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

(**USAT**)

Requirements Document

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

8 February 2012

Table of Contents

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

Preface: Defined words are **bolded** at their first occurrence and a glossary can be found near the end of this document for reference.

Introduction

The Astrogeology Science Center located in Flagstaff, Arizona currently uses a software package called **ISIS** (Integrated Software for Imagers and Spectrometers) to manipulate historical and recent imagery data from planetary missions. ISIS takes historical data and cleans up errors made by humans or camera distortions. A combination of programs is required to extract the information out of raw distorted data. However, each program is run through system command line calls or rudimentary **Graphical User Interfaces (GUI)**. The time it takes to learn over 300 different programs makes training problematic and processing imagery an overly tedious task. ISIS is a growing entity, constantly changing and evolving. A solution is imperative; ISIS is becoming progressively gigantic with time.

The solution is to incorporate the complex power of ISIS with the simplicity of a **centralized** GUI, having it act as the middle man between the scientist and numerous manipulation programs. A system that utilizes an all-encompassing GUI will provide elegant organization by combining programs into one process flow, which can be run on a local machine or **cluster**. This system can also provide the ability to save the history and process recommendations to be exported to other users for future use. Furthermore, in order to reduce the training duration, the GUI will need a "help" center that demonstrates and annotates what each ISIS program embodies.

Most of this project is design based, and the implementation will provide a visual representation of the proposed abstraction. Team USAT will provide the inspiration and creative ideas for use in the future development of GUIs.

Problem Statement

The ISIS project has been evolving since the 1970's. It began as the Flagstaff Image Processing System (**FIPS**), where program size was limited, and then changed to the Planetary Image Cartography System (**PICS**), where there weren't any restrictions on program size. The Integrated System for Imaging Spectrometers was developed in the late 1980's, after involvement with the Galileo NIMS mission and then Clementine mission, where ISIS 2.1 was forged between ISIS 2.0 and PICS. The current version, the Integrated Systems for Imagers and Spectrometers 3.0, sprang up in 2001 as the old technologies previously used gave way to newer, more efficient methods.

There is a need for ISIS to undergo another evolution. ISIS 3.0 introduced a simple GUI. However, each of the 300-and-growing number of subroutines that comprise ISIS has its own interface, and is still run via command line prompts and hand written scripts. There is not only a tediousness to using this kind of setup, but also a learning curve. A new ISIS employee needs multiple weeks of training before being able to effectively use ISIS. Even a trained user can encounter errors, but ISIS 3.0 only presents the errors after the whole process--sometimes a week-long affair--is complete.

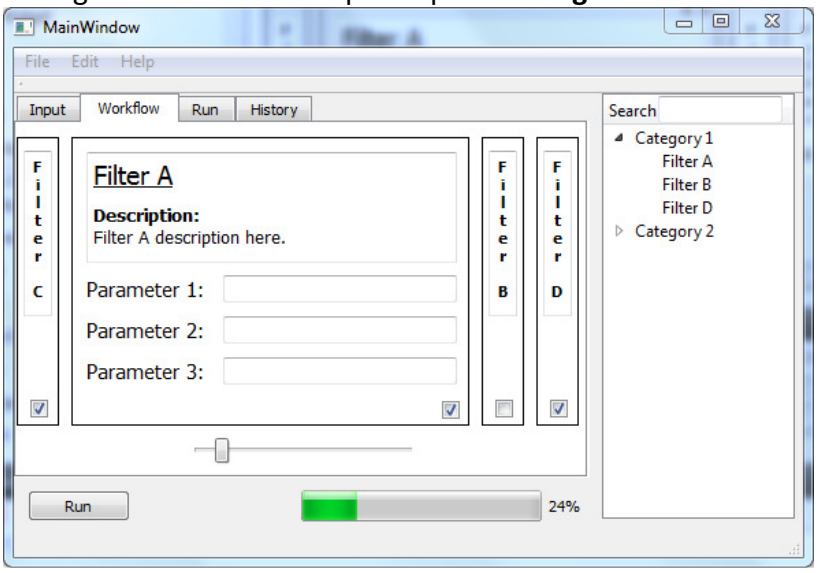
The creation of a more robust and user-friendly interface is a top concern to the progression of the ISIS system. Besides the consolidation of the numerous ISIS programs, the ability to have an accompanying work flow would be a much welcomed feature. The less-immediate future of ISIS should also be addressed, and not just that of regular maintenance. Features such as support for cloud processing to help scientists out in the field still have access to ISIS and the ability to export work flows into other programming languages for portability will provide the ISIS system with far more accessibility and usability than currently exists.

