User **S**ystem of **A**strogeology **T**echnologies

(**USAT**)

Final Report

Kyle Andrew McGinn, Megan Backus, Zack Ellett, Mikal Ustad

Rev 1.2

5/8/2012

Table of Contents

Website:
http://www.cefns.nau.edu/Research/D4P/EGR486/CS/12-Projects/USAT/

Note: Defined words are superscripted [a-z] at the first occurrence and can be found listed alphabetically in the glossary near the end of this document for reference.

Problem Statement

This project was brought to the Department of Computer Science at Northern Arizona University by the Astrogeology Research Program at the United States Geological Survey (USGS[q]) in Flagstaff, Arizona. The sponsors from USGS are Trent Hare and Moses Milazzo. The Astrogeology Research Program is responsible for processing data from missions to planetary bodies in the solar system. This data is actually thousands of images, each with extensive metadata[i] that can be many gigabytes in size. The default data formats are not very useful for human analysis, and many of the missions use different formats altogether. To help assist scientists in the analysis of this massive amount of data, the USGS created a suite of tools called ISIS[e], the Integrated Software for Imagers and Spectrometers.

The current version of ISIS is composed of almost 300 different programs. Each of these programs adjusts the given image for analysis. For example, some ISIS programs convert the image file type; others sharpen or blur the image. The interface for ISIS is the command line. There are rudimentary graphical user interfaces (GUI[d]), but these are used more for training purposes. There is also no way to link multiple ISIS programs together without writing a shell script[m].

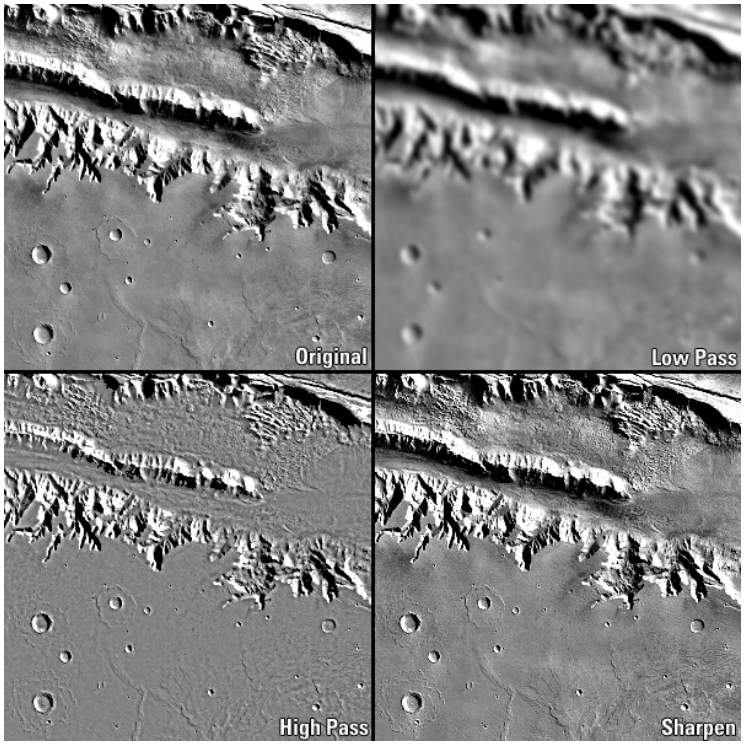

Figure 1: Example of three ISIS image filters applied to the same image

The USGS Astrogeology Department desires a facelift for ISIS. The proposed project is to create a single, centralized[a] GUI. The interface needs to have access to all ISIS programs as well as the capability to link them together in workflows[s]. The project would mostly help training new users as well as offer enhanced functionality for more experienced users.

Process Overview

Design Process

The project began by following the waterfall design methodology[r]. The stratified steps and focus on documentation appealed to the schedule of the Capstone class. However, once implementation was underway, a more agile process was required. The waterfall method was transformed into something more akin to scrum[l]. The scrum method promoted weekly sprints and made it possible to always have a working prototype. The weekly sprints also corresponded nicely with the weekly team meetings. As documentation was still a required part of the design process, each deliverable was incorporated into the sprint schedule.

