Steve Cahill, Marcus Reese, Zac Kruger

Including the name of our application and a short description here.

Programmer Manual

CSC442B - Group 8 - Simpletons

# Requirements

## Introduction

### Purpose

This section describes the functional requirements of the application.

### Scope

This project seeks to develop an English language simplifier, where one can input English text, and then receive output of a simplified version of that text.

### Definitions, Acronyms, and Abbreviations

To be included as needed

### Overview of this Section

This section describes the functional requirements of the application.

## General Description

### Product Perspective

English is a complex language, and deciphering word meaning is a tough undertaking. A recent Harvard University/Google joint study pointed to a count of words in the English language at over 1,000,000, and that the size of the language has more than doubled in the last 100 years. Given that the average vocabulary for a native speaker is between 5,000 and 10,000, and the vocabulary for a non-native English speaker is much lower, participants in this language will frequently encounter words that must be simplified in order to improve understanding.

### Product Functions

The simplification will occur using some defined lists of words, their synonyms, and properties such as “reading level.” Some of these lists already exist, such as “Simplified English,” the “Oxford 3000,” and “Common 5000.” The user will input a word or a series of words, and the application will compare those words (and their synonyms) to these lists, identifying words that can be simplified for the user.

### User Characteristics

#### Non-Native English Speakers

#### Educators and Students

### General Constraints

#### Web Browser

Google Chrome, most recent version

#### Internet

The user must have Internet access

#### Server Uptime

The application server must be up and reachable over the Internet

### Assumptions and Dependencies

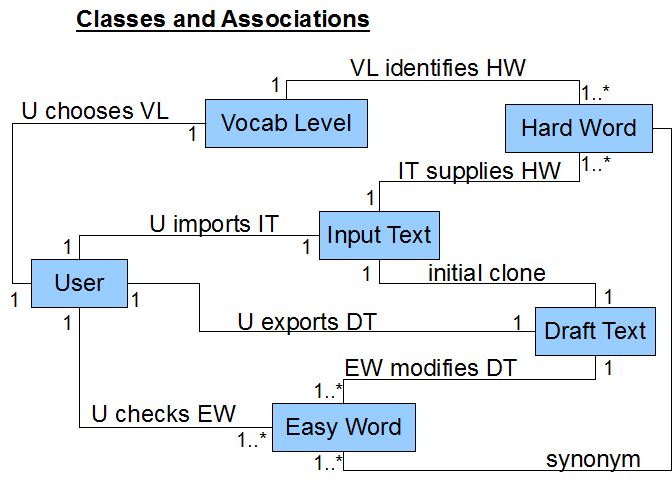
## Specific Requirements

|  |  |  |
| --- | --- | --- |
| Number | Requirement Text | Priority |
| **1** | **START** | **N/A** |
| 1.1 | The first start tool shall be labeled "1.1 Importing method" | 1 |
| 1.1.1 | The first option/item in the "importing method" tool shall be to type or paste text into a textarea | 1 |
| 1.1.1.1 | The textarea option shall be selected by default. | 1 |
| 1.1.1.2 | When the textarea option is selected, a textarea shall be displayed in the second importing tool (1.2). | 1 |
| 1.1.2 | The second option shall be to import text from a text file--if this feature is available. | 2 |
| 1.1.2.1 | If this option is selected for the first time, the user shall be allowed to browse files and select a text file for import. | 2 |
| 1.1.2.1.1 | If a user has selected a file, the contents of the file shall be displayed in the second importing tool (1.2). | 2 |
| 1.1.3 | The third option and subsequent options shall be available for previously saved projects--if this feature is available. | 3 |
| 1.1.3.1 | If the project saving feature is available but unused, there will be no third option in the first importing tool, but in its place below the tool shall be a note that says "(No saved projects)" or something to that effect. | 3 |
| 1.1.3.2 | If there are saved projects, each of these shall have its title displayed as an item in the first tool, starting in third position. | 3 |
| 1.1.3.2.1 | When a saved project item is selected, its text shall be displayed in the second importing tool (1.2). | 3 |
| 1.2 | The second start tool shall be labeled "1.2 The text to simplify" or something to that effect. | 1 |
| 1.2.1 | The first item in the "text to simplify" tool shall include a check box or some similar device indicating whether or not the program will "remember" the state of the project if/when the window is closed. The word "save" would be ambiguous in this context as the file-saving process is distinct from this remembering. | 3 |
| 1.2.1.1 | The "remember" option shall be checked by default. | 3 |
| 1.2.2 | The second item in the "text to simplify" tool shall contain the title and content of the text to be processed, if any. | 1 |
| 1.2.2.1 | If the method of input is to be typing or pasting in a textarea (1.1.1), this item shall include a text field for the title and a textarea for the content. | 1 |
| 1.2.2.2 | If the method of input is to be opening a text file, this item shall contain the file name followed by the text of the file--if the size of the file does not impede normal operation of the system. | 2 |
| 1.2.2.3 | If the method of input is for the system to remember a previous project, the remembered name and text shall be used as defined in 1.2.2.2. | 3 |
| **2** | **SIMPLIFYING** | **N/A** |
| 2.1 | The first simplification tool shall have the title "2.1 Vocab Goal" or some equivalent title. | 1 |
| 2.1.1 | The system shall compare submitted text against at least one default list of simple words/phrases to determine which submitted words are too hard. In this context, "list" implies that these words/phrases can be represented as a list, not that it is actually represented that way in the system. | 1 |
| 2.1.2 | The system shall offer the user five to eight vocabulary levels (lists of simple words labeled according to number of words allowed) and adjust the collection of hard words accordingly. | 2 |
| 2.1.3 | In addition to the basic five to eight vocabulary levels, the system shall offer options (e.g., check all that apply) for target locales (e.g. Australia, Bahamas, etc.) | 3 |
| 2.1.4 | When data is fully processed, each Vocab Goal (including both level items and locale options) shall have a number displayed near it, indicating the number of changes that will be required for the document to comply with that goal. | 1 |
| 2.1.5 | Selecting a Vocab Goal item or option or deselecting an option shall result in the updating of the display of tools of a higher number, allowing the user to see quickly the results of aiming for various levels of vocabulary. | 1 |
| 2.2 | The second simplification tool shall have the title "2.2 Result Type" or some equivalent title. | 1 |
| 2.2.1 | The list shall have dividers labeled "Unchecked" and "Checked" or the equivalent. | 1 |
| 2.2.1.1 | Under "Unchecked", there shall be two items labeled "Low-Risk" and "High-Risk" or the equivalent. "Low-Risk" means results of data that fit into the expectations of the system. "High-Risk" means that something results of data that the system was not fully prepared for. Low-Risk is listed first to give the user a good impression of the results before focusing on the possibly embarrassing "High-Risk" results. | 1 |
