Requirements Specification

Team Osiris

2020-11-07

Version 1.1

Sponsored by Scott Akins and Trent Hare

Advised by Fabio Santos

Daris Dumel
Evin Dunn
Milton Ibera
Rebecca Leggett
Sara Huber

By signing below both parties agree that the requirements listed in this document are to be the baseline requirements for the USGS ASC Capstone project at NAU.

Client Signature: X_____ Date: 11 /19 /2020

Team Lead Signature: X_____ Date: 11 /20/2020

Table of Contents

Introduction

The exploration of space beyond our own planet has fascinated geologists, physicists, engineers, and the general public for decades. Exploration of the solar system is driven by interest in the makeup of worlds besides our own, the forces that shape our universe, and the possibility of life beyond our earth. Aside from questions as broad as these, there is evidence that space exploration directly benefits the daily lives of the general public. According to NASA[1], launch propulsion research has contributed to cheaper airfare, space energy storage provides a model for more efficient solar power, and satellite communication systems have led to improved WiFi here on earth.

It takes more than the efforts of astronauts to explore space. At the U.S. Geological Survey's Astrogeology Science Center (ASC) in Flagstaff, Arizona, scientists explore the surfaces of other planets via images from satellites and space probes. These images are often stitched together to create 3D and topographic maps of planetary surfaces, which is vital work in ensuring the safety of eventual landing craft, whether robot or human. One such spacecraft was the Curiosity rover, which landed on the surface of Mars in 2011. ASC played a critical role in ensuring the safe landing and operation of Curiosity by determining the composition and layout of the surface of Mars. Since its landing, Curiosity has transmitted over 700,000 images of the planet's surface back to earth[2], providing even more data for ASC to analyze.

The most vital piece of software used in the analysis and transformation of planetary image data is the Integrated Software for Imagers and Spectrometers suite, or ISIS. However, ISIS is a set of command-line tools initially developed in 1971 with no support for cloud processing. As an initial solution to this issue, ASC developed a wrapper around the ISIS suite. Using this utility, available at astrocloud.wr.usgs.gov, scientists can submit images along with a set of ISIS commands to run on those images. Users then receive the output images via email.

The main concern that ASC has with respect to the Astrocloud image processing pipeline is that the set of available ISIS commands is limited and static. Users can only perform certain ISIS commands on input images, and the order of those commands is fixed. The ability to configure commands is limited, the ability to specify multiple sources for input images is restricted, and output is only available via email.

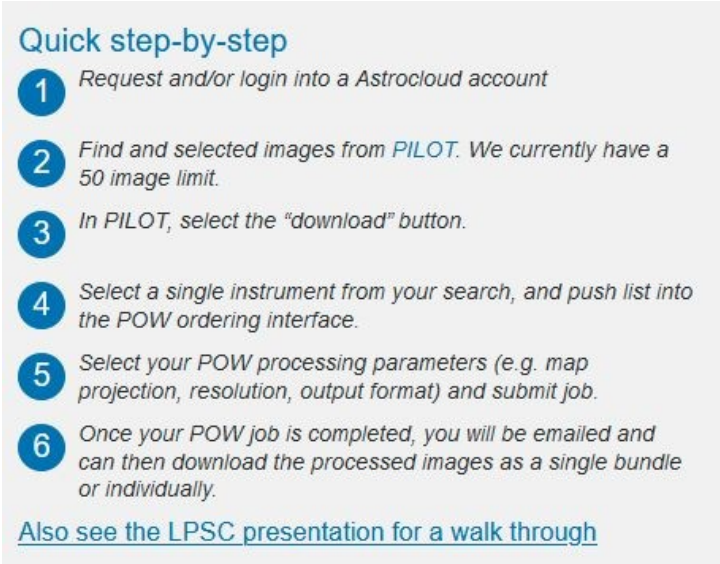
Team Osiris will develop a directed, acyclic graph of ISIS image processing commands. Users will be able to select one or more input images, choose processing nodes from a catalog of ISIS commands, configure and customize those processing nodes, then link them together in any order. Each node will take the result of the previous command as input, allowing for a dynamic image processing pipeline which can stream output to a configurable destination.

