Project Requirements Document

November 21st, 2022

<u>USGS AirFlow Processing Pipeline</u>

Team ARES

<u>Sponsored By:</u>
Trent Hare

<u>Team Mentor:</u>
Vahid Nikoonejad Fard

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

Table of Contents

I. Introduction

With each success that NASA accomplishes, landing rovers onto the surfaces of foreign planets and conducting groundbreaking research, it's easy for the average person to overlook the crucial underlying sciences that support such efforts, especially in the early stages. Questions such as: “How can this rover land safely on planet Y?” or “How will this rover navigate the surface of planet Y once it is landed?” require reliable answers before any technology can set foot in outer space. NASA is not alone in its effort to answer such questions.

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.

Our team has been tasked with assisting in the development of a workflow management system. That is a system that makes the process of organizing, executing, and monitoring a workflow more intuitive and accessible to more users. To our sponsor, this would be best accomplished with some sort of graphical user interface. This interface would allow users to interact with their workflows much like a flowchart. This Directed Acyclic Graph (DAG) would consist of nodes that represent an individual application, and the connections between these nodes show the direction of data input and output.

This report will document the client’s specific problem, our proposed solution, our plan of implementation, all the various technical requirements of the project, and the potential risks that are inherent within the solution. All of these culminate in a successful implementation of our client’s Minimum Viable Product (MVP) in addition to any possible stretch goals.

II. Problem Statement

The main problems the USGS is facing in its current system are convenience, efficiency, and adaptability. With ISIS3 consisting of more than three hundred individual applications that each handle unique roles in the image processing workflow, it is no small feat to manually navigate this workflow and obtain the desired results. This problem quickly grows out of hand if the desired pipeline has not been previously scripted and uploaded to the ISIS3 repository. Should this be the case, a researcher, who may or may not have coding experience, would have to write the Python script themselves. To make matters worse, if one of the applications within the script needs to be changed out for another application, then a new script would have to be written. The final hurdle that a user would face would be the execution of the pipeline. These scripts are run through a UNIX terminal, rather than a polished user interface, making it a far less accessible method of getting the desired output. These four sub-problems are what help define the solutions required for our Minimum Viable Product.

In addition to the Minimum Viable Product, the USGS has asked us to look into ways for researchers to build pipelines with minimal to no coding involved. In addition, the ability to export cloud-optimized GeoTIFF and STAC metadata files using the *Geospatial Data Abstraction Library* (GDAL) would be preferred, in conjunction with the ability to connect the final output to the USGS's Leaflet web-map interface.

To briefly summarize the above, the problem we are seeking to solve is:

- Minimum Viable Product:
    - Access to prebuilt common pipelines within the workspace of the project
    - A way for researchers to create unique workflow pipelines efficiently
    - A user interface to be used in the creation and execution of the pipeline
    - A container to be able to efficiently deploy the project on any number of systems.
- Stretch Goals:
    - A minimal or non-coding solution to pipeline creation
    - Export support for cloud optimized metadata files
    - Export support for main output directly to web services hosted by the USGS

III. Solution Vision

Each year, thousands of images from the planets and moons within our solar system are captured by various imaging satellites and other data collection hardware. These images, to be useful for scientific applications, require processing. Currently, this processing is done by a package of image processing tools called ISIS3 - *Integrated Software for Imagers and Spectrometers* - which is created and managed by the USGS and NASA. While an incredibly powerful tool, ISIS requires a high level of conceptual and practical knowledge of computer systems and software to operate. This can be alienating for researchers who have specialized in other areas of expertise.

Our team has been tasked with developing a new way to better manage the image-processing workflow with the ISIS3 suite. This would involve the implementation of intuitive and easy-to-use graphical tools, rather than terminal commands. To accomplish this, we will be building off of Apache AirFlow, a unique tool for data science that allows DAGs to be run in predetermined configurations (pipelines). Our goal for the project is to port the DAGs defined in the ISIS suite over to AirFlow. Combined with graphical plugins available for AirFlow, this will allow researchers to construct their processing pipelines with minimal effort.
Functionally, this solution will be a massive improvement over the original workflow for generating processed images from DAGs. By implementing a graphical user interface, as well as intuitive tools such as drag-and-drop DAG creation, we will significantly decrease ISIS3's barrier of entry. Researchers will be able to process images without having to go through extraneous or specialized training. This will decrease the time from project start to project end, and significantly improve the throughput of fully processed imagery.

