

# **Software Design Document**

**for**

## **Activation-Selection Model Simulation of Human Memory**

**Version 1.0**

**Prepared by:**

**Group Three**

**Reza Moghtaderi Esfahani  
Jonathan Sekela  
Graham Strong**

**682169  
723192  
1053882**

**moghtr12@highpoint.edu  
sekelj13@highpoint.edu  
strong14@highpoint.edu**

**Client:** Dr. Kimberly Wear, HPU Psychology Department  
kwear@highpoint.edu 336-841-9246  
**Course:** Software Design and Engineering  
**Date:** April 10, 2016

# Contents

## Table of Contents

- 1. INTRODUCTION
  - 1.1 PURPOSE
  - 1.2 SCOPE
  - 1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS
- 2. REFERENCES
- 3. DECOMPOSITION DESCRIPTION
  - 3.1 MODULE DECOMPOSITION
    - 3.1.1 MODULE 1 DESCRIPTION
    - 3.1.2 MODULE 2 DESCRIPTION
  - 3.2 CONCURRENT PROCESS DECOMPOSITION
  - 3.3 DATA DECOMPOSITION
    - 3.3.1 DATA ENTRY 1 DESCRIPTION
    - 3.3.2 DATA ENTRY 2 DESCRIPTION
  - 3.4 DESIGN CONSTRAINTS
- 4. DEPENDENCY DESCRIPTION
  - 4.1 INTERMODULE DEPENDENCIES
  - 4.2 INTERPROCESS DEPENDENCIES
  - 4.3 DATA DEPENDENCIES
- 5. INTERFACE DESCRIPTION
  - 5.1 MODULE INTERFACE
    - 5.1.1 MODULE 1 DESCRIPTION
    - 5.1.2 MODULE 2 DESCRIPTION
  - 5.2 PROCESS INTERFACE
    - 5.2.1 PROCESS 1 DESCRIPTION
    - 5.2.2 PROCESS 2 DESCRIPTION
- 6. DETAILED DESIGN
  - 6.1 MODULE DETAILED DESIGN
    - 6.1.1 MODULE 1 DETAIL
    - 6.1.2 MODULE 2 DETAIL
  - 6.2 DATA DETAILED DESIGN

# **1. Introduction**

## **1.1 Purpose**

This document describes the design decisions and modules for a Psychology research simulator based on the ASM paper references in SRS section 1.

## **1.2 Scope**

See SRS section 1.2

## **1.3 Definitions, Acronyms, and Abbreviations**

See SRS section 1.4

# **2. References**

At this point, we do not have any other references.

# **3. Decomposition Description**

## **3.1 Module Decomposition**

### **3.1.1 Help/Tutorial Module**

This module will be in the form of an independent HTML page that will show as a separate window in the desktop app. The content will include how to import data, how to use the main simulation and

priming modules, links to support and publication source, and any other static data requested by the client.

### **3.1.2 Main Simulation Module**

This module will be written in html and JavaScript. It executes most of the backend logic of the ASM project. It consists of two arrays of an equal number of attribute objects, along with roughly a dozen other global variables and arrays that make up the JavaScript Attribute class.

Each attribute object contains a weight, which is number between 0 and 1, where 0 is inactive and 1 is very active. The simulation module's input is the output from the priming module (3.1.3). It uses this input to select a certain number of attributes (see 3.1.6 settings) and increase their multiplier value. This multiplier value change affects the probability of the activated attributes being chosen upon pressing of the "select meaning" button.

### **3.1.3 Priming Module**

This module contains 3 buttons: Prime Dominant (PD), Prime Secondary (PS), Select Meaning (SM). PD causes the simulation module to activate mostly dominant nodes, PS causes the simulation module to activate mostly secondary nodes.

SM simulates the brain actually choosing a meaning when given a homophone. It adds up all live weights of all attributes, selects a given number of attributes based on the probability of each one being chosen, and updates the static weight of each chosen attribute to a newer, higher value. This essentially skews the ratio between dominant and secondary meaning based on what attributes were randomly chosen by SM.