[1] https://www.nasa.gov/sites/default/files/files/SEINSI.pdf
[2] https://mars.nasa.gov/msl/mission/overview/

Problem Statement

At USGS Astrogeology Science Center, researchers need to provide a science ready copy of the data they have collected. One of the tools used to get this data ready is an image processing suite that is ISIS3. ISIS3 is a very complex set of tools and requires extensive knowledge of each of 300+ commands to use elegantly. So, to help aid researcher's use of these tools and those alike, USGS has put in place the data pipeline POW (Map Projection on the Web) that is hosted at https://astrocloud.wr.usgs.gov/. POW will take in images requested by the researcher and provide them with calibrated cartographic images that can be used in geologic mapping or used in other scientific applications such as MatLab and ArcMAP.

POW employs a 6 step process in order to use the data pipeline which they have provided a quick overview of on the homepage of the POW. The first and second steps listed on the diagram are preliminaries to getting to the core of the pipeline and require integrated tools Pilot and Astrocloud. USGS is currently working on removing the need to log in through AstroCloud. Once the researcher has collected their photos from PILOT, PILOT will redirect the user to the POW portal where they can select the process or defined workflow that they will be using from the ISIS suite, select the map projection type, and the output format. Once the job is submitted the user's request is then converted into a xml document and sent off to a processing cluster of 132 CPUs to be run through the ISIS 3 processes. After the final process is done the user will be notified by email that they can download the final images.



The problems with this current workflow are the following:

- ISIS processes have to be run individually with this implementation. This means if a researcher needed to have different processes done they would have to submit each job individually instead of chaining the tasks automatically.
- The current graphic user interfaces do not provide any means to set up a multiple program process in order for the user to create a dynamic data pipeline.
- ISIS consists of 300+ separate tools which greatly increases the difficulty for the user to complete sequential tasks manually
- currently the output only goes to one location where the user can download the documents

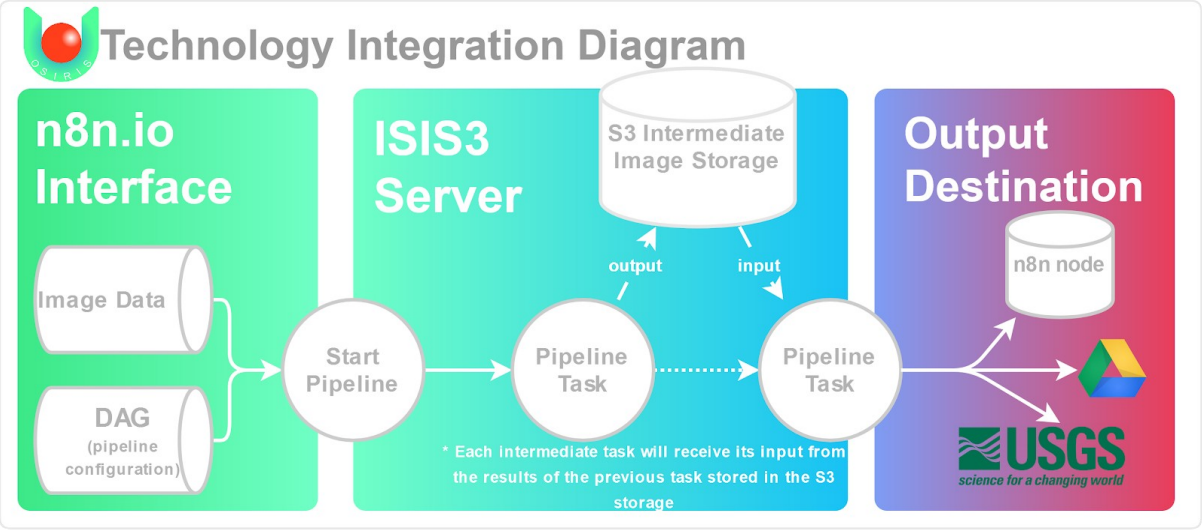
- The entire process is tied to the server system at USGS with no way of a researcher to run ISIS locally on their systems.

The USGS ACS wants to focus on these problems in order to have a faster, more user friendly way of using ISIS tools for researchers and those alike. By fixing these issues USGS believes that they can lower overall costs both in money and time and greatly increase the reach of the ISIS3.