All of these features aim toward furthering the development of ISIS into the most optimal program for the processing of images. Future prospects aside, the integration of all of the ISIS programs into one cohesive interface is the next step is the system's evolution.

Solution Statement

The main concern of the project to improve upon the ISIS system is the integration of all of its programs into one centralized interface. This means providing a way to easily implement such programs onto a given image and to be able to get the desired program without hassle. Having all the needed components at one's fingertips, or at least no more than a click away, is the key.

The image below is a mock-up that presents a **generalized** solution.



The main section shows a list of filters that represent the ISIS programs to be used. One of them is enlarged to view and edit any details pertaining to the program, while the others are minimized. As more programs are added, a user can scroll side-to-side to navigate through them all. The right pane is a list of the programs broken down into predefined categories. A user can scroll through this tree to find the desired program or just use a search field for quickness. This is just the main bulk of addressing the solution to ISIS's centralization.

As indicated by the image above, other features can be accessible through tabs. A "Run" tab would provide a screen where error messages are presented while the image processing is currently running and where a user can pause the processing. The "History" tab is a place where a user can manage **macros**, which are a series of steps that are commonly run together, including exporting, saving, and importing files containing such information. Some recommended macros are also available in "History". The "Input" tab is the place where a user manages the images to be processed. Buttons at the bottom of the window also provide shortcuts to some of the more common tasks found in the tabs.

The whole GUI will embody a uniform design. The tabs, shortcut buttons, and overall formatting are to remain consistent for less confusion and distraction to a user. Different tabs

can easily be added to enhance the solution if the need arises, as well as the subtraction and possible regrouping of the tabs.

Functional Requirements

The **USAT** project aims to improve upon the existing interface for the ISIS suite of programs. Currently, ISIS is just a collection of nearly 300 separate programs that each has its own graphical interface. USAT will be a single, centralized graphical interface for interacting with all of these programs, as well as other related resources. Drawing from the **API**s and **XML** definitions of the existing programs, USAT will be able to **dynamically** recreate all the functionality of these programs based on which are required for the current workflow.

1.0 The centralized graphical interface will provide a consistent way to interact with the application, allowing a steeper learning curve for new users and powerful tools for experienced users.
- **1.1** Improved error checking and progress reports will result in a smoother user experience
- **1.2** Currently, no provisions are in place to report on the progress of the image processing execution.
- **1.3** The ability to preview the results of a workflow by testing it on a subset of images will result in less wasted **computation time**.

2.0 Expected use environment
- **2.1** Hardware
  - **2.1.1** Personal workstations
- **2.2** Distributed systems
  - **2.2.1** Cluster at **USGS** and others
  - **2.2.2** Possible **cloud computing**
  - **2.2.3** Mobile devices
    - **2.2.3.1** Submit jobs to the cluster from a laptop in the field
    - **2.2.3.2** Smartphone

3.0 Software
- **3.1** Unix/Linux based operating systems, including Mac OSX
- **3.2** Existing ISIS package and all required **libraries**

4.0 Operators
- **4.1** Users should have an existing knowledge of the basic concepts of image processing
  - **4.1.1** Experience with software titles like Adobe Photoshop, ENVI, ERDAS, GIMP, etc.
- **4.2** Basic shell and scripting knowledge
- **4.3** Relevant knowledge of the field
  - **4.3.1** Scientists studying astrogeology and related fields
  - **4.3.2** Astrogeology enthusiasts

