Team Penobscot

Seismic-radar toolbox GUI

System Design Document

Dr. Chris Gerbi

Team Penobscot
Adam Farrington, Alex Thacker, Jens Hansen, Nathan Gazey

November 12, 2019

Table of Contents

1. Introduction

1.1 Purpose of This Document

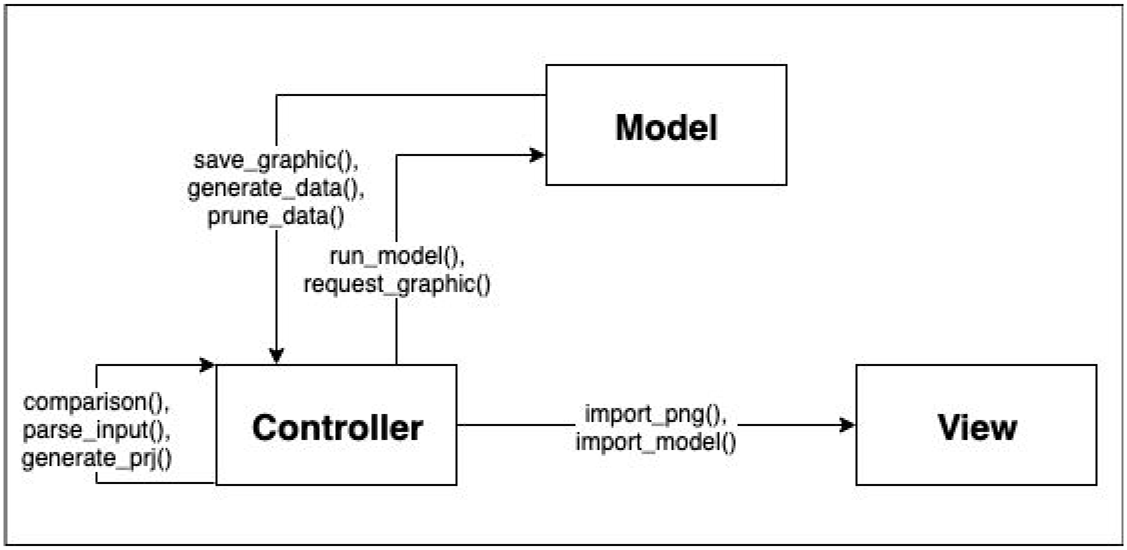
This is the formal agreement between Dr. Gerbi and Team Penobscot that describes in detail the design of the system being developed by Team Penobscot. This project is a user interface for a software system, SeidarT, currently accessed through command lines only. Included in this document are the proposed system architecture, plans for persistent data, and a summarizing table comparing system requirements which the corresponding architectural components. The remainder of this document is used to describe the plans for how this interface will be designed.