#### **3.1.4 Import/Export Data Module**

Contains word-specific settings, such as the spellings of the desired homophone pair, the activation threshold for word selection, and the starting ratio between dominant and secondary meaning at the start of the simulation. Upon updating of the import/export data module, all attribute weights are reset based on the new information.

#### **3.1.6 Parameters Module**

This module allows the manipulation of the settings of the primary simulation module. It defines how many nodes get activated upon pressing a button in the priming module, how long nodes stay activated upon priming, how many nodes there are in the simulation, how much a weight increases on priming, how much the activation multiplier increases on priming. Upon updating of the parameters module, all attribute weights are reset based on the new information.

### **3.2 Concurrent Process Decomposition**

Electron uses a main process and renderer processes. No two renderer processes can talk to each other directly; they have to go through the main process. The main page is our application's main process, while the parameters, tutorial, and import/export pages are renderer processes.

### **3.3 Data Decomposition**

All the the data for this program will be stored in the form of JavaScript variables. This means that it resides in volatile memory/RAM on the user's computer. Therefore, any data that is not saved using the export module, will be lost upon termination of the program.

### **3.4 Design Constraints**

Must work on all common platforms including Linux, Windows, and OS X. To accomplish this, we will use electron packager (<https://github.com/electron-userland/electron-packager>) It is expected that the finished product takes less than 200 MB so that all versions could be included on a CD-ROM.

## **4. Dependency Description**

### **4.1 Inter-module Dependencies**

The Parameters and I/O modules change simulation variables housed in Main. Prime Dominant and Prime Secondary change node variables, which are also housed in Main.

The tutorial is accessed via a link on the main page, but it changes nothing in any other module. It is just an information page.

### **4.2 Inter-process Dependencies**

There is a timer running constantly in the background, executing an algorithm on an array containing all recently-activated attributes. This timer decays their activation at a constant rate until they are completely inactive. This timer must be paused during the priming module's meaning selection process, as that process involves live weights and this process manipulates live weights.

## **4.3 Data Dependencies**

### **4.3.1 User Attribute Interaction**

When the user interacts with the attributes in the program, the attributes will need to be stored so that they can be displayed in the program and the user can save the data to be viewed later. Saving the data is a stretch goal, along with file I/O. At this point, displaying the data is enough.

## **4.4 To Be Implemented**

### **4.4.1 Imported Data**

Imported data will be a dependency for the program because if the user wants to use a specific case they they will need to supply it otherwise it will just use a base case which is all of the nodes set to neutral.

### **4.4.2 Exported Data**

Data will be exported in the form of a CSV file, openable in Excel. The file will contain the priming sequence along with every node's weight, activation level, and the current ratio between dominant and secondary meaning.