| 2.2.1.2 | Under "Checked", there shall be two items labeled "Successes" and "Exceptions". Successes are changes that the user has approved in which the new word or phrase qualifies as simple according to the selected vocabulary goals (level and locales). Exceptions are the results (changes or non-changes) that the user has approved in which the word or phrase is still outside the limits of the Vocabulary Goals. Having these types allows the user to revisit past decisions. | 1 |
| 2.2.2 | Selecting a Result Type item shall result in the updating of the display of tools of a higher number, allowing the user to focus first on a few Low-Risk results for a good impression, then to focus on High-Risk results for priority validation, then to review more Low-Risk results, double-check Exceptions, and double-check or admire Successes as time allows. | 1 |
| 2.2.3 | When the data is fully processed, the system shall display next to each of the four items a number indicating how many results are in each type or section based on the selected Vocabulary Goals. | 2 |
| 2.2.3.1 | After that, those numbers shall be updated each time the user approves a result. | 2 |
| 2.2.3.2 | If a user approves a default replacement in a High-Risk word or phrase, all results related to that original word or phrase shall be moved to the Low-Risk type/category with the default replacement applied. | 3 |
| 2.3 | The third simplification tool shall have the title "2.3 Hard Word" or some equivalent title. | 1 |
| 2.3.1 | The items in this list shall consist of words or phrases in the submitted text that are too difficult according to the selected Vocabulary Goals and categorized as belonging to the selected Result Type. | 1 |
| 2.3.2 | Selecting a Hard Word item shall result in the updating of the display of tools of a higher number, allowing the user to focus on one hard word or phrase at a time. | 1 |
| 2.3.3 | When the user imports text, and simplifying tools are automatically displayed, the system shall find the first four hard words, if that many exist, and make these and their associated results available for the user to work with before processing the rest of the original text. This is to avoid delays for the user. | 1 |
| 2.3.3.1 | The display shall be updated with all currently processed data when the user selects any of the instructions. | 2 |
| 2.3.3.1.1 | If the user selects the first or third main instruction (see 1.1) and the data has not been fully displayed, the system shall inform the user that the data is still being processed and suggest that the reader return to step 2. | 2 |
| 2.3.3.2 | The display shall be updated with all currently processed data when the user has selected every available hard word. | 2 |
| 2.3.3.3 | The display shall be updated when all data has been processed. | 1 |
| 2.3.4 | If the Hard Word list is longer than can be viewed on the screen, the GUI shall provide above it a filter text-input bar, and the system shall use input text from that bar to reduce the number of displayed items so that the user can quickly find a certain word, especially one that may be brought to mind by the text in 2.4 Sentence/Instance or 2.5 Old Context. | 2 |
| 2.4 | The fourth simplification tool shall have the title "2.4 Sentence" or "2.4 Instance" or some equivalent title. | 1 |
| 2.4.1 | Each item in this list shall consist of a sentence in which the selected Hard Word occurs in the original text. | 1 |
| 2.4.2 | Selecting a Sentence/Instance item shall result in the updating of the display of tools of a higher number, allowing the user to focus on one sentence at a time. | 1 |
| 2.5 | The fifth simplification tool shall have the title "2.5 Old Context" or some equivalent title. | 1 |
| 2.5.1 | There shall be three items in this list. The middle one shall be the selected sentence. The first item shall be the sentence that precedes the selected sentence in the original text. The third item shall be the sentence that follows the selected sentence in the original text. | 1 |
| 2.5.2 | The system shall allow the user to optionally view higher or lower sentences. | 3 |
| 2.6 | The sixth simplification tool shall have the title "2.6 Part of Speech" or "2.6 Sense" or some equivalent title. | 1 |
| 2.6.1 | The primary items in this list shall include "Noun", "Adjective", "Verb", "Adverb", and "Other". | 1 |
| 2.6.2 | The secondary items in this list shall include senses within each part of speech. | 3 |
| 2.6.3 | Selecting an item shall result in the updating of higher numbered lists (2.7 and 2.8) as needed. | 3 |
| 2.7 | The seventh simplification tool shall have the title "2.7 Easier Word/Phrase" or "2.7 Replacement" or some equivalent title. | 1 |
| 2.7.1 | The first items in this list shall consist of a selection of simple synonyms (words or phrases) that can be replacements for the selected hard word or phrase, and each of these items shall have its associated level and locale listed with it. | 2 |
| 2.7.1.1 | The Easier Word/Phrase items shall have the same inflection as the original word so that they can be automatically reinserted into the sentence in the place of the hard word or phrase. | 3 |
| 2.7.1.2 | When the user selects a replacement word or phrase, the system shall use that to create or update a simplified draft text and thus update 2.8 New Context. | 3 |
| 2.7.2 | The system shall provide a place for the user to enter and submit an easier word as replacement for the hard word in focus (see 3.3). | 1 |
| 2.7.2.1 | When the user submits a replacement word or phrase, the system shall use that to create or update a simplified draft text and thus update 2.8 New Context. | 1 |
| 2.7.2.2 | The system shall store the user-submitted word and offer it as an alternative replacement for other instances of the hard word in focus. | 2 |
| 2.7.3 | Underneath or above the Easier Word/Phrase list, the GUI shall provide a "set as default" option, though not necessarily with those exact words. | 3 |
| 2.7.3.1 | The "set as default" option shall be selected by default. | 3 |
| 2.7.3.2 | If the "set as default" option is selected when a change is approved in 2.8, the system shall associate the selected Easier Word/Phrase with the selected Hard Word and update any Unchecked results accordingly. | 3 |
| 2.7.3.2.1 | If an Easier Word/Phrase is set as default for a High-Risk Hard Word, the system shall recategorize that Hard Word and all associated instances as Low-Risk. | 3 |