Solution Vision

Team Osiris aims to address these problems by essentially rewriting from step 3 of the original workflow up to the final steps to generate an output for a user of the workflow. This rewrite will change the POW into a graphical user interface in which a user can connect tasks using nodes and customize the parameters within each task. This interface will also allow the user to oversee the tasks as they are completed. Once the tasks are complete the user will be able to choose an output destination for their files instead of having to download a bundle of files from the USGS servers. The end goal is to give the user the ability to choose from most of the ISIS tools individually as well as choose from predefined workflows that are commonly used to bring into their workflow anyway they seem fit. This method of having a drag and drop interface greatly reduces the need for extensive knowledge on the ISIS3 tools.

To get an idea of what the workflow will look like compared to the current, we have created an overview diagram of how the system will connect together.



At the start of the workflow we will have a graphic user interface powered by n8n.io's DAG creation interface. This interface will be hosted on the USGS website similar to the current UI. The user will have a url to the image data provided by PILOT and will have a list of nodes that they can drop into the DAG creator and link them as they see fit. Once the user has started the process the image data will serve as the input for the first node. A node will start its task within

the ISIS server with the given input and optional parameters and dump the output into an intermediate storage so that it can be fetched by the next node. the final node the user provided will simply fetch the final data from the storage and send it to an output of the users choice. Once the pipeline is completed all the data generated and stored within the intermediate storage will be deleted.

The main computational process and data transformations will take place on the ISIS server hosted within USGS similar to that of the current implementation. This allows the user to not have to rely on their own hardware as some tools provided with ISIS can use a lot of processing power. Using the already established processing cluster will also allow multiple different pipelines to be run at a time.

Using this implementation we hope that our sponsor will be able to better their workflow with only improvements to the system in place. Our sponsor would only need to simply replace the POW piece of the workflow with our implementation with only changing the destination of where PILOT sends the information for the image data. Once in place our solution will solve issues with creating multiple processes at once as well as getting the ability to change output destinations. minor step backs for our client will be integrating the logs of each process and data pipeline as ours will not play well with the current log system.

Before this current solution, we had explored the idea of creating a user interface from scratch using html css and javascript plugins. this interface would communicate with an apache airflow based pipeline with all the ISIS commands stored within it. This solution had the benefit of being able to be deployed locally, but creating such a solution would require much more time and knowledge that is out of the constraints of our project deadlines. installing this locally would also require more storage space than we think is worth. Both of these problems were pivotal in the construction of our current solution.

We at Team Osiris believe that with this solution we will not only be able to adhere to the problems with USGS's current system, but also as a result, the solution will be able to open up the ISIS tools to a broader user base. Researchers independent of USGS will be more attracted to the user friendly interface and be more encouraged to contribute to the ever growing collections of science ready images available to us. This surge in data computation can take our steps into understanding our cosmic world and turn them into leaps.

Requirements

Domain-level Requirements

Team Osiris needs to develop a flexible interface for processing planetary image data

1. Images will be processed as a pipeline, or a directed, acyclic graph of image processing steps
2. Users will build custom graphs in order to represent the full process of manipulating their image data
3. Graphs will be send to a processing server, along with input images
4. Users will upload one or more images as input
5. Users will choose a customizable output area

Functional Requirements

Images will be processed as a pipeline, or a directed, acyclic graph of image processing steps: This domain-level requirement will be implemented in part by the n8n[3] pipeline design framework, with significant customizations by Team Osiris.

1. Be able to drag and drop commands: Each ISIS command will have a corresponding node in the n8n user interface.
2. Be able to chain commands together in any order: By using n8n and making the commands into nodes, we can take the output of one function as the input to another. The drag and drop interface will allow users to chain the commands together.
3. Be able to store data between each step
    a. This will allow the output of one function to be used as the input to another
    b. Intermediate data will be stored in a minio[4] S3 server
    c. Nodes will push to S3 after processing
    d. Nodes will pull from S3 before processing
4. Output data to the specified location and in the proper format
    a. Using already built ISIS commands, we can export data into the proper format. N8n has nodes already that will allow for exporting files.
5. Provide an easily accessible user interface
    a. Web interface through n8n
6. Keeping the interface and software secure
    a. Prevent users from getting access to data they aren't supposed to be able to see
    b. Prevent users from running system commands other than those within ISIS
    c. Disable functionality built-in to n8n that is not needed for the project

[3] n8n.io
[4] min.io

Users will build custom graphs in order to represent the full process of manipulating their image data: This domain-level requirement will also rely heavily on Team Osiris customizations to the n8n pipeline framework.