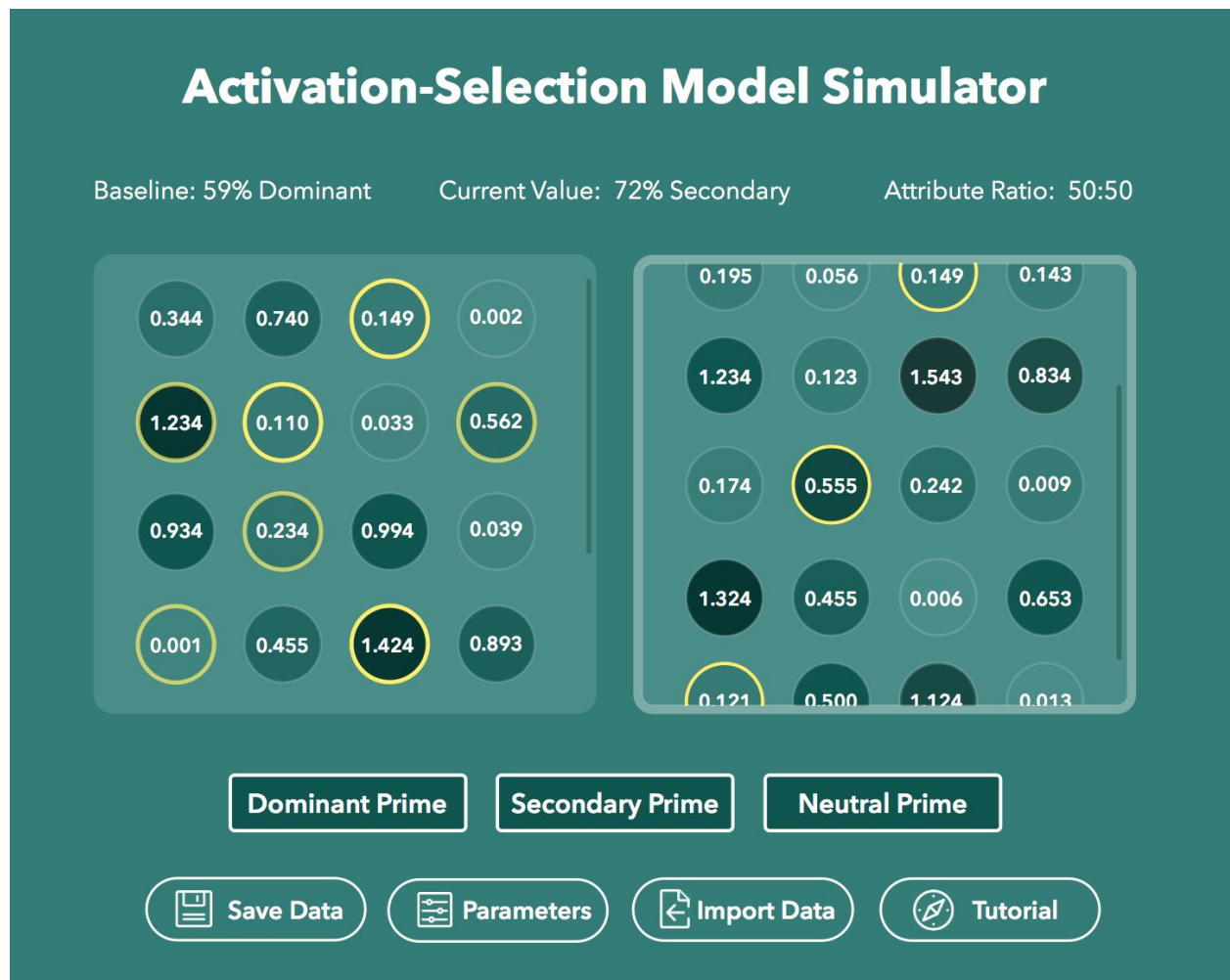
## **5. Interface Description**

## **5.1 Module Interface**

### **5.1.1 Main Window**

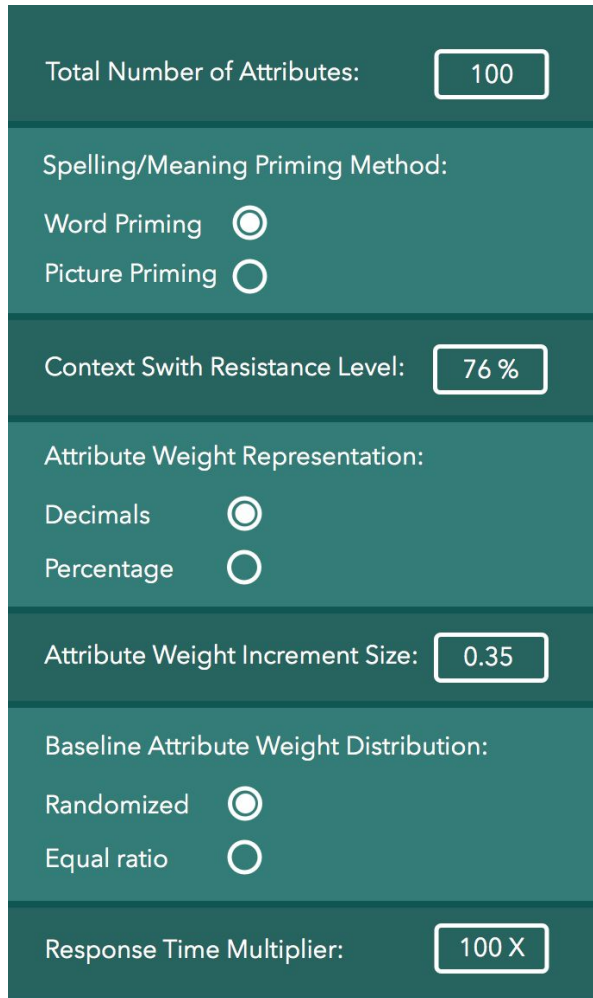
The main window should contain the data representation of the program as well as buttons to access other modules. The picture

