
SOFTWARE REQUIREMENTS SPECIFICATION

for

ICEBERG - Rivers use case

Version 1.0

Prepared by Ioannis Paraskevakos
RADICAL
Brad Spitzbart
Stony Brook University

September 28, 2018

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 5 |
| 1.1 | Purpose | 5 |
| 1.2 | Document Conventions | 5 |
| 1.3 | Intended Audience and Reading Suggestions | 5 |
| 1.4 | Project Scope | 5 |
| 1.5 | References | 5 |
| 2 | Overall Description | 6 |
| 2.1 | Product Perspective | 6 |
| 2.2 | Product Functions | 6 |
| 2.3 | User Classes and Characteristics | 6 |
| 2.4 | Operating Environment | 6 |
| 2.5 | Design and Implementation Constraints | 7 |
| 2.6 | User Documentation | 7 |
| 2.7 | Assumptions and Dependencies | 7 |
| 3 | External Interface Requirements | 8 |
| 3.1 | User Interfaces | 8 |
| 3.2 | Hardware Interfaces | 8 |
| 3.3 | Software Interfaces | 8 |
| 3.4 | Communications Interfaces | 9 |
| 4 | System Features | 10 |
| 4.1 | Analyzing images in bulk | 10 |
| 4.1.1 | Description and Priority | 10 |
| 4.1.2 | Stimulus/Response Sequences | 10 |
| 4.1.3 | Functional Requirements | 10 |
| 4.2 | Output seasons | 10 |
| 4.2.1 | Description and Priority | 10 |
| 4.2.2 | Stimulus/Response Sequences | 11 |
| 4.2.3 | Functional Requirements | 11 |
| 5 | Other Nonfunctional Requirements | 12 |
| 5.1 | Performance Requirements | 12 |
| 5.2 | Safety Requirements | 12 |
| 5.3 | Security Requirements | 12 |
| 5.4 | Software Quality Attributes | 12 |
| 5.5 | Business Rules | 12 |

| | | |
|----------|---|-----------|
| 6 | Other Requirements | 14 |
| 6.1 | Appendix A: Glossary | 14 |
| 6.2 | Appendix B: Analysis Models | 14 |
| 6.3 | Appendix C: To Be Determined List | 14 |

Revision History

| Name | Date | Reason For Changes | Version |
|---------|-----------|--------------------|---------|
| Initial | 9/28/2018 | | 0.1 |

1 Introduction

1.1 Purpose

The purpose of this document is to capture the requirements of the ICEBERG: Rivers Use Case. It will include functional, non-functional and User Interface requirements. It will be used as the reference document between the RADICAL Team and the Stony Brook and UCSB teams for the Rivers use case development.

1.2 Document Conventions

The requested features are listed in section 4 and the non-functional requirements are listed in section 5. Each of these requirements have a priority from the set HIGH, MEDIUM, LOW. Based on the number of requirements and their priority, a timeline will be created with each requirement and its expected time-to-completion.

1.3 Intended Audience and Reading Suggestions

The document is edited and iterated between users and developers. It is intended to provide the developers as well as the project managers a complete understanding of the requirements as they are expected by the users.

An early use case document is provided in [1]. The current status of the project is provided by the use case Github repository [2].

1.4 Project Scope

We provide a detection algorithm to extract the location of water features from high-resolution imagery.

1.5 References

[1] https://github.com/iceberg-project/Rivers/blob/master/HOWTO_StreamExtraction.txt

[2] <https://github.com/iceberg-project/Rivers>

2 Overall Description

2.1 Product Perspective

ICEBERG is a multi-disciplinary, cyberinfrastructure, integration project to (1) develop open source image classification tools tailored to high-resolution satellite imagery of the Arctic and Antarctic to be used on HPDC resources, (2) create easy-to-use interfaces to facilitate the development and testing of algorithms for application specific geoscience requirements, (3) apply these tools through four use cases that span the biological, hydrological, and geoscience needs of the polar community, (4) transfer these tools to the larger (non-polar) EarthCube community for continued community-driven development.

2.2 Product Functions

The main functions of the Rivers pipeline are conversion of orthorectified imagery into NDWI: Normalized difference water index images, extracting large stream profiles, and extracting lake profiles.