1. Be able to build a graph via the n8n user interface
    a. Built-in n8n nodes must be disabled and replaced with ISIS command nodes
    b. n8n must be branded for ASC
2. Be able to save and share a graph designed in the n8n user interface
    a. n8n has the functionality to save a graph
    b. Files are formatted in JSON and can be published as a simple JSON file
3. Be able to load a previously saved graph into the interface
    a. n8n can load a previously-saved JSON
    b. Create a custom node for loading a workflow via URL

Graphs will be sent to a processing server, along with input images: By decoupling the web UI server used for creating graphs from the actual processing server, we can ensure scalability.

1. Nodes within the graph must send an ISIS command to an ISIS processing server
2. Nodes within the graph must send an input image to intermediate S3 storage
3. ISIS server must parse and execute an ISIS processing command for each node in the graph
4. ISIS server must pull input images from intermediate S3 storage
5. ISIS server must push output images to intermediate S3 storage

Users will upload one or more images as input: Selecting one or more images to process involves custom n8n nodes. Preparing images for processing involves uploading them to S3. Processing the images involves having processing nodes download them from S3.

1. Be able to upload a single image file
    a. Create a custom node for n8n in which users can drag-and-drop or upload an image file via form
    b. Send image to temporary S3 storage
    c. Make the S3 URL available to subsequent processing nodes
2. Be able to upload a CSV with URLs of multiple files
    a. Create a custom node for n8n in which users can drag-and-drop or upload a CSV via form
    b. Validate the CSV, make sure it contains valid image URLs
    c. Pull in the images, upload to S3 storage
    d. Make S3 URLs to the images available to subsequent processing nodes

3. Be able to choose one or more images from pilot.wr.usgs.gov and process them
    a. Create a custom n8n node for interfacing with PILOT
    b. Have the node communicate with PILOT, pulling relevant image URLs into S3 once they're selected via the PILOT interface
    c. Make the S3 images pulled from PILOT available to subsequent processing nodes

Users will be able to specify processing options for each step of the pipeline: When a node is selected the user will be able to choose the type of map projection and the file format that the finished results will be converted to.

1. Be able to select map projection for each task
    a. each node will need to have a list of possible map projections
    b. ISIS server must be able to parse and add map projection into the parameters of the task.
2. Be able to select file format for each task
    a. each node will need to have a list of possible file formats for the output
    b. ISIS server must be able to parse and add file formats into the parameters of the task

Users will choose a customizable output area: Images are stored in a minio S3 server between processing steps. Outputting them involves transferring images from the S3 server to a desired output location.

1. one form of output will be to connect to the Leaflet map viewer.
    a. output needs to be the Cloud-optimized GeoTIFF (COG)
    b. output needs to also include a Spatio Temporal Asset Catalog(STAC) JSON record.
        i. this record will need to describe the files from the output

Performance Requirements

Usability:

- Usability of the user interface should be easy to understand and use right when you open the interface. The layout should be labeled enough that the user can see which of the blocks they want and a simple click of connecting the blocks together in order to create the pipeline. Having a layout that is simple to see which blocks are the most common in creating a pipeline and can drag and drop right into the pipeline. The blocks themselves should be distinguishable from each other so the user knows which block is which. In addition, when clicking on it have the labels that describe to the user in what the block needs.

- Training required of the user interacting with the pipeline should have the basic knowledge of drag and drop interfaces. A simple click/drag of one of the nodes and dropping it in the desired order to create a pipeline. The user should also have a basic understanding of ISIS functions and how to use them to create the end result that they want.

- Responsiveness of the web interface should be highly responsive in that it should take less than a second. When the user drags and drops blocks into the diagram, there should not be a delay of more than the second. In addition, the web interface should show the current steps and progress so the user can see where the pipeline is currently at in the process.