below is the initial sketch and is subject to change upon customer recommendation. This window will also have a selector for word pairs that are read from the input file. In addition, it will show the current sequence of priming that the user has executed.





### 5.1.2 Parameters Window



The Parameters Window is a dark teal interface with white text and controls. It contains several sections, each with a label and a set of controls. The controls include text input fields, radio buttons, and a percentage input field.

Total Number of Attributes:	<input type="text" value="100"/>
Spelling/Meaning Priming Method:	
Word Priming	<input checked="" type="radio"/>
Picture Priming	<input type="radio"/>
Context Switch Resistance Level:	<input type="text" value="76 %"/>
Attribute Weight Representation:	
Decimals	<input checked="" type="radio"/>
Percentage	<input type="radio"/>
Attribute Weight Increment Size:	<input type="text" value="0.35"/>
Baseline Attribute Weight Distribution:	
Randomized	<input checked="" type="radio"/>
Equal ratio	<input type="radio"/>
Response Time Multiplier:	<input type="text" value="100 X"/>

### 5.1.3 Tutorial Window

This screen would contain an HTML based tutorial on how the program works, what each button or parameter controls and the format that is accepted for input as CSV files.

## 5.2 Process Interface

No process interface at this time.

## 6. Detailed Design

### 6.1 Module Detailed Design

- **randomizeWeights**

Called in initial setup of nodes

Input: initial number of nodes, node arrays

Puts nodes in 2 groups of equal number

Assigns a random number between 0 and 1 to each node, so that in the end the 2 groups have an equal weight among all nodes.

- **primeDominant**

Takes number of nodes to activate  $m$  (defined in settings module) and number of increase to weights  $n$  (also defined in settings module).

If  $\text{currentNodeWeight} + n > 1$ ,  $\text{currentNodeWeight} = 1$ .

Else,  $\text{currentNodeWeight} += n$  for all  $m$  randomly selected.

Random selection is such that dominant weights are more likely to be chosen than secondary weights.

- **primeSecondary**

Takes number of nodes to activate  $m$  (defined in settings module) and number of increase to weights  $n$  (also defined in settings module).

If  $\text{currentNodeWeight} + n > 1$ ,  $\text{currentNodeWeight} = 1$ .

Else,  $\text{currentNodeWeight} += n$  for all  $m$  randomly selected.

Random selection is such that secondary weights are more likely to be chosen than dominant weights.

- **activateAttribute**

Takes: index of node to activate: 0 to  $n$  (dominant),  $n+1$  to  $2n$  (secondary)

Javascript: lights up borders of nodes to indicate visually that they have recently been activated and that they are more likely to be activated again if priming is done in the near future. Also increases activated nodes' probability to be activated by `primeDominant` or `primeSecondary`, regardless of whether the node is dominant or secondary.

## **6.2 Data Detailed Design**

Data will be stored in the form of local and global variables in JavaScript. There will also be files created from the Export modules. Depending on implementation, storage of simulation parameters is subject to change.