1.2 References

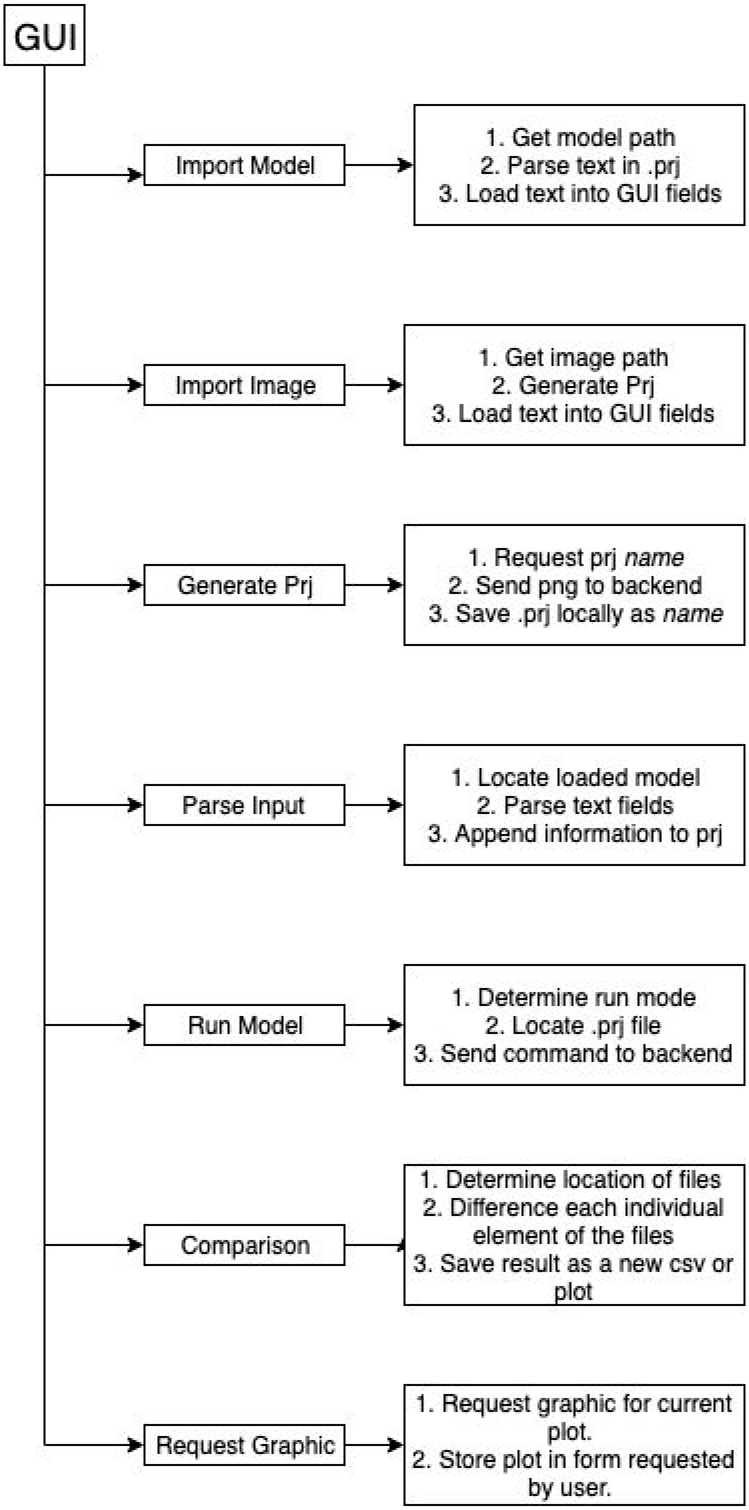
1. A slideshow describing the purpose of the backend software.
2. A GitHub Repository of the backend software.
3. Team Penobscot's own Software Requirements Specification.
4. Team Penobscot's own User Interface Design Document.

2. System Architecture

2.1 Architectural Design



2.2 Decomposition Description


GUI
Import Model
1. Get model path
2. Parse text in .prj
3. Load text into GUI fields
Import Image
1. Get image path
2. Generate Prj
3. Load text into GUI fields
Generate Prj
1. Request prj *name*
2. Send png to backend
3. Save .prj locally as *name*
Parse Input
1. Locate loaded model
2. Parse text fields
3. Append information to prj
Run Model
1. Determine run mode
2. Locate .prj file
3. Send command to backend
Comparison
1. Determine location of files
2. Difference each individual element of the files
3. Save result as a new csv or plot
Request Graphic
1. Request graphic for current plot.
2. Store plot in form requested by user.

For our architecture, we are using a Model-View-Controller pattern. This pattern best serves our purpose as we are creating a graphical user interface for an already created backend for data processing. The backend will serve as the model in this interpretation.

Our Graphical User Interface will split into the View and Controller portion of the MVC pattern. The View will reflect all user supplied parameters for a given plot. The Controller will take parameters and model geometry provided by the user and send it to the backend for processing.

3. Persistent Data Design

3.1 Database Descriptions

The software requested does not require the use of a database.

3.2 File Descriptions

This application operates using or generating seven different files.

1. Model control files (prj): These are text files which represent a model that is to be run (at some point) on the backend to generate a data files.
2. Input images (png): These are graphics supplied to the front end in order to generate the .prj files.
3. Meta Data (.txt): Files which contain information about the run parameters for a specific instance of running a model file on the backend.
4. Fortran Binaries (.DAT): These are artifacts generated by the backend which are stored locally. Users have no access nor need to access these files.
5. Export graphics (.svg): A graphic produced by running a plot on the backend.
6. Export graphics (.png): A graphic produced by running a plot on the backend
7. Resultant CSV (.txt): A subset of the information contained within the .DAT files.

4. Requirements Matrix

Below is a table showing the relation between the use cases stated in the System Requirements Specification document and the functions/methods satisfy those requirements. Descriptions of the functions are listed in the following section.

