Technical Feasibility

11/8/18

Team: Orion
Sponsor: USGS

Jun Rao

Brandon Kindrick
Chadd Frasier
Yuxuan Zhu

Table of Contents

Introduction:

NASA has made leaps and bounds in planetary science in the recent decade, but NASA is not solely responsible for the success. NASA works very closely with the USGS for all their data crunching needs, so much so that the USGS has spent thirty plus years developing the ISIS software. The Integrated System for Imagers and Spectrometers is a binary data analysis tool for reading NASA's data files that have been generated from Deep Space Imagers. We are Team Orion, and we have been given the pleasure of tackling the Planetary Image Caption Writer Project from the United States Geological Survey. Team Members include Brandon Kindrick, as the team leader, Yuxuan Zhu, as code designer and Chadd Frasier, as version controller. Using ISIS, our client Dr. Laszlo Kestay takes data files from NASA and prepares them for publication using the ISIS3 software. The problem is that our client's process could take anywhere from 1 to 2 hours for a single publication image, making figures very difficult to add to research papers quickly. This is slowing down the speed at which articles can be published and intern slowing the speed at which scientific knowledge spreads. He finds himself often not wanting to add images to publications He would like to have a much easier way of cropping and adding important icons, like a scale bar or a north arrow, to the image. Dr. Kestay is also very keen on the idea of making the ISIS3 software available on a Windows machine which is not possible with the current version. In order to create a fast and easy tool that works on all platforms, we will be designing a web application that will allow the server to do the grunt work of running ISIS3 while the user can have the speed of a web app.

The purpose of this document is to analyze the technical challenges of this project that our team will have to overcome in order to succeed. This document will analyze and report the reasoning behind our chosen solutions and possible alternatives. Section 1 will include the Technological Challenges where we will be presenting the problems that we currently face in all parts of the system. Section 2 will be the Technology Analysis where we will be discussing in detail our first choices for software tools and frameworks as well as possible alternatives. We will provide some evidence that the solutions presented are truly possible and will work. The last thing for this section will be the explanation about what we plan to do for the Technology Demo later in the development process. Section 3 will contain the overall discussion of how the tools presented in Section 2 will cooperate to form a working product. Lastly, in Section 4 we will be a comprehensive summary of everything that was said in this document.

Section 1: Technical Challenges

- We will need ISIS3 to communicate with our web app:

  We will need a user-friendly tool(web app) to call ISIS3 commands in real time, and have their results displayed in the web app. The reason we will need a server for this is that ISIS3 decodes binary data files and presents the metadata and images in USGS specific files. We would have to read through the ISIS algorithms and copy hundreds of thousands lines of C++ into a web framework. We simply would not have enough time to research the NASA file format and write a program for decoding these huge files. ISIS also uses complex physics and math equations. To remake these, it would require tons of research to implement it properly. Instead, we will be able to utilize the work that has already been done.

- We will need a UI that can add publication quality graphics to the image.

  Metadata will need to be able to be exported to a text file as well as exported as a whole figure with the image. Our UI will needs to be able to add scale bars that follow the guidelines outlined here: https://ngmdb.usgs.gov/fgdc_gds/geolsymstd/fgdc-geolsym-sec35.pdf. The scale bar needs to have the ability to add a Digital Number Value (DNV) to the bar. Our UI will also need to be able to add indications for Sun azimuthal direction and Sun elevation. Using symbols found here: https://solarsystem.nasa.gov/resources/680/solar-system-symbols/. Our UI will need to be able to add indications for Observer azimuthal direction and Observer elevation.

- We will need the product to run on Unix and Windows Machines, using the ISIS software.

  Currently, ISIS3 can only be installed on Linux and MacOS machines and is incompatible with the Windows operating systems. And to use the corresponding functions of ISIS3, the user must install the complete ISIS3 system on the computer and know the corresponding command line operations. Many scientists do not have professional computer experience, which brings them a lot of trouble. The technical challenge facing our team is to explore how easy it is to use the convenience of ISIS3 and eliminate the tedious task of installing a complete ISIS3 system. If we achieve the above goals, we will continue to pursue perfection, just as Dr. Kestay is also very keen on the idea of making the ISIS3 software available on a Windows machine, we will continue to improve and strive to make it available on Windows.