This final product will make use of multiple technologies operating in unison. Apache AirFlow will be the underlying base of the software, forming the skeleton of the system. Supplementing AirFlow will be Elyra, a plugin for AirFlow that allows for the implementation of a drag-and-drop functionality for building pipelines. Additionally, ISIS3 and Kalysiris will be implemented to allow their Directed Acyclic Graphs to be imported into the final product. This will fulfill the requirements for our Minimal Viable Product.

There are several stretch goals for this project we hope to add to the project once it is completed. The largest of these goals is the implementation of a bridge between AirFlow and the client's existing GDAL and CartoCosmos system, which allows the user to immediately display their finished product on a live map of whatever planetary body the image is from. Additionally, these finished products will be able to be exported as cloud-optimized GeoTIFFs to local machines or cloud storage.

IV. Project Requirements

Functional Requirements

1. **Users will be able to track and watch their progress**
    a. For the user to be able to track their progress, our team will need to implement a system wherein, through the use of Airflow's built-in GUI and the Elyra plugin, the user will be able to visualize the workflow that they are interacting with.
    b. This visualization will allow the user to seamlessly visualize the DAGs. With this functionality, the user can create, delete, and edit DAGs as they see fit, using the python-wrapped ISIS3 command "nodes" as the individual pieces of the DAG.
    c. The user must also be able to visually watch the product of the DAG come to fruition. Therefore, the entire process of taking individual nodes and forming a pipeline/DAG with them must be visualized in the Airflow graphical user interface, including the finished map being visualized after the process terminates.

