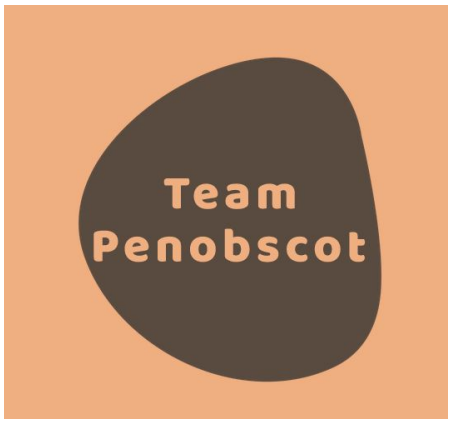

Seismic-radar toolbox GUI

Code Inspection Report

Dr. Chris Gerbi

Team Penobscot
Adam Farrington, Alex Thacker, Jens Hansen, Nathan Gazey
March 9, 2020
Code Inspection Report, Version 1.0.0

1. Introduction

1.1 Purpose of This Document

This document covers the programming standards that are to be upheld by the members of team Penobscot throughout the development cycle of SeidarT. Team Penobscot, as part of this plan will conduct a number of code inspection during this cycle. During these code inspections our team will be inspected to ensure that these standards are upheld. This document also contains details of the team's first code inspection meeting. This will be used as a template to be used as a guide for future inspections.

1.2 References

1. A Github repository which contains all relevant code and documentation regarding this project.
2. Team Penobscot's SRS, found in the above Github repository.
3. Team Penobscot's SDD, found in the above Github repository.

1.3 Coding and Commenting Conventions

We are using the PEP-0008 standards that were set by Python in 2001. These standards are fully embraced except functional and variable names are to be camel case. This difference would change the standard typical "get_image()" to "getImage()". This change is made as the entire team agreed that they are used to this style and find it easier to read. This standard was chosen due to the ability for static analysis capabilities and industry-wide adoption.

1.4 Defect Checklist

Figure 1 - Defect Table by Type

Defect #	Module Location	Type of Defect
1	getImage()	Correctness User Friendliness
2	getImage()	Correctness
3	Kivy File	Coding Convention

4	Kivy File	Coding Convention
5	SeidarTGUI.py Imports	Coding Convention
6	SeidarTGUI.py imports	Coding Convention
7	SeidarTGUI.py configurations	Correctness
8	build()	Coding Convention
9	build()	Coding Convention
10	getImage()	Correctness
11	getImage()	User Friendliness
12	getImage()	Coding Convention
13	getImage()	Coding Convention
14	getImage()	Correctness
15	getImage()	Coding Convention
16	getImage()	Coding Convention
17	run()	User Friendliness
18	run()	Coding Convention
19	run()	Coding Convention
20	Throughout	Commenting Convention

2. Code Inspection Process

2.1 Description

Our team's first code inspection took place at Fogler Library on February 29th, 2020. The plan was to all meet to perform a walkthrough review session of everyone's contribution to the code. Unfortunately Nathan was not able to be there in person but he was able to participate remotely by using Discord. During the meeting Adam took the role of Recorder while Jens acted as the groups Moderator. We decided it would be best to go through our code chronologically, based on how the program would operate during normal operations. Due to how our group distributed work on the software we did

not have any issues in terms of jumping back and forth between multiple authors during the inspection.

We started by going over the general appearance of the Graphical user Interface. Alex was in charge of everything having to do with the kv file, which managed all of the visual properties of displayed objects. There were not any major defects found in the kv file. The most notable issue was with the files coding/formatting convention. After that Jens and Adam took over as they walked through the main section of the program. They worked together on it through the extreme programming style of development. Similar to the previous section, most of the issues again were based on coding conventions. Other defects that were found were focused on code correctness and user friendliness. Nathan did not have code to present to the group since he was trying to set up Travis CI for continuous integration testing. That said, Nathan and the members who were not currently presenting their code were all contributed to the inspection. The code inspection was a success and helped our group identify many things that we needed to address in our software.

2.2 Impressions of the Process

The walkthrough process of reviewing the teams code worked well for our group. We were able to find 20 defects during our teams inspection review. We went through each person's code, taking the time to make sure that everyone understood what their code does. It was a great way to get everyone up-to-date with the current state of the entire program. A number of the defects that we discovered during the inspection process were pointed out by their author before others noticed them. This goes to show how important personal code reviews can be. The majority of the defects that were found during the code review were code convention violations. Our group never set up a standard coding convention before we started programming. These were small defects that didn't impact the operation of the code, most were naming conventions. Nevertheless they still needed to be addressed. The group code inspection was very beneficial for our group and is something that we should do more often.

Of our modules listed in section 3, we would wager to guess that the most defect-free modules we have would be build(). While GUI building is imperative for the creation of our software, it is not an inherently debug filled process. The defects in this module are all about coding conventions and commenting conventions, no defects in that module point to correctness. Of our modules we would also wager that getImage() is our most defect-filled module. This module was the first built, and as such was built before most of our coding conventions were put into place. This in addition to the ever-difficult process of searching file locations and reading image files leads to it being the module most prone to error.

2.3 Inspection Meetings

As of the creation of this document our team has only held one full team inspection review. It was held on the 29th of February with all team members present. It started at 10am on a virtual call in the Folger Library. This was moderated by Jens Hansen with Adam Farrington as a scribe. We completed a module by module walkthrough with Alex and Nathan acting as the code reviewers.