| 2.8 | The eighth and final simplification tool shall have the title "New Context" or some equivalent title. | 1 |
| 2.8.1 | There shall be three items in this list. The middle one shall be the selected sentence. The first item shall be the sentence that precedes the selected sentence in the new draft text. The third item shall be the sentence that follows the selected sentence in the new draft text. | 1 |
| 2.8.2 | The system shall allow the user to optionally view higher or lower sentences. | 3 |
| 2.8.3 | Underneath or above the New Context list, the GUI shall provide an "approve" option, though not necessarily with that exact word. | 1 |
| 2.8.3.1 | The "approve" option shall not be selected by default. | 1 |
| 2.8.3.2 | If the user selects the "approve" option, the system shall recategorize the current results under Checked as either one of the Successes or the Exceptions. | 1 |
| **3** | **EXITING (see points under 1.2)** | **N/A** |
| 3.1 | The first exit tool shall have "3.1 Status: " as the first part of its label, but the rest of the label shall depend on the application state. | 1 |
| 3.1.1 | If there are unprocessed results, the label shall warn the user that it is still processing and recommend returning to Step 2. | 1 |
| 3.1.2 | If the results are processed but there are unchecked results, the label shall warn the user that some results are unchecked. | 1 |
| 3.1.3 | If there are unchecked results, the first selectable item in the Status tool shall say "Keep checked replacements only" or something to that effect. | 1 |
| 3.1.3.1 | If the user chooses to keep checked replacements only, the output in the second exit tool shall reflect that choice. | 1 |
| 3.1.4 | If there are unchecked results, the second selectable item in the Status tool shall say "Keep both checked and unchecked replacements" or something to that effect. | 1 |
| 3.1.4.1 | If the user chooses to keep both checked and unchecked replacements, the output in the second exit tool shall reflect that choice. | 1 |
| 3.1.5 | If the results are all checked, the status label shall congratulate the user on completing the task. | 1 |
| 3.1.6 | If there are no unchecked results, there shall be no selectable list items under the status label. | 1 |
| 3.1.7 | Regardless of the status, under the status tool shall be a check box or some similar device allowing the user to choose whether or not the system should remembering/saving the application state, in order to resume after closing and restarting the application--if this feature is available. | 3 |
| 3.1.7.1 | The default state of 3.1.7 shall match the corresponding state of 1.2.1. | 3 |
| 3.1.7.2 | If the project state is saved/remembered and the option to remember is selected, there shall be a notice underneath the option stating that the saving/remembering was successful. | 3 |
| 3.2 | The second exit tool shall be labeled "1.2 The simplified text" or something to that effect. | 1 |
| 3.2.1 | The first item in the "simplfied text" tool shall include a check box or some similar device indicating whether or not the program should save the output to a file--if this feature is available. | 2 |
| 3.2.1.1 | The save to file option shall be unchecked by default. | 2 |
| 3.2.1.2 | If the save to file option is selected, the user should be allowed to browse to a folder and save. | 2 |
| 3.2.1.3 | If the save to file action is successful, the option shall remain checked and a success notice shall appear beneath it. | 2 |
| 3.2.2 | The second item in the "text to simplify" tool shall contain the title and content of the output text, if possible. | 1 |
| 3.2.2.1 | The title shall be in a text field so that it can be modified by the user. | 2 |
| 3.2.2.1.1 | The default title shall be a slight modification of the name of the original if the original is from a file, or the same as the original if it did not come from a file. | 2 |
| 3.2.2.1.2 | The default title shall be used in the save to file dialog. | 2 |
| 3.2.2.2 | Below the title field shall be the output text, if such exists and if it is not so large as to interfere with the normal functioning of the system. | 1 |
| 3.2.2.3 | If the method of input is to be opening a text file, this item shall contain the file name followed by the text of the file--if the size of the file does not impede normal operation of the system. | 2 |
| **4** | **GENERAL REQUIREMENTS** | **N/A** |
| 4.1 | Tools in the GUI shall consist of lists that exhibit "inline-block" behavior (fitting as many as possible in a window but lining up one under the other in the case of a small window) so as to adapt to various screen/window sizes. | 1 |
| 4.2 | Every page/view of the GUI shall provide at least the following three numbered main instructions or steps: 1) Start, 2) Simplify, and 3) Finish (not necessarily in those words). | 1 |
| 4.2.1 | Step 1 (Start) shall be visually different than the other instructions at the beginning, indicating current focus. | 1 |
| 4.2.2 | The system shall allow the user to shift focus manually to any of the three steps. | 1 |
| 4.2.3 | Whenever focus shifts (whether manually or automatically) to a new step, the new current step shall be visually different than the other two steps. | 1 |
| 4.2.4 | The current step shall determine some of the other elements on the screen. | 1 |
| 4.2.4.1 | If and only if the focus is on the Step 1, the screen should hold one or more tools for importing text and possibly recovering an earlier project. | 1 |
| 4.2.4.2 | If and only if the focus is on the Step 2, the screen should hold tools for simplifying text. | 1 |
| 4.2.4.3 | If and only if the focus is on Step 3, the screen should hold one or more tools for exporting text and possibly saving project state. | 1 |
| 4.3 | There should be three main colors for the lists/tools: one for titles (and possibly other non-selectable items), one for unselected items, and one for selected items. | 1 |
| 4.4 | If a list or collection of options is too long, scroll bars shall be used to keep all content small enough for a typical desktop or notebook screen. This is to prevent a long list effectively hiding the options below it and also to make the whole program more usable in a mobile environment. | 2 |
| 4.5 | On a typical notebook or desktop screen, all simplification tools must be at least partly visible when the second main instruction is selected and the web page scroll bar is in its highest position. On mobile screens this is likely to be impossible. | 1 |

