TrawlExpert Requirements Specification Document

Trawlstars Inc. (Group 11)
Lab section: L01
SFWRENG 2XB3

Christopher W. Schankula, 400026650, schankuc
Haley Glavina, 001412343, glavinhc
Winnie Liang, 400074498, liangw15
Ray Liu, 400055250, liuc40
Lawrence Chung, 400014482, chungl1

February 25th, 2018 (Last Updated: April 11th, 2018)

1 Domain

The TrawlExpert is a tool that will provide statistical data on fish populations throughout the Great Lakes. This data will be presented in a concise and specific manner to simplify the task of data retrieval for users. The intended users of TrawlExpert are environmental and climate researchers, however stakeholders include all communities that interact with the Great Lakes ecosystem or rely on its fish for survival. This tool will provide users with fish population data that is dependent on fish species, time, and geographical information. The TrawlExpert will allow users to intelligently filter and extract data that pertains to their research area of interest and generate historical graphs of populations of a given fish species.

For the purposes of this project, a dataset of Great Lakes trawl surveys from 1958-2016 will be used. These surveys include dated information on fish of the Actinopterygii class observed throughout the Laurentian Great Lakes and the geographical coordinates each fish was found. Additional information includes biological classification of fish species and the depth at which each fish was observed. These criteria will allow researchers to customize their search preferences when using the TrawlExpert to ensure that data relevant to each user?s highly specialized research is generated.

2 Functional Requirements

TrawlExpert will provide researchers with several tools, including basic dataset searching as well as some advanced tools for extracting historical and location-based statistics from the dataset. Each subsection details some key functional requirements for TrawlExpert to be a successful piece of software.

2.1 FR1. Reading input file; dealing with corrupt data

Input: .csv dataset file
Process: read, clean dataset, convert to Java Record objects
Output: array of Java Record objects.
Details: TrawlExpert should be able to read from the 280,000+ line dataset detailed in the original project proposal, as well as recover data from lines which are corrupt (a lot of lines in the dataset are missing certain (unneeded) columns of data.

2.2 FR2. Generate output file

Input: array of records (from a search, for example)
Process: convert records to string representation delimited by ",".
Output: .csv file of results (same format as input file)
Details: Writes a file containing the dataset entries in the array. These could come from a search detailed in section 2.3, for example.

2.3 FR3. List of species by family, order, genus, etc.

Input: dataset (tree generation), biological classification (family, order, genus, etc), requested output (family, order, genus, species – ancestor of previous input in terms of classification tree)
Process: find all of the requested nodes in the tree
Output: list of nodes found in tree
Details: The dataset contains not only the species but the higher-order classifications (kingdom, order, family, genus, etc.) that encompass the species. This will be used to generate a tree structure. For example, a user may request for a list of all the species under a certain genus in order to study similar species and their appearance in the dataset.

2.4 FR4. Basic search by family, genus, species, time, location, etc.

Input: search criteria (family, genus, species, time range, location range)
Process: find all dataset records matching the search

Output: array of Record objects
Details: sorts and searches the dataset to find entries matching the search criteria, returns an array of matching records.

2.5 FR5. Historical distribution of records

Input: array of Record objects, histogram binning (yearly, monthly)
Process: create histogram of yearly / monthly occurrences within the array
Output: totals by year or month for the given input array
Details: For a given search, output a histogram showing the number of occurrences by year or month (could be a .csv file and / or a plot).

2.6 FR6. Geographical subgroupings

Input: array of Record objects, radius of similarity
Process: construct a graph by attaching Records if within a certain radius, then determine connected components
Output: array of connected components and their sizes
Details: Allows the researcher to find if certain species tend to be spotted at certain locations.

2.7 FR7. Plotting / mapping tools [time allowing]

Input: array of Records, type of plot / map
Process: create the plot from the data
Output: plot / map of data
Details: If time allows, these tools will allow the user to plot things like heatmaps to visualize data in their input.

2.8 User story example

A researcher is studying fish species in the family Cottidae for the entire time range of the dataset. She wishes to determine the historical distribution on records of species in this family, in order to determine whether there is evidence that there has been a decline in the populations. She first uses Trawlexpert to read the dataset from the .csv file (section 2.1) she has downloaded from a government repository. She then performs a basic search (section 2.4) to locate all the records in this family. The basic search module first uses the biological tree module (section 2.3) to get a list of species, then searches for all the records of these species. This search returns an array of Records from the dataset. She uses the historical distribution module (section 2.5) to turn this array of Records into a yearly histogram of fish found during the trawl surveys in the dataset. In just a few simple, fast steps, she has gained valuable output to inform and motivate her research going forward.

