

# Senior Project

## Abstract

The objective of this project is to revamp my Dictionary Cleaner program by adding more features, overhauling existing features, and redesigning the interface. There are four major issues affecting the usability and usefulness of Dictionary Cleaner 2.0 $\beta$ , all of which will be addressed by this project. By completing this project I will hone my existing Cocoa and Objective-C programming skills and expose myself to a few new technologies that I have little to no experience with.



## Dictionary Cleaner

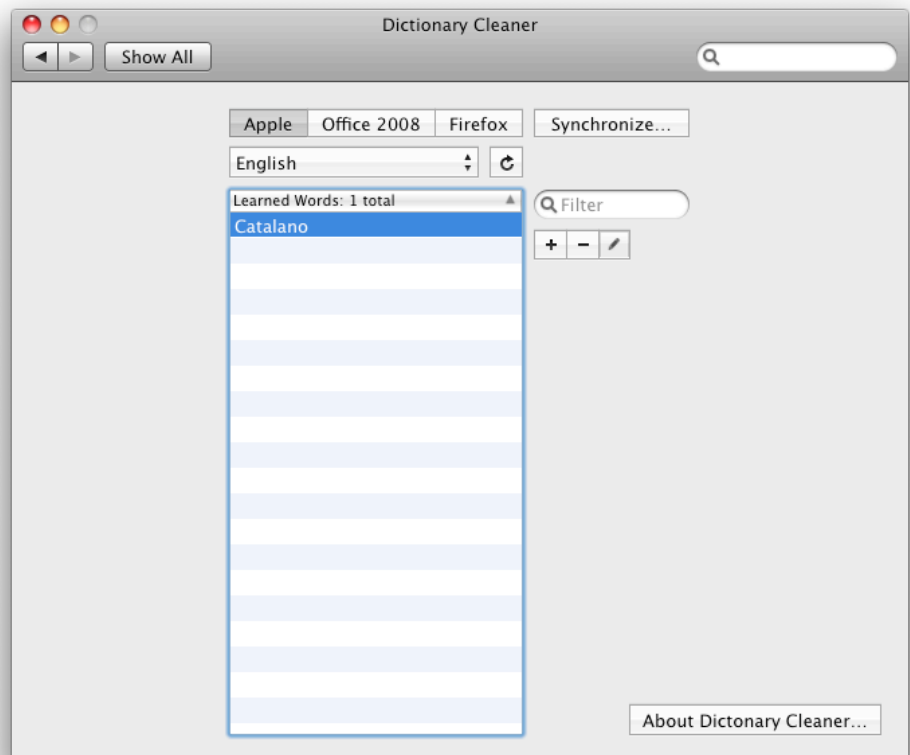
Dictionary Cleaner is a Mac OS X System Preferences pane which allows the user to view, edit, add and remove words in system-wide, Office, and Firefox user-defined dictionaries. It's a free program that I wrote when I realized that there was no easy way to view the words that have been added to the system-wide dictionary. Originally it only supported the system-wide dictionaries, but the current beta of version 2.0 includes support for Office 2004/8 and Firefox dictionaries as well.

Over 2,500 people have downloaded Dictionary Cleaner to date from [MacUpdate](#) alone. The program is also available at [CNet](#), [Apple](#), [VersionTracker](#), and my own website, [TwoAMSoftware](#).

## Issues

There are four main issues affecting the most recent version of Dictionary Cleaner:

1. While the dropdown dictionary selection menu worked well with where there was only one source, the addition of Office and Firefox dictionaries added another layer to browse through.
2. There is far too much "glue code" connecting the interface and models. Things like filtering, sorting, and even keeping track of which dictionary is displayed could all be automated via Cocoa bindings in Interface Builder.



**Dictionary Cleaner 2.0 $\beta$  interface**

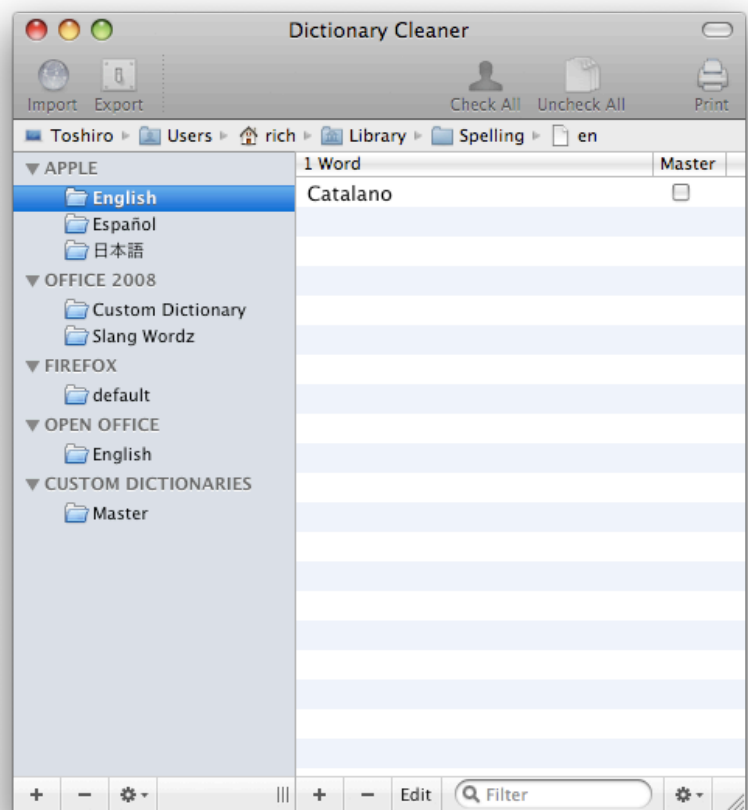
3. Synchronization between dictionaries is currently limited at best and lousy at worst. I want all of this program's functionality to be contained in the main window and not to have a separate sheet just for synchronization.
4. The program has stepped outside of the boundaries of a System Preferences pane in terms of functionality and now deserves to be its own application.