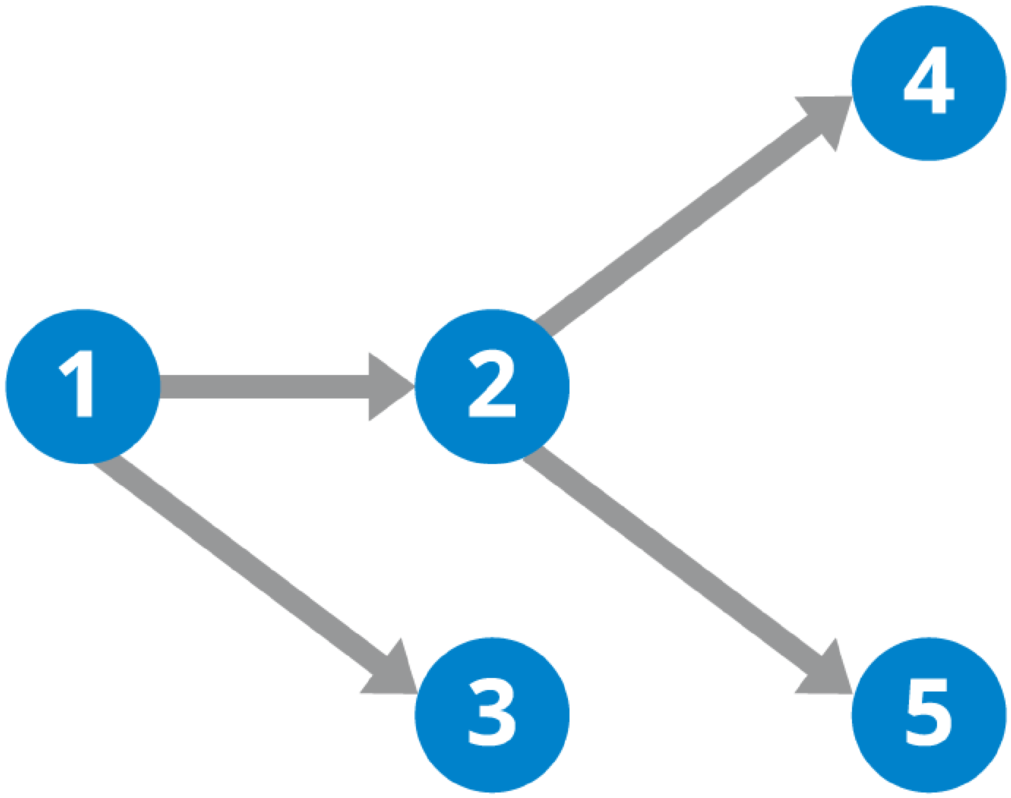


Fig.1: Example of a visualized DAG

2. **Users will be able to interact with their environment**
    a. To make this product as easy to use as possible, interactability with the Apache Airflow and Elyra environment is required. Instead of executing individual ISIS3 commands, users will be able to utilize the functionality of the Airflow environment to create their DAGs.
    b. Likewise, to assist with the readability of the Airflow environment, our team will add the Elyra plugin to Airflow's native environment. Through Elyra, users will be able to "drag and drop" Python-wrapped ISIS3 command nodes into their DAG pipelines seamlessly. This will allow for maximum efficiency with the Airflow pipeline creation, as users will run minimal, if any, commands to achieve the same result as the previous iteration of the ISIS software.
3. **Users can save and create new pipelines**
    a. In the previous section, it was discussed how users should be able to create, edit, save, and remove their pipeline creations. To achieve this, Apache's workflow pipeline creation tool Airflow will be utilized.
    b. In Airflow, users will be able to take individual ISIS3 commands, which our team will be wrapping in python to be accessible to Airflow, and create DAGs. These DAGs will be able to take a base image input, then run through the pipeline's commands in the order that the user chose, to generate an actionable image product.
    c. Crucially, to reduce the time overhead involved with DAG creation, users will be able to save their pipeline instance for later use. They will also be able to delete pipelines they don't deem fit and replace them with entirely new or existing ones. This will also be handled by the Airflow framework, with Elyra assisting users in the creation of DAGs using drag and drop, and storing their pipelines on the local Airflow instance.

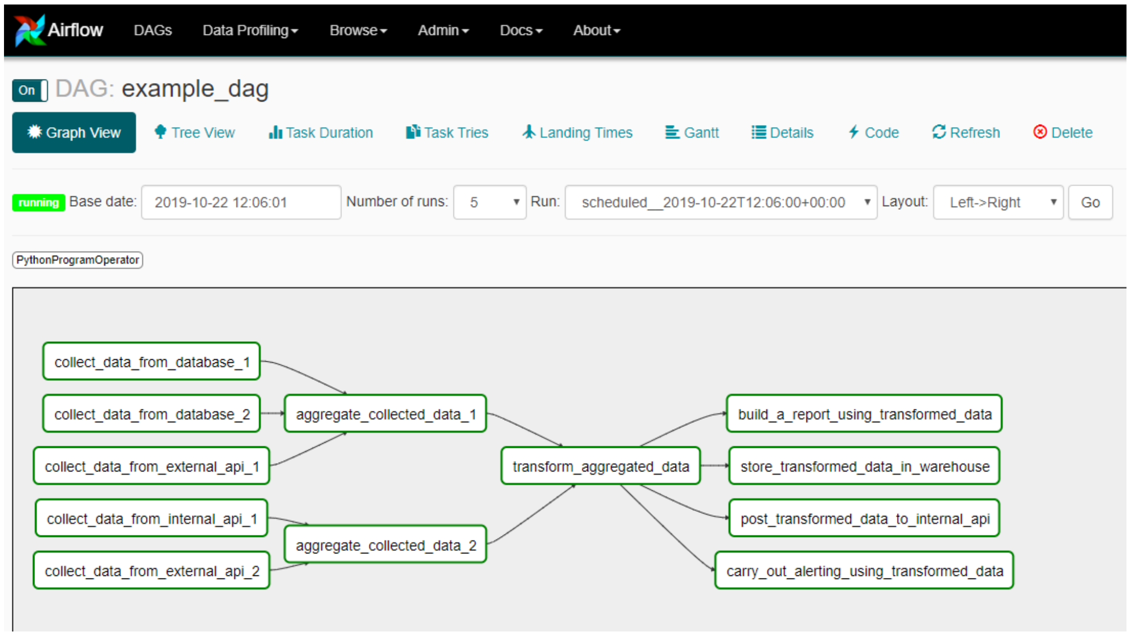

Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
On
DAG: example_dag
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh
Delete
running
Base date: 2019-10-22 12:06:01
Number of runs: 5
Run: scheduled__2019-10-22T12:06:00+00:00
Layout: Left->Right
Go
PythonProgramOperator
collect_data_from_database_1
collect_data_from_database_2
collect_data_from_external_api_1
collect_data_from_internal_api_1
collect_data_from_external_api_2
aggregate_collected_data_1
aggregate_collected_data_2
transform_aggregated_data
build_a_report_using_transformed_data
store_transformed_data_in_warehouse
post_transformed_data_to_internal_api
carry_out_alerting_using_transformed_data

Fig.2: Example of a pipeline in the Airflow GUI

4. **The program will be run in a single instance/environment**
    a. For this program to be as easy to use as possible, our team and client decided it would be best to have the program run in as small of an instance as possible, preferably a single instance. To accomplish this, we will be employing a Conda environment or docker environment to keep the entire program as one runnable instance.
    b. By employing this strategy, we keep the required number of running instances to a minimum, which makes for overall easier readability and understanding of what is going on inside the application.
5. **Users will be able to customize DAGs**
    a. To be able to fully render the former ISIS3 process obsolete, the DAGs must be fully customizable with the python-wrapped ISIS3 “nodes”. To accomplish this, we are again implementing the use of Apache’s Airflow and Elyra.
    b. By combining the GUI and “drag and drop” functionality of Elyra with Airflow’s database-like system for storing and saving DAGs, we are able to create a system wherein a user can visualize the DAG that they want to edit, drag and drop nodes as they see fit to create an actionable pipeline, then save and re-use these graphs.