Deliverables

The deliverables for the Capstone class consisted of two major documents as well as weekly reports. The weekly reports covered what was completed, assigned tasks, and any difficulties that occurred for that week. Besides allowing the team's advisor to monitor the progress of development, it helped keep the focus on the project and minimize cascading difficulties. The first document focused on the requirements of the project and included some initial design ideas and mockups. The second document covered more in-depth design aspects.

The project also includes documentation within the interface. A user is able to inspect each tool and read a description about what the tool does, including example uses. An accompanying installation guide was also created to assist the clients in setting up and administrating the new system once the project is complete. The guide depicts how to use the system as well as how to adjust current ISIS standards to be compatible with the new system.

Timeline

The schedule for the project was purely enforced by the Capstone class deadlines. The project began mid-January and needed to be complete by the end of the semester at the beginning of May. Requirements and design documents were due before the end of February. In addition to the written documentation, there were two design reviews which required the team to present the status of the project to the entire computer science capstone class and faculty.

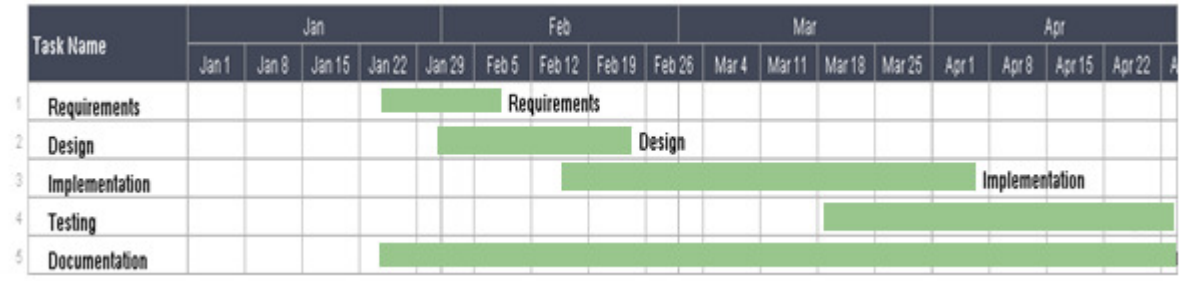

Figure 2: Project Schedule

Implementation began with prototyping and proof-of-concept mockups, which is why Figure 2 shows the requirements phase overlapping with design. Documentation was persistent throughout the project, although it took many forms. The most prominent of these were the deliverables for the capstone class, but documentation also included commenting within the code and ISIS tool descriptions within the interface.

Requirements

The main acquisition of the project requirements was through meetings with the sponsors. When scheduling became difficult, emailing turned out to be the main form of communication. Throughout these conversations, the sponsors' requirements were sorted into primary and secondary goals. The difference between these two categories was defined as follows:

> **Primary goals** were features that the final project needed to have
> **Secondary goals** were features that would be nice to have and would enhance the project

With those two major categories in mind, this is how the project requirements were characterized:

Primary Goals

The primary goal was the creation of the graphical interface. Its required features include:

- Have access to all current ISIS programs
- Be able to create workflows to link ISIS programs
- Save and share workflows
- Include documentation on each ISIS tool within the user interface
- Be able to run on Unix/Linux environments (as ISIS does currently)

Secondary Goals

There were many secondary goals, some of which were out of the scope of the current project. However, all of these were things that were wanted for the future of ISIS:

- Adapting the GUI for mobile environments (e.g. smart phones, laptops, etc)
- Compatibility with cloud computing
- Having an image viewer within the interface
- Improved error checking and notification
- Having a web-based implementation (for accessibility and portability)

Constraints

The images that ISIS works with are several gigabytes in size and the computers the Astrogeology Research Program uses are already built to handle all of the processing needed. This project is only a frontend for all of the processing that ISIS does, so size was not a constraint. The project will be small in comparison to what already exists. There was also freedom in the design of the interface. The only real constraint was that the new system must be compatible with the current ISIS system.