Given these issues, it is time for Dictionary Cleaner to be completely overhauled.

## Solutions

### Interface

- I've received many questions regarding the location of the files that Dictionary Cleaner edits, so I put an instance of NSPathControl at the top of the window. I may decide to let the user toggle it on and off.
- I chose to go with a Mail or iTunes-style source list view on the left because it allows the user to see all of the user-defined dictionaries at once and interact with one dictionary while viewing another.
- The addition of custom dictionaries will be a new feature. The Master dictionary will automatically be created on first launch and cannot be removed. Every word in this custom dictionary will be automatically added to all of the other dictionaries as well. The user will be free to create and remove as many other custom dictionaries as he or she wants as well.
- The actual table containing the words is largely unchanged. The only addition here is that of a second column, Master. If a word is checked, it is automatically added to the Master custom dictionary. Likewise, if a word is unchecked, it is removed from the master. The gear menu below will contain options to check and uncheck all of the entries in the table.



**Dictionary Cleaner 2.0 interface mockup.**

## Interaction

- The program itself will no longer be a System Preferences pane, instead becoming a full-fledged application.
- Users will be able to add words to dictionaries in four different ways:
  1. Clicking on the add button below the word table.
  2. Clicking the gear below the source list and selecting the option to import a comma delimited text file, adding its words to a new custom dictionary.
  3. Dragging and dropping selected words in the word table to any other dictionary in the list.
  4. Dragging and dropping a dictionary in the list onto any other dictionary to add all of the words from the dragged to the recipient.
- Users will be able to export any dictionary to a comma delimited text file by selecting a dictionary in the list, clicking on the gear icon below and selecting the export option.
- Support will be added for OpenOffice.Org user-defined dictionaries.

## Old+New

In this project I will use many technologies available to Cocoa programmers that I have either only implemented in useless demo apps or have no experience with at all, including:

- Cocoa bindings (NSTreeController)- Zero-code approach to automatically pass events and data between the view and model layers.



- NSOutlineView - Tree-like alternative to NSTableView.

- NSPathControl - Clicking on the path will take the user to the indicated folder.



- Archiving - To store the custom dictionaries.



- Drag and Drop - To copy words between dictionaries.

I will also be able to re-use a significant amount of of my model classes, including everything that reads, writes, and stores the various dictionaries. Virtually of of my synchronization code and glue code will be throw out.

## [Rich spendHours: int hours onTask: NSTask\* task];

### Interface

#### Feb 3rd

Setting up the interface with appropriate actions and outlets shouldn't take more than an hour or two.

## **Bindings**

### **Feb 4th - Feb 25th**

One of the most time consuming parts of this project will be to implement the bindings for the outline view. Bindings involve no code and are constructed entirely in interface builder, but they're also really hard to learn. Because outline views use trees as their underlying data source, they're quite a bit more complex than table views (which use arrays) or single-object bindings. I will need to set up bindings for both the outline view and the table view and all of their applicable functions (add, edit, remove, filter, sort, etc.). I need to go through a few tutorials, so I could easily see myself spending twelve to fifteen hours on this.

## **Archiving**

### **Feb 26th - Mar 4th**

Archiving doesn't look to be too difficult to implement, but I've never done it with Cocoa before so I should leave myself about five or six hours.

## **Drag and Drop**

### **Mar 5th - Mar 25th**

This is going to be another big one. I've read through the Apple documentation and examples, but I've never implemented anything like this before. I don't anticipate any major roadblocks here, but I have to allow ten to twelve hours for drag and drop.

## **Import and Export**

### **Mar 26th - Apr 1**

Should be fairly similar to archiving, but with more string parsing! This will take five to six hours.

## **Incidentals**

### **Apr 2 - Apr 10**

Check all buttons, testing, bug fixing, etc.

## **Evaluation**

I expect to be evaluated on whether or not I complete all of the goals listed in the solutions section and use all of the new technologies mentioned above. Don't hold me to the specifics of the interface mockup; I may have a brilliant idea down the road that will force me to change a few things around.