Scalability:

- Scalability of the storage, processing, and user interface these three needs to be able to scale independently from each other. The storage needs to be able to store a little data to a large amount of data during each of the steps in the pipeline. The user interface should be able to help create a one step pipeline to a multi-step pipeline in less than a minute. The scalability of the processing is the most important since it deals with both the storage and user interface. Scaling the processing should take an average amount of time of 5 mins when doing a short task and a long task

Environmental Requirements

There are a few constraints on this project that must be followed in order to ensure the clients' desires for the final product have been met. One constraint is that any workflow must have the capability to process multiple images at once. Previously, any image handling could only be done one-at-a-time, which resulted in a longer wait-time for any user that wanted to process many images. USGS would like up to 100 images to be processed in the new workflow.

Another constraint for the project is that the workflow must be accessible and adjustable for users. Not only should the UI for the workflow be easy to navigate, it must also allow the user to adjust the nodes (and therefore commands) that the images can be run through based on their own desired data outcome for the images.

The last major constraint is to make sure that the workflow functions with existing USGS software. As there is already a site that contains all of the image data, the workflow Team Osiris creates must connect and cooperate with that site cohesively so that everything functions seamlessly and smoothly.

Potential Risks

Potential risks are defined as the possibility of danger or problems. They can range from a small risk that is overlooked like a small math error that calculates direction to big risk that is too important that we need to carefully observe like a math error that calculates the amount of oxygen for deep sea divers. These are always on the back of any team's mind when building a project. For team Osiris, there are several potential risks to think about. Some of these risks are image data input, valid ISIS3 commands, connections between the pipeline, and output storage.

		Impact				
Probability		**Trivial**	**Minor**	**Moderate**	**Major**	**Extreme**
	Rare	Low	Low	Low	Medium	Medium
	Unlikely	Low	Low	Medium	Medium	Medium
	Moderate	Low	Medium	Medium	Medium	High
	Likely	Medium	Medium	Medium	High	High
	Very Likely	Medium	Medium	High	High	High

One of these potential risks is when we accept image data input of 30 or more images into the pipeline. The fact is if one or more of the images are not the correct image type and the pipeline does not handle that correctly it would hold up the entire pipeline and not continue or might crash part of the pipeline. In addition, having the user search through the images looking for which image is incorrect using up valuable time to compile the images. The probability of this happening is moderate to likely and given that it will stop the entire pipeline before it starts we put this as a major impact. This puts this risk at a high risk value. To minimize this risk we will put in place a filter within n8n front end or the back end server to make sure that all file types are valid for the task before it is run.

Second potential risk is validating ISIS3 commands and passing them through. The risk with this is that if an invalid ISIS3 command is passed through then the user would have access to part of the ISIS3 server that they should not have, which is a breach in accessibility. Depending on what the ISIS3 command is and what it outputs. The output could crash the next steps in the pipeline task. The probability of this risk occurring is rare as there will be a set of predefined ISIS3 tasks for the user to choose from. If the user did manage to run restricted commands though it could cause extreme damage to the back end server giving the risk a medium risk level. In order to try to avoid this we already have a predefined set of task nodes the user can choose from and if the user tries to inject their own commands we have a list of valid commands on the back end server that when checked will return a 400 error if the command is not on the list.

Third potential risk is connection between the n8n DAG to the ISIS3 server to the output. The risk with this is that if the connection is not properly set and secure then it would break down the whole entire project, which is part of the purpose of creating this connection between these. Having the connections fail during the pipeline process would in turn create a problem for the client to fix these connections on their time. If the client could not fix the failing connections the client would think about abandoning the project in search for a different way. In addition, it would be a waste of time for everyone involved. The probability of this happening is moderate as people always have connection issues and like stated before the impact this would be major as the user would not be able to complete their pipeline. If this is happening while the user has the project running locally (without the use of USGS servers) we will direct them to use the USGS site instead. If this is happening on the USGS servers, it would be something that USGS would have to handle. jobs that fail during this time will still have files from their last completed task and will pick up from there.

Fourth potential risk is that having a cloud based storage Amazon s3. This risk with this is that it is operated by a third party and its cloud based. Having it cloud based is good in the sense that you have more local storage for other things but when having no access to the internet would mean no access to the data at all. Including this risk is it being from a third party, so if the third party Amazon were to have a crash in its system. Users would have to wait on Amazon to get their system back up which would cost time and potentially money on the users waiting to access the data again. This is a low probability extreme impact risk that we would be passing on to the amazon s3 cloud services to handle.