3 Non-functional requirements

Along with its functional requirements, TrawlExpert will also meet a set of standards that deal more with performance. The following subsections indicate such performance aspects and describe metrics that the team will use to measure such metrics

3.1 NR1. Accuracy

The tool must always find the correct pieces of information that the user is inquiring. The team will use a prototype, along with a sample data set and run the algorithm, based on different inputs, to determine if the outcome is correct. 100

3.2 NR2. Robustness

The tool must be able to take many different types of input and still provide the user with some output. For example, if the user enters an invalid type of input (i.e. type int instead of expected type str), the tool will inform the user that input is of the wrong type. Using the prototype of the tool, the team will provide different types of input to ensure that the appropriate message is given.

3.3 NR3. Speed

The speed is determined by the time taken for queries to be completed. The tool takes the input, runs through the search, sort, graphing algorithms and prints. All the jobs combined should take less than 3 seconds. The team will implement a timer that will output the amount of time taken to achieve the task.

3.4 NR4. Memory usage

The tool should effectively use memory. It was decided that 1GB of RAM is the limit that the tool will use, as most modern computers should have much more than 1gb of memory with which to run the program. This measurement will be taken using a profiling tool such as JConsole, JProfiler or another tool that is found to best fit the needs of the project.

3.5 NR5. Scalability

The tool must be able to handle large amounts of data, all while being able to complete queries at a high speed. Currently, the dataset is 200,000 lines of data, but the tool should be able to meet the speed requirement no matter how big the data set. Using sorting algorithms such as QuickSelect to build a K-D tree, the tool should complete the tree construction in linear time.

3.6 NR6. User-friendly

The tool must be easy to use for a user who is well-versed in the scientific principles related to using Trawl surveys as a basis for research. This include making it easy to perform searches and display data that is relevant to the user?s research.

4 Requirements on the development and maintenance process

To ensure the robustness and ease of development of TrawlExpert, the team has outlined several principles and conventions that must be adhered to throughout the development process. These are in two categories: software engineering principles and development conventions and standards.

4.1 Software engineering principles

TrawlExpert will be developed while following several software engineering principles and some conventions that will aid in maintaining these principles during the development process. The following subsections discuss the principles and their benefits towards the development process of TrawlExpert.

4.1.1 Modularization

The software will be broken down into modules to reduce complex functions into simpler subfunctions. The strategies applied to each modules only concerns itself with a single set of responsibilities. This would increase the maintainability of the program and its ability to adapt to future changes. The modules must have high cohesion and low coupling.

4.1.2 Encapsulation and Immutability

The csv dataset file will be processed and stored in different abstract data types. These ADTs will be encapsulated and immutable by declaring certain state variables and methods as private and / or final.

4.1.3 Generality

In general, methods should be designed with generality and reuse in mind, in order to reduce code duplication and increase maintainability. For example, sort and search functions will designed with the principle of generality by accepting a compare function as a parameter, which will allow the same sort and search to be applied to objects of different data types.

4.2 Development conventions and standards

The TrawlExpert team shall follow the following procedures and standards while developing and maintaining the software, in order to support the software engineering principles listed in section 4.1.

4.2.1 Git usage conventions

The CAS GitLab server will be used as the primary means of code sharing and version control. Subteam specific tasks should be committed to their respective branches. Branches should not be merged into master until a module is complete and has been tested. The team captain must oversee any and all merges into the master branch. If a merge conflict arises, developers of that piece of code must be present to resolve the issue. Team members are expected to document changes appropriately and to commit regularly.

4.2.2 Naming conventions

The TrawlExpert team will follow standard Java conventions as stated below. In addition, efforts should be made to ensure that variables performing similar tasks in different modules have standardized names, which will be decided upon by the team or subteams as work on the software progresses. No non-temporary variable names should be repeated. This means that a variable used in one part of the code should not be repurposed later on in the code as this will cause confusion and decrease maintainability.
Packages

- fully lowercase name

Classes

- UpperCamelCase
- Abbreviations should be avoided unless well known.

Methods

- lowerCamelCase

Variables

- lowerCamelCase
- Should not start with an underscore
- Should be descriptive. Single character names should only be used for temporary or index variables in loops.

4.2.3 Testing and refactoring requirements

All modules should have accompanying testing using the Java JUnit 4 testing framework. Each module should be tested enough to adequately show that the module performs we expected. To guarantee that a module meets specifications, code that have undergone changes - functional or not, should be retested with prior test cases. Changes should not be made to a module's name or parameters unless absolutely necessary. Such changes are to be made though the Eclipse IDE's refactoring feature.

4.2.4 Documentation requirements

Each class and method should be adequately documented using JavaDoc so that another programmer could easily understand the method and in the case of public methods, use it in their own client code. When applicable to the method, the following tags must be included:

- A one- or two-sentence description of the class / method
- For classes, @author documents the main author of the method: the one that should be contacted for questions pertaining to the method
- @param tag for each input parameter (e.g. @param x description)
- @throws tag for each exception that is thrown (e.g. @throws IndexError description)
- @return tag for non-void functions
- Additionally, particularly obscure or difficult-to-understand code must have accompanying in-line comments. All comments and documentation should be made as soon as possible after code is written, preferably as code is written, so as to ensure that comments are accurate and representative of the author?s thought processes while coding.