Software Testing Plan

March 29th, 2023

Version 2.0

USGS AirFlow Processing Pipeline
Team ARES

Sponsored By:
Trent Hare

Team Mentor:
Vahid Nikoonejad Fard

Team Members:
Hunter Woodruff (Team Lead)
Quinton Jasper
Chris McCorkle
Isaiah Raspet
Richard McCormick

Table of Contents

I. Introduction

The United States Geological Survey (USGS) Astrogeology Science Center is a government research center based in Flagstaff, Arizona. Their mission is to collect, analyze and publish cartographic and geological data, which helps both researchers and the general public to better understand the surface of not just Earth, but also other celestial bodies within our solar system. The USGS team accomplishes this via multiple pieces of software designed for specific tasks, such as file conversions, data compression, image rendering, etc. For the course of this project, Team Ares will be working with a specific package of USGS software, collectively known as ISIS3.

ISIS3 is a package of more than three-hundred individual applications that each handle unique roles in the image processing workflow. The sponsor for this capstone project, Mr. Trent Hare, has informed the team that, while ISIS3 is a complete product and works for their needs, its fragmentation makes it difficult for users to manage. To follow the workflow properly, users have to locate and understand each required application from within a Linux terminal. It is also expected that users will need to establish custom workflows, where each may consist of many different combinations of applications based on the user's needs. From the perspective of a researcher or an enthusiast, the current structure involves too many steps with too many complications that increase the time and energy required to obtain the desired outputs.

As a brief overview of this project, our team has been working closely with the USGS to develop a *pipeline visualization and control front-end* for ISIS3. Over the semester, our team has completed a version of this solution that meets our client's acceptance criteria. However, before our project can be handed-over, a system must be established to ensure our work meets quality

standards before it is released to production. As a team, we've developed a plan for how to test our product best, via the creation of Unit Tests, conducting Integration Testing, and Usability testing, each method serving a unique purpose and ensuring the quality of our work. Within the correlating sections, each type of testing is explained, as well as, an in-depth look at each planned test criteria needed to accept the software as a successful product.

II. Unit Testing

Unit testing is a software testing method in which individual units or components of a software system are tested in isolation from the rest of the system. The goal of unit testing is to validate that each unit of code performs as expected and meets the design requirements. A unit test typically consists of a small piece of code that tests a specific function or method of the codebase. The unit test is written by the developer, and it can be automated to run every time the codebase is changed. The test checks for errors or unexpected behavior in the unit of code being tested, and it can help identify and fix bugs early in the development process. Unit testing is a crucial part of the software development process as it helps ensure the reliability, maintainability, and scalability of the code. By identifying issues early on, developers can save time and effort in the long run, and deliver high-quality software to users.

Due to the relative simplicity of our project, as well as the fact that ISIS and Kalasiris have been in use for decades, our plans for unit testing are simple as a result. With our project revolving mostly around linking different softwares into a working system the most rigorous unit testing has been completed by these separate softwares. As such, our plan revolves around

ensuring that each portion of the code we write is working completely as expected. This boils down to ensuring that the ISIS and Kalasiris Code is wrapped into a usable form for Apache AirFlow, as well as verifying that our JSON parser can parse the ISIS and Kalasiris scripts efficiently and without error.

1. **Python Wrapper Program Unit Testing Plan**

- Input Validation:
    - Test that the Wrapper program can handle invalid input data and returns appropriate error messages.
    - Test that the Wrapper program can handle different types of input data.
- Functionality:
    - Test that the Wrapper program can perform the desired function or functions on the input data.
    - Test that the Wrapper program returns the correct output data based on the input and function performed.
- Performance:
    - Test that the Wrapper program can handle large amounts of input data efficiently.
    - Test that the Wrapper program can handle multiple concurrent requests without crashing or slowing down.
- Error Handling:
    - Test that the Wrapper program can handle unexpected errors and return appropriate error messages.
    - Test that the Wrapper program can recover from errors and resume normal operations.

