United States Geological Survey
Interactive Point Visualization
Testing Document

Prepared by: Erin Bailey, Curtis Bilbrey, Alex Farmer, Tim Velgos
Prepared for: Dr. James Dean Palmer
Date Submitted: May 4th, 2015

Project Sponsor: Jay Laura
Faculty Mentor: Dr. James Palmer

Last Updated: May 3rd, 2015
Creation Date: March 23rd , 2015
Version: 2.0

Table of Contents

Introduction

This document outlines the testing plans for the Interactive Point Visualization project, sponsored by USGS. In this document, we will address our plans for unit testing, integration testing, and user usability testing. Our testing begun according to these guidelines, and was originally estimated to be completed by April 30th. Testing concluded on May 1st.

Our unit testing for the project addresses individual sections of the project that may be tested independently. The integration testing portion focuses on our major components of the project, which are MongoDB, the Django framework, and the OpenLayers JavaScript library. Finally, we will document our user usability testing. This includes gathering the reactions of users as they interact with the user interface, and also their satisfaction with the ease of use and accessibility of the web application.

Unit Testing

The unit testing for this project will focus on the 3 main modules by testing their most relevant functions and including their return values. The 3 main modules to test for Interactive PointViz are Python Scripts, OpenLayers, and MongoDB in order to test for the desired result of each function. Along with the modules we test, we also list

Testing Procedures

The procedures for our unit testing section will be completed in their own Python modules in order to individually test functions.

Items to be Tested

The MongoDB module has several functions that will be tested:

Python Scripts:

1. The Query Script
    a. The query that obtains the data to be visualized will be tested with every possible range of values that it could possibly take to make sure the desired data is actually being extracted.
2. The Population Script
    a. This is the script that performs the necessary operations to store all of the point data within a MongoDB instance. It will be tested to make sure that the data that is put into a MongoDB instance is the same as the actual data received from the Kaguya Spectral Profiler.
3. The JSON Parsing Script
    a. This is the script that parses extracted information queried from MongoDB and converts it to the GeoJSON format which is necessary for the use of OpenLayers. This script will be tested to make sure all possible sets of data are being successfully

converted into the GeoJSON such that all data remains accurate and there is no data loss.

Openlayers:

1. mousePositionControl
    a. This function will be tested with various coordinate systems to make sure that it returns with accurate coordinates in regards to our map.
2. projection
    a. This function will be tested with different projection identifiers to verify that the coordinates are properly transformed.
3. heatmapLayer
    a. This function will be tested with many sets of point data to be sure it is accurately visualizing each data set.
    b. This function will be tested with various datasets to see if the points can be properly represented on our map.
4. htmlControls
    a. This function will be tested to ensure that the map is in fact updating properly within the limits of the HTML controls.

Items not to be Tested

This project integrates existing technologies whose failures are beyond our control.

1. Kaguya Spectral Profiler
    a. We have no way of testing if the data received from the Kaguya Spectral Profiler is actually accurate. Any faults that occur within this data are outside our control.
2. Openlayers Library
    a. We will not be testing the Openlayers library outside of the scope of our code. Any failures of Openlayers itself are beyond our control.
3. Django Library
    a. We will not be testing the Django library since it is outside the scope of our code. Any failures of Django itself are beyond our control.

Unit Testing Pass/Fail Criteria

The interpretation of the data is either correct or incorrect.

Possible unit tests:

-MongoDB Queries

-Test that a query is returning data from the correct bound/range

-We Will be testing pymongo queries since that is what the product actually uses. But at the core pymongo are MongoDB queries.

Integration Testing

For our integration testing, we will be focusing on the three major components that comprise our project: the MongoDB database, Python scripts, and the OpenLayers JavaScript visualization code.

Example Outline of Integration Test

The MongoDB database is our most critical component, it is where the information that the PointViz application is interpreting will be stored. It will be tested as a standalone module.

