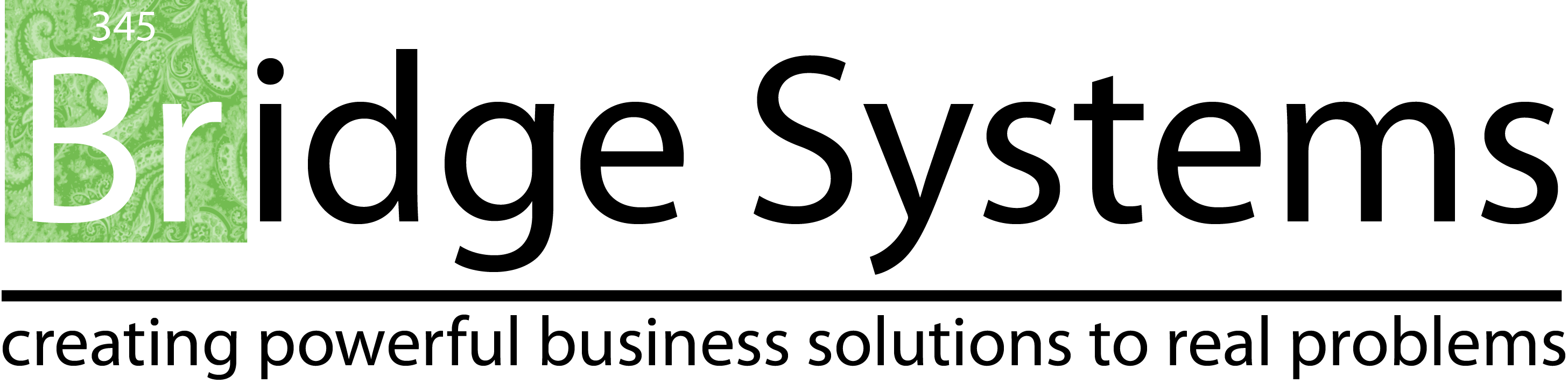
System Design Specifications

for

Bio-Mimicry Search Engine

Version 1.0 approved

Prepared by Mike Phillips, Richard Field, Mason Smith and Brett Long



Tuesday February 26, 2013

Table of Contents

Table of Contents 1

Revision History 2

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 References 1

2. Overall System Description 1

2.1 Overview 1

3. System Components 2

3.1 Component Overview 2

3.2 Program Input 2

3.3 Searching 2

3.4 Program Output 2

4. System Architecture 2

4.1 Architecture Overview 2

4.2 New Software Architecture 2

4.3 Hardware Architecture 3

5. Other Nonfunctional Design Specifications 3

5.1 Interface Upgrade 3

Appendix A: Glossary 3

Appendix B: To Be Determined List 3

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| All Authors | 2/21/2013 | Initial Commit | 0.1 |
| All Authors | 2/24/2013 | Content updates | 0.2 |
| Mike Phillips | 2/26/2013 | Edits to First Draft | 0.3 |
| All Authors | 2/26/2013 | Edits to First Draft | 0.4 |
| All Authors | 3/12/2013 | Edits to Final Draft | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 

# Introduction

## Purpose

The designs set forth in this document are to describe the software upgrade to the functionality and other specifications of the bio-mimicry search engine for Dr. Jacquelyn Nagel, physics department, James Madison University. The design and architecture are laid out to the best of the team’s current knowledge and may change as work continues on the project.

## Document Conventions

This document will follow the conventions set forth by the IEEE 830-1984 guide [1]. It is consistent to the writers’ best knowledge and traceable through the revision history on the previous page. Version numbers (for the application and this document) will follow the ‘major.minor.patch’ labeling heuristic and only major releases will break functionality. All previous releases will fall under the 0.1.\_ minor release. Project completion is satisfied by launch and the release of version 1.0.0.

## Intended Audience and Reading Suggestions

The readership this document is intended to include the developers, project managers, users, testers, and documentation writers. The developers are to include Mike Phillips (also the project manager for the development team), Mason Smith, Richard Field, Brett Long, and any collaborators. The project manager for whom this project is being completed is Dr. Jacquelyn Nagel. Professor Nancy Harris will also act in this role as well as be an intermediary between the project managers and general overseer of the project.

The rest of this document will complete the introduction by listing references. Following that will be an overall system description, a list and description of the design components, and a detailed look at the architecture.