2. Json Parser Unit testing plan

- Parsing valid JSON:
    - Test the parser with a small JSON file that contains simple and complex key-value pairs.
    - Test the parser with a larger JSON file that contains simple and complex key-value pairs
    - Test the parser with a JSON file that contains non-ASCII characters.
- Handling invalid JSON:
    - Test the parser with a JSON file that has a missing opening or closing brace.
    - Test the parser with a JSON file that has a missing comma or colon between key-value pairs.
    - Test the parser with a JSON file that has an invalid data type such as a string value for a number key.
    - Test the parser with a JSON file that has a malformed string value such as missing quotes around a string.
- Performance:
    - Test the parser with a large JSON file to ensure it can handle parsing the file efficiently.
    - Test the parser with a variety of JSON files with different sizes and complexity to ensure it can handle different scenarios.
- Edge Cases:
    - Test the parser with a JSON file that has a key or value that is an empty string or null.

- Test the parser with a JSON file that has a key or value that contains whitespace or control characters.

III. Integration Testing

Integration testing is a critical part of the development process for our software project, which aims to provide users with a clean, user-friendly, and visually informative tool for designing Directed Acyclic Graphs (DAGs) using graphics-based tools, and easily deploying and running those graphs in a pipeline. The software will be installed on a user's workstation or in a cloud environment and will integrate Apache AirFlow, ISIS3, Kalysiris, and Anaconda to achieve the desired functionality.

Our approach to integration testing will focus on the interfaces between the major modules and components of the software, ensuring that interactions and data exchanges between these modules take place correctly. We will be particularly diligent in testing the "plumbing" of the system, i.e., ensuring that all components are wired together correctly to achieve seamless integration. This will include thorough testing of the boundaries between important modules, such as those involved in accessing databases, handling data exchanges in a web-based interface, and managing network-based communication. Our goal is to verify that these interfaces are working correctly and that the overall system functions smoothly and reliably. Additionally, we will conduct extensive testing to identify and resolve any potential issues that may arise due to the integration of multiple components, ensuring that the software performs as intended and meets the requirements of our client's vision for the final product. Integration testing will be a

crucial step in ensuring the overall quality and reliability of the software package being developed.

1. **ISIS3 and Kalysiris Integration:**

   The first major integration point in the software package is the integration of ISIS3 and Kalysiris. ISIS3 is the core software for applying filtering and processing modules to raw data collected from remote sensing arrays, and Kalysiris provides a Python-based interface for each ISIS3 module. The integration testing for this component will involve the following:

- Developing test harnesses to invoke ISIS3 modules through Kalysiris and verifying that they return the expected results.
- Testing various scenarios with different input data and parameters to ensure that the integration of ISIS3 and Kalysiris is correctly handling the data exchanges and producing accurate results.
- Verifying that all contract assumptions, such as data required to invoke a function and data expected to be returned, are respected during the integration process.

2. **Apache AirFlow and ISIS3/Kalysiris Integration:**

   The next major integration point is the integration of Apache AirFlow with ISIS3 and Kalysiris. AirFlow is the software that allows users to design and run directed acyclic graphs (DAGs) for managing workflows. The integration testing for this component will involve the following:

- Developing test harnesses to create DAGs in AirFlow that include ISIS3/Kalysiris modules and verifying that they can be executed without any errors.
- Testing various scenarios where different DAGs with different configurations of ISIS3/Kalysiris modules are executed to ensure that the integration between AirFlow and ISIS3/Kalysiris is functioning correctly.
- Verifying that the data exchanges between AirFlow and ISIS3/Kalysiris are accurate and all contract assumptions are met.

3. **Overall System Integration:**

   Once the integration of ISIS3/Kalysiris with AirFlow is successfully tested, the overall system integration testing will be performed. This will involve testing the interaction and data exchanges between all major modules in the software package, including ISIS3, Kalysiris, and AirFlow, to ensure that the system is working seamlessly as a whole. The testing will include:

- Testing end-to-end scenarios where users design DAGs using Elyra (excluded from this plan), which includes ISIS3/Kalysiris modules, and run them in AirFlow.
- Testing different workflows with various combinations of ISIS3/Kalysiris modules to verify that the overall system is handling the data exchanges and interactions correctly.
- Verifying that all contract assumptions, data exchanges, and interactions between the modules are working as expected.