Solution Statement

The Product

In order to address all of the project's primary goals and cover as many secondary goals as possible, the USAT[o] team went through many ideas on how to implement the GUI. After exploring possibilities of using Java Swing[f], Nokia QT[j], and Python[k], the decision was made to use Galaxy[c].

Galaxy

Galaxy is an open source project of the Nekrutenko Lab at the Center of Comparative Genomics and Bioinformatics at Penn State as well as the Taylor Lab at Emory University. It is a web-based platform designed to assist in the creation of bioinformatics workflows.

The main reason for using Galaxy as a basis for the USAT project was that it allowed the team to focus on additional features. Galaxy provided a foundation for the GUI, and allowed more time to be partitioned for tool integration and description. Galaxy tool interpretation is done using XML[t] files to define how each tool appears within the overall interface. This is an added benefit because ISIS currently uses a similar system. The two XML schemas were not compatible, and converting tools from ISIS to Galaxy represented one of the major challenges of the project. The secondary goal of having a web-based implementation was also handled with the use of Galaxy. Furthermore, Galaxy also has provisions for tool documentation, a primary goal of the USAT project.

Functional Specifications

Workflows

In Galaxy's Analyze Data section, a user can view the details of each tool and run the tool as well. Using Galaxy's workflow creation, a user can view and use multiple tools (Figure 3). These workflows create a script or plan of execution for the tools used. Such scripts can be saved and shared through Galaxy. The sharing is facilitated through Galaxy's user system, but the workflow can also be exported and sent via email.

