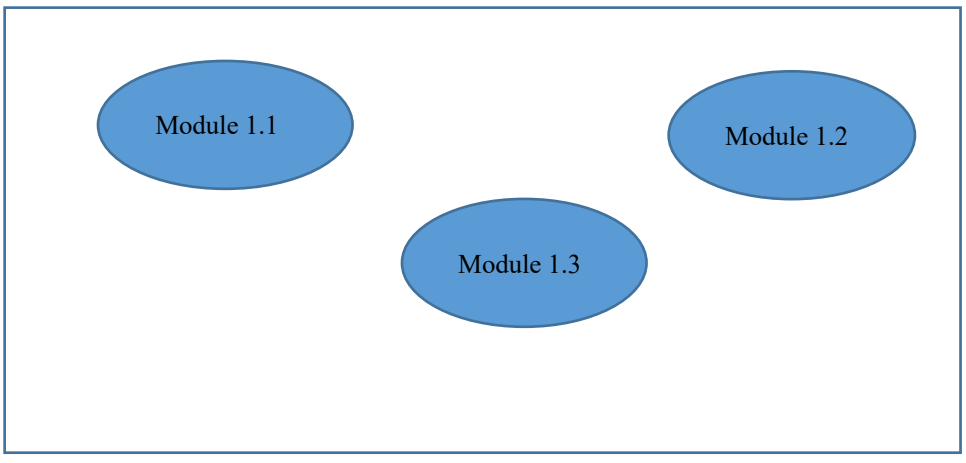
4. System Architecture

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.

This is where the level 0 DFD will probably work best.

At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its modules/components/subsystems (identifying each top-level modules/component/subsystem and the roles/responsibilities assigned to it).

Each subsection (i.e. “4.1.3 The ABC Module”) of this section will refer to or contain a detailed description of a system software component.



Level 1 DFD

Level 1 Data Flow Diagrams (DFD) and Control Flow Diagrams (CFD) should probably go here.

Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them. Diagrams that describe a particular component or subsystem in detail should be included within the particular subsection that describes that component or subsystem.

5. Policies and Tactics

Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). Such decisions might concern (but are not limited to) things like the following (Must include 5.1, 5.2, and 5.3. The rest of these categories or custom ones can be added as needed.):

5.1 Choice of which specific products used

- IDE
    - PyCharm
- Software
    - 3D Slicer
    - Blender
    - Python
    - Jupyter Notebook
- Programming Language
    - Python
- Import/Export Files
    - STL
    - IPYNB

5.2 Plans for ensuring requirements traceability

We will be using GitHub because it's fast and accessible. Every time one of the group members makes an "update," the other group members will know and have access to it.

5.3 Plans for testing the software

The plan for testing the software is to aim for professionalism. The goal is to combine AI and human anatomy together basically; that way, we will have more extended accessibility to it.

5.# Engineering trade-offs

…Describe…

5.# Coding guidelines and conventions

…Describe…

5.# The protocol of one or more subsystems, modules, or subroutines

…Describe…

5.# The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

…Describe…

5.# Plans for maintaining the software

…Describe…

5.# Interfaces for end-users, software, hardware, and communications

…Describe…

5.# Hierarchical organization of the source code into its physical components (files and directories).

…Describe…

5.# How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

…Describe…

5.# Describe tactics such as abstracting out a generic DatabaseInterface class, so that changing the database from MySQL to Oracle or PostGreSQL is simply a matter of rewriting the DatabaseInterface class.

For this particular section, it may become difficult to decide whether a particular policy or set of tactics should be discussed in this section, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. You will have to use your own "best" judgement to decide this. There will usually be some global policies and tactics that should be discussed here, but decisions about interfaces, algorithms, and/or data structures might be more appropriately discussed in the same (sub) section as its corresponding software component in one of these other sections.

6. Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

This is where Level 2 (or lower) DFD's will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.

6.x Name of Component (Module)

6.x.1 Responsibilities

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

6.x.2 Constraints

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

6.x.3 Composition

A description of the use and meaning of the subcomponents that are a part of this component.

6.x.4 Uses/Interactions

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass's, and metaclasses.

6.x.5 Resources

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers,

databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

6.x.6 Interface/Exports

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.

7. Detailed Lower level Component Design

Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. “7.1.3 The ABC Package”.) Note that there isn't necessarily a one-to-one correspondence between packages and components.

7.x Name of Class or File

7.x.1 Classification

The kind of component, such as a subsystem, class, package, function, file, etc.

7.x.2 Processing Narrative (PSPEC)

A process specification (PSPEC) can be used to specify the processing details

7.x.3 Interface Description

7.x.4 Processing Detail

7.x.4.1 Design Class Hierarchy

Class inheritance: parent or child classes.

7.x.4.2 Restrictions/Limitations

7.x.4.3 Performance Issues

7.x.4.4 Design Constraints

7.x.4.5 Processing Detail For Each Operation

8. Database Design

Include details about any databases used by the software. Include tables and descriptions.