Project Plan

In order to achieve the final goal our team has set for this project, we plan to break things up into the following five aspects: making an accessible server, creating intermediate storage for the workflow to utilize while processing data, making the uploading to/downloading from the workflow and processing in between nodes function smoothly, programming nodes that will contain ISIS3 commands, and enforcing security throughout everything. These five steps will be completed simultaneously for the most part; some parts may need to be worked on/finished before we can move on to others, but a majority of the portions of this project can be worked on at the same time.

A server is needed to host the workflow. The team lead, Evin, has the most familiarity of the group with Node.js and n8n, so he will take the responsibility of working with those for the server. In this project, the server is a key aspect; the server needs to be accessible, and is the “base” from which we will build the rest of the pipeline.

While the images are being processed by the pipeline, the image data requires a storage location for it to remain in. Rebecca has familiarity with the USGS image data we will be working

with, so she will be in charge of creating the intermediate storage space, which will likely be a database.

There needs to be a way to get the image data to and export it out from the pipeline. Darius will take over the I/O protocol of the processing pipeline, ensuring that data can flow seamlessly into and out of the workflow, as well as in between each of the nodes in the workflow.

The nodes that will be used in the workflow are a vital part of the overall project, as being able to use the workflow is impossible without the nodes containing the ISIS3 commands. Milton and Sara, both with a comfortable confidence regarding working with Python, will work to wrap the existing commands to function within the pipeline.

Security will need to be enforced throughout all parts of the project that Team Osiris will be developing. Security in this case will mean making sure that no one (outside of the team) can alter any portion of the workflow and render it unusable.

		01/11/21	01/18/21	01/25/21	02/01/21	02/08/21	02/15/21	02/22/21	03/01/21	03/08/21	03/15/21	03/22/21	03/29/21	04/05/21	04/12/21	04/19/21	04/26/21	04/30/21
Milestones		Kickoff!				Domain Req 1 & 2: Graph-based Processing				Domain Req 3: Processing Server				Domain Req 3 & 4: Image Input/Output				Release!
Task	Assigned To																	
ISIS node development (n8n)	Evin																	
Frontend Testing / Validation	Sara, Milton																	
S3 Frontend I/O	Rebecca																	
Image input (upload)	Darius																	
ISIS command development	Sara, Milton																	
Backend Testing / Validation	Evin, Darius																	
S3 backend I/O	Rebecca																	
Image output (download)	Darius																	
ISIS node integration (n8n ↔ backend)	Evin, Sara, Milton																	
Backend Integration Testing / Validation	Rebecca																	
Frontend Integration Testing / Validation	Rebecca																	
Integration with USGS Pilot	Evin, Milton, Sara																	
Integration with USGS Storage	Darius, Rebecca																	
Final Testing	All																	

Functional Requirement: Graph based processing (UI)
Functional Requirement: ISIS processing server
Functional Requirement: Pipeline I/O
Functional Requirement: S3-based Storage (UI Server)
Functional Requirement: S3-based Storage (ISIS Server)
Non-functional requirement -based testing
Integration Task

Conclusion

At USGS Astrogeology, scientists create maps based off of satellite and telescope images. These images are processed through a program called ISIS, which is run by using the command line. Currently the astrocloud pipeline is used to help with image processing, but it is limited in commands that it can run, and the commands must be run in a specific order. Our goal is to develop a directed, acyclic graph of ISIS image processing commands that will make it easier for users to retrieve the images, give those images to functions, and reach the end result.

The user will be able to use the supported commands in any order, using the output of one function as the input to another, and retrieve the final processed images. To accomplish this, we will be creating nodes for each function which can be connected with arrows to the next function. The end result of our product will be responsive when users drag and drop nodes, with no visible delay for the user adding nodes, and will display the current pipeline progress. The new pipeline should be easily usable for anyone with basic ISIS knowledge.

We will be using n8n to create the user interface, which will allow us to create the nodes and display the current progress of the pipeline. Any data that needs to be stored between steps will be stored using a minio S3 server. We will use Python to communicate between the S3 server and the pipeline. The ISIS server will complete the tasks specified in the pipeline. In addition, users can save and modify pipelines so they do not have to recreate the pipeline steps if they want to run the same program again for new images.

Glossary

ASC - Astrogeology Science Center
ISIS - Integrated Software for Imagers and Spectrometers suite
PILOT - Planetary Image Locator Tool