## References

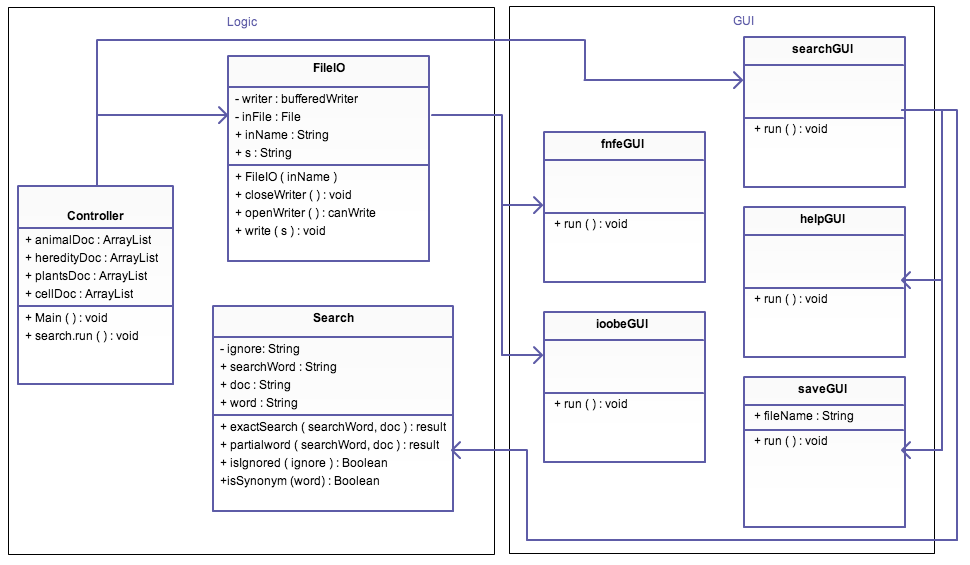
[1] http://standards.ieee.org/findstds/standard/830-1998.html

[2] Nagel, Jacquelyn K. Bio-Mimicry – An Approach to Engineering Design.

# Overall System Description

## Overview

The system overview can best be described by the UML presented in Figure 2.1.



**Figure 2.1.** A UML diagram of the class structure contained in the BioSearch Program. Notice that the classes can be divided into two modules, a logic module and an interface (GUI) module.

The controller class starts the program when the .jar/exe/whatever is called. From there the controller takes in the names of the files from inside of your src document using the DocumentReader class. These files should be named "Animal.txt", etc. If there is any problem with reading in these files the fileNotFoundException or IndexOutOfBoundsException will be called and an error screen will appear on the screen informing you of such. The searchGUI class is then called by passing in the four documents to the searchGUI constructor. searchGUI then initializes the main GUI frame you will see upon start. Each file is then sent to a specified search function in the Search class. The search function that is called depends on which panel is chosen at the top of the GUI structure. (i.e Exact, Partial, Derivative)

Inside the Search class is where all of the main functions of the program are performed. Each search function takes in the search word String, the Document which it is searching, as well as two booleans, representing the radio buttons at the top of the GUI. If the synonym button is pushed then an if statement executes, checking to see if the search word has synonyms. If it does then it will search through the document using the root word as well as its synonyms and returns the results. If this is not the case, a regular search function is performed. There is then another if statement that is executed if the second radio button (sentences) is pushed. If that button is pushed, the search returns a list of sentences as opposed to paragraphs.  Each of these search functions returns an ArrayList of Strings that have been found to hold the search word. The program then jumps back up to SearchGUI where the write function is called. The write function takes in the textArea that the results will be written to, the results from the document which was searched as well as the search word. In the write function, the contents of the results of the search are formatted to break at every 90th character to fit onto the screen in an appealing manner. A highlighter for the textArea then checks through the index of which the search word was found and highlights each of the search words. The write function is called a total of four times per run, once for each document that has been searched.

Other non-linear classes include the error GUI classes that execute if a major problem appears while running the program. There is also a class that is executed if the save button is pushed on the main GUI panel that is called saveGUI. This class opens a box that will ask what you would like to name your file. If no file name is chosen, a txt file by the name of saved.txt will be saved to your src folder. Another class such as the error or save GUI's is the helpGUI class. If the user pushes the help button, a help window will pop up on their screen. The help window contains basic instructions on how to run the program if the user does not find the controls very intuitive.

The program can be exited at any time by pushing the X button on the top right (or left, if mac) of the main GUI window.

# System Components

## Component Overview

The system will have three major components. The first is the input to the program, which will be accomplished by presenting the user with a text box for the search term, and a dropdown box to select the type of search. The next component is the search algorithm where the user will press the search button and the application will compute the results of the search. The final component is the output of results.

## Program Input

The user will be presented with a text box that prompts the user to enter the search terms. These will be stored in a variable when the user initiates the search. The user will also be presented with a drop down list to select whether they want to search the full search terms as input, a derivation of the search terms, or a partial derivation of the search terms.