Functional Requirements:	**Relating Functions:**
1. Import Model	import_model()
2. Import Image	load_png(), generate_prj()
3. Input Parameters	parse_input()
4. Run Model	
4a. Single Shot Run	parse_input(), run_model()
4b. Wide Angle Run	

4c. Common Offset Run	
4d. Common Midpoint Run	
5. Preview Plot Metadata	import_model()
6. Generate a Comparison Plot	comparison()
7. Generate a Graphic	generate_graphic()

4.1 Function Descriptions

- import_model:
  - This function allows the user to import information contained within a .prj file to be loaded into the appropriate fields within the GUI.
- import_png()
  - The user will be directed to import a png file which will be used to construct a .prj file.
- generate_prj
  - Passes the image to the backend which generates the template for a .prj file to be expanded later.
- parse_input:
  - This function will take the user supplied information and add it to the .prj file currently imported.
- run_model:
  - run_model will operate in 4 primary modes: single shot, wide angle, common offset, and common midpoint. Each of these will take as a parameter a finalized .prj file. The mode will be selected by the user via the GUI and will be performed accordingly.
- comparison:
  - This function will take two model output sets (.dat or .csv) as inputs. It will then, value by value, take the difference of each pixel region. The result of this differencing will then be stored as a third plot for later usage.
- generate_graphic()
  - Based on pruned CSV text data, a graphic is produced by the backend and stored locally on the host’s machines. They have the option of storing it as a .png or as a .svg

Appendix A – Agreement Between Customer and Contractor

By signing below, the team members understand and agree to complete the functional and non-functional requirements listed and the delivery of the deliverables listed above. By signing below, the customer understands and agrees that the completion of the functional and non-functional requirements listed and the delivering of the deliverables listed above is equivalent to completion of the software project as a whole.

Should one or both parties desire a change to this document, a new draft of this document will be formed. This new draft will be written through collaboration between both parties, to elicit shortcomings of previous versions of this document and new functional and/or non-functional requirements. This new version will be signed and dated, replacing the previous version of this document only when both parties have signed and dated the document. Should signature dates occur on different days, the new version of the document will replace the previous version on the latest date specified below.

Team Member #1:

Name (Printed): Adam Farrington Date: 14 November 2019
Signature: Adam Farrington

Team Member #2:

Name (Printed): Nathan Gazey Date: 14 November 2019
Signature: Nathan Gazey

Team Member #3:

Name (Printed): Jens Hansen Date: 14 November 2019
Signature: Jens Hansen

Team Member #4:

Name (Printed): Alex Thacker Date: 14 November 2019
Signature: Alexander Thacker

The Customer:

Name (Printed): Christopher Gerbi Date: 14 November 2019
Comments:

Signature:

Appendix B – Team Review Sign-off

By signing below all team members agree they have reviewed this document. Signing below, team members agree to all content in this document aside from any comments in the space provided below. Signing below, team members agree that the format used in this document is agreeable aside from any comments in the space provided below. Singers acknowledge that the comment area below is not a place to voice major points of contention, only minor points of disagreement in this document.

Team Member #1:

Name (Printed): Adam Farrington Date: 14 November 2019
Comments:

Signature: Adam Farrington

Team Member #2:

Name (Printed): Nathan Gazey Date: 14 November 2019
Comments:

Signature: Nathan Gazey

Team Member #3:

Name (Printed): Jens Hansen Date: 14 November 2019
Comments:

Signature: Jens Hansen

Team Member #4:

Name (Printed): Alex Thacker Date: 14 November 2019
Comments:

Signature: Alexander Thacker

Appendix C – Document Contributions

- Adam - 25.0%
  - Attended meeting to establish expectations of customer
  - Initial draft of Introduction section
  - Initial draft of architecture diagram
  - Recreated digital architecture diagram
  - Review and approval of all document parts
  -
- Alex - 25.0%
  - Initial draft of Requirements Matrix section
  - Initial draft of section 3.1
  - Formatting document
  - Review and approval of all document parts
- Nathan - 25.0%
  - Attended meeting to establish expectations of customer
  - Initial draft of architecture diagram
  - Review and approval of all document parts
- Jens - 25.0%
  - Attended meeting to establish expectations of customer
  - Initial draft of Persistent Data Design section
  - Review and approval of all document parts