Performance Requirements

1. Performance Requirements are not reliant much on Airflow Processing Pipeline and rest more on ISIS3, which is a compiled binary. It is where most of the program execution time will take place.
2. This software is not intended for rapid use, nor anything real time. This software is also not designed for use on any specific machine but rather for use to generate maps and images that may be used in future projects.
3. The Ares project will utilize a simple architecture that creates faster deployment of the already developed ISIS software. The performance hitches on user accessibility, so the majority of streamlining will not be in software optimization but rather in clarity and ease of use for a user.
4. The Airflow Processing Pipeline will allow for quick redeployment of common pipelines, which should cut down massively on use time, and should not greatly impact runtime performance.

Environmental Requirements

1. Within the Airflow Processing Pipeline exists only one Environmental Requirement, which is the use of ISIS3 and Kalasiris for the processing of the data within the pipeline. ISIS was developed by USGS, and no other software can fit their needs nor be adapted to fit any new requirements as seamlessly. As a result, ISIS3 and Kalasiris are the only software that is required, by Mr. Hare, to be used within the Airflow Processing Pipeline.

V. Potential Risks

There are inherent risks that come from the use of any software. The Airflow Processing Pipeline is no different, and, as such, the team will have to carefully consider those risks. The main risk is bad data being provided by the user. Bad data is defined as *"any data that is inaccurate, inaccessible, poorly compiled, duplicated, or missing any number of key features that are required by the system for computation."* In the case of The Airflow Processing Pipeline, bad data can have sweeping consequences, ranging anywhere from execution errors to hundreds of millions of dollars lost on a failed mission to Mars.

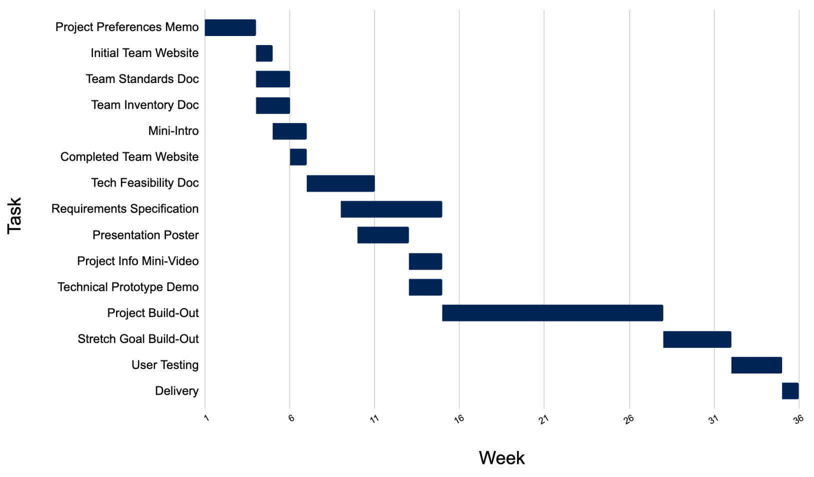
Starting with the more inconsequential risks, we will discuss the risk of user error. In this section, it is assumed that bad data will result in an error and not reach the end of the pipeline. User error can stretch from a misunderstanding of how to deploy, build, or use the pipeline to a simple mistake made by an expert, for example, the use of one ISIS3 application instead of another. In any of these cases, it is most likely that the execution of the pipeline will fail and ISIS3 will provide its error-handling mechanisms. Thus, terminating the process with only time wasted; is by far the best-case scenario. Following that would be a scenario where the pipeline software fails to exit and *"hangs"* indefinitely. Hanging, or freezing, is when a program stops responding to inputs. Apache Airflow has the capability of displaying which nodes of the pipeline have been completed and which nodes have not. While hanging can similarly lead to wasted time, it introduces an element of confusion, as to whether or not a process is properly executed or, rather, frozen in place. Airflow's features can be leveraged to identify which situation the program may find itself in. Lastly, a scenario in which the pipeline completes, but the output data is incorrect. This can happen if there is a mistake in the construction of the pipeline or data corruption that results in inaccurate outputs. For the moderate to worst-case scenarios, the risks escalate exponentially. If that output is used anywhere within the USGS's ecosystem, that data could be duplicated, resulting in further incorrect outputs. The more times the incorrect output is used, the more time it will take to revert and populate those areas with correct data. In the worst-case scenario, as unlikely as it is, it is still possible that incorrect data is used in the planning and execution of a mission to Mars. At present, if an unmanned mission to Mars fails, the price is under one billion dollars. With a manned mission to Mars, not only would the cost be near or over a billion dollars, but also the loss of lives of a crew of experts.