## Searching

Upon initiating of the search by pressing a “Search” button, the program will store the input information and begin the search. The program will search the word according to the setting it is on (i.e. partial word, full word) and begin its search from there. If the word is a “key word”, meaning it is known to have multiple synonyms then the word will be searched as well as all of its synonyms.

Upon completion of the search, the results will be stored for output.

## Program Output

The program will output the results of the search in the output box, as formatted by the program. In the case of key words as defined above, the results of the search will be shown underneath the search word that was used to find the definition.

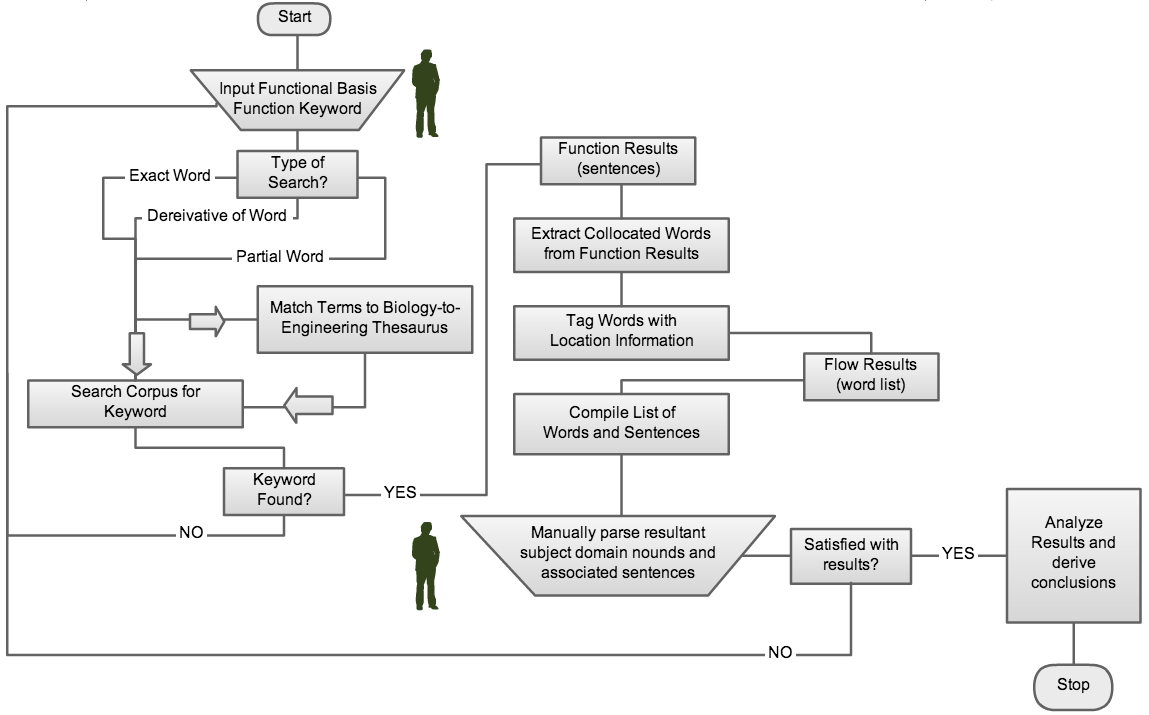
# System Architecture

## Architecture Overview

The architecture of the program will follow the basic structure of the flowchart in the system requirements document (see System Requirements Document for Bio-Mimicry Search Engine, Figure 2.1).

## New Software Architecture

While the design of the upgraded system (as described in the requirements document) closely follows the old flowchart, we have upgraded it’s content to reflect our changes. This can be seen in figure 4.1.



**Figure 4.1.** A logic diagram of the new application flow. Note the addition of search term replacement with words from the engineering-biology thesaurus.

As one can see, we have updated this figure to include the main design requirement, search term replacement with words from the engineering-to-biology thesaurus. This allows the program to intelligently substitute words that the user might not think of in order to deliver better results that are more useful to the user.

## Hardware Architecture

The hardware architecture will not change, as there were essentially no hardware components to the old system, nor are any necessary to complete the requirements of the upgrades. The only hardware necessary is a computer, and this has been expanded to most operating systems (as opposed to just Macintosh in the original system) as per the requirements specifications.

# Other Nonfunctional Design Specifications

## Interface Upgrade

As noted in the requirements document, it may be decided by the developers that during the process of rewriting the application code, an interface requirement might be desirable or even necessary. In this case, best design practices will be put into effect so that the interface is markedly improved upon and still fulfills the necessary functions of the program with out being distracting or taking away from the usability of the program.