5.0 External data
- **5.1** API signature of every individual programs
- **5.2** XML definitions for each program

6.0 Internal data

- 6.1 User **preferences**
- 6.2 User-defined macros
    - 6.2.1 Previous session
    - 6.2.2 Pre-defined macros

7.0 Product behavior
- 7.1 The USAT interface will be laid out in a logical and intuitive manner
    - 7.1.1 User testing will help determine and evaluate design decisions

8.0 Product lifetime
- 8.1 Current ISIS interface has been around in some form since 2003
- 8.2 Interface definitions are created as needed when a new program is added to the suite
- 8.3 Ideally, the USAT interface will be in use for 5-10 years
    - 8.3.1 It will sit on top of the underlying programs, so it can be modified as needed.
    - 8.3.2 Additional features can be added without affecting the original specifications and use cases
    - 8.3.3 Each instance of USAT will have access to all resources required for any individual use case
    - 8.3.4 Some resources will be shared across the network

9.0 Map data
- 9.1 Multiple instances could be run on the same machine
- 9.2 Actual processing should be done on a machine with the necessary hardware capabilities.

10.0 Physical/environmental constraints
- 10.1 Operating environment
    - 10.1.1 Linux/Unix based operating system
- 10.2 Hardware capable of processing the workflow.
- 10.3 Physical or network connection to a machine capable of processing workflow.

11.0 Client requirements
- 11.1 New interface
    - 11.1.1 Centralized
- 11.2 Unique perspective on interface and workflow design
- 11.3 Ease of use
- 11.4 In-depth documentation and examples
- 11.5 Powerful tools for experienced users
    - 11.5.1 Ability to export and save workflows
    - 11.5.2 **Shell scripts**
    - 11.5.3 Macros

12.0 Client “wants”
- 12.1 Mobile accessibility
    - 12.1.1 Laptop/smart phone job submissions
    - 12.1.2 Cloud computing options
- 12.2 Access to additional resources

- 12.2.1 NASA image archives
- 12.2.2 Web based information

13.0 Existing products

- 13.1 USAT is specifically unique
  - 13.1.1 Only existing option is the nearly 300 separate interfaces currently in use
- 13.2 USAT will be one unified user interface
- 13.3 No accommodations for new/inexperienced users
  - 13.3.1 USAT will have extensive help and documentation features.
- 13.4 Existing resources must be gathered manually
  - 13.4.1 USAT will have provisions for scraping data from external sources.

Functional Specification

Some of the features of the GUI have already been fleshed out as a quick extension of developing the solution. The following points represent these features and how a user might interact with them.

1.0 Input
- **1.1** Switches focus to input tab
- **1.2** Allows a user to select the input files for the program
  - **1.2.1** Target body
  - **1.2.2** Preferred data server
  - **1.2.3** Specific mission snapshot

2.0 Workflow
- **2.1** Switches focus to workflow tab
- **2.2** Allows a user to build the workflow of the process
- **2.3** Shows all currently selected ISIS programs in the workflow
  - **2.3.1** A user may select and deselect these programs to enable and disable them
  - **2.3.2** Scrolling will shift the focus of the workflow window left or right
  - **2.3.3** Selecting a program allows the user to edit the parameters of that program
  - **2.3.4** Clicking on the "X" or pressing "Delete" will remove the program from the workflow
- **2.4** Clicking the "+" or pressing "Insert" will show the user all possible programs to add to the current workflow
- **2.5** Typing the first few letters of a desired program will narrow the options to match that given string of characters
- **2.6** A user can change the type of any program by clicking "Change" and selecting a different program
- **2.7** A user can change the order of the filters by dragging them within the workflow portal

3.0 Run
- **3.0** Switches focus to the run tab
- **3.1** Shows current running job(s)
- **3.2** Shows list of pending jobs
- **3.3** Shows list of recently completed jobs

4.0 Import
- 4.1 Displays options for importing data describing an existing workflow
- 4.2 Focus is returned to previous window once workflow is imported or cancelled