The final and most relevant risk The Airflow Processing Pipeline is facing is the relative fragility of the system. Our solution relies upon ISIS3, Apache Airflow, Elyra, and the Conda environment it will be housed in. If any of these dependencies were modified in such a way that compromises compatibility, it is likely the team at USGS will be forced to return to their original, inefficient workflow for data processing. Once again, this outcome is unlikely but not outside the realm of possibility and, as such, must be taken into consideration.

VI. Project Plans

As a team, we have established rough time periods by which we expect to have completed our set development milestones. Below is a Gantt chart that presents each of our established milestones, as well as a timeline indicating estimated start and end dates for each.

At this point and time, the primary milestone of our team is to create a simple product prototype. This will demonstrate both the feasibility of the desired product and give the client a point of reference for upcoming meetings and discussions. Furthermore, this prototype will allow our



team to evaluate each of the technical aspects of the product and reevaluate troublesome areas where necessary.

With this milestone achieved, focus will shift towards full product buildout. This is the most important milestone for our team and will represent the culmination of a semester of research and planning. This product will, at a minimum, demonstrate all attributes of the client's Minimum Viable Product. Included in these requirements are the ability for users to use a Graphical User Interface to construct DAGs, as well as to run and monitor pipelines using Apache AirFlow. We estimate that the completion of these requirements will take the longest amount of time, and are thus estimating at least 1 month of development time for each portion. Additionally, we will be working throughout both of these requirements to ensure that the product is packaged and can be easily and quickly distributed via Conda. Once this milestone is achieved, we can continue to our final milestones.

With the MVP in hand, the team will begin developing the product stretch goals. These will include additional functionalities that the client has identified as desirable but not necessary. The biggest stretch goal is the implementation of the CartoCoSMoS LeafLet map into our product. This will allow users to instantly display their finished product. Furthermore, completing the MVP will allow us to also begin the user testing phase of our product. We are very happy to be able to have this option. User testing over the last few weeks of development will allow us to polish and optimize the product before final delivery.

VII. 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.

Before work on this project can begin, our team had to conduct thorough research to determine the most effective technologies and tools to use in the development of this project. By the end of this research, we've made final decisions as to what tools will be used. *Anaconda* will act as our environment and dependency management tool, *ISIS3* is the core software package that provides tools for researchers to process planetary data, *Apache Airflow* will be the framework that is used for managing and executing DAGs, and lastly, *Elyra* will allow users to construct DAGs without the need for software programming with its 'drag and drop interface.

Though the team will be primarily working with established technologies, one can not be too careful when considering what risks may be introduced to the system. From our analysis, we determined that most issues with the system itself are not a major concern in the short term. We identified that most worst-case scenarios involved the introduction of human error while constructing DAGs, or providing data inputs. Although, when considering the long-term effects of mal-constructed data we express a growing concern for the integrity of future space missions that may make use of this data, and ultimately fail as a result.

Overall, our team has made a great effort to understand the client's problem and flesh out a solution that works best for them. With constant collaboration, communication, and dedication within the team and with the client, the individuals of Team ARES hope to produce a piece of useful software that will be used by the USGS team in the years to come.

VIII. Glossaries and Appendices

Project RFP
https://www.ceias.nau.edu/cs/CS_Capstone/Projects/F22/Hare_T-AirFlow-ISIS-FY23_v04.pdf

Team Website
https://www.ceias.nau.edu/capstone/projects/CS/2022/TeamAres_F22/

Team GitHub
https://github.com/Team-ARES-Airflow-Processing-Pipeline/

Jenkins Pipeline
https://www.jenkins.io/doc/bookapipeline/

Elyra
https://github.com/elyra-ai/examples/tree/main/pipelines/run-generic-pipelines-on-apache-airflow

Rabix Composer
https://github.com/rabix/composer

Chartis Plug-In
https://github.com/trejas/chartis

Rabix + Chartis Visual DAG Editor Demonstration
https://airflowsummit.org/sessions/demo-visual-dag-editor/