2.3 User Classes and Characteristics

- Community users - Will use a web interface to access the capabilities of ICEBERG
- Expert users - Will use ICEBERG via command line interface to execute experiments and their use cases.
- Developers - Are users that are able to develop additional pipelines and/or change/update existing pipelines. They will be able to use the CLI or directly the resource interfaces.

2.4 Operating Environment

The software's middleware should be able to use Unix-based Operating Systems, such as Linux and MacOS. The software has library dependencies as listed in Table 2.1. *HARD* dependency to a library is restricted to the version shown. *SOFT* dependency to a library requires as a minimum version the one depicted.

The Command Line Interface should be used from a Virtual Machine that is in the Cloud and has constant Internet connection.

The Web interface should be hosted on resources with constant operation.

| Library | Version | Executable | Type |
|------------|---------|-------------|-------------|
| Python | 3.5 | All | <i>SOFT</i> |
| matplotlib | 2.2.2 | Verify | <i>SOFT</i> |
| EnTK | 0.7 | entk_script | <i>SOFT</i> |

Table 2.1: Software Dependencies.

2.5 Design and Implementation Constraints

Access to the VM should be through SSH. The web interface should be under HTTPS protocol.

Users should have their image data uploaded to the resources via a secure data transfer system, like sftp/scp.

Users should not execute from the login nodes, unless otherwise specified explicitly by the resource documentation.

2.6 User Documentation

Users will be provided on-line documentation and help. Syntax, options, and error messages will be displayed via the web or command line interfaces.

2.7 Assumptions and Dependencies

OMPI is installed and provided via a module for the launch method of RP. Python 2.7 and 3.5 should exist in the resource.

3 External Interface Requirements

3.1 User Interfaces

Provide an interface where users can upload a set of images and get back a table with the water features.

The CLI interface should have the arguments based on table 3.1

| Argument Name | Argument Flag(s) | Argument Type | Value | Required/Optional |
|------------------|------------------|---------------|----------------------|-------------------|
| Resource | -r/--resource | String | xsede.bridges | Required |
| Output Directory | -op/--output_dir | String | '.' | Optional |
| Input Directory | -ip/--input_dir | String | /home/iparask/images | Required |
| Path to Model | -m/--model | String | model file | Optional |
| Walltime | -w/--walltime | Integer | 60 | Optional |
| User Name | -u/--username | String | 'iparask' | Required |

Table 3.1: Command Line Interface Arguments.

3.2 Hardware Interfaces

The software system requires High Performance Computing (HPC) resources for execution and needs at least one GPU. The HPC resources should provide CPU and GPU node. Until now, PSC Bridges and SDSC Comet are the possible candidates.

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software

components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4 System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Analyzing images in bulk

4.1.1 Description and Priority

The Rivers component should take as an input a set of images. These images will be organized in a single folder. Each analysis is independent of each other.

Priority: 9

4.1.2 Stimulus/Response Sequences

The stimulus of this feature will be the path to a folder where all the images will exist. The response sequence is a table with the water features per image.

4.1.3 Functional Requirements

The following list displays the requirements of this feature. All requirements have the same priority unless otherwise stated.

REQ-1: Identify the individual images that exist in a single folder

REQ-2: Each image should have an individual output file

REQ-3: Provide an aggregated output file for all images with two columns: Image file-name, id of water feature

4.2 Output seasons

***brad: Shape files for each image as they are produced and an CSV with the results for one season.

4.2.1 Description and Priority

The Rivers component should output results per season if need be.

Priority: 5

4.2.2 Stimulus/Response Sequences

The stimulus is a flag during runtime. The response would be to output the results of images as they are produced.

4.2.3 Functional Requirements

The following list displays the requirements of this feature. All requirements have the same priority unless otherwise stated.

REQ-1: Create an input defining a season. The images of the season are prioritized over the rest of the images.***giannis: @Brad: how do we understand which images belong to a season?

REQ-2: Create a file with the results of the season with two columns: Image filename, id of water feature

5 Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

All Digital Globe (WorldView) imagery is proprietary and cannot be released publicly. Use of imagery must be in accordance with the guidelines and requirements of the Polar Geospatial Center and the NGA NextView License.

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional require-

ments in themselves, but they may imply certain functional requirements to enforce the rules.>

6 Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

6.1 Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

6.2 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

6.3 Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>