Integration testing is a critical step in ensuring smooth integration and interaction between major modules and components in the software package being developed. By thoroughly testing the integration points and verifying that all interactions and data exchanges are taking place correctly, the software package can be ensured to be functioning as expected. The integration testing plan outlined above provides a comprehensive approach for testing the integration of ISIS3, Kalysiris, and Apache AirFlow, and will help ensure the successful integration of these components in the final software product. Additionally, further testing may be conducted during the development process to address any issues or bugs that may arise, ensuring a robust and reliable software package that meets the client's requirements of being user-friendly, visually informative, and capable of constructing, running, and monitoring workflow pipelines.

IV. Usability Testing

Usability testing is a process of evaluating a product, such as a website or mobile application, by testing it with real users. The goal of usability testing is to measure the usability of a product and identify any problems that users may encounter while interacting with it. This type of testing is essential in ensuring that a product is easy to use and meets the needs of its intended audience.

Usability testing can take many forms, including in-person testing, remote testing, and automated testing. In-person testing typically involves bringing participants into a lab and observing their interactions with the product. Remote testing is conducted online, allowing participants to test the product from their own devices. Automated testing involves using

software to test the product's usability automatically. The process of usability testing typically involves several stages. First, the team conducting the test must define the goals and objectives of the test. This may involve developing a set of tasks for participants to complete while using the product. Once the goals and objectives are defined, participants are recruited for the test. These participants should be representative of the product's target audience.

Our group will be conducting the remote variation of usability testing to be better able to observe how our product is being used by researchers at USGS. As diligent researchers already, the team at USGS will provide the perfect candidates for our usability testing run as they're fully capable and qualified to use the software to conduct research. To be able to run accurate tests, we will be recording and analyzing testing sessions with our users wherein we will track their flow and thought process when using the Apache Airflow interface, and how they utilize the new features our product brings to their current workflow. Since our testing is all online, recording our results and sessions will be easy, as any recording software will suit our needs. These tests will be run as often as possible with our Sponsor and his team, to ensure a surplus of data for us to analyze to better understand the researchers' process and make adequate adjustments. To analyze these insights, we'll be employing the use of our current knowledge of the software, and how we feel it should be utilized, with how the reality of the situation is. If the researchers deviate from our 'optimal workflow' then we'll be able to measure why they made the choices they did, and if the choices they made better suited the work environment and workflow target our software is trying to enhance.

Usability testing is a vital process for ensuring that a product is easy to use and meets the needs of its intended audience. By testing a product with real users, we can gain valuable insights into how the product is used and identify areas for improvement. Usability testing will be an

integral part of the product development process, and allow us to realize the goals we set out to achieve with the team ARES project.

V. Conclusion

The hard work provided by the team at the USGS Astrogeology facility has far-reaching applications that will affect the future of human space travel and exploration. However, before large-scale goals can be accomplished, optimizations must be made at a lower level in the system to increase efficiency and effectiveness. In this case, the USGS's original workflow system for processing and publishing gathered planetary data has consisted of a complex, manual execution process for all steps involved. Team ARES is tasked with developing a *pipeline management system* to allow researchers at the USGS to streamline their process of image processing and data publishing.

To ensure the quality of the product before it is released to production we have developed a plan to test all functionality of the *pipeline management system*. The plan includes unit testing, integration testing, and usability testing. A detailed plan for unit testing the Python Wrapper Program and Json Parser to test these individual components of the system in isolation from the rest of the system, with input validation, functionality, performance, error handling, and edge cases. The integration testing will ensure the smooth integration of each component into the entire system, while the usability testing will evaluate the user experience of the software. Overall, our efforts are geared towards making the software easy to use, reliable, and scalable, thus delivering high-quality software to users.