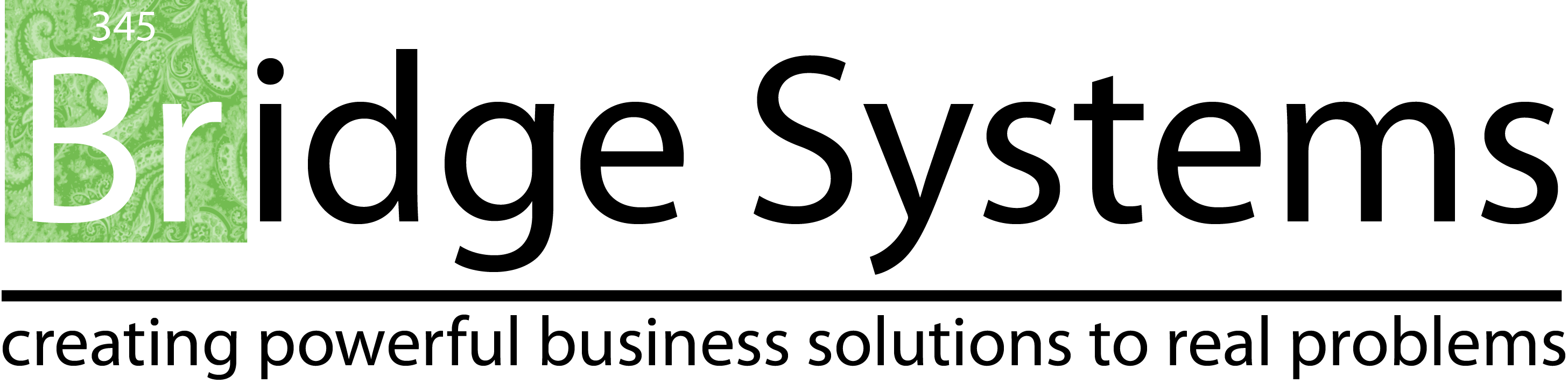
Software Requirements Specification

for

Bio-Mimicry Search Engine

Version 1.0 approved

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



Tuesday February 12, 2013

Table of Contents

Table of Contents i

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 3

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Functions 4

2.3 User Classes and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3. External Interface Requirements 5

3.1 User Interfaces 5

3.2 Hardware Interfaces 5

3.3 Software Interfaces 5

3.4 Communications Interfaces 5

4. System Features 5

4.1 Platform Independence 5

4.2 Integration of the Engineering-to-Biology Thesaurus 6

4.3 Improving the User Interface 6

5. Other Nonfunctional Requirements 6

5.1 Performance Requirements 6

5.2 Safety Requirements 7

5.3 Security Requirements 7

5.4 Software Quality Attributes 7

5.5 Business Rules 7

6. Other Requirements 7

Appendix A: Glossary 7

Appendix B: Analysis Models 7

Appendix C: To Be Determined List 7

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Mike Phillips | 2/11/2013 | Initial Commit | 0.1 |
| All authors | 2/12/2013 | First Draft | 0.2 |
| Mike Phillips | 2/13/2013 | Edits to First Draft prior to submission | 0.3 |
| Mike Phillips | 2/19/2013 | Edits before Final Draft | 0.4 |
| Richard Field | 4/22/2013 | Edits for Final Draft | 1.1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 

# Introduction

## Purpose

The requirements 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 changes are to include, but are not limited to the following. The search engine will be expanded for use on multiple platforms (it currently only works on Macintosh). The upgrade will also allow the engine to integrate an engineering-to-biology thesaurus, which would intelligently expand the search terms by converting engineering language to biology language through a prescribed corpus of terms. The search results would appear the same way, but would provide better answers by recursively searching through multiple terms. The upgrade would also continue to allow the user to receive results for the exact word, derivation of word or partial words.

These changes may require the final product to come in the form of a complete application rewrite. All original functionality will be either retained, or modified to fit the aforementioned upgrades. In the event of a total rewrite, the release will have the version number 1.0.0 at launch.

## 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 2.0.0.

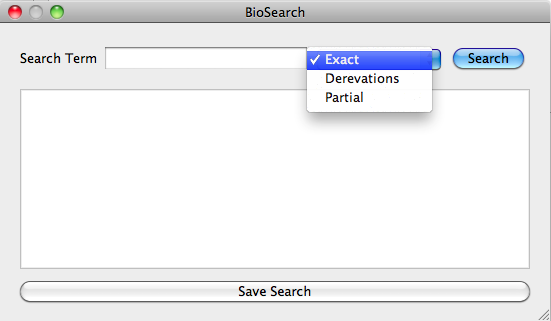
## Intended Audience and Reading Suggestions

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 outlining scope and references. Following that will be an overall description of what requirements the product will satisfy upon completion, a description of external interface requirements, a list of features, and lastly, a list of other nonfunctional requirements. The reader should begin with the overview sections and proceeding through the sections that are most pertinent to each reader type.