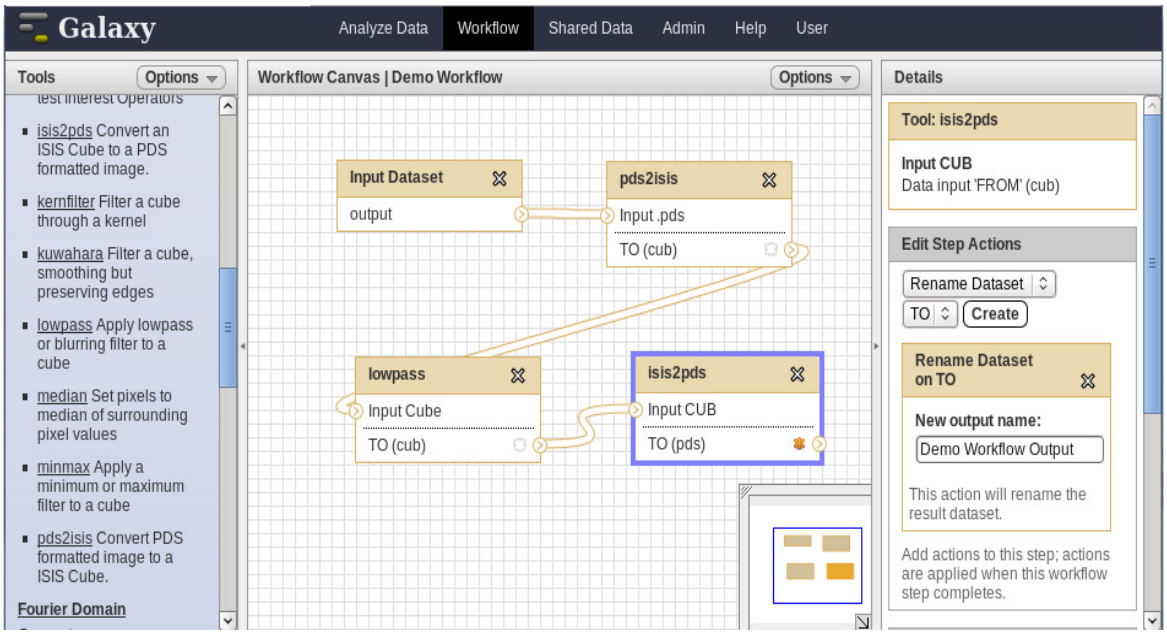

Figure 3: An example of workflow creation in Galaxy

Tool Documentation

There is a consistent list of organized tools on the left side of the Galaxy interface (as seen in Figure 3). A short description accompanies each tool in the list. In the Analyze Data section, a user can select a tool to view more information on it. At the top of each tool window are the parameters for the tool, which can be modified by the user as needed. Below the parameters is a detailed description of the tool. Depending on ISIS's documentation, this can include image examples as well as text.

Users

Galaxy supports a user database. Users have access to previously saved workflows and data. As previously mentioned, a user can also share these workflows with other users. An administrator can also use the user database system to determine access privileges (Figure 4). This includes limiting access to certain tools or data sets.

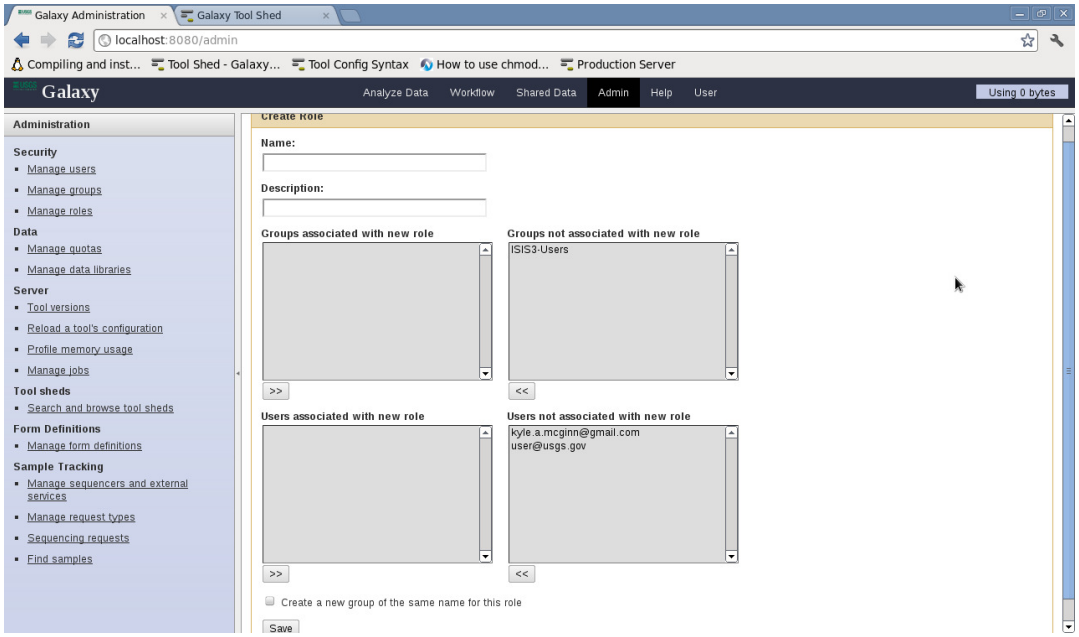

Figure 4: An administrator determining access privileges

Programming Languages

The Galaxy GUI uses JavaScript[g], XML, and Python. JavaScript is used during workflow creation, controlling the drag-and-drop functionality and other visual effects. The XML files handle the dynamic creation of each tool and its parameters. Python does the main computation and calling of the necessary scripts that actually run the tools.

Environments

Because Galaxy is web-based, it can be run on any operating system. The USAT team worked on the project through a Linux virtual machine, but also used Windows for testing compatibility. Galaxy can also be used with a smart phone that can use the web. The current implementation is not optimal, but there is already a basis for further development.