## Appendices

# Design

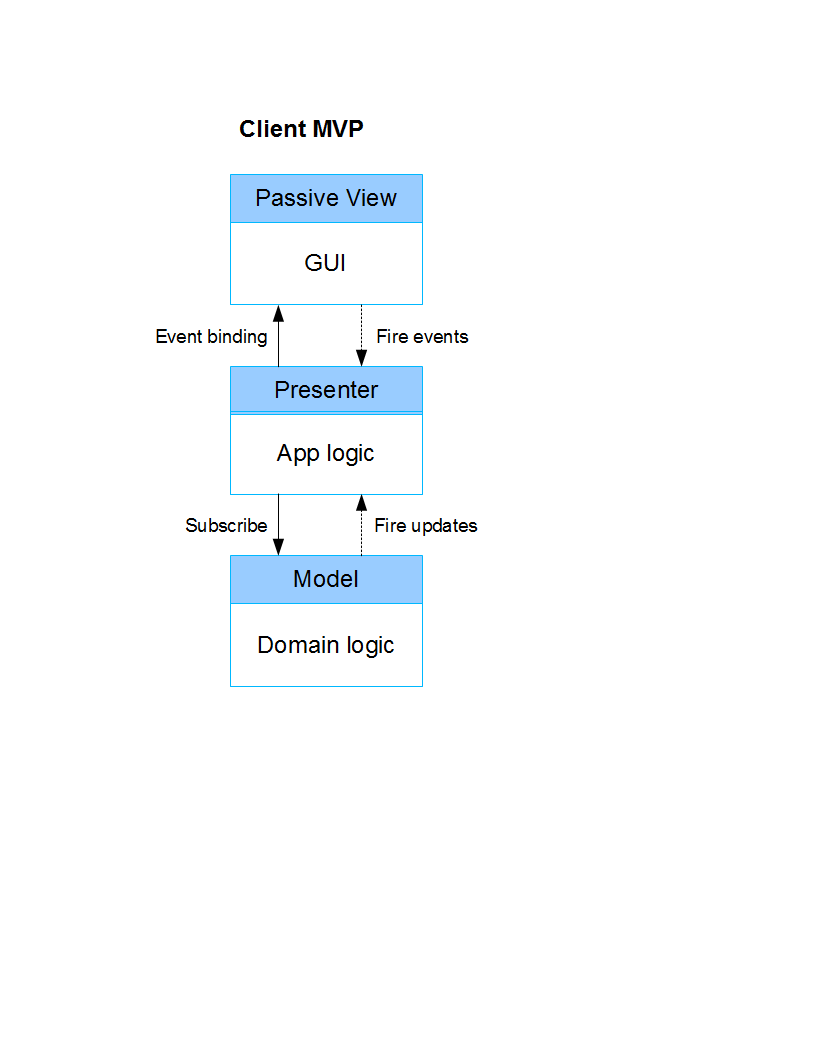
## Class Diagram



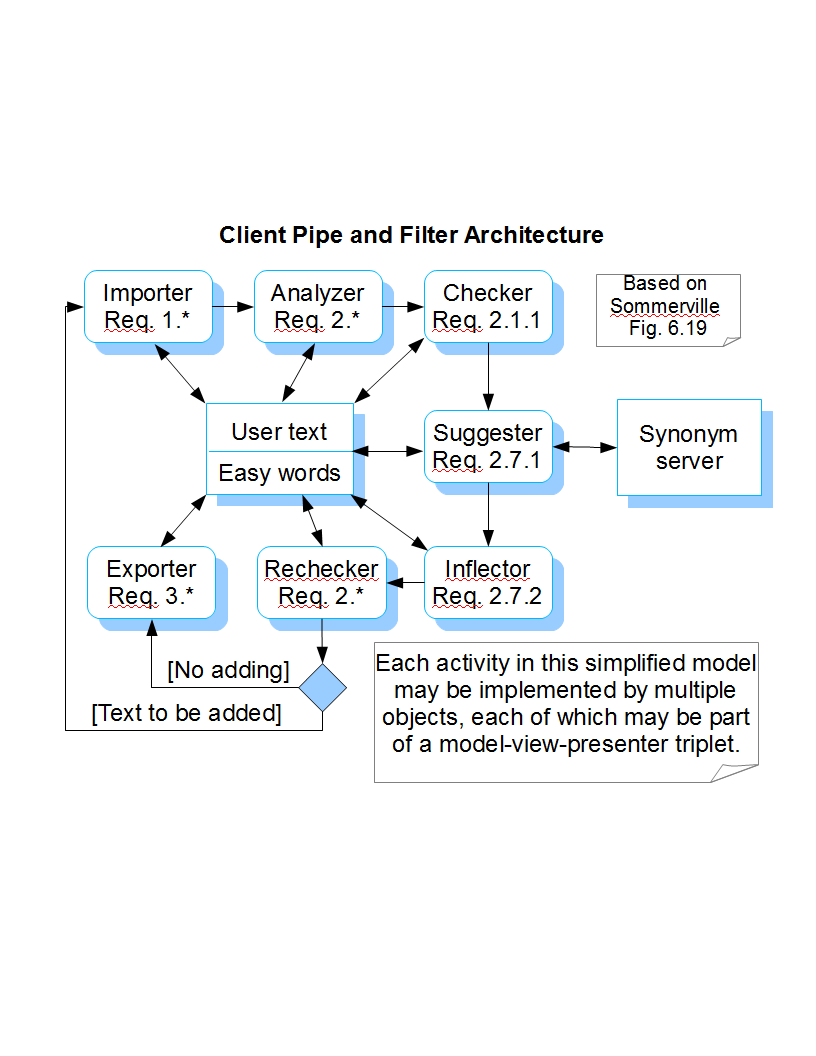
## Process and GUI Diagram

## 

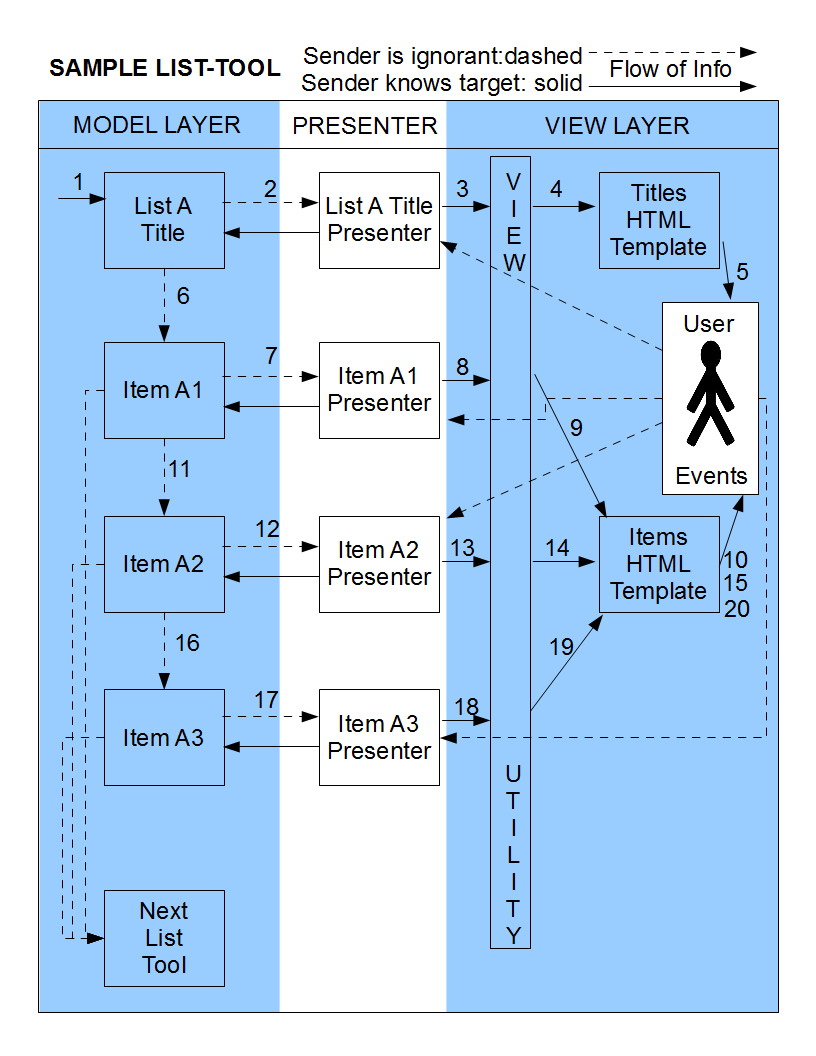
## Client MVP Diagram



## Client Pipe & Filter Diagram



## List Tool Diagram



## Structural Design of Data

STRUCTURAL DESIGN OF DATA

Note the role of quotation marks in defining some keys as string literals (one per instance), as opposed to variables (may be many per instance).

Variable keys have no siblings in this document, and leaf node values are JavaScript primitives, and other node values are containers.

This document was not done before all coding, but it was done before the sprint that coded most of this structure.

level: object (max of 6 levels)

type: object (4 types)

hardWord: object

"definitions": object

partOfSpeech: definition (string)

"defaults": object

partOfSpeech: suggestion (string)

"sentenceInfo": object

sentenceId: object (sentenceId can be inputSentenceId or outputSentenceId)

textArrayIndex: object (this and contents can best be added in Analyzer if needed since a hard word may be found more than once in a sentence)

"replacement": suggestion (string)