Section 2: Technical Analysis

This section aims to address the technical challenges outlined in the previous section and the possible solutions to those problems. Our team has carefully reviewed and analyzed each of the technical challenges that lie ahead. We have researched some different solutions and based on that, we have decided on a few solutions that we feel are the best to pursue.

- Issue 1: ISIS communication with Web App

  The ISIS3 software has built-in functions for extracting metadata, making calculations for sun location and also essential functions for editing images. ISIS3 is crucial for us to implement our app without “remaking the wheel.” Some of the functions require a strong knowledge of trigonometry, calculus, and astrophysics to calculate. Remaking this alone would be a project unto itself, and that is why we think it’s necessary to use ISIS3 and its accompanying functions in our web app. Our web app will need to take input from the user and then call the respective ISIS commands. We feel that the best way to do this is to run ISIS on a server for the backend.

  There are many options when it comes to server hosting. Servers range from free trials to $50 a month. Of all of the options, we narrowed it down to four, we could use the CEFNS server hosting, we could pay for cheap server hosting on Digital Ocean for ~$10 a month or we could buying and create a server from hardware at the NAU’s Property Surplus (~$25). CEFNS server space is free which is a huge plus. On the downside, we have no control over speed and need to get permission to run ISIS on the server. The second option, paying for a DigitalOcean server allows us to run any program and do not require prior permission. We would need to pay monthly for this service though relatively cheap still not the best option. The last option is the least realistic, but it is an option if the other two didn't work out. It allows us total freedom to run any program but at a high cost. We would have to buy the server and then pay to run the server at someone’s house. Lastly, we could use server space at the USGS office.

  Of the four choices, we decided to host the server on the USGS office’s server. Our client, Dr. Laszlo Kestay has informed use we could easily gain server space on the USGS server. This will be cost effective and will not require a new installation of ISIS and we already know the USGS servers will permit ISIS to run at reliable speeds. Hosting the server at USGS will yield massive gains for our project and will cut down on cost of development.

- Issue 2: Web App communication with ISIS

    Our front-end web app will serve as a tool for editing photos that are uploaded by researchers. The web app will only be the GUI for uploading images, selecting what data to display, etc. All of the calculations will be done on the back end and then relayed back to the front end web app. We ideally will have a server written in Node JS which will be running ISIS3 and the front end web app would also be written in Node JS. This will make interaction much simpler. The backend will get input from the front end and then pass it into ISIS. The results will then be given back to the backend and then to the front end to be rendered/ displayed to the user.

    The first option is using Node JS. The node can function as a backend server and also can run other programs on the said server. We also may be using Node or base JS for our front end, so this would eliminate the need to use multiple languages in our project. Not our whole team is familiar with JS which means we will need to set time aside to make sure all team members are comfortable writing/ reading it. Another option would be using Python Flask. All members are familiar with Python so it would be easier for all of us to work on. It also can run other programs on its servers. This would require us to use different frameworks and different languages and mesh them together. This could cause issues in the future that we are currently unaware of.

    We decided to go with Node JS. We decided that using JS for our frontend and backend was the best way to go. It also simplifies communication between the backend server and the front end web app. Node JS has C++ add-ons which allow for Node JS to execute C++ commands in a natural JS manner. Node JS is also very easy to pick up, and all members are willing to learn it. If Node Addons do not seem fit, we can go with a server message queueing software to translate commands between C++ and Node.JS.

    To demonstrate that the concept will work, we will build a server in Node. We will also build a simple GUI with one or two buttons. These buttons will call functions on the server. These functions will then in turn call a simple function on ISIS that the server is running. This will then be returned to the front end GUI.