Architecture Overview

With the use of Galaxy, the architecture for this project is client-server[b]. Communication between these two entities is done through HTTP. Figure 5 is a graphical overview of the system's architecture.

The Client

The client side of the server-client architecture handles two things: authentication and the Galaxy GUI interpretation. Authentication involves the retaining the information of log-in status. The Galaxy GUI is interpreted and displayed in the client's browser.

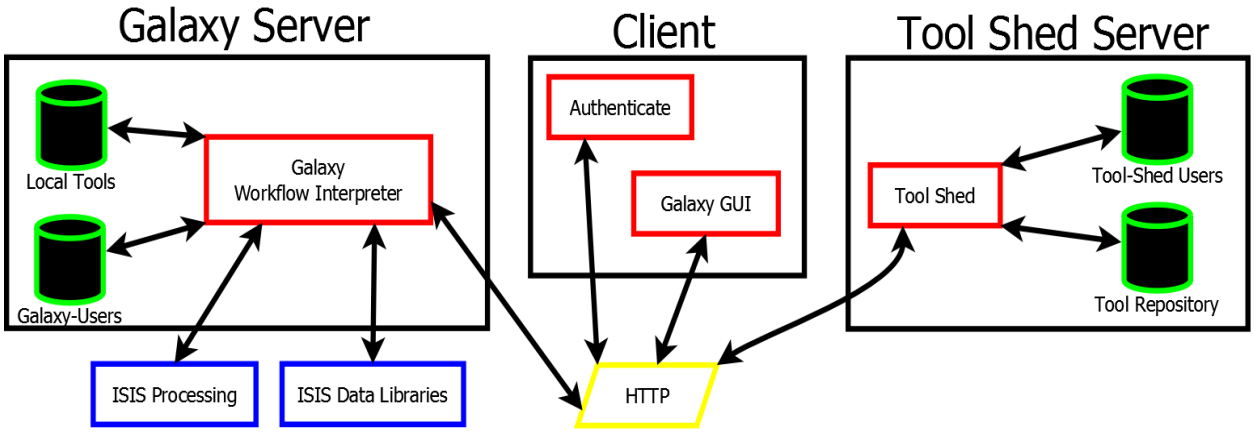

Figure 5: Module overview of the product's architecture

The Galaxy Server

The main Galaxy server can be run locally or remotely. When it is locally hosted, ISIS processing and Data Libraries can be accessed locally as well. The database of Galaxy users and local tools are also stored on the Galaxy server. Workflow interpretation deals with the under-the-hood processing of the tools and their relationship with the processing data.

The Tool Shed Server

The Galaxy Tool Shed server can also be run locally or remotely, and even on the same system as the main Galaxy server. Like the Galaxy server, the Tool Shed server holds information on its users and the tools. However, the tools are the main purpose of this server; it is where they are primarily stored. The tools on the Galaxy server are loaded from the Tool Shed server.

Product Specifications

Galaxy Workflow Interpreter

As mentioned before, Galaxy uses JavaScript for its interactive visuals and Python to process commands in the background. Galaxy also uses JSON[h] for the workflow saving feature. In order to dynamically create tool information, Galaxy interprets XML files. This was one reason why Galaxy was appealing to the USAT team, because ISIS tools also have XML files. However, the mapping between the ISIS and Galaxy files was not perfect. Each ISIS XML file will need to be manually translated to fit the Galaxy XML schema.

Tool Shed

Originally, the plan was to create a program that would automatically convert the ISIS XML files to match Galaxy's XML schema. The imperfect mapping between the schemas created a large hurdle for this tool builder. A lot of Galaxy's features went unused when an ISIS file was converted. The ISIS tools were also created by many different authors and not entirely consistent. In the end, the USAT team utilized another of Galaxy's features, the Tool Shed.

The Tool Shed is essentially a tool repository and management instrument. Tools are stored within the Shed, and the Galaxy servers can connect to it to access tools. The Tool Shed handles update notifications, letting a user know that a new tool can be downloaded or an old tool can be updated. An administrator can also determine who has access to groups of tools through the Tool Shed.