"partOfSpeech": string

"suggestions": object

partOfSpeech: array

number: object

"suggestion": string

"vocab": string

"synset": number

inputSentenceId: object

"textArray": array

number: string

outputSentenceId: object

"textArray": array

number: string

## Pseudocode

Pseudocode for client of simplifier program

ITEM INTERFACE

+getLocalId():String

+getDataEtc():Object

+setDataEtc(dataEtc:Object):Boolean

+checkSubscriberId(subscriberId:String):Booleanish Number

+checkPublisherId(publisherId:String):Booleanish Number

+addSubscriberId(subscriberId:String, priority:Number):Boolean

+removeSubscriberId(subscriberId:String):Boolean

+addPublisherId(publisherId:String):Boolean

+removePublisherId(publisherId:String):Boolean

+update(publisherId:String):Boolean

+getUpdateFunction():Function

+setUpdateFunction(update:Function):Boolean

ITEM MANAGER INTERFACE

+createItem(core:Object):Item

+deleteItem(item:Item):Boolean

+getItem(localId):Item

+setItem(item:Item):Boolean

ITEM CORE INTERFACE

localId:String

DataEtc:Object

DataEtc.data

DataEtc.dataType:String

SubscriberIds:Array<String>

PublisherIds:Array<String>

Update:Function(publisherId:String):Boolean

ACCEPTED DATE: 6 MARCH 2013

//+createItem(core:Object):Item

//check given core for localId

//if core.localId,

//check that it is valid

//if not string, to string

//if > 64, chop it off there

//check if localId is used

//if used, modify, saving originalLocalId in dataEtc.

//chop make max 62 to allow for underscore and number

//add underscore and increasing numbers until unused id is found

//every time mod10 is 0, chop off a letter from the 64 max?

//if no core.localId, check for dataEtc.data

//if none, not a true core

//convert to string

//if no string, create date+time string

//create core

//put string in dataEtc.data

//use dataEtc.data to create localId

//accept first 15 letters (enough for date string)

//then, if space, see if used

//if not, add more letters (not ending with space until unused id is found

//if up to 33(enough for time string), add underscore and number

//+getLocalId():String

//+getDataEtc():Object

//+setDataEtc(dataEtc:Object):Boolean

//send this to the update functions of subscribers

//if subscriber does not exist, console.log warning

//if unchecked publishers still do not exist, console.log warning

//if unchecked publishers now exist but do not have this listed as subscriber, console.log warning and subscribe

//delete unchecked publishers list

//create private subscribers and publishers arrays and also indices for easy lookup?

//check for subscribers

//if subscribers, check that typeof each === string

//if not, throw exception

//check for publishers

//if subscribers, check that typeof each === string

//if not, throw exception

//check if publishers have this listed as subscriber

//if not, console.log warning and subscribe

//if publishers do not exist, keep a list of unchecked publishers

//+checkSubscriber(subscriberId:String):Boolean

//+checkPublisher(publisherId:String):Boolean

//+addSubscriberId(subscriberId:String, priority:Number):Boolean

//no immediate update because may cause unwanted behavior, and they just looked at us

//somewhere publisher needs to be added--here or elsewhere?

//+removeSubscriberId(subscriberId:String):Boolean

//somewhere publisher needs to be removed--here or elsewhere?

//+addPublisherId(publisherId:String):Boolean

//+removePublisherId(publisherId:String):Boolean

//inner update function should be in core

//check if it's already there

//if so, ensure typeof === function or some such check, otherwise throw exception

//+update(publisher:Item):Boolean (this is where others update this; for this updating others, see setDataEtc.)

//outer update function

//if this did not already have publisher listed,

//console.log warning

//add publisher

//if unchecked publishers still do not exist, console.log warning

//if unchecked publishers now exist but do not have this listed as subscriber, console.log warning and subscribe

//delete unchecked publishers list

//run supplied core update function

//+getUpdateFunction():Function

//+setUpdateFunction(update:Function):Boolean

//ensure typeof === function or some such check, otherwise return boolean

//+deleteItem(item:Item):Boolean

//I think setting value to null will be sufficient

//+getItem(localId):Item (

//no false needed in interface def, as long as it returns something falsey

//+setItem(item:Item):Boolean

//if no item.getId, throw exception

Vertex interface

+getId():String

+setId(id:String)

+getDataEtc():Object

+setDataEtc(dataEtc:Object)

+getSubscribers():Array

+getPublishers():Array

+createVertex(core:Object):Vertex

+addSubscriber(vertex:Vertex, priority:Number)

+removeSubscriber(vertex:Vertex)

+update(publisher:Vertex)

+beDeleted()

ACCEPTED DATE: 5 MARCH 2013

BELOW THIS IS OLD, MUCH MAY BE DELETED

do Vertex constructor

check core, possibly supply defaults

core should be something like...

{

id: 'Step\_1\_Import\_Text",

dataEtc :

{

dataType: 'text',

data: 'Step 1: Import Text',

created: '5 Mar 2013',

upPoints: '1',

downPoints: '0'

}

subscribers:

[

idX,

idY,

idZ

]

publishers:

[

idV,

idW,

idX

]

//update: ?

}

//Req. 4.2: Steps on every virtual page

//Create Steps tool model (at first with title and orientation),

//Create Steps tool views (one for invisible tool container, one for title to be used if title data exists)

//Create Steps tool presenter

//Connect them via subscription

//Changing model data to active to display (with first but finally without title)

//Try separating presenter from its functions, for reuse

//Create Step1 model

//Create Step1 view, if not reusing

//Create Step1 presenter, try to reuse functions

//Connect them together to display

//Create Step2 model

//Create Step2 view, if not reusing

//Create Step2 presenter, try to reuse functions

//Connect them together to display (and ensure switching steps changes state via console.log)

//Create Step3 model

//Create Step3 view, if not reusing

//Create Step3 presenter, try to reuse functions

//Connect them together to display

//Create 1.1 Import Method tool model

//Create 1.1 Import Method view, if not reusing

//Create 1.1 Import Method presenter, try to reuse functions

//Connect them together to display

//Create textarea button model

//Create textarea button view

//Create textarea button presenter

//Connect it with tool/title and Step1 to display (and ensure clicking other steps makes 1.1 disappear)

//If time, try dev mode 0.1 by looping through allVertices and displaying a crud interface

//Then move on to completing importer, analyzer (chunking), checker(comparison with simplicity standard), rechecker (manual), exporter

//algorithm for list:

//record time in processedWordList

//split by new lines

//split by seperators

//in each line:

// convert first into id in core

// but if parentheses, create a quick item that says variantOf:

// stem? or do that lazily?