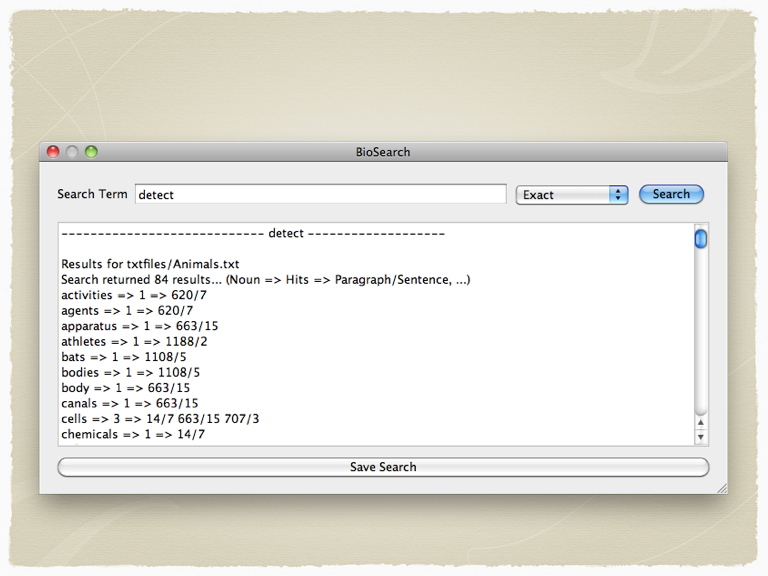
## Product Scope

The software described by this document is an upgrade to a search engine that helps engineers brainstorm designs by comparing mechanical problems to those already solved in the natural world. The user enters a search term or terms and the application sorts thorough a corpus to find references. This functionality is already present in the current version of the software. A screenshot of the main screen is provided in Figure 1.1. The small text box at the top is for the input of the search term.



**Figure 1.1.** Screenshot of the current version of the application showing the various buttons and inputs to the program. [2]

The dropdown shows the options for the exact term, a derivation of the term or a partial word form of the term. The output appears in the larger text box as illustrated in Figure 1.2. The original program was written in C. The project will include an expansion of the current functionality and a possible rewrite to provide cross-platform implementation. The goal of this project is to implement all of the customer’s desired features while minimizing additional runtime and including possible interface upgrades.



**Figure 1.2.** Screenshot of the application showing the output of the program. The software finds references to the search terms in the corpus and presents the user with the hits and their locations in the file. [2]

## References

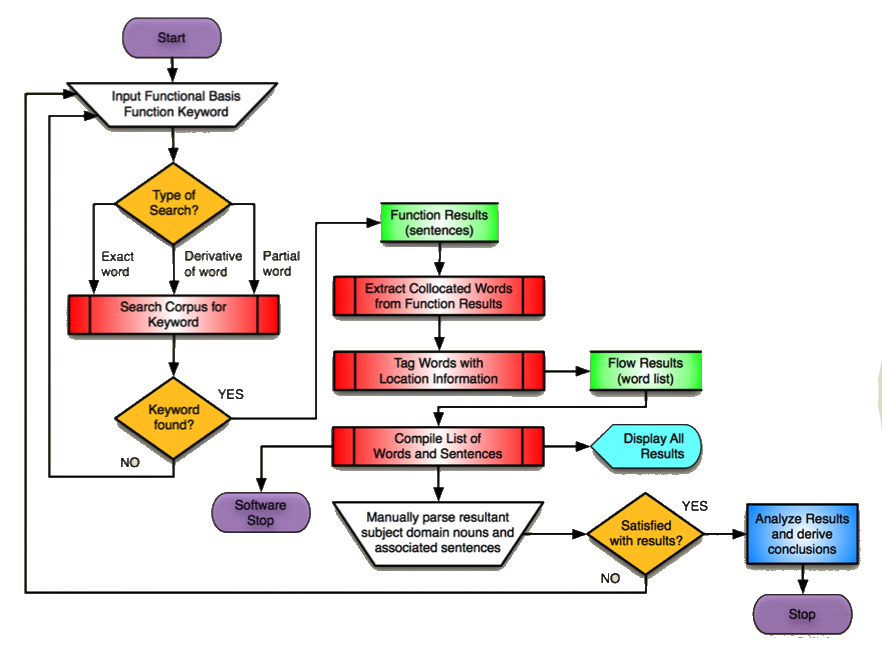
[1] http://standards.ieee.org/findstds/standard/830-1998.html

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

# Overall Description

## Product Perspective

This project is to fulfill the need to extend and service the application originally built by Dr. Nagel. It is a follow up to her work and this document describes the changes and additions to functionality and interface that she has requested. This is not a component to a larger system, but will rely on the ability to load a corpus of data for the search engine to parse. The current system diagram appears in Figure 2.1.



**Figure 2.1.** A logic diagram for the current version of the program. The extensions to this will include the program running cross platform, as well as search term replacement between the user input and the engineering-to-biology thesaurus. [2]

The functionality outlined in Figure 2.1 will, for the most part, not change. The search algorithm will stay the same. The major difference will be the addition of term replacement using the engineering-to-biology dictionary to intelligently expand the results. This will hopefully deliver more accurate and useful searches to the user.