3. Modules Inspected

3.1 `SeidarTGUI.py` Imports

SeidarTGUI.py is where a majority of our magic happens. It's where *getImage()*, *run()*, and *build()* live and as such uses the same imports across all three. While no functioning code exists outside another module, as all three modules work under this wrapper it is important to be reviewed as well for enforcing conventions.

3.2 `getImage()`

getImage() is the module that handles the user inputting the name of an image file and subsequent importing of that file into the GUI. This module both imports the image and creates an empty PRJ based on the image chosen. The module also reads a pre-existing PRJ file, and imports into the GUI all information found there that was inputted on previous runs. *getImage()* satisfies several functions listed out in the SDD. As it creates a blank PRJ, it works alongside *run()* to satisfy Generate PRJ, and handles Import Model and Import Image in completeness.

3.3 `run()`

run() is the module that handles the reading of user inputs into a PRJ and the running of computation models SeiDartT is known for. Like *getImage()*, *run()* is a workhorse module that handles a majority of the SDD's functions. It works in tandem with *getImage()* to create completed PRJ files, satisfying Generate PRJ. As it interacts with the Kivy File it also handles Parse Input, Run Model in completeness.

3.4 `build()`

build() works primarily alongside the *Kivy File* to create the GUI that the user interacts with. The *Kivy File* handles some aspects of input parsing, but *build()* alone

doesn't engage in any functions listed in the SDD. Build simply imports the pieces build in the *Kivy File* and gives them names that other functions can find.

3.5 Kivy File

The Kivy File is another half of the GUI alongside *build()*. Kivy allows parts of a GUI such as a text input or a button to have labels that can be grabbed later via Python code. As such, it plays a passive yet still integral part in the Parse Input function specified in the SDD.

4. Defects

Figure 2 - In Depth Table of Defects by Module

Defect #	Module Location	Type of Defect	Description of Defect
1	getImage()	Correctness User Friendliness	If a file name not found is given by the user, the software crashes.
2	getImage()	Correctness	The first material input is inaccessible due to a layout issue.
3	Kivy File	Coding Convention	'Radar' and 'Seismic' are less descriptive than our conventions demand.
4	Kivy File	Coding Convention	Our convention is to put objects/modules in order or relevance. This convention is not followed in the Kivy file.
5	SeidarTGUI.py Imports	Coding Convention	Imports should be done as an on-need basis, where we have imported entire libraries.
6	SeidarTGUI.py imports	Coding Convention	Kivy imports are explicit and lead to legacy imports as modules change.
7	SeidarTGUI.py configurations	Correctness	Minimum height and width are not being enforced.
8	build()	Coding Convention	Panels are named panel1-4, not descriptive of their purpose.

9	build()	Coding Convention	Input forms are not unified in naming style. “Seismic_stuff” for instance.
10	getImage()	Correctness	The image imported before is not erased, leading to image overlap.
11	getImage()	User Friendliness	Image scale is not controlled at the moment, leading to a massive image potentially overlapping everything.
12	getImage()	Coding Convention	System calls in coding are correct, but not recommended by our convention.
13	getImage()	Coding Convention	colorsTuple may not need the float type casting it is given.
14	getImage()	Correctness	“The spacing on this is too large, but decreasing it causes some rows to disappear.”
15	getImage()	Coding Convention	Logical misordering of getting materials, making the materialBox, and scraping colors.
16	getImage()	Coding Convention	Several variables (‘content’, etc) are named in nondescript ways.
17	run()	User Friendliness	Before running, we re-build the prj to protect the user from mistakes such as editing the prj file. However this is computationally heavy. A decision needs to be made regarding if we want to protect the user from a niche example or save the time of re-building the prj file.
18	run()	Coding Convention	Variables are not named descriptively.
19	run()	Coding Convention	Duplicate code potential in writing to file.
20	Throughout	Commenting Convention	Comments are insufficient throughout several modules. Comments primarily separate by group, but these group comments require more description.

Appendix A – 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: 3/12/2020
Comments:

Signature: Adam Farrington

Team Member #2:

Name (Printed): Alex Thacker Date: 3/12/2020
Comments:

Signature: Alexander Thacker

Team Member #3:

Name (Printed): Jens Hansen Date: 3/12/2020
Comments:

Signature: Jens Hansen

Team Member #4:

Name (Printed): Nathan Gazey Date: 3/12/2020
Comments:

Signature: Nathan Gazey

Appendix B – Document Contributions

Adam:

- Contributed to the Code Inspection Review
- Review and edited whole Code Inspection Report
- Wrote Modules Section (3)
- Wrote Figure 1 and 2

Alex:

- Contributed to the Code Inspection Review
- Wrote draft of Purpose of Document (1.1)
- Wrote Code Inspection Process Description (2.1)
- Wrote Code Inspection Process Impression (2.2)
- Review and edited whole Code Inspection Report

Jens:

- Contributed to the Code Inspection Review
- Review and edited whole Code Inspection Report

Nathan:

- Contributed to the Code Inspection Review
- Review and enforcement of document style
- Wrote Commenting and Code style convention section (1.3)
- Wrote Inspection Meetings section (2.3)
- Review and edited whole Code Inspection Report