The python scripts are mainly used as a communicator between MongoDB and Openlayers. There is also a script that is used to populate a MongoDB instance with the data received from the Kaguya Spectral Profiler. The communication scripts are responsible for taking the requested information out of the PointViz application, querying a MongoDB instance based on that information, and converting the queried data into a GeoJSON format which can be understood by Openlayers. Each of these scripts will be tested individually based on their functionality and goals.

OpenLayers controls the visualization of our data and allows the user to manipulate the queries.

Testing Specifications of Modules

MongoDB

1. Check the data inputted into the database against the actual data. We will first populate our MongoDB instance, then check all of that with the data received from the Kaguya Spectral Profiler.
2. Check to make sure a connection the the proper MongoDB instance has actually been made, and if not throw an error.
3. Check to make sure each pymongo query is being constructed properly based on the query inputs.

Django

1. Check to make sure that the python portion of the project and the javascript/html/css are interacting with each other in the proper way.
2. Each of these instances will be tested separately first in the unit testing portion in order to evaluate whether or not they work properly as independent entities before we can assess whether or not they will coexist together using Django.

OpenLayers

1. Check that the lat/lon points are displayed on the corresponding lat/lon points on the map
2. Check that the visualization can pull the array of lat/lon points and successfully map the visualization to it.
3. Check that the visualization is accessible and interactable with the user interface

Measurements of Success

MongoDB

1. All the data is accurately stored in a MongoDB instance.
2. Proper connection information will be known to the user.
3. Every pymongo query is being formed exactly to the desired inputs.

Django

1. All the code from our javascript, html, css, and python files will be working together efficiently
2. The interface for the web application is an interpretation of the previous files mentioned, integrated together using Django.

OpenLayers

1. The visualization is displayed on the screen
2. Point data is displayed on the visualization

Outcome

We will have solidified our confidence in the accuracy and usefulness of the Interactive PointViz application. We also have increased comfort in knowing all the modules are communicating soundly.

Usability Testing

The Point Visualization group will develop several scenarios that have users perform specific tasks. During this process each user will be timed and recorded. After the scenario is complete each user will fill out a survey with questions regarding the process.

Participants

The participants will be selected from a diverse group of volunteering NAU students ranging from people with strong computer skills to others who have only a basic computer skills. At minimum a participant must know how to operate a web browser. After selection the participants will be required to accomplish a set of tasks using the Point Visualization tool that represent a real life scenario. Afterwards they will be asked to fill out a short survey regarding the experience.

Procedure

The usability testing process will take place on campus at the NAU Engineering building. Each participant will use a computer with recording software setup to record their screen and face. Each user will be handed a printed set of instructions with a set of goals they must accomplish. After completing the goal the participant will be asked to participate in a short survey pertaining to the experiment.

The Test

Each scenario will be developed by the Point Visualization team and will aim to accomplish different goals. Each scenario will attempt to accomplish three goals:

Goal 1: Can the user easily navigate the interface to accomplish the goal?
Goal 2: Did the user produce the desired image after performing the given tasks.
Goal 3: Was the scenario completed within a reasonable amount of time.

The testing scenarios will vary depending on the user's experience with web browsers and computers in general. While the interpretation of the data will depend on user's background information from a geological perspective, the actual process of testing the interface and learning how to use the PointViz web app will not rely on any previous experience with geology.

Measurement

Each participant will be timed and compared with other users with their given scenario. The participants time will be weighted together with their score on the survey to give us a value for how usable the product was.

Evaluation and Questions

A short survey will be issued to each usability tester and they will be evaluated on their performance. The evaluation process involves a measurement of success and timing how long it took for the user to complete their tasks. The survey will have various questions asking how comfortable they were with the interface, stress levels while operating the interface, suggestions for future tests, and other similar questions.

Reporting

The recordings of each test will be carefully reported and examined. The results will then be used by the Point Visualization team to adjust the interface and add any extra features to increase usability.