## Product Functions

* The user can launch the program (in most operating systems)
* The user can input a search term(s)
* The user can select whether to search for the exact term, derivations of the term, or partial derivatives of the term
* The user can begin the search
* The application can intelligently substitute search terms within the engineering-to-biology thesaurus to deliver better results
* The application will display the search results with information on where the hit appeared in the corpus
* The user can select to save the search results in a text document.
* The user can minimize or exit the application at any time

## User Classes and Characteristics

There is only one user class, as the application does not have any authentication restrictions in place. Any user with any level of access should be able to open the application and use it just as any other in the manner of Section 2.2.

## Operating Environment

One key goal of this project is for the application to be platform independent. As it is now, the software only runs on Macintosh. For success of the project, it should be able to run on Windows and Linux, with the possibility of some functionality on other less used systems.

## Design and Implementation Constraints

The software is constrained in very few ways as defined by the project managers. The design can remain fairly similar to the current implementation, using only a few buttons and inputs. It is not necessary for the interface to be over designed, just that it allows the user access to the defined functionality. As implemented, it must carry out the functions defined in this document and should not go beyond these requirements, unless such implementation is necessary to the success of the application.

## User Documentation

Documentation will come in the form of the final report turned in to the project managers upon completion of the project. This will likely include any information the user might need to operate the program. However, since the software is so simplistic and straightforward, this should not be necessary for the average operator to use it.

## Assumptions and Dependencies

This application and its requirements are dependent on the assumption that the data being fed into it is correct and properly formatted. Without this, the output of the program is meaningless.

# External Interface Requirements

## User Interfaces

The user will be able to type in a word in the user interface that will then use the software so search for that word, partial word, and derivation of that word and will also show the exact paragraph and line that they are located. You will also be able to save search

## Hardware Interfaces

It will be displayed on the screen in graphic user interface and will be able to be accessed from both Macintosh and PC operating systems.

## Software Interfaces

These software components will be accessible for MAC and PC operating systems. It will be a basic search engine that search for specific synonyms of engineering languages that laterally coincide with biological languages.

## Communications Interfaces

Our program is self-contained. It will not be communicating with anything but the direct user and the search engine software.

# System Features

## Integration of the Engineering-to-Biological Thesaurus

4.1.1 Description and Priority

Engineering terms will automatically be swapped with a corresponding biological term then searched. Recursive searches when there are multiple terms. There will be an option to search the exact terms, partial terms, and derivative terms.

This is a high priority function (9)

4.1.2 Stimulus/Response Sequences

The user will enter the desired search word, the program will check then to see if that word is one that should be recursively searched with its synonyms. The program will then give all of the search results from each word.

4.1.3 Functional Requirements

REQ-1.1: Able to check if a word is actually a word

REQ-1.2: Able to identify key biology words.

REQ-1.3: Search Exact terms.

REQ-1.4: Search Partial terms.

REQ-1.5: Search Derivative terms.

## Platform Independence

4.2.1 Description and Priority

This feature will allow the program to be run on all operating systems. This is of high priority because it is one of the main features desired.

This is a high priority function (9)

4.2.2 Stimulus/Response Sequences

* Read through the code and determine what makes it dependent on Mac.
* Rewrite parts of the code to make it work on all platforms.

4.2.3 Functional Requirements

The code should be written in C because it is already platform independent. Product should flawlessly run between multiple operating systems.

REQ-2.1: Run on multiple platforms.

## Improving the User Interface

4.2.1 Description and Priority

The implementation of cross platform functionality may for coding reasons require the user interface to be redesigned. Even if this is not the case, some attempt to improve the look and feel of the program might be added to the upgrade.

This is a moderately low priority function (4)

4.2.2 Stimulus/Response Sequences

* Examine the user interface as it is in its current version
* Redesign the user interface to meet developers’ needs and desires

4.2.3 Functional Requirements

REQ-3.1: Redesign the user interface

# Other Nonfunctional Requirements

## Performance Requirements

As long as the program works then we have completed the requirement. Dr. Nagel has stated that with the addition of the recursive search algorithm, speed loss is to be expected. This is not a necessary function of the program however. The program should be able to run without crashing and deliver the appropriate results. Beyond that, there are no speed requirements.

## Safety Requirements

The program should not affect anything outside the scope of the program. The original documents should not be harmed, deleted, or rewritten. Beyond this, we expect the program not to pose any safety threats to other software, the operating system, hardware, or people.

## Security Requirements

There are no security requirements. As stated in Section 2.3, any user can perform the same operations within the program. It is such a simple program that authentication and security become a non-issue. The only requirement for safety is that the application should not change or disseminate any information stored on the computer or attached network devices.

## Software Quality Attributes

The product should be just like a search engine. It should be very simple and easy to learn. It should work consistently, the search process should take as little time as possible, and it should deliver accurate results every time to the best of the programs ability.

## Business Rules

This program is for personal use and will not break any laws or rules.

# Other Requirements

The project has no other requirements at this time. Any unforeseen requirements will be integrated according to direction provided by the project managers and done so in a way that does not interfere with the primary requirements.