- Issue 3: GUI with ability

    A user-friendly "viewer tool" for interactively exploring ISIS images and converting sub-images into publication-ready data and figures. Specifically, the tool will allow users to load one or more ISIS images, and from there the user can extract and calculate geospatial metadata for these images with simple button clicks. The GUI will allow users to view and interact with the data, and export images and metadata in an easy-to-use, standardized format. This will facilitate clear and complete communication to technical and non-technical audiences.

    To make it user-friendly, we will try to build a Responsive or Mobile-First Navigation, keep our navigation simple, make our function logo easy to find, make our website fast and make content accessible to any audience. Also, we will try to use web2.0 design by using bright, cheerful colors to dominate the website. We will use new CSS techniques for achieving rounded corners because the friendliness of rounded corners is in keeping with the comfortable, informal tone of many web 2.0 sites. Many JS libraries interact with maps/images and allow for editing and superimposing graphical symbols. OpenLayers and Leaflet are the two most popular ones, and both have extensive documentation and tutorials.

    We have decided to use OpenLayers over Leaflet, due to more of our group members having experience using OpenLayers. It can add scale bars, crop images, add graphical images on top of a map and has beneficial documentation. To demonstrate this for the in class demo, we will build a simple GUI with Openlayers that will be hosted by a USGS server. This app will have a basic map and will demonstrate map editing functionality.

- Issue 4: Run on Non-Unix machines

    ISIS3 can only be installed on Linux and MacOS machines at the moment. Users on Windows cannot even install ISIS3. This is an issue that limits the user base, and our sponsor want us to consider a solution that would allow users to use ISIS3 without a full installation of ISIS3 on their host machines. There are complex trade-offs in picking the most viable path to make this tool and our team will explore how this could be accomplished most easily.

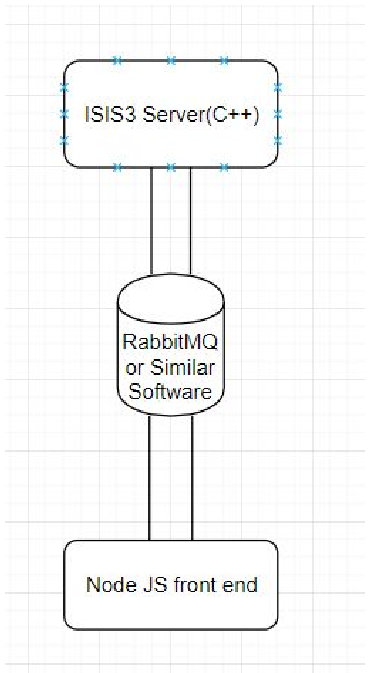
    The first option is to use the web application; the web application is an application program that is stored on a remote server and delivered over the Internet through a browser interface. As issue 1 mentioned, we will run ISIS on a server. Our web application takes input from the user and then calls the corresponding ISIS command that are run on a server. The most important advantage of this option is “Zero install” because all PCs have a browser. But web application relies heavily on the speed of the Internet connection. In the absence of the Internet or its poor connection, it can cause performance problems in web applications.

The second choice we come up with is to design a desktop application for our users. Desktop applications are installed on a personal or work desktop. Word processors and media players can be considered to be typical desktop applications, while an online shopping cart on an ecommerce website can be considered as a web application. Some people argue that web applications are superior for various reasons, while other people say that desktop applications will always reign supreme. Compared with the web application which can be influenced by internet connectivity, desktop applications are inherently independent, so there is no barrier to Internet connectivity. But the disadvantage for a web application is our team has to develop different desktop applications for each platform. Because different platforms have different requirements, this will significantly increase our workload.

In conclusion, the most critical requirement for this issue is that we need to run it on all operating systems, and that is why we are going with a web application. As a result, we should focus on user convenience, we know desktop applications are confined to the physical machine that has it installed and hence have usability constraint. Whereas a web application makes it convenient for users of an OS to access the application from any location as long as they have a computer and an Internet connection.