9. User Interface

The user interface is the application, from the point of view of the users. Do your classes and their interactions (the logical and process views) impose restrictions on the user interface? Would removing some of these restrictions improve the user interface? Use some form of user interface flow model to provide an overview of the UI steps and flows. Don't go into too much refinement. You should include screen shots or wireframe layouts of significant pages or dialog elements. Make sure to indicate which of the system level modules or components that each of these user interface elements is interacting with.

9.1 Overview of User Interface

Describe the functionality of the system from the user's perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback Information that will be displayed for the user. This is an overview of the UI and its use. The user manual will contain extensive detail about the actual use of the software.

9.2 Screen Frameworks or Images

These can be mockups or actual screenshots of the various UI screens and popups.

9.3 User Interface Flow Model

A discussion of screen objects and actions associated with those objects. This should include a flow diagram of the navigation between different pages.

10. Requirements Validation and Verification

Create a table that lists each of the requirements that were specified in the SRS document for this software.
For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.
For each entry describe the method for testing that the Requirements Validation and Verification ment has been met.

11. Glossary

An ordered list of defined terms and concepts used throughout the document. Provide definitions for any relevant terms, acronyms, and abbreviations that are necessary to understand the SDD document. This information may be listed here or in a completely separate document. If the information is not directly listed in this section provide a note that specifies where the information can be found.

3D Slicer - a free and open-source software package for image analysis and scientific visualization.

Anatomy - the study, classification, and description of structures and organs of the body.

Artificial Intelligence (AI) - It is the science and engineering of making intelligent machines, especially intelligent computer programs.

Automate – to convert to automatic operation.

Bladder – a membranous sac serving as a receptacle for secretions, such as the gallbladder.

Blender - a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, motion graphics, interactive 3D applications, virtual reality, and computer games.

Biology – the scientific study of life.

DICOM (digital imaging and communications in medicine) - the standard used for the electronic transferring of digital image data.

Dimension - a measure of the width, length, or height of a space, usually described in units of a linear scale.

Dummy data - used as a placeholder for both testing and operational purposes.

Female - pertaining to the sex that has the ability to become pregnant and bear children; feminine.

Geometry - a branch of mathematics that deals with the measurement, properties, and relationships of points, lines, angles, surfaces, and solids.

Google collab - allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more.

Image smoothing - a key technology of image enhancement, which can remove noise in images.

Image thresholding - the simplest method of image segmentation, that replace each pixel in an image with a black pixel, if the image intensity is less than than some fixed constant, or a white pixel if image intensity is greater than the constant.

MRI (Magnetic Resonance Imagining) - medical imaging based on the resonance of atomic nuclei in a strong magnetic field and it has also become an important tool in musculo- skeletal and pelvic imaging. Page 1081 from Mosby Medical Dictionary

Nvidia - designs graphics processing units for the gaming and professional markets, as well as system on a chip units for the mobile computing and automotive market.

Nvidia AIAA - enables you to serve the models you trained to speed up annotation process.

Patient - a recipient of a health care service.

Pelvis – the lower part of the trunk of the body, composed of four bones, the two innominate bones laterally and ventrally and the sacrum and coccyx posteriorly.

Organ – a structural part of the system of the body that is composed of tissues and cells that enable it to perform a particular function.

Rectum – the lower part of the large intestine, about 12 cm long, continuous with the descending sigmoid colon, proximal to the anal canal.

Render - the process of generating a photorealistic or non-photorealistic image from a 2D or 3D model by means of a computer program.

Rendering - the process involved in the generation of a two-dimensional or three-dimensional image from a model by means of application programs.

Standard Triangle Language (STL) - is a file format native to the stereolithography CAD software created by 3D Systems and is widely used for rapid prototyping, 3D printing and computer-aided manufacturing.

Two Dimension (2D) – having only two dimensions, such as width, length, or height.

Three Dimension (3D) – having only three dimensions, such as width, length, or height.

UI (User Interface) - the point of human-computer interaction and communication on a device, webpage, or app.

Vagina – the part of the female genitalia that forms a canal from the orifice through the vestibule to the uterine cervix.

12. References

<List any other documents or Web addresses to which this SDD refers. These may include other SDD or SRS documents, user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Community, B. O. (2018). *Blender - a 3D modelling and rendering package*. Stichting Blender Foundation, Amsterdam. Retrieved from http://www.blender.org

Kikinis R, Pieper SD, Vosburgh K (2014) 3D Slicer: a platform for subject-specific image analysis, visualization, and clinical support. Intraoperative Imaging Image-Guided Therapy, Ferenc A. Jolesz, Editor 3(19):277–289 ISBN: 978-1-4614-7656-6 (Print) 978-1-4614-7657-3 (Online)

Mosby's Medical Dictionary. 9th ed. St. Louis, MO: Mosby Elsevier, 2013. Print.