5.0 Save
- 5.1 Displays options for importing data describing an existing workflow
- 5.2 Focus is returned to the previous window once workflow is saved or cancelled

6.0 Export
- 6.1 Displays options for exporting current workflow
- 6.2 Focus is returned to previous window once workflow is exported or cancelled

7.0 Search
- 7.1 Switches focus to search options
- 7.2 A user may type into a search box
- 7.3 Results are filtered and displayed in real time

8.0 Menu buttons
- 8.1 File
  - 8.1.1 Contains most common input/output and system functions
    - 8.1.1.1 Open
    - 8.1.1.2 Save
    - 8.1.1.3 Import
    - 8.1.1.4 Export
    - 8.1.1.5 Exit
- 8.2 Edit
  - 8.2.1 Contains most common editing functions
    - 8.2.1.1 Undo
    - 8.2.1.2 Redo
    - 8.2.1.3 Preferences
- 8.3 Help Center
  - 8.3.1 Opens the help window or switches focus to help window if open already
    - 8.3.1.1 Documentation
      - 8.3.1.1.1 Lists every available ISIS program
        - 8.3.1.1.1.1 Listed by category
        - 8.3.1.1.1.2 Program information contains a description, author(s), and a usage example
      - 8.3.1.1.2 Lists predefined and user-defined macros
    - 8.3.1.2 Glossary
      - 8.3.1.2.1 Contains definitions of common terms and resources
      - 8.3.1.2.2 Links to external resources when applicable
    - 8.3.1.3 Additional resources
      - 8.3.1.3.1 Contains links to further information and interesting resources that are related

9.0 Errors

9.1 Errors are written to an error log

9.2 Window focus will shift to an error dialogue when an error occurs

- 9.2.1 Processing will pause
- 9.2.2 “Ignore” option closes dialogue, resumes processing, and ignores any possible malfunction
- 9.2.3 “Details” option shows a more detailed error message
- 9.2.4 “Cancel” option closes the dialogue but processing remains halted
- 9.2.5 “Abort” option closes the dialogue and aborts processing
  - 9.2.5.1 The user is returned to the previous window

Non-Functional Requirements

1.0 Development Process
- 1.1 Technologies
  - 1.1.1 Use **QT** GUI framework
  - 1.1.2 Use QT Creator development environment
  - 1.1.3 Use **C++** as primary programming language
  - 1.1.4 Use XML to dynamically generate parts of the user interface.
  - 1.1.5 Use **UNIX** system calls to call ISIS processes
- 1.2 Process
  - 1.2.1 Essentially waterfall process model with additional prototyping and flexible design changes during implementation phase

2.0 **Maintainability** and **Expandability**
- 2.1 Source Code Documentation Standards
  - 2.1.1 Comments
    - 2.1.1.1 Each class will include a brief description as well as the last time it was edited and who made those changes
    - 2.1.1.2 Functions that do not have self-explanatory names will include a comment explaining their function
  - 2.1.2 New processes are continuously being added to ISIS and the developer(s) will be able to ingrate these new processes in less than an hour

3.0 **Understandability**
- 3.1 Learnability
  - 3.1.1 Users of the current system
    - 3.1.1.1 95% of users will be able to perform all the tasks they need to within hours of starting with the GUI
    - 3.1.1.2 80% of users will be able to use new functions and shortcuts after working with the USAT GUI for a week
  - 3.1.2 New USGS employees
    - 3.1.2.1 Reduce ISIS training for new employees to less than a month
- 3.2 **Ease of Use**
  - 3.2.1 The help center will assist users who do not have prior experience with ISIS
  - 3.2.2 The User Interface will be designed to make it easy for users with experience to be able to deduce how it works without constantly referring to the tutorial or help sections

4.0 Performance
- 4.1 ISIS is a **resource intensive** application as is, so the USAT GUI shall perform well enough to have negligible increase in run time

5.0 **Reliability**

5.1 Prevention

5.1.1 Use of a well-developed and documented GUI framework to add stability to the pieces of the program that are not written by the USAT team