Section 3: Technology Integration

There are a few key parts to the system that will need to be demonstrated before we can start developing the WebApp front end. Since ISIS is a key piece to the solution we will need to be able to run the ISIS software on a UNIX server because it is already implemented for UNIX. Plus running the backend on a server will allow for our app to be completely portable to any OS. Now in order to call the commands that are housed in the backend we will need some type of internet based messaging system, whether we write our own or if we use a third party software like RabbitMQ.



RabbitMQ is an encrypted message passing software that is compatible to send messages back and forth between many different language frameworks. For example we could send a message using the Rabbit Node.JS commands which can be caught by the Rabbit server software and is queued, the message then is passed to the backend for the C++ Rabbit library to decode, and execute the commands. Lastly, we will have to return the necessary data.

When the data is given to the front end we will need to be able to freely adjust the image's size, color, and orientation. We will also need to add scale bars by using the metadata returned by the server to calculate pixel density per meter and then creating an image scale bar. Our client also really would like a way to add on-screen indicators such as the Sun's location in the image and a north direction indicator. Since the ISIS3 software creates new files we will need a way to reliably pass that data in a way that won't corrupt it. That is another reason for using software like RabbitMQ because it freely supports HTTP and AMQP. Both of these Application layer protocols can be used for reliable data transfer from host-to-host or from server-to-client. The rest of the challenges can all be handled in the Node JS front end from simple data crunching to visible adjustments to images that were returned from the server.

Conclusion:

In this technical feasibility document, we mainly discuss the analysis regarding technology. Through our review of the project, our team hopes to grasp the possible feasibility limits in the project as early as possible; and will not ignore these when doing requirements acquisition and early design. While analyzing technologies, we must have a strong understanding of our project, such as why we make certain design decisions and what kind of results do we hope to get. In the first part, we give a simple introduction to our project. Currently, we are serving for USGS who is under NASA. Because scientists always have to spend 3-4 hours to manage a straightforward image, which causes figures very difficult to add to papers quickly; as a result, we want to design a web application that can interact with a server doing the grunt work of ISIS3 while the user can have the speed of a web app. After that, we entered the most crucial analysis stage of the article. We present the main technical "challenges" foreseen in the project, and then carefully analyze each challenge: discuss the overall solution that needs to be done and briefly outline the possible alternatives in response to the challenge, describe what we have done to test these alternatives, and finally we determine which method we will use. To summarize our findings clearly, I will use a table to describe it:

Technical Challenges	**Proposed Solution**	**Confidence Level(10)**
Running ISIS on a server	Hosting on USGS server	9 It is free and we know it is fast already. This also will ensure that there is no install issue with the current ISIS software

Web App communication with ISIS	Using Node JS Addons or Server Message System	7 It simplifies communication between the backend server with C++ and the front end web app in Node.JS
GUI with ability	Node JS Front End	9 The Node JS front end is solely responsible for the UX. It is for this reason that we will need to work very hard to make a fast and easy to use GUI
Run on Non-Unix machines	Web Application	10 If we use a web application we will solve the issue of portability with less dissection of ISIS3

The above table clearly explains our technical challenges and possible solutions. Fortunately, most of our solutions have a high confidence level. Using a C++ backend and a JS front end, we are confident that there will be no issues in communication between the two. After comparing a digitalocean’s paid services and CEFNS servers with no fees, we decided on the CEFNS server. This is because our client, Dr. Laszlo Kestay has not discussed an available budget and we would likely need to pay for it out of pocket. We will address these problems with our client Dr. Laszlo Kestay next week and gather more information.

Finally, in the Technology Integration section, because we have introduced individual challenges and our plans to solve them. This approach allows us to combine all of these "micro-solutions" come together into a coherent overall system in this section. To better support our ideas, we used the "system diagram" of the envisioned system to show the interrelationships between the main elements. First, ISIS3 is the key to the solution. So, we need to run ISIS software on a UNIX server, because ISIS itself is operating on UNIX. In the front-end and back-end links, to invoke the back-end commands, we need an Internet-based messaging system. This is the bridge in our "system diagram." By considering the encryption in the message passing process, and compatibility with sending messages back and forth between many different language frameworks, we decided to use RabbitMQ or similar software.