// second: use as key for new object

// in new object, state level as 3

// remainder of line are "extra" synonyms--for use when this turns up as a synonym

// save core with others in processedWordList for human checking

// createItem

//record ending time in processedWordList

//break up user text

//find index of first alphabetical character

//if at beginning (0), disregard index, else split and take first as non-word

//for every non-word, add to current sentence

//for every word, check first letter and remember whether capitalized or not

//also check for the word in simple list

//consider stemming, affixes, etc.

//if not, add it as key in dataEtc.hardWords, if not already, and push next sentence id to value array

//for every non-word, keep track of first char, last (and length?)

//if last word was not cap (i.e. avoid "Mr. X"), last non word started with sentence-ending punctuation and ended with space, and next word starts with cap

//then finalize the sentence, save it with next numbered id

updateFrom: function updateFromForHardWordsPresenter(publisher) {

//if publisher is Hard Words tool model, get hard words and turn them into an array for displaying

//(before they were in object for lookup by the word itself when encountered in the text, but present view prefers numbers)

//if isDisplayable and not isDisplayed, give them to view utility one by one and set isDisplayed to true

//if publisher is Vocab Level tool model, check if hardWordsArray length indicates Hard Words tool model has processed

//if so and isDisplayed is false, give hard words to view utility one by one and set isDisplayed to true

//regardless, set isDisplayable to true, probably

//if update is run by view markup, it has set selectedIndex here, so fetch and set selectedWord in model for future observers

//affixes, derivational (from Specialized English) (not yet added to the simplifier word list(s))

//if it starts with one of the following prefixes, cut that off and check the rest; let user see both the prefix meaning and the other meaning

//anti- - against, opposed to (anti-government)

//dis- - not (dishonest, disobey)

//pro- - for, supporting (pro-government)

//re- - to do again (reorganize, reunite)

//self- - acted upon /by self (self-declared, self-appointed)

//un- - not (unusual, unhappy)

//If it ends with one of the following suffixes, cut that off and check the rest; let user see both the prefix meaning and the other meaning

//These four may be added to a word to mean â€œthe person or thing that does the action suggested by the wordâ€-

//-ist - (art >artist), -er - (paint > painter), -or - (act > actor), -ant - (serve > servant)

//These five can change a verb into a noun closely related to the verb-

//-ment - (agree > agreement), -age - (marry > marriage), -tion - (protect > protection), -sion - (invade > invasion), -ing - (I like to read > I enjoy reading. A noun that uses -ing in this way is called a gerund.)

//These suffixes can change some nouns into adjectives-

//-ful - (care > careful),-al - (tradition > traditional)

//These suffixes can change some verbs to adjectives-

// -ive - (protect > protective), -ing - (save > saving)

//And this one can change some adjectives into nouns-

//-ness- (willing > willingness)

Req 1.1

Create Item 1.1 Importing Method

Create Item 1.2 The text to simplify

Create Item 2.1 Vocab Goal

Create Item 2.2 Result Type"

2.2.1 The list shall have dividers labeled "Unchecked" and "Checked" or the equivalent. Priority: 1

2.2.1.1 Under "Unchecked", there shall be two items labeled "Low-Risk" and "High-Risk" or the equivalent. "Low-Risk" means results of data that fit into the expectations of the system. "High-Risk" means that something results of data that the system was not fully prepared for. Low-Risk is listed first to give the user a good impression of the results before focusing on the possibly embarrassing "High-Risk" results. Priority: 1

2.2.1.2 Under "Checked", there shall be two items labeled "Successes" and "Exceptions"

Create Item 2.3 Hard Word

Create Item 2.4 Sentence

Create Item 2.5 Part of Speech

//Create Item 2.6 Replacement

//Change title to "2.6 Replacement (Level, P.o.S.)" to function as column headings

//When user clicks on a part-of-speech, use that to replace the existing part-of-speech for that word

//Req. 2.6.1.2, Req. 2.6.2 Copy the selected word, but not pos/level into the text field

//Req. 2.6.2.1 Use the selected word to populate the replacement field for the hardWord-sentence combo

//Loop through the words in the sentence (if hard, using replacements where they exist), and use them to create an output sentence.

//Also, if user changes whole sentence, save it as an output sentence, but don't add it to checked category; Check it for hard words and add those to unchecked.

//For Output, loop through sentences, and for each, check for a parallel output sentence

//Place a "default for [hardWordX]" checkbox next to the text field

//Req. 2.6.3.1 Keep a default field for each hard word (with sentences and suggestions), and check it before using replacement

//Add original word with level as (last?) replacement suggestion.

//for each suggested word, get level.