5.1.2 Unit testing and usability testing will be used to find errors from special use cases

5.2 Protection

5.2.1 The save features protect users from system failures by allowing them to return to the last save point

6.0 **Robustness**

6.1 **Parameters** are well defined for ISIS process so that the GUI application will have to check to ensure incorrect parameters cannot be passed in

6.2 Error reports related to GUI applications will be labeled as such and contain valuable information about the error

Constraints

Software

The GUI application will have to run on UNIX systems, as the current version of ISIS does, and the first version of the GUI application will make system calls to execute the processes. Future versions may be able to run on mobile devices as well as other desktop operating systems such as Microsoft Windows. A cross platform GUI framework that spans all of these systems will be used.

Currently, the GUI will need to be compatible with Apple and Linux operating systems. Also, the GUI will need to integrate into ISIS3, an already existing API.

USGS Astrogeology will be providing a **virtual machine image** to the USAT team for development and testing to ensure compatibility with their systems.

Hardware

The ISIS software has much more strict hardware constraints than the GUI application, so any system that can already run ISIS will be able to implement the GUI as well. The future versions of the GUI application with the ability to send jobs to be processed remotely will have specific hardware requirements.

Project Plan

January 2012

27th Team Standards complete
Team website created

February 2012

1st Requirements & Execution Plan draft
7th Requirements & Execution Plan complete
9th Presentation: Plan & Requirements
16th Software Design Specification draft
23rd Software Design Specification complete
Prototype Complete

March 2012

1st Integration of XML documents complete
Help Center Documentation complete
14th Understandability and logical placement of buttons complete
28th Design fully implemented with minor modifications needed

April 2012

5th Presentation: Design Review
User Testing Planning
Implementation complete
15th User Testing complete
27th Capstone Conference Final Project Presentation

May 2012

4th Website & software documentation complete
7th Final report

Glossary

API: abbreviation for, application programming interface

C++: general-purpose intermediate level language that provides code reuse through Object Oriented Paradigm.

Centralized: one interface that combines all other interfaces.

Cloud computing: computation as a whole service, therefore giving accessed to shared resources, software, and information.

Cluster: loosely connected computers that can be viewed as a single system.

Code: text written in the format and syntax of the programming language.

Computation time: total time for a processor to complete processing instructions.

Dynamically: responding to change.

Ease of Use: the quality of the user experience through the entire GUI.

Expandability: the ability to accommodate additions to the programs capacity or capabilities.

FIPS: Flagstaff Image Processing System; the beginning system from which ISIS eventually spawned.
Source: http://isis.astrogeology.usgs.gov/documents/IsisHistory/IsisHistory.html

Framework: a reusable set of classes (or, sections of code) for a software system.

Generalized: a general form for example use.

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.

ISIS: Integrated Software for Imagers and Spectrometers; an image processing software package.
Source: http://isis.astrogeology.usgs.gov/documents/Overview/Overview.html

Libraries: a collection of resources used to develop software.

Maintainability: the ability of a computer program to retain its original form, and to be restored in that form in the event of a failure or error.

Macros: a pattern of sequences based upon a defined procedure.

Parameters: a variable given to a value to be executed within a program.

Preferences: individual user settings.

PICS: Planetary Image Cartography System; the second generation of what would eventually become ISIS

QT: A framework with tools designed to streamline the creation of applications for specific phones and desktop applications.

Source: http://qt.nokia.com/products/

Resource intensive: methods that affect the overall performance of a program's execution.

Reliability: the probably of a failure-free software option for a specified period of time.

Robustness: a program that performs well under ordinary but also unusual conditions.

Shell scripts: a script written for a command line interpreter.

Understandability: the ability for which a human reader can understand each module without having previous knowledge.

UNIX: multitasking, multi-user operating system.

USAT: User System of Astrogeology Technologies

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

USGS: United States Geological Survey

Virtual machine image: implementation of an operating system with software emulation.

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.

Source: http://www.w3schools.com/xml/xml_whatis.asp\