The Tool Shed solved the problem of using tools, but not of creating them. With the tool builder not being an optimal solution, it was determined that manual conversion for each tool was necessary. The initial time investment would be large, but once it is complete new tools can be built without need for conversion.

ISIS Data Libraries

Besides using ISIS tools to analyze data, this project also needs data to work on. Within Galaxy is the ability to link to data libraries. ISIS has many libraries from various planetary missions. The USAT team modified Galaxy so that it can link to these libraries and use the data within them.

Testing

Most of the testing on this project has been functional and unit testing[n]. User testing[p] was limited. Ideally, employees in the Astrogeology department would use the product and provide feedback. However, scheduling proved difficult. Most user responses came from demonstrations.

Functionality testing still proved informative however. It ruled out the tool builder (mentioned above in the Solution Statement). To facilitate testing, the team was able to create and run versions of the Galaxy and Tool Shed servers both locally and remotely. This provided insight into how the clients will use the product, as well as many potential issues that may arise.

Future Work

The USAT team has only been able to manually convert less than 30 of the 300 existing ISIS programs. The first step for the future of this project is to adapt the rest for Galaxy. The USAT team has developed a document to help facilitate this process, with explanations on all the needed conversions.

One of the secondary goals that was not completed was an image viewer within the GUI. One of the ISIS tools currently does this, but needs to be processed as ISIS tool. The main problem is that the images that ISIS works with are very large, sometimes around 12 gigabytes. The developers of Galaxy have been contemplating supporting an image viewer, but are hesitant to do so because of

potential issues related to allowing file uploads. In short, they don't want to become an image hosting service.

The product can currently be viewed with a smart phone that has internet access. However, it does not have a version formatted for smaller screens. Polishing a mobile version of the product is a future project that would further enhance the work done by the USAT team.

Conclusion

The finished USAT project has successfully established a foundation for the future of ISIS. With its built-in documentation and intuitive design, this project will help the USGS employees do more work and require less training. Along with the core functionality, the many additional features implemented by the USAT team provide a solid stepping stone for even more improvement in the future of astrogeology research.

This project has also helped introduce Galaxy into other domains besides bioinformatics. Documentation will be given to the Galaxy developers to show them the USAT team's setbacks and accomplishments.

Glossary

[a]Centralized: one interface that combines all other interfaces.

[b]Client-Server: a computer model where tasks and workloads are partitioned between a provider (server) and requester (client).

[c]Galaxy: an open source, web-based program that provides a scientific workflow, data integration, and data analysis persistence and publishing platform.

[d]GUI: provides the ability to interact with a computer using pictures and symbols, rather than having to memorize many complicated commands and type them in exact form.

[e]ISIS: Integrated Software for Imagers and Spectrometers; an image processing software package.

[f]Java Swing: the primary Java GUI widget toolkit

[g]JavaScript: a scripting language that enables design of interactive sites

[h]JSON: JavaScript Object Notation is a text-based open standard designed for human-readable data exchange

[i]Metadata: a set of data that describes and gives information about other data

[j]Nokia QT: Nokia's cross-platform application framework for GUIs

[k]Python: an interpreted, interactive, object-oriented, extensible programming language.

[l]Scrum: an agile methodology for software development focused as weekly sprints

[m]Shell Script: a Unix shell for a series of commands

[n]Unit Testing: individual units of code/functionality is tested separately from the whole system.

[o]USAT: User System of Astrogeology Technologies

[p]User Testing: tests that involve having an actual user test the system.

[q]USGS: United States Geological Survey

[r]Waterfall Design Method: a sequential software design process focused on documentation

[s]Workflow: a sequence of connected steps that flow from one to another.

[t]XML: a programming language with a set of rules for encoding documents. The difference between HTML and XML is that XML was designed to transport and store data, and HTML was designed to display data, with the focus on how the data is shown.