//sort suggested words by (in increasing priority): level, PoS, default

//When part-of-speech is known, use that to order the suggestions

//Use level to order the suggestions within part-of-speech (PoS has priority over level, so do level first)

//Req. 2.2 in each level container, keep four containers of hard words: uncheckedHighRisk, uncheckedLowRisk, checkedSuccesses, checkedExceptions

//Req. 2.1, Req. 2.2 the collection of four containers referenced in Type will be only one of the fours/levels/collections referenced in Level

//Req. 2.2, Req. 2.3 the container of hard words referenced in hard words tool will be only one of the containers referenced in Type

//Req. 2.3, Req. 2.4 the sentence referenced in sentence will be only one of the sentences referenced in hard words

//level object can keep a collection of hard word collections for each level, with lower levels only including what's not in higher levels

//level object can keep a record of selected levels (lower level selection implies higher level also selected)

//type object can keep a reference to the selected level(s)

//hard words object can

// 2.6.1 Most of the items in this list shall be simple synonyms (words or phrases) that can be replacements for the selected hard word or phrase, and each of these items shall have its associated level (and possibly locale and/or part of speech) listed with it. Priority: 2

// 2.6.1.1 Whenever it is feasible, the Easier Word/Phrase items shall have the same inflection as the original word so that they can be automatically reinserted into the sentence in the place of the hard word or phrase. Priority: 3

// 2.6.1.2 When the user selects a replacement word or phrase, the system shall use that to create or update a simplified draft text and thus update 2.7 Result. Priority: 3

// 2.6.2 The system shall provide a place for the user to enter and submit an easier word as replacement for the hard word in focus (see 3.3). Priority: 1

// 2.6.2.1 When the user submits a replacement word or phrase, the system shall use that to create or update a simplified draft text and thus update 2.7 Result. Priority: 1

// 2.6.2.2 The system shall store the user-submitted word and offer it as an alternative replacement for other instances of the hard word in focus. Priority: 2

// 2.6.3 Underneath or above the Easier Word/Phrase list, the GUI shall provide a "set as default" option, though not necessarily with those exact words. Priority: 3

// 2.6.3.1 The "set as default" option shall be selected by default. Priority: 3

// 2.6.3.2 If the "set as default" option is selected when a change is approved in 2.7, the system shall associate the selected Easier Word/Phrase with the selected Hard Word and update any Unchecked results accordingly. Priority: 3

// 2.6.3.2.1 If an Easier Word/Phrase is set as default for a High-Risk Hard Word, the system shall recategorize that Hard Word and all associated instances as

Create Item 2.7 Result

Create Item 3.1

Create Item 3.2

Create Item 1.1

//Req. 4.2.2, Req. 4.2.4 If one of the main step buttons are clicked, hide unneeded tools and show needed tools

//Req. 4.2.2 If user goes back to Step 1 and actually changes input, remove all and start completely from scratch

//Req. 4.2.4.1 If user clicks Step 1 and is not already there, hide Step 2 and Step 3 tools and show Step 1 tools.

//Req. 4.2.4.2 If user clicks Step 2 and is not already there, hide Step 1 and Step 3 tools and show Step 2 tools.

//Req. 4.2.4.3 If user clicks Step 3 and is not already there, hide Step 2 and Step 1 tools and show Step 3 tools.

# Testing

Source code includes a Mocha script which provides for and self-documents many automated test cases

# Known Bugs and Issues

Known Bugs:

base.createItem(' ') or base.createItem({dataEtc: {data: ' '}}) etc. will create ids that consist only of spaces

base.createItem is programmed to give up looking for unique id after about 10 trillion attempts. Is this too many or too few?

base.createItem is programmed to limit localIds to 64 chars. Is this too many or too few?

base does not deal with foreignIds (i.e. server ids). Can this be taken care of in dataEtc?

when item.setDataEtc() is passed an array, it accepts it, but maybe this is okay?

items have subscribers and publishers ids each stored twice, once in array and once in object to use hash instead of search. Will size matter?

The following test produces an unexpected id, which is 'faith\_9\_10'

describe("when item.addPublisherId is called with an invalid id", function () {

it("ensures item returns false", function() {

var item = base.createItem({localId: 'faith', publisherIds: ['hope', 'love']});

var result = item.addPublisherId([]);

It doesn't seem secure: getItem and getItems?

Need a way to export core?

Right now it does console.log when an id is requested but denied because already used or >64chars. Only user will see that if user opens browser console.

Better to have a log, perhaps item(s) that both user and server/webmaster can have access to?

Base forces me in presenter to record modelId twice: once in publisherIds and once as model in dataEtc. This leads to unexpected program behavior if one is set to one id and the other to another (i.e. allows programmer more scope for error). Perhaps both publishers and subscribers could be listed with associated keywords. But then why not just give them each their own data area and avoid one module/object interfering with another? Because then how could they talk to one another? I suppose they all need read access, but only write access where they themselves created the keyword within dataEtc.

On a related topic, when a change occurs, all subscribers are updated, making debugging difficult if there are many subscribers? Better to have them subscribe to certain "topics"? (though some may need all changes--eg. undo log)? As in pubsubjs?

On a related topic, it notifies for EVERY change. I guess that's good for logging, but sometimes I wish we could stipulate suppressMostPublishing or require manual publish() for most publishing, i.e. that which is not important for undo, etc.

body parts are defined as 'part of the body'

The following notes indicate known bugs:

//TODO add stemming here, trying again to make it easy if it presently fails the isEasy test (e.g. 'Did')

//TODO add affixes here, trying again to make it easy if it presently fails the isEasy test (e.g. 'undo', because do is in list and un- is easy)

1.2 "The text to simplify" tool GUI--the blue does not connect visually to the gray the way it should.

This program is not going to highlight words that can be easy or hard depending on the sense.

For example, the word "over" is in the easy list, but if you use it in the specialized Cricket sporting sense (e.g. "after fifteen overs"), it won't be easy to some readers.

This program needs to highlight the focus word in the displayed sentence and display a sentence more than once when there are multiple instances of the same hard word in it. That way the user can choose a different replacement for each. Presently the user can only choose a single replacement to cover